THE DEVELOPMENT OF DATA-DRIVEN METHODS FOR MODELLING AND OPTIMISATION OF CHEMICAL PROCESS SYSTEMS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2022

Max Mowbray
Department of Chemical Engineering

Contents

Word count 77887 words

Abstract

In this thesis, data driven approaches to sequential decision making problems within process systems engineering (PSE) are developed. Specifically, the use of model-free Reinforcement Learning (RL) is considered for process control, online optimisation, online production scheduling and supply chain management problems. Model-free RL methods are purely data-driven approaches to identifying an optimal control policy for an uncertain, decision process. These policies are identified independently of assumptions on system dynamics and associated uncertainties. Incentives for the use of RL and its challenges in application to PSE is presented in Chapter 2. These challenges include: improving the sample efficiency of policy identification; ensuring safety through satisfaction of operational constraints; as well as robustness via risk-sensitive decision making. A framework for the use of RL is also proposed, which all of the work items in this thesis adhere to. This framework relies on initial policy identification via simulation of an approximate, uncertain system model; and then subsequent transfer of the policy to the real system for online decision making and policy improvement.

In Chapter 3, we explore how best to leverage existing process knowledge expressed by process operators and control schemes in the form of process data to aid offline policy learning. We propose a methodology to first extract a parameterisation of this knowledge in an offline simulation model in the form of a control policy. This removes the requirement to tune the policy manually and improves learning efficiency. This policy parameterisation is then transferred to the real process to provide online control and for subsequent policy improvement. A case study is provided via a tracking problem in a linear, uncertain dynamical system and existing data is provided by a proportional-integral-derivative controller.

In Chapter 4, an entirely data driven methodology is proposed to ensure the probabilistic satisfaction of state constraints in online optimisation of uncertain, nonlinear process systems. The approach is benchmarked to a nonlinear model predictive control scheme on a lutein photo-production process, demonstrating improvements in constraint satisfaction and 30% improvements in the expected performance.

In Chapter 5, a zero-order optimisation approach to RL policy identification is proposed for online scheduling of an uncertain sequential production environment. The approach is able to robustly handle common restrictions on these problems. Additionally, the framework inherits the benefits of posing risk-sensitive formulations, such as optimising for the conditional value-at-risk (CVaR). The method is benchmarked to an online mixed integer linear programming formulation and is demonstrated to be competitive with a performance gap of at most 5%, but identifying online decisions orders of magnitude more efficiently.

Finally, in Chapter 6 we explore the application of a zero-order RL framework to an uncertain multi-echelon supply chain, inventory management problem. We benchmark the approach to a popular first-order RL method. We highlight the relative sample efficiency of our method and demonstrate improved performance in the objective. Benchmark is also provided to mathematical programming, with the method demonstrating competitive performance in the objective, but gaining the ability to incentivise worst-case performance by constraining the CVaR.

As described, the work explores the development of RL methodologies and tailors them specifically to PSE problems. Additionally, many of the open challenges associated with the use of RL are addressed. Conclusive summary is provided in Chapter 7.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy, in any relevant Thesis restriction declarations deposited in the University Library, and the University Library's regulations.

Acknowledgements

This thesis is dedicated to my family, friends and supervisors, and has been made entirely possible through their support and advice. In particular, I would like to thank Dr. Dongda Zhang, Dr. Ehecatl Antonio Del Rio Chanona and Prof. Robin Smith for their investment into my development. The experience of the last three years has been enriched by their dedication to academia and learning, as well as their friendship. Additionally, I would like to thank Dr. Panagiotis Petsagkourakis for his encouragement and collaboration, but more importantly for a number of coffees.

I would also like to thank the members of Dongda's group at The University of Manchester including Bovinille Anye Cho, Fernando Vega Ramon, Alex Rodgers, Alex Norman, Harry Kay and Sam Kay; also Antonio's group at Imperial College London including Ilya Orson Sandoval, Thomas Savage, Zhengang Zhong, Damien van de Berg, Akhil Ahmed, Miguel Angel de Carvalho Servia and Haiting Wang. Our collective group meetings have been one of the most enjoyable parts of my week for nearly three years.

Additionally, to Dr. Giannis Zacharopoulos and soon to be Dr. Bovinille Anye Cho for their friendship. Our memories of fun in Manchester at the end of a busy week will stay with me. I am also thankful to the rest of the Centre for Process Integration at The University of Manchester and particularly Taicheng Zheng for interesting conversations shared within and outside of the office.

Completion of this work could not have been possible without Campbell Minogue, John Pilgrim, and especially Madison Brownley. Time spent with you provided valuable distance from study as well as plenty of fun.

Finally and most importantly, I would like to thank my mother and sister, Elizabeth and Imogen, for their unconditional love, support and perspective. The last few years have been full of life for all of us, and I have been lucky to experience them with you.

List of Tables

List of Figures

List of Symbols

This section describes the notation used throughout the chapters of this thesis. If notation is redefined between different chapters then it is included here for reference.

Chapter 2

$\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$	The physical system states
$\mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$	The control inputs
$t \in \{0, \ldots, T\}$	Index within the finite discrete time horizon of length T
$\varphi : \mathcal{X} \times \mathbb{U} \times \mathcal{X} \to \mathbb{R}$	Stage cost function
γ	Discount factor
G_0	Discounted sum of stage costs
$\boldsymbol{\tau}$	Process trajectory
$\mathbf{x} \in \mathbb{X} \subseteq \mathcal{X}$	The physical system states that satisfy state constraints
$\hat{\mathbb{U}}(\mathbf{x}) \subseteq \mathbb{U}$	The control inputs that satisfy state dependent input constraints
$f_r(\mathbf{x}, \mathbf{u}, \cdot)$	The hypothesised real process dynamics
$f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_s} \to \mathbb{R}^{n_x}$	The approximating discrete time process dynamics
$\mathbf{s} \in \mathbb{S} \subseteq \mathbb{R}^{n_s}$	General uncertain model parameters
$\pi : \mathcal{X} \to \mathbb{U}$	A feedback control policy
$\mathbb{E}[Z]$	Expected measure of arbitrary random variable, Z
$V_\pi : \mathcal{X} \to \mathbb{R}$	State value function
$Q_\pi : \mathcal{X} \times \mathbb{U} \to \mathbb{R}$	State-action value function
$\mathcal{T}_\pi(\cdot)$	Bellman operator
$\mathcal{T}_{\pi^*}(\cdot)$	Bellman optimality operator
α =(0,1]	The learning rate
$\delta(\cdot)$	The temporal difference error
$\delta^{QL}(\cdot)$	The Q Learning error

$\delta^{SA}(\cdot)$	The state-action-reward-state-action (SARSA) error
$L(\cdot)$	The squared Q learning error
$\theta \in \mathbb{R}^{n_\theta}$	Parameters of a function approximation
$\boldsymbol{\phi} : \mathcal{X} \times \mathbb{U} \times \mathcal{X} \to \mathbb{R}^{n_w}$	A set of basis functions
$\mathbf{w} \in \mathbb{R}^{n_w}$	A vector of *learnable* weights
$\boldsymbol{\upsilon}(\boldsymbol{\tau})$	Basis feature counts
$\mathbf{r}(\cdot)$	Testable information

Chapter 3

$\mathbf{y} \in \mathbb{R}^{n_y}$	Noisy observation of physical system state
$p(\mathbf{y} \mid \mathbf{x})$	The exact description of noisy observation
R_t	The reward allocated (by the cost function)
$p(\boldsymbol{\tau} \mid \theta)$	Probability density function describing the process trajectory under a policy
$\bar{\rho}(\mathbf{x}_0)$	Initial state distribution
$J(\tau)$	Expected trajectory cost
$\nabla_\theta J(\tau)$	The policy gradient
b	Policy gradient baseline
$\boldsymbol{H}$	Hidden state of LSTM neural network
$\boldsymbol{\alpha} \in \mathbb{R}^{n_\alpha}$	A vector of *learnable* cost function weights
$\boldsymbol{T}$	Existing dataset of process trajectories
ξ	Dataset acquired through sampling the approximate process model via Monte Carlo under a control policy
$h(\cdot)$	Linear process dynamics
$\delta(\cdot)$	Magnitude of disturbance
W	Wiener process
C_A	Concentration of reagent
T	Temperature of reactor
φ_i	Basis function of type i
C_{A0}	Influent concentration of reagent
T_E	Temperature of heated jacket

Chapter 4

$\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$	System state

$\hat{\mathbb{X}} \subseteq \mathbb{X}$	Set of states that satisfy state constraints
$\boldsymbol{\upsilon} \in \mathbb{R}^{n_x+n_u}$	State and control pair
$\boldsymbol{\Upsilon}$	Dataset of state and control pairs
$\mathbf{Y}$	Dataset of (future) state components
ω	Additive white noise
$f_{GP}(\cdot)$	Gaussian process prior
K	Gram matrix
K_*	Covariance of a test datapoint
α	User defined probability of joint chance constraint violation
$\mathbb{P}(\cdot)$	Probability of event
ι_j	Probability of violation of constraint j
$\boldsymbol{\mu}(\cdot)$	Mean of Gaussian process state space model posterior
$\boldsymbol{\Sigma}(\cdot)$	Covariance of Gaussian process state space model posterior
$\bar{\mathbf{x}}$	Nominal system state
$\Sigma[\mathbf{x}]$	Covariance of system state
$\epsilon_{j,t} \in \mathbb{R}$	A Cantelli-Chebyshev backoff
$\boldsymbol{\xi} \in \mathbb{R}^{n_g}$	Constraint tightening multipliers
$\bar{\mathbb{X}} \subseteq \hat{\mathbb{X}}$	Tightened set of states that satisfy state constraints
$\varphi_p(\cdot)$	Penalty function solving for tightened constraint set
$\kappa \in \mathbb{R}$	Constraint violation (penalty function) weighting
$\zeta \in \mathbb{R}^{n_x \times n_x}$	Penalty weighting for Gaussian process state space model uncertainty
π_C	Constraint satisfying policy
$F_X(\cdot)$	Cumulative distribution function
$F_{SA}(\cdot)$	Empirical cumulative distribution function
$F_{LB}(\cdot)$	Robust empirical cumulative distribution function
$\boldsymbol{U}$	Least squares in desired joint chance constraint satisfaction
$J_{BO}(\cdot)$	Objective for desired constraint tightening
$\beta = [0, 1]$	Hyperparameter for J_{BO}

$c \in \mathbb{R}^+$	Hyperparameter for J_{BO}
$\sigma_J \in \mathbb{R}$	Variance in policy objective performance
$f_{AF}^{EI}(\cdot) \in \mathbb{R}$	Expected improvement acquisition function
Chapter 5	
$Z \sim p_\pi(z)$	Cumulative sum of rewards (a random variable described by a probability density function)
$\mathbb{X} \subseteq \mathbb{R}^{n_x}$	Set of physical system states
$\mathbb{U}^{(l)} \subseteq \mathbb{Z}$	Input constraint set for unit l
$\mathbf{w} \in \mathbb{W} \subset \mathbb{R}^{n_u}$	Relaxed, continuous equivalent to discrete control space
$\bar{\mathbb{U}}(\mathbf{x}) \subseteq \mathbb{U}$	Those controls, which satisfy constraints identifiable via propositional logic
$f_r : \mathbb{W} \to \bar{\mathbb{U}}$	Rounding policy to map from continuous relaxed control space to discrete controls that satisfy propositional logic constraints.
$g(\mathbf{u})$	Constraints on control inputs not handled via f_r.
$\phi(\cdot)$	Penalty function
$Z^\phi(\cdot)$	Cumulative sum of penalty function allocations
κ_g	Penalty function weight
$F_\pi(z^\phi)$	Cumulative distribution function associated with policy performance
z_β^ϕ	Value-at-risk defined for a given probability level, β
μ_β^ϕ	Conditional value-at-risk defined for a given probability level, β
Z_{MC}^ϕ	Evaluations of policy performance
π_β	Risk-sensitive policy
I_i	Inventory associated with product i
ω_j	the task/product index processed within unit j over the previous time interval
δ_j	A forecast of remaining processing time of production task in unit j
ρ_i	A forecast of discrete time indices remaining until orders (tasks) are due

Chapter 6

$\mathbf{x} \in \mathbb{X}$	State space
$\mathbf{u} \in \mathbb{U}$	Control space
$P : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to [0, \infty]$	State transition probability matrix
$R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}$	Reward function
G_t	Cumulative sum of rewards observed from time index t
$F_{fitness}(\cdot)$	Fitness function to evaluate the policy
V_α	Value-at-risk associated with policy performance for probability level $1 - \alpha$
$CVaR_\alpha$	Conditional value-at-risk associated with policy performance for probability level $1 - \alpha$
D	Evaluations of policy performance
$V_\alpha(D)$	Monte Carlo estimator for value-at-risk
$CVaR_\alpha(D)$	Monte Carlo estimator for conditional value-at-risk

Chapter 1

Overview

This thesis investigates the development of data-driven and machine learning methods for the modelling and optimisation of chemical process systems.

1.1 Connecting Machine Learning and Process Systems Engineering

The application of machine learning techniques is a key research direction within the process systems engineering, control and operations research communities. The establishment of machine learning within these fields has been assisted by the engagement of the public with paradigm redefining achievements in the fields of computer vision and decision making, as well as its integration with conceptual frameworks such as Industry 4.0. Industry 4.0 is billed to play a significant part of the fourth industrial revolution (Thomas and Nicholas, 2018). First proposed by the German manufacturing community (Lasi et al., 2014), this conceptual contribution motivates the use of process data, digitization and automation in achieving a more sustainable and resilient framework for industry. Machine learning techniques comprise a key component of this concept, primarily because of their capacity to identify, parameterise and make predictions based on patterns in potentially highly dimensional process data (Fuentes-Cortes et al., 2022). The following explores the fundamentals of machine learning and provides comment on its prospective use in chemical and process systems engineering (PSE).

The identification of patterns within data has been standard practice within the chemical engineering community since the early 1800s and the initial lectures provided by George E. Davis at the University of Manchester in 1887 (Reed, 2020). This has typically been achieved either using first principles reasoning, the construction of semi-empirical expressions (often derived from the use of dimensional analysis) or classical system identification methods. Instead, machine learning approaches the problem of pattern identification through a black-box approach, typically by combining statistical estimation procedures with highly flexible classes of nonlinear models. More specifically, machine learning can be divided into three primary subfields: supervised learning, unsupervised learning and reinforcement learning. Other subfields, such as semi-supervised and contrastive learning exist, however, discussion of these methods is out of the scope of this overview.

Supervised learning is a subfield that focuses on identifying a function mapping between an input and output space, given a number of noisy observations of the true underlying mapping. Appropriate identification of the defining model parameters may proceed via either Frequentist or Bayesian estimation techniques as usual (Oden, 2018). Traditionally, this is known as the inverse problem, where model construction procedures are deployed to identify an approximation of the mapping concerned (De Vito et al., 2005). Common examples of supervised learning include image classification, which has been applied in the setting of healthcare diagnostics (Fujisawa et al., 2019); time-series forecasting, which is a common problem in modelling the dynamics associated with process systems (Bradford et al., 2018; Wei et al., 2022; McBride and Sundmacher, 2019); soft-sensing (Sun and Ge, 2021; Ge et al., 2017); and data-driven optimisation (Demirhan et al., 2020; Beykal et al., 2022; van de Berg et al., 2022). The major attraction for the use of ML in these settings is the lack of explicit knowledge regarding the underlying system. For example, there is no map that can be derived from first principles reasoning that will classify whether or not a collection of pixels displays evidence of a tumour. However, one can use human expertise as well as chemical and biological diagnostics to help label images and then identify such a map in a black-box manner (Banbury et al., 2019; Horgan et al., 2021). Similarly, in industrial process systems it is often difficult to measure certain process variables, an example is estimating the viscosity of a non-Newtonian fluid via conventional process instrumentation

(Haroon et al., 2020). In the case that these process variables require monitoring, it becomes crucial to estimate them. If sufficient mechanistic knowledge is unavailable, this can be achieved by a black-box approach, as demonstrated in (Memarian et al., 2021). The flexibility of parametric modelling together with statistical estimation techniques can also allow one to consider robust formulations that, for example, account for outliers (Sadeghian et al., 2022). The example of modelling time series sequences essentially reduces to identifying discrete-time process dynamics. This can be challenging, especially in processes characterised by complex reaction kinetics, such as fermentation systems. In these cases, the reaction mechanism is fully described by the possibly thousands or millions of reactions associated with cell metabolism. This has led to the historic dependence on semi-empirical model structures such as the Monod model (Koch, 1998; See et al., 2018). These structures are lumped approximations to the process kinetics, which are often capable of describing the process over a range of process conditions, but are often subject to mismatch (Meraz et al., 2022). In these cases, it is desirable to use black-box or hybrid machine learning approaches to capture what is not well described by the mechanistic approach (Bradley et al., 2022). Supervised learning has also been used in data driven optimisation frameworks. The general idea here is to build a surrogate model of the function one wants to optimise or of an unknown constraint (van de Berg et al., 2022; Beykal et al., 2020). This may help overcome complex or intractable model-based optimisation problems (Dias and Ierapetritou, 2020). The construction of these models may utilise parametric and non-parametric methods including, for example, artificial neural networks (LeCun et al., 2015) and Gaussian processes (Williams and Rasmussen, 2006), respectively.

Where supervised learning is interested in identifying input-output mappings, unsupervised learning is instead interested in identifying statistical relationships between data points within a given space. Often these relationships are exploited to identify lower dimensional[1] representations of the original input data. Such representations can then be used for analysis of the input data or in the creation of subsequent models. For example, principal component analysis (PCA) is commonly thought of as a form of unsupervised learning. PCA has been used within the process industries for statistical process analysis and monitoring (Nomikos and MacGregor, 1994), and also

[1]These are also known as latent representations.

within the context of reduced order modelling where one is interested in identifying lower dimensional models of a dynamical system (Audouze et al., 2009). The main idea for the use of PCA within process systems monitoring is that the data that processes generate is generally highly dimensional with process variables of different orders and variation. Hence by using PCA, one is able to project this information into a low dimensional latent space, which a) preserves the information expressed in the original data space, and b) enables analysis of the reduced data representations via visual inspection and statistical analysis (Jiang et al., 2019). For example, Hotelling's T^2 test is often used to characterise whether or not the process is moving away from the on-specification distribution of data and as a result whether or not there has been a shift in dynamics or a fault (Thebelt et al., 2022). PCA is also highly interpretable given that it is essentially a linear transformation of the original data. More recent work has investigated the use of generative modelling and generative adversarial networks to make predictions as to whether a process has shifted or there is a fault, directly in the original data space (Qin and Zhao, 2022). This approach is likely to be an effective alternative to PCA-based monitoring when the process data is non-Gaussian in the reduced space. However, this comes generally at the price of model interpretability.

Reinforcement learning (RL) is the final subfield that we will discuss here. RL has generated excitement within the public domain, particularly due to its achievements in sequential decision making under uncertainty problems. As a subfield, it presents an avenue to learn an optimal decision policy in a purely data-driven manner (Sutton and Barto, 2018a). Furthermore, many RL algorithms promise to achieve this independently of any explicit knowledge about the dynamics of the system, provided one can gain observations of how the discrete-time system state responds to control inputs. This is particularly promising for PSE, given that generally, we are unable to identify perfect models of processes via finite-dimensional expressions and in many cases this does limit current model-based solution methods. For example, Google DeepMind recently demonstrated the development of the first super-human Go system, known as AlphaGo Zero (Silver et al., 2017b). The game of Go has long been designated as the perfect forum to demonstrate the potential of RL, primarily due to its complexity. It has of the order of 10^{170} different discrete system states and over 360 discrete control inputs (Silver, 2009). The AlphaGo Zero system utilised a set

of RL methods, together with Monte Carlo tree search and simulation to identify an approximately optimal parameterization[2] of a decision policy. Importantly, this policy acts conditional to the current state of the Go board, enabling the decisions to account for realizations of uncertainty derived from the counter moves of the opposing player. There are many instances of decision making under uncertainty within the process industries. For example, as will be outlined in the subsequent section, the areas of process control (Spielberg et al., 2019; Lawrence et al., 2022), online optimisation (Yoo et al., 2021a; Oh et al., 2022), as well as online production scheduling (Hubbs et al., 2020a; Zhang et al., 2020a) and supply chain management (Peng et al., 2019) are all instances of sequential decision-making problems under uncertainty. As a result, the potential impact of RL within the process industries is substantial and this has been demonstrated in a number of preliminary investigations (Lee et al., 2018a; Yoo et al., 2021a).

1.2 Decision-making in industrial process systems

Decision-making processes are present in almost all operations common to the process industries. Conventionally, industrial decision-making is thought of according to a hierarchical framework, known as a distributed control system (DCS). Modern DCS have been developed in accordance with the international standards founded upon and including ISA-95 (ISA, 2022) and are widely applied in predominantly continuous, but also batch process systems. The decision-making levels outlined by the DCS include enterprise resource planning (ERP), manufacturing execution systems (MES), supervisory control and data acquisition (SCADA), programmable logic controllers[3] (PLC), and field and process sensors.

All of the decision-levels mentioned generally deal with multi-stage or sequential decision making problems, which necessitate interaction with the system of concern at discrete instances in time. ERP is concerned with business level decisions that primarily pertain to the planning interactions within the supply chain as well as management of risk, sales, governance, finances and human resources (Shehab et al., 2004).

[2]The implementation used an artificial neural network as a mapping from states to control inputs.
[3]PLC are solid state computers that execute the control system's decisions based on observations of the process.

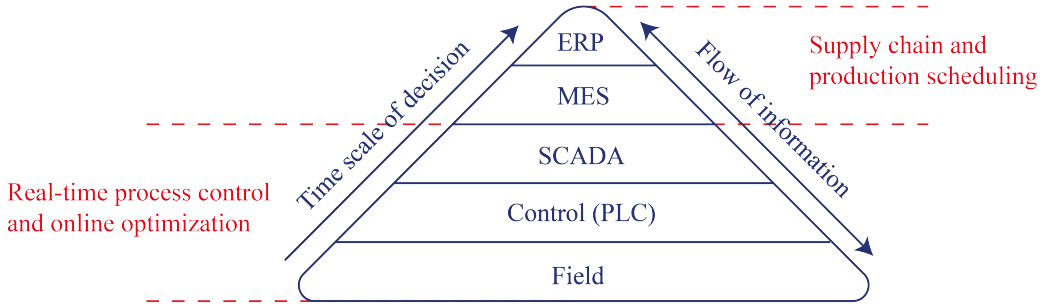


Figure 1.1: The structure of industrial distributed control systems, which consists of enterprise resource planning (ERP), manufacturing execution systems (MES), supervisory control and data acquisition (SCADA), control and sensing structures.

Whereas, MES is focused on ensuring productive transformation of raw materials to marketable products within and across plants (Saenz de Ugarte et al., 2009). From the perspective of process systems engineering (PSE), ERP and MES can be jointly and more coarsely thought of as those decision-making functions that dictate a business' supply chain and production scheduling operations. Meanwhile, the latter levels of SCADA, PLC and field and process sensors can be thought of as those systems, which enact process control and online optimisation (Trentesaux, 2009). The respective levels and their relations are expressed by Fig. 1.1.

As Fig. 1.1 details, there is heterogeneity between the timescales upon which decisions are required at the respective levels. Typically, ERP decisions occur on the scale of weeks to months; MES at days to weeks; and, milliseconds to hours across the SCADA, PLC and field levels (Darby et al., 2011). There is also is a degree of information flow between each layer. For example, in process control structures measurements of the physical process inform PLCs to provide fast acting supervisory control. Other information passed between layers may relate to, for example, process economics (Grossmann, 2005), production targets (Maravelias and Sung, 2009) or other physical parameters of the system (Uraikul et al., 2007).

Information flow is a key requirement for decision-making. From the perspective of decision theory, the decision-making process is often thought of as the identification of a function mapping from an observable variable to a decision variable. The most advanced methods in PSE, achieve this mapping via numerical optimisation. Information flow between decision-making layers can therefore either constitute the observable variable or to inform the construction of the decision mapping itself. However, it

should be noted that decisions are not necessarily made immediately upon receipt of new information. For example, the MES layer typically does not continuously update production plans based on feedback from the plant. Determining when to compute decisions, especially from higher level functions is a central theme in PSE research. For example, identifying optimal, event-based production planning (Okpoti and Jeong, 2021), rescheduling (Gupta et al., 2016) and control policies (Åarzén, 1999; Aström, 2008; Liu et al., 2014) has been well explored to approach the ambiguity regarding the frequency with which control decisions should be provided.

This decision theory interpretation discussed above also allows for many other PSE problems (beyond those constituting the DCS) to be viewed as decision-making problems. For example, both mechanistic model structure and parameter identification, as well as the manual tuning of controllers (as often assisted by an operator's process knowledge) are both decision-making problems. As we will see, the identification of decision rules is central to the motivation of all of the research objectives presented in this thesis.

In the following subsections, we will explore the high level ideas that constitute real-time process control, online optimisation, production scheduling and supply chain operations.

1.2.1 Process control and online optimisation

The enactment of process control is generally handled by the lower three levels of the DCS outlined in Fig. 1.1. For continuous and batch process systems, these three layers are commonly constituted by real-time optimisation (RTO), model predictive control (MPC) and supervisory control structures such as proportional-integral-derivative (PID) controllers[4] (Darby et al., 2011).

A traditional RTO layer provides a mapping (i.e. a decision rule) from information pertaining to process economics and plant variables to optimal process set points of controlled variables. In order to do this, RTO leverages nonlinear programming by incorporation of finite dimensional descriptions of the steady state plant, and constraints imposed upon it[5]. Optimisation algorithms can then be deployed to identify

[4]It should be noted that alternative structures are also used within the (bio)chemical process industries.

[5]There are other forms of RTO, which are not discussed further here. We direct the interested

a decision variable that satisfies the Karush-Kuhn-Tucker (KKT) conditions under an economic objective (Darby et al., 2011). Such an objective may correspond to plant operational cost, an appropriate minimiser of which is widely known as a KKT point. After identification by the RTO layer, the KKT point is fed to a regulatory control structure. This is typically either an MPC module combined with lower level PID control or simply PID control. Both MPC and PID modules track the optimal process operating conditions online, and react to uncertain process dynamics on the basis of state feedback (Rawlings et al., 2017). Similar to RTO, MPC represents a decision rule as expressed by a nonlinear programming model, however it makes use of a finite dimensional description of system dynamics and a tracking objective. If an MPC scheme is not implemented, typically lower level regulatory control is instead enacted by a PID controller that identifies control actions to meet the new set point. The identified regulatory control is then implemented through field instrumentation, e.g. a valve actuator. Once the plant has reached a new steady state or a batch has terminated[6], plant measurements together with renewed process economics are used to update the decision mapping of the RTO layer (Krishnamoorthy and Skogestad, 2022). The manner in which these measurements are used to update RTO decisions is an active area of research (Chachuat et al., 2009; Marchetti et al., 2016; del Rio Chanona et al., 2021), discussion of which is beyond the scope of this thesis - please see Krishnamoorthy and Skogestad (2022) for more information. However, the effective utilisation of plant state feedback is certainly a common feature with MPC.

The shared conceptual ground between RTO and MPC has led to the development of an alternative online optimisation framework known as economic MPC (EMPC) (Ellis et al., 2014). EMPC attempts to directly optimise an ongoing process online to maximize an economic objective via the use of state feedback. This enables one to merge RTO and MPC layers by identifying process set points and conducting predictive control within one decision layer. This has been well detailed in Ellis et al. (2014) and examples within the batch processing paradigm have been provided by Markana et al. (2018); Bradford et al. (2021a); Kim et al. (2022). Major efforts within

reader to (Krishnamoorthy and Skogestad, 2022)

[6]The conditions used to trigger a new RTO iteration detailed here correspond to continuous and batch processing, respectively. In the case of batch processing, RTO is often viewed as 'batch to batch optimisation'.

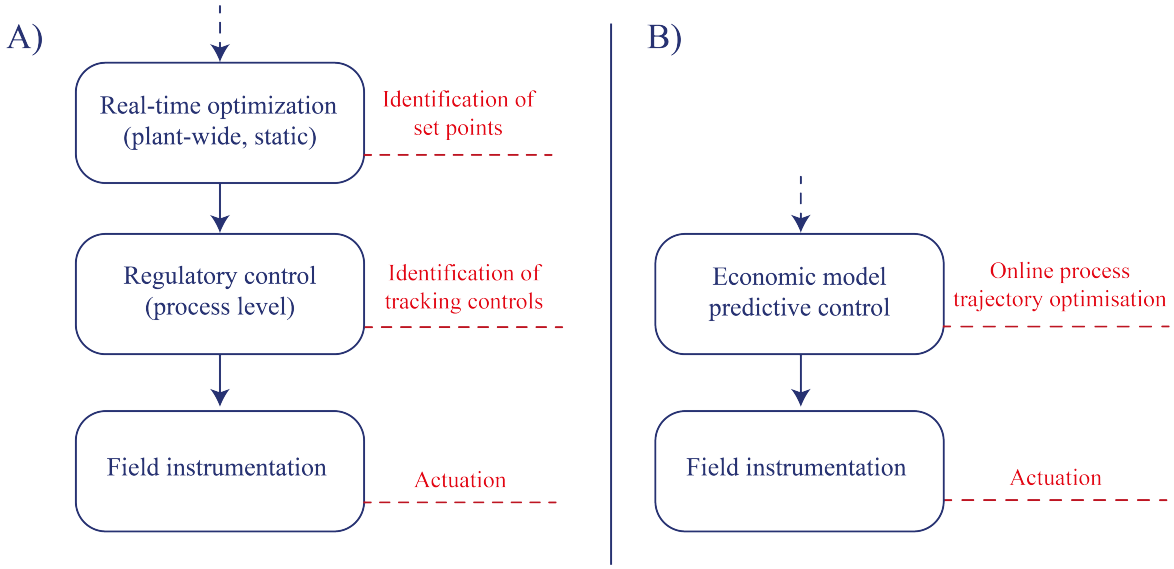


Figure 1.2: The structure of established frameworks for real-time control and optimisation of industrial process systems. A) The traditional real-time optimisation frameworks, and B) the more recently proposed economic model predictive control framework. The arrows with dashed lines indicate vertical integration with higher planning and scheduling functions.

EMPC research have been directed towards providing control-type stability and performance guarantees given certain assumptions on the underlying process dynamics and uncertainties. However, the major practical benefit of the EMPC approach is that it can account for potentially time-varying process economics and utilise process state feedback[7] within one mathematical programming model. This is something RTO hierarchies based on static process optimisation fail to achieve directly.

Once one has identified optimizing control inputs via the EMPC decision layer, the relevant information is fed to field and control instrumentation associated with a process. The structural interactions of the two real-time optimisation and control frameworks outlined within this section are detailed by Fig. 1.2.

1.2.2 Production scheduling and supply chain operations

Production scheduling and supply chain decision making problems constitute much of the upper two levels of the DCS (ERP and MES). These problems can be more broadly separated as long-term strategic planning; medium term tactical production

[7]Informally, state feedback is a beneficial mechanism because it contains often un-modelled information regarding realizations of process uncertainty and disturbance. This allows one to identify control decisions according to existing process knowledge and new information descriptive of process uncertainties.

planning; and, short term scheduling operations (Maravelias and Sung, 2009).

Generally, strategic operations are focused on determining the best operational structure for a supply chain. Such a problem may consider the placement of new production and transportation infrastructure based on that which is already existing, geographical locations of raw materials, permits to access these raw materials, and forecasts of costs and revenues that could be observed for a given supply chain design (Cafaro and Grossmann, 2014). These decision problems are typically constructed on a timescale of years, and 'solved' within an offline decision-making paradigm (i.e. without chance for recourse decisions to be made). This problem class is generally constituted by mixed integer programs (MIPs).

Whereas more medium term supply chain operations consider the tactical assignment of production targets to facilities or stages within an existing supply chain infrastructure. For example, in the case of an oil and gas supply chain, allocation decisions about production rates from a particular field may be informed by medium-term forecasts of oil and gas prices, the implied depletion rates of the respective fields and the capacity of the available transportation networks (Attia et al., 2019). Clearly, the structure of the supply chain ultimately provides a constraint for the tactical planning, which is why strategic planning is essential. Further, coordination between each of the production nodes within the supply chain is absolutely required to ensure the supply chain can satisfy demand. If this is not achieved, phenomenon such as the *bullwhip effect* can often be observed (Forrester, 1997). Such a phenomena describes 'demand amplification' within the supply chain, where initial small imbalances or changes in supply and demand down the chain, propagate through its constituent stages, such that large imbalances are observed further up the chain (Wang and Disney, 2016). These decisions generally occur on a timescale of weeks to months, with mathematical programming approaches typically in the form of linear programs (LPs) or MIPs that utilise nominal point values for model parameters.

The shortest term planning functions observed within the process industries are production scheduling operations. The decisions identified assign production tasks to equipment items, as well as determine the sequencing and sizing of those tasks within the equipment. There are many different classifications of production scheduling problems, which ultimately reflects the heterogeneity in industrial production environments

(Harjunkoski et al., 2014). As a result, there exists a diversity of modelling and solution approaches. Furthermore, decision-making may be set within an online or offline paradigm. The existence of the online decision-making paradigm again reflects the presence of uncertainty, not only within plant dynamics (e.g. task processing times) but also variations in immediate product demand. Due to the timescale required for decisions in these problems, solution approaches in online production scheduling often involve the use of heuristic rules and occasionally the operational knowledge of plant staff (Letsios et al., 2021). This is primarily because the mathematical programming models descriptive of production scheduling problems take the form of mixed integer linear programs (MILP), which can often be characterized by large solution times. As will be discussed in the following chapters, there is large interest in the development of various frameworks that reduce the computational burden of online scheduling (Hubbs et al., 2020a).

In the following section, the motivation and objectives of the thesis are defined. The thesis contributions and structure are then outlined subsequently.

1.3 Motivation and objectives

In the previous sections, we discussed the fundamental sequential decision making problems common to PSE, the most applied solution approaches and the basics of modern machine learning. It is clear that modern solution methods based on mathematical programming ultimately require finite dimensional descriptions of uncertain process models. In reality, when we have uncertain variables influencing the system dynamics, that are described either by continuous or large discrete supports these problems at best become very large or become infinite dimensional. This means that typical model-based approaches are either required to make considerable approximation to the uncertain dynamics via stochastic programming, to handle the uncertainty robustly or to neglect the uncertainty altogether and utilise a nominal description of the uncertain parameters. Furthermore, computing a control decision online via optimisation has disadvantage in that either a) a control may not be identified in the time frame required due to the size or nature of the problem at hand, b) there is potential for the optimisation solver to fail, based on infeasibility in the underlying approximate

model, and c) the development and maintenance of advanced optimisation schemes can often be expensive both in cost and technical expertise (Gopaluni et al., 2020).

In this thesis, we will look to explore the application and development of ML framework, with particular focus on RL, for application to a variety of PSE problems. RL has the potential to handle the three points mentioned above regarding exact mathematical programming because: a) it can identify optimal policy function approximations within a feedback control framework independently of explicit assumptions as to the form or propagation of uncertainty over the time horizon, this is explored in Chapter 2 and has some theoretical benefits in terms of optimality; b) because decisions are identified through predictions of a policy function, a control will always be identified and in a time frame orders of magnitude shorter than common optimisation formulations; and c) RL is based on intuitive model-free and data-driven update rules, which has the potential to automate controller tuning and remove the requirements for expertise heavy periodic recalibration. Additionally, the use of model-free update rules means that RL does not make approximation to the uncertain system, but rather requires samples from it. Typically, this means if there is enough data, the RL will not have to make approximation to the system or model uncertainty. Clearly, real-world data is expensive, but synthetic data could instead be gained from simulation of an uncertain process model. This is discussed more extensively in Chapter 2.

However, there are some outstanding research challenges associated with the use of RL. The first is the efficiency of RL policy learning. Generally, RL requires many *learning* (optimisation) iterations to identify an approximately optimal control policy. In this thesis, we will look to improve the efficiency of the policy learning process. Specifically in Chapter 3, we explore how to best use process knowledge expressed in process data from the action of existing control schemes and process operators in order to hot-start the RL process. Then, in Chapter 6, we demonstrate that in instances where the policy function approximation has a relatively small parameter space, metaheuristic and stochastic search optimisation methods provide improved efficiency in policy identification over well established RL methods such as the policy gradient.

The other areas of research focus in RL considered in this thesis include a) handling state and control constraints, and b) improving policy robustness. These two areas

are of interest to this work, because a) typical RL policy learning methods have no inherent means of satisfying state constraints; and b) typically, one optimises for the expected performance of an RL decision policy. Given that there will be a distribution of policy performances due to the uncertain nature of the underlying process, it is important to consider other risk-sensitive formulations. This is again explored in two chapters of this thesis. In Chapter 4, we explore handling state constraints within fed-batch process systems and in Chapter 5, we demonstrate a methodology for identifying risk-sensitive policies as well as handling common input constraints present in online production scheduling problems.

Further details of the research contributions follows.

1.4 Research contributions and thesis structure

The work is presented via the Journal format with permission from the supervisory team associated with the project. Chapter 1 and 2 provide introduction to the thesis, and background and literature review on the identification of decision rules within the process industries, respectively. The research conducted throughout the course of the period spanning from September 2019 to July 2022 is then summarised by four distinct chapters (3-6). High level description of these works is provided subsequently. Finally, the thesis is brought to a close in Chapter 7 with presentation of conclusions and directions for future work.

1.4.1 Chapter 3 - Research Objective 1

In Chapter 3, a novel research contribution on the abstraction of operational knowledge in process control is presented. In this work, we assume the availability of data pertaining to the enactment of a control task (e.g. a set point change) from an existing control scheme. Specifically, this data set is assumed to be constituted by measurements of the system state and control inputs at discrete instances in time. We propose an operational framework to extract the knowledge of that control scheme into a function parameterization of the control policy expressed within the data, and then to improve it further. This is enabled by two data driven methodologies known

separately as inverse RL (IRL) and RL. IRL is used to synchronously extract the control objectives guiding the action of the existing control scheme and identify a policy function parameterization of it. RL is then utilised to improve the policy function parameterization (i.e. the decision rule expressed by the existing scheme) further by learning on the real process system.

This research item is published in the American Institute of Chemical Engineers (AIChE) Journal, and is accessible via the following reference:

Mowbray, M., Smith, R., Del Rio-Chanona, E.A. and Zhang, D., 2021. Using process data to generate an optimal control policy via apprenticeship and reinforcement learning. *AIChE Journal*, 67(9), p.e17306.

1.4.2 Chapter 4 - Research Objective 2

Chapter 4 presents a novel research contribution on the development of a completely data-driven offline framework for the identification of a safe RL policy for online optimisation of constrained fed-batch process systems. The work utilises the uncertainty prediction from simulation of an offline Gaussian process state space model to synchronously identify a constraint tightening mechanism for state path constraints and handle process-model mismatch. This enables the *learning* of a safe RL policy offline, which is then deployed to optimise the real process online. The benefits of this approach are evaluated against a nonlinear model predictive control scheme in a computational study descriptive of an uncertain, fed-batch lutein photo-production process. This contribution is aimed at a) handling the lack of explicit mechanism to consider state constraints with RL and b) the uncertainty associated with model construction for bioprocess systems where little mechanistic knowledge exists.

This research item is published in the Computers & Chemical Engineering (CACE) Journal, and is accessible via the following reference:

Mowbray, M., Petsagkourakis, P., del Rio-Chanona, E.A. and Zhang, D., 2022. Safe chance constrained reinforcement learning for batch process control. *Computers & Chemical Engineering*, 157, p.107630.

1.4.3 Chapter 5 - Research Objective 3

In Chapter 5, a novel research contribution on the identification of an online scheduling policy is provided. In this work a parallel, sequential chemical production environment is assumed to be completely reactive to the decisions of the scheduling policy function. Uncertainty is present in the unit specific processing times of each of the production tasks as well as the due dates for shipping of the products to clients. We assume that this problem can be modelled as a Markov decision process and a novel stochastic search based RL methodology is proposed. The approach relies on the implementation of an additional logic based mechanism to ensure that control selection satisfies both disjunctive and precedence constraints, which are common restrictions on production scheduling problems. The benefits of this approach is that the resultant policy explicitly accounts for plant uncertainties in online decision making, and provides scheduling decisions via inference (orders of magnitude faster than the most efficient MIP models). Further, the framework enables the incentivisation of risk-sensitive measures, such as the conditional value-at-risk, in offline policy identification. This enables the identification of a scheduling policy, which is risk averse, something which is appealing in industrial practice.

This research item has been submitted to the American Institute of Chemical Engineers (AIChE) Journal. A revised version has been submitted for final decision. The paper is accessible on arXiv via the following reference:

Mowbray, M., Zhang, D. and Chanona, E.A.D.R., 2022. Distributional Reinforcement Learning for Scheduling of Chemical Production Processes. arXiv preprint arXiv:2203.00636.

1.4.4 Chapter 6 - Research Objective 4

In Chapter 6, a novel research contribution on the identification of a tactical planning policy for supply chain management is presented. In this work a multi-echelon supply chain inventory management problem is considered. A stochastic search based RL methodology is proposed to ensure coordination between the re-order policies of the constituent stages within the chain. This is in view of the potential for realization

of various phenomena such as the *bullwhip effect* if a sufficient amount of coordination is not achieved. Again, the framework enables the incentivisation of risk-sensitive measures, such as the conditional value-at-risk, in offline policy identification. This is highly desirable in industrial practice. The approach is benchmarked to a state of the art policy gradient method as well as mathematical programming. The method demonstrates improved sample efficiency over policy gradient methods and demonstrates the ability to pose risk-sensitive formulations, something which is generally not considered in mathematical programming approaches.

This research item has been submitted to the Industrial Engineering and Chemistry Research (IECR) Journal:

Wu, G., de Carvalho Servia, M. A., Petsagkourakis, P., Zhang, D., Del Rio Chanona, E.A., Mowbray, M., 2022. Distributional reinforcement learning for inventory management in multi-echelon supply chains, Submitted to Journal, 2022:

The paper was produced through supervision of a student (Guoquan Wu) who interned with the research group after completing his MSc at the Department of Chemical Engineering, Imperial College London. Max Mowbray is the corresponding author of this paper and responsible for the original draft.

Chapter 2

Background and Literature Review

In the following, common approaches to the identification of decision rules in the process industries will be outlined. Specific attention will be directed towards sequential decision-making under uncertainty problems. Major discussion will be directed towards methods, collectively classified as Reinforcement Learning (RL). RL is a major topic within this thesis, and has strong potential to provide solution to the class of problems of interest. As a result the basic theory and its application direct much of this Chapter.

2.1 Identification of decision rules

Currently, the most advanced and utilised approach to the identification of decision rules at all levels of the distributed control systems hierarchy is encompassed by mathematical programming. In the domain of process control and optimisation, for example economic model predictive control (EMPC) is typically reduced to iteratively solving a nonlinear programming problem (Nocedal and Wright, 2006; Bazaraa et al., 2013; Rawlings et al., 2017) at discrete instances through time. Similarly, supply chain planning and management operations, as well as production scheduling, typically utilise (mixed integer) linear programs (Papageorgiou et al., 2007; Lodi, 2010; Vanderbei et al., 2020; Maravelias, 2021a). The current state-of-the-art provided by mathematical programming approaches is based on a long history of development and application. However, its application to problems characterised by uncertainty, and either non-smoothness or nonlinearity in the underlying system can be challenging.

In this section, we explore the potential applications of Reinforcement Learning (RL) and the basic theory associated. As will be outlined through the course of this section, RL has large potential for its ability to identify decision rules in a purely data-driven manner. This enables the decision rule to flexibly account for uncertainty in general discrete-time dynamics. However, its practical application, and hence algorithmic development, is still in relative infancy. Discussion will also explore methodologies that can be used to extract existing decision rules from existing industrial datasets. More immediately, however, the attention of the review explores the construction of general sequential decision making problems observed within the process industries.

2.1.1 Sequential decision making problems and uncertainty

Sequential decision making problems arise frequently in the process industries. Unlike single-stage decision making problems, such as model parameter estimation, sequential decision making problems are concerned with identifying multiple decisions that optimise an ongoing process through time (Bertsekas et al., 1995; Sutton and Barto, 2018b). Almost all of the decision problems that constitute the hierarchical decision structure are sequential in nature. For example, in production scheduling, one is concerned with allocating production tasks to available equipment items at discrete time points over a given time horizon in order to satisfy e.g. production demand. Similarly, at the level of process control and online optimisation, regulatory control structures and EMPC schemes identify control inputs to a system through time to minimise an operational objective. The identification of a globally minimising sequence of decisions for a given problem in the process industries is obviously desirable, given the societal requirement for efficient, productive and sustainable industrial operations (Grossmann et al., 2016).We can state a general sequential decision making problem, over a finite horizon, as follows:

$$
\begin{aligned}
\mathbf{u}_\tau^* = & \arg\min_{\mathbf{u}_\tau} G_0 \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{x}(0) \\
& \mathbf{x}_{t+1} = f_r(\mathbf{x}_t, \mathbf{u}_t, \cdot) \\
& \mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t) \\
& \mathbf{x}_t \in \mathbb{X}_t \\
& \forall t \in \{0, \ldots, T-1\}
\end{aligned}
\tag{2.1}
$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ are the true system states; $\mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ are control decisions identified by the decision rule; t represents the discrete time index within the control horizon (of length T); $\mathbf{u}_\tau = (\mathbf{u}_0, \ldots, \mathbf{u}_{T-1})$ is the sequence of controls identified over the control horizon; $G_0 = \sum_{t=0}^{T-1} \gamma^t \varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ is the discounted sum of objective costs as provided by a cost function, $\varphi : \mathcal{X} \times \mathbb{U} \times \mathcal{X} \to \mathbb{R}$, accrued over the discrete time horizon from the initial system state, $\mathbf{x}_0$; with $\gamma = (0, 1]$ representing a discount factor[1]; $\mathbb{X}_t \subseteq \mathcal{X}$ and $\hat{\mathbb{U}}(\mathbf{x}) \subseteq \mathbb{U}$ represent constraint sets on the system states and control decisions, respectively; and, $f_r(\mathbf{x}, \mathbf{u}, \cdot)$ represents the real process dynamics, as indicated by the subscript, which one could conceptualise to be a function of the current system state, $\mathbf{x}$, control, $\mathbf{u}$, and potentially further unknown variables.

It is clear that if appropriate assumptions are made regarding the form of the cost function, real process dynamics, and constraint sets, a mathematical program can be recovered and then a decision making problem may then be solved. Assumptions as to the form of cost function, and the control and state constraint sets, are generally well informed by the objective of operations, restrictions on control decisions[2], and operational constraints[3]. These are all objects that are typically known *a priori* to the identification of operational decisions. However, the identification of the *real* process dynamics, f_r, is generally not tractable. Instead approximations are made through the identification of process systems models (Pan et al., 2022).

The construction of systems models proceeds on the basis of identification of an

[1]The use of a discount factor can be thought of as a weighting that places greater importance on costs accrued earlier in the horizon, or equivalently a net present value interpretation of future costs.

[2]Such restriction could be derived from upper bounds on value actuation, or plant production in control tasks and tactical supply chain planning, respectively. This description also extends to precedence constraints in production scheduling problems.

[3]Operational constraints may be derived from e.g. safe limits on the operating temperature for a reactor or the requirements of downstream equipment items in (bio)chemical processing

appropriate model structure. A model structure may be defined through applications of first principles reasoning, and for example in reactions systems via the identification of kinetic mechanisms (Bradley et al., 2022). Alternatively, one could utilise a black-box, data-driven approach where instead highly flexible model classes are utilised to approximate complex functions (Thebelt et al., 2022). Hybrid approaches also exist that combine the advantages of first principles and data-driven modelling (Von Stosch et al., 2014). Once a model structure has been identified, statistical estimation procedures may be deployed to identify the associated model parameters that are not available via process knowledge.

However, whichever modelling paradigm one exploits, the exercise itself is simply an effort to identify an approximation to the complexity of the underlying reality. Therefore, the model construction procedure is subject to various uncertainties, which reflect a) the lack of information available to characterize the system, and b) the natural variance of the underlying data generating process. More formally, this is known as epistemic and aleatoric uncertainty, respectively (Mukhoti et al., 2021). In view of the presence of these uncertainties, general approximations to the true dynamics can be defined in state space form as follows:

$$f_r(\mathbf{x}, \mathbf{u}, \cdot) \approx f(\mathbf{x}, \mathbf{u}, \mathbf{s}) \tag{2.2}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_s} \to \mathbb{R}^{n_x}$ is the approximate process model[4]; and, $\mathbf{s} \in \mathbb{S} \subseteq \mathbb{R}^{n_s}$ denote general model uncertainties, which may be estimated by the modeller[5]. For example, $\mathbb{S}$ could describe the support of a distribution over uncertain parameters (Stuart, 2010) (see Fig. 2.1a)), or simply a set of possible parameter values that could be observed with some confidence (Oden, 2018; Arridge et al., 2019) (see Fig. 2.1b)). The identification of an uncertain process model implies that the discrete time process evolution is probabilistic. For any given state and control decision pair, $(\mathbf{x}_t, \mathbf{u}_t)$, the model will describe a distribution over next states via a conditional probability density function (cpdf), $X_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, such that the state at a given time index is indeed a random variable. This also assumes that the evolution of the uncertain state is dependent only on the current state-control pair, a characteristic widely known as

[4]This discrete-time mapping could be identified through integration of continuous-time models or by constructing this mapping explicitly

[5]Model uncertainties could be descriptive of parametric uncertainty or terms descriptive of general disturbance

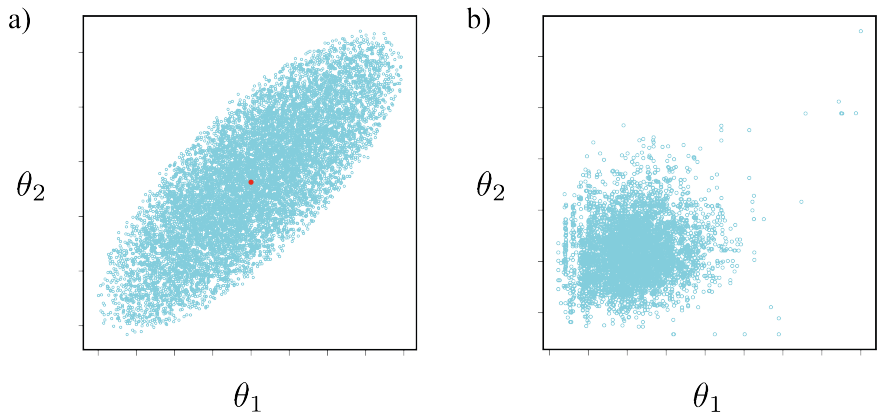


Figure 2.1: Parameter estimates generated via a) Frequentist practice and b) Bayesian estimation.

the Markov property. This is expressed by Fig. 2.2.

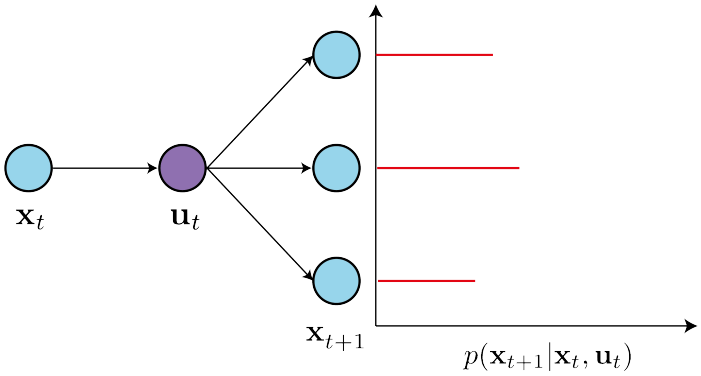


Figure 2.2: The single-stage scenario tree associated with a given state-control pair and the associated probability masses of observing each scenario according to an uncertain process model. Here the model represents a discrete conditional probability mass function, exhibiting the Markov property.

As a consequence of uncertain state evolution, the performance of the decision-maker, i.e. the objective cost from the initial state, G_0, is a random variable distributed according to some probability density function and dependent on the sequence of decisions input to the system. If the process model is required for sequential decision making, this typically necessitates the decision-maker to optimise for various measures of the cost distribution. Generally, these measures are chosen based on the risk exposure tolerated by the process operation. See Rockafellar et al. (2000) for further discussion on risk-sensitive measures. A common objective for optimisation, given an initial state, is the expected performance (also known as the cost), $\mathbb{E}\big[G_0|X_0 = \mathbf{x}_0\big]$.

This enables redefinition of Eq. 2.1 as follows:

$$\mathcal{P}(\mathbf{u}_{\boldsymbol{\tau}}) := \begin{cases} \min\limits_{\mathbf{u}_{\boldsymbol{\tau}}} \mathbb{E}\big[G_0 | X_0 = \mathbf{x}_0\big] \\ \text{s.t.} \\ \mathbf{x}_0 = \mathbf{x}(0) \\ \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t), \quad \forall \mathbf{s}_t \in \mathbb{S} \\ \mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t) \\ \mathbf{x}_t \in \mathbb{X}_t \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{2.3}$$

In the context of mathematical programming, the field of stochastic optimisation provides many solution approaches to this dynamic optimisation problem. However, implementation of such an 'open loop' decision strategy, as expressed by Eq. 2.3, is sub-optimal. Given that we have a finite sequence of decisions to optimise the uncertain system at hand, it is important to introduce a necessary and sufficient condition for the optimality of the decisions made along the horizon. This condition is essentially contextualised by the *The Principle of Optimality*, which was developed by Richard Bellman (Bellman and Lee, 1984).

The general statement of the principle is provided as follows, which applies specifically to those systems *without* state constraints imposed: *given* an uncertain sequential decision making problem with the Markov property; the optimal sequence of decisions is made by a policy, $\pi^* : \mathcal{X} \rightarrow \mathbb{U}$, which provides a decision rule and a mapping to controls, $\mathbf{u}_t$, conditional to the realization of state, $\mathbf{x}_t$. Specifically, the optimal policy, π^*, minimises the expected cost from a given state (Sniedovich, 1978). The cost for an arbitrary policy π, in a given state is defined:

$$\mathbb{E}_\pi\big[G_t | X_t = \mathbf{x}_t\big] = \mathbb{E}_\pi\Bigg[\varphi\big(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{x}_{t+1}\big) + \sum_{i=t+1}^{T-1} \gamma^{i-t} \varphi\big(\mathbf{x}_i, \pi(\mathbf{x}_i), \mathbf{x}_{i+1}\big) \mid X_t = \mathbf{x}_t\Bigg] \tag{2.4a}$$

$$= \mathbb{E}_\pi\Big[\varphi(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{x}_{t+1}) + \gamma G_{t+1} \mid X_t = \mathbf{x}_t\Big] \tag{2.4b}$$

where $\varphi : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \rightarrow \mathbb{R}$ defines the stage cost and $\gamma = (0, 1]$ is the discount factor as before in Eq. 2.1. From analysis of this statement, the optimal sequence of decisions

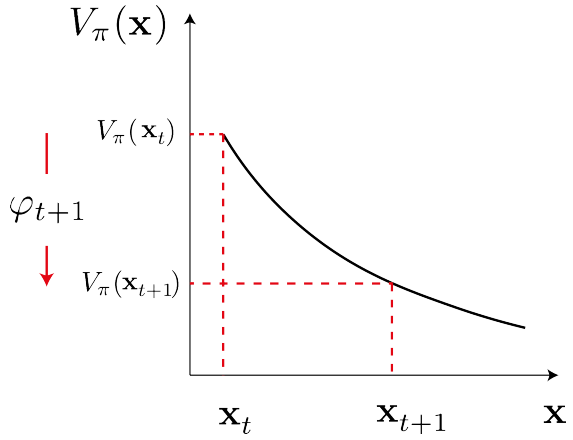


Figure 2.3: Intuition behind the recursive definition of the state value function as defined by Eq. 2.5. Here, we have one state variable, only one control action available and deterministic dynamics. The stage cost is defined as φ_{t+1}, for conciseness.

is identified via a closed-loop decision making process[6] (i.e. a policy). We can work from this principle to provide a condition to certify the optimality of the policy via definition of the Bellman equation:

$$V_\pi(\mathbf{x}_t) = \mathbb{E}_\pi\big[G_t \mid X_t = \mathbf{x}_t\big] \tag{2.5a}$$

$$V_\pi(\mathbf{x}_t) = \mathbb{E}_\pi\big[\varphi(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{x}_{t+1}) + \gamma V_\pi(\mathbf{x}_{t+1}) \mid X_t = \mathbf{x}_t\big], \qquad \forall \mathbf{x}_t \in \mathcal{X} \tag{2.5b}$$

where $V_\pi : \mathcal{X} \to \mathbb{R}$ defines the state value function, which provides a recursive definition of the expected cost of a policy, π, from a given state. For example, Eq. 2.5b, is composed of the expected immediate stage cost from the current state (i.e. the first term within the expectation), and the discounted expected state value at the next time index (i.e. the second term within the expectation). This means the cost of the decisions taken immediately are dependent not only the realisations of uncertainty but also the decisions taken at future time indices. A derivation is provided in Appendix A.1 for clarity and further intuition is provided by Fig. 2.3.

Although, the Bellman equation is able to recursively define the cost of a policy, it does not quantify whether the policy is optimal. However, from the principle of optimality we know that the optimal policy satisfies the following:

$$\pi^* = \arg\min_{\pi \in \Pi} V_\pi(\mathbf{x}_t) \tag{2.6}$$

[6]Note that feedback control policies are a rich class of decision making functions, which also contains the decision rules described by open-loop control sequences (Bertsekas, 2012).

where Π is the set of policies that satisfy control input constraints. Given that we have stated π as a deterministic function of the state, $\mathbf{x}$, one can equivalently define the Bellman optimality equation, which certifies the optimality of a policy:

$$V_{\pi^*}(\mathbf{x}_t) = \mathbb{E}_{\pi^*}\Big[\varphi(\mathbf{x}_t, \pi^*(\mathbf{x}_t), \mathbf{x}_{t+1}) + \gamma V_{\pi^*}(\mathbf{x}_{t+1}) \mid X_t = \mathbf{x}_t\Big] \tag{2.7a}$$

$$= \min_{\mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t)} \mathbb{E}_{\pi^*}\Big[\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \gamma V_{\pi^*}(\mathbf{x}_{t+1}) \mid X_t = \mathbf{x}_t\Big] \tag{2.7b}$$

Eq. 2.7a is simply a definition of Eq. 2.5 under the optimal policy and Eq. 2.7b is the direct result of applying the principle of optimality (i.e. Eq. 2.6) to the Bellman equation (i.e. Eq. 2.5). It is worth highlighting that both Eqs 2.7a and 2.7b are equivalent definitions of the Bellman optimality equation and define the state value function for the optimal policy. There are two further things to draw attention to here. Firstly, Eq. 2.7b suggests one can handle control constraints directly by selecting those controls, which satisfy the constraint set for a given state (Puterman, 2014a). Secondly, by applying the minimum operator over the immediate controls to the optimal state value function, one does not alter the value of the function itself (Bertsekas, 2022). This is highlighted by the equivalence of the two expressions on the right-hand side of Eqs. 2.7a and 2.7b. This is significant because it: a) it indicates the optimal state value function, $V_{\pi^*}(\mathbf{x})$, is a fixed point[7] of Eq. 2.7b; and b) specifies that the optimal policy greedily minimises the state value function at each decision interaction. The observations a) and b) are the basis for various algorithms, which fall under the umbrella of dynamic programming (DP), which aim to identify the optimal state value function. These methods shall be discussed in more detail in Section 2.1.2 and at length in Appendix A.2.

Leveraging Eq. 2.6, and provided state constraint sets are also imposed, then instead of identifying open loop controls as in Eq. 2.3, one can attempt to solve the stochastic optimal control problem (SOCP):

[7]The fixed point, $\mathbf{z}^*$ of a function, $H(\mathbf{z})$, is defined such that $\mathbf{z}^* = H(\mathbf{z}^*)$. The identification of fixed points is common to numerical optimisation too. For example, in unconstrained optimisation of a continuous objective function, a minimising decision variable, is the fixed point of a Newton-Raphson scheme (i.e. $H(\mathbf{z})$). The difference here is that optimal state value function is the fixed point of Eq. 2.7b (i.e. $H(\mathbf{z})$).

$$\mathcal{P}(\pi) := \begin{cases} \min_{\pi} V_{\pi}(X_0 = \mathbf{x}_0) \\ \text{s.t.} \\ \mathbf{x}_0 = \mathbf{x}(0) \\ X_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{u}_t = \pi(\mathbf{x}_t) \\ \mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t) \\ \mathbf{x}_t \in \mathbb{X}_t \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{2.8}$$

Provided that the state and control space is discrete, finite and small, and one has closed form expressions for probabilistic, discrete time dynamics, then an optimal policy for Eq. 2.8 may be identified via exact DP (Sniedovich, 1978). Significantly, although not explicitly considered by the principle of optimality, DP can handle state constraints by allocating infinite cost to those states, which incur violation[8] (Sundström et al., 2010). Due to the recursive definition of value provided by Eq. 2.5, infinite cost then characterises all of those policies that violate (Bertsekas, 1971). Further discussion in this vein is provided by Bertsekas (2005). However, DP becomes computationally infeasible as the state and control spaces become large or continuous,[9], and generally probabilistic process dynamics are not known in closed form. These are both characteristics common to process systems engineering applications, which ultimately provides a barrier to solving Eq. 2.8 exactly via the use of DP for general systems of concern to this work. As a result, few works have explored the use of exact DP within PSE within the last 20 years, although approximate DP schemes have been investigated instead (Luus, 1993; Zhao and Mi, 2021).

There has also been significant development of approximate policy identification schemes via mathematical programming, including but not limited to stochastic, robust and chance constrained online schemes (Heirung et al., 2018; Mesbah, 2016; Ben-Tal et al., 2009; Mayne, 2016). For example, in online optimisation of nonlinear batch

[8]This allocation is known to artificially reduce the feasible region of the problem and a number of methods have been developed to consider this. The conventional approach is to allocate a sufficiently large cost instead (Sundström et al., 2010).

[9]This is widely known as the curse of dimensionality, as coined by Bellman (Bellman, 1956). In fact, when the state becomes continuous this problem is a function space optimisation problem, which is infinite dimensional and cannot be solved exactly via mathematical programming either.

production processes, model predictive control re-solves NLP to identify $\mathbf{u}_t = \pi(\mathbf{x}_t)$ upon observation of a new state, $\mathbf{x}_t$ (Rawlings et al., 2017). Alternatively, in online production scheduling, MILP schemes are often used due to the non-smoothness of the underlying model (McAllister et al., 2022). All of these methods are adept at handling both continuous and discrete control and state spaces, as well as state, control and dynamics constraints. However, they are often required to make approximations to the model uncertainties because mathematical programming ultimately relies upon a finite dimensional description of the uncertain system (Li and Floudas, 2016; Holtorf et al., 2019). These approximations can be significant if the discrete-time system dynamics are nonlinear functions of the current state and control. These existing methods are discussed further in Section 2.2 and 2.3.

Recently, a set of methods known as Reinforcement Learning (RL) have been demonstrated as an attractive approach to identifying optimal policies for sequential decision making problems. This directs attention in the next section.

2.1.2 Reinforcement Learning

We will first provide an overview to the main frameworks for RL algorithms, before discussing their implementation and application within process systems engineering more thoroughly.

An overview of Reinforcement Learning

The field of RL is host to a wide diversity of ideas and concepts, however, more widely these concepts can be thought of as heuristic, data-driven stochastic optimisation algorithms. There are three major subfields - direct and indirect model-free methods and model-based RL methods. It is worth differentiating our interest as specifically directed to model-free RL, primarily because model-based RL shares significant similarities to model predictive control (Kim et al., 2021b). Hence from here, reference to RL will indicate model-free RL methods.

Both direct and indirect approaches are comprised by three key components: model-free learning rules, simulation and function approximation. These elements shall be explored in more depth in the following sections, however, direct and indirect methods differ in their learning rules and in what objects they identify function approximation

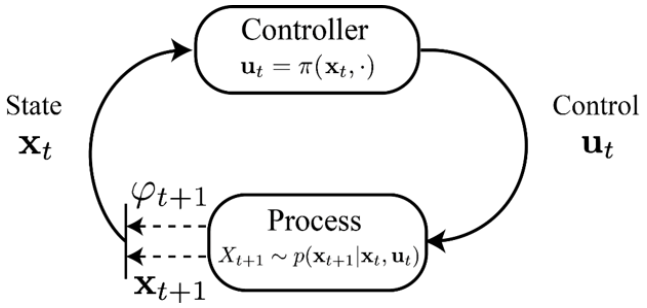


Figure 2.4: The Markov decision process framework. A description of the stochastic optimal control problem without state constraints.

to. In direct approaches, RL attempts to identify a policy function approximation, i.e. $\pi(\mathbf{x}) \approx \pi(\mathbf{x}; \theta)$, where $\theta \in \mathbb{R}^{n_\theta}$ are function parameters, that minimises the state value function (Eq. 2.6) (Schulman et al., 2017b). Whereas indirect methods aim to identify a policy by parameterising the state value function, $V_\pi(\mathbf{x}) \approx V_\pi(\mathbf{x}; \theta)$ and solving the necessary and sufficient condition provided by the Bellman optimality equation (i.e. Eq. 2.7). Optimal decisions may be identified thereafter by greedily minimising the optimal state value function approximation. Indirect methods achieve this by using model-free learning rules closely related to DP (Bertsekas, 2022). Further differences between these two RL approaches will be made clearer in discussion provided later in this section, however, we can characterize both approaches as providing solution to a problem known as a Markov decision process (MDP). The MDP is a formal description of Eq. 2.8, but defined *without* state constraints (Chang et al., 2007). Please see A.3 for formalisation of MDPs and Fig. 2.4 for intuition.

Indirect and Direct Reinforcement Learning

In this section, we will first provide background on ideas common to DP. This will provide foundation to discuss indirect RL methods and their application.

The main idea of DP is to iteratively identify a better policy, π^{k+1}, given the current policy, π^k, such that:

$$V_{\pi^{k+1}}(\mathbf{x}) \leq V_{\pi^k}(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathcal{X}$$

with equality achieved when the Bellman optimality equation is satisfied (i.e. Eq. 2.7a). In DP, we do not *a priori* know the optimal value function, but we do have knowledge of the conditional probability mass functions descriptive of the discrete-time

probabilistic dynamics, some randomly initialized policy, as well as the cost function. With this knowledge, identification of the optimal state value function, V_{π^*}, occurs through recursive application of a policy evaluation step, known as the Bellman operator, $\mathcal{T}_\pi(\cdot)$ and then a policy improvement step, defined by the Bellman optimality operator, $\mathcal{T}_{\pi^*}(\cdot)$.

Before providing further detail of these two operations, we first introduce the state-action value function, $Q_\pi : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$. Under the assumption of a stochastic policy, $\pi(\mathbf{u} \mid \mathbf{x})$, which is a conditional probability density function over controls, the state-action value function, Q_π, is related to the state value function, V_π, as follows:

$$V_\pi(\mathbf{x}_t) = \sum_{\mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t)} \pi(\mathbf{u}_t \mid \mathbf{x}_t) \mathbb{E}_\pi \big[\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \gamma V_\pi(\mathbf{x}_{t+1}) \mid X_t = \mathbf{x}_t, U_t = \mathbf{u}_t \big] \tag{2.9a}$$

$$V_\pi(\mathbf{x}_t) = \sum_{\mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t)} \pi(\mathbf{u}_t \mid \mathbf{x}_t) Q_\pi(\mathbf{x}_t, \mathbf{u}_t) \tag{2.9b}$$

Analysis of Eqs. 2.9a and 2.9b enables us to define the state-action value function, Q_π, as the expected cost of choosing a control, $\mathbf{u}$, in state, $\mathbf{x}$, and then following the policy, π, thereafter. The motivation for definition in terms of a stochastic policy will be made clearer in the following; however the use of stochastic policies is a key concept in RL and it is a general way to think about decision making given that a deterministic policy can also be described by e.g. placing all probability density on one control. The definition of Q_π is particularly useful in the context of control because it provides a ranking over the quality of controls in a given state. This is reinforced by making a connection between the state-action value function and the Bellman optimality equation (see Eq. 2.7b):

$$V_{\pi^*}(\mathbf{x}) = \min_{\mathbf{u} \in \hat{\mathbb{U}}(\mathbf{x})} Q_{\pi^*}(\mathbf{x}, \mathbf{u}) \tag{2.10}$$

which defines that the optimal policy chooses the control, which greedily minimises the state-action value function, Q_π, in each state. Due to the utility of the state-action value function, estimated values are initialised in memory for each state and control and then DP proceeds by applying the policy evaluation and policy improvement steps to identify the optimal state value function (i.e Eq. 2.10).

It follows that the Bellman operator, $\mathcal{T}_\pi(\cdot)$, and Bellman optimality operators,

$\mathcal{T}_{\pi^*}(\cdot)$, are defined via Eq. 2.11a and Eq. 2.11b, respectively:

$$\mathcal{T}_{\pi}(Q_{\pi}(\mathbf{x}_t, \mathbf{u}_t)) = \mathbb{E}_{\pi}\big[\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \gamma V_{\pi}(\mathbf{x}_{t+1}) \mid X_t = \mathbf{x}_t, U_t = \mathbf{u}_t\big], \tag{2.11a}$$

$$\mathcal{T}_{\pi^*}(Q_{\pi}(\mathbf{x}_t, \mathbf{u}_t)) = \min_{\mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t)} Q_{\pi}(\mathbf{x}_t, \mathbf{u}_t) \tag{2.11b}$$

The intuition behind these operators is that the evaluation of Eq. 2.11a improves the state-action value function estimates, Q_{π}, of the current policy (i.e. provides policy evaluation). Whereas Eq. 2.11b improves the optimality of the state-action value function, and is equivalent to selecting the control that greedily minimises the state-action value function (i.e. it provides policy improvement).

For the sake of conciseness, formalisation of two specific DP algorithms, known as value iteration and policy iteration, as well as the technical details of the operators defined in Eq. 2.11a and 2.11b is provided in Appendix A.2. However, in parallel to continuous optimisation methods that take Newton steps on a KKT system; the application of the Bellman operator, $\mathcal{T}_{\pi}(\cdot)$ (Eq. 2.11a), and then subsequently applying the Bellman optimality operator, $\mathcal{T}_{\pi^*}(\cdot)$ (Eq. 2.11b), to all states and controls within the respective sets, can be thought of as taking a newton step on the current state-action value function estimate, Q_{π}, in order to identify the optimal state-action value function, Q_{π}^* (Grand-Clément, 2021; Bertsekas, 2022). By iterating through policy evaluation and policy improvement operations, DP methods start from some initial sub-optimal estimate, Q_{π}, to converge towards the optimal state-action value function, $Q_{\pi^*}(\mathbf{x}, \mathbf{u})$. This intuition is reinforced by Fig. 2.5a) and 2.5b).

Commonly in PSE, we lack of closed form expression describing probabilistic discrete-time dynamics. To approach this, RL utilises two of the three key components: simulation and model-free learning rules. In indirect methods, these mechanisms are inspired by policy evaluation and improvement steps.

We will start by introducing two of the most widely applied model-free learning rules, known under the umbrella of TD-learning (Sutton and Barto, 2018a). The main concept of policy evaluation is to iteratively apply the Bellman operator, $\mathcal{T}_{\pi}(\cdot)$, to the current estimate of the state-action value function, Q_{π}, to minimise the difference between the estimate and the true function. The family of TD-learning updates achieves this in a slightly different, data-driven manner (Meyn, 2022). For a discrete-time state transition, $\boldsymbol{\tau}_{t:t+1} = (\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$[10], collected under policy, π, the general TD-learning

[10]Note that this transition may be representative of process data or synthetic data gained from a

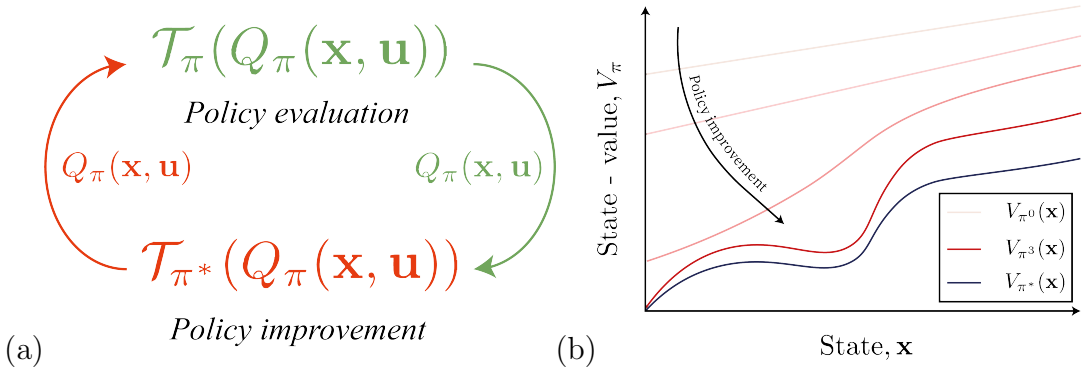


Figure 2.5: Figurative description of the combined action of policy evaluation and policy improvement steps a) on the state-action value function, Q_π and b) on the state value function V_π in a continuous state space with one state variable. b) Starting from a given value function, V_{π^0}, that could be randomly initialized, the combined action of these operators act to iteratively improve the current state value function to satisfy the Bellman optimality equation (Eq. 2.7a).

update has the following structure:

$$Q_\pi(\mathbf{x}_t,\mathbf{u}_t) = Q_\pi(\mathbf{x},\mathbf{u}) + \alpha\delta(Q_\pi) \tag{2.12}$$

where $\alpha = (0,1]$ is a learning rate, which weights the current estimate of the state-action value function, $Q_\pi(\mathbf{x}_t,\mathbf{u}_t)$, against the TD error, $\delta(Q_\pi)$. The TD error is defined as a biased, bootstrapped estimate of the error in approximating the cost under the policy, π, in a given state. We can differentiate the form of this update, as either *on policy* or *off policy*. Here, we detail two such rules, known as expected SARSA, $\delta^{SA}(Q_\pi)$, (Van Seijen et al., 2009) and Q learning, $\delta^{QL}(Q_\pi)$, (Watkins, 1989), respectively:

$$\delta^{SA}(Q_\pi) = \varphi(\mathbf{x}_t,\mathbf{u}_t,\mathbf{x}_{t+1}) + \gamma V_\pi(\mathbf{x}_{t+1}) - Q_\pi(\mathbf{x}_t,\mathbf{u}_t) \tag{2.13a}$$

$$\delta^{QL}(Q_\pi) = \varphi(\mathbf{x}_t,\mathbf{u}_t,\mathbf{x}_{t+1}) + \gamma\mathcal{T}_{\pi^*}\big(Q_\pi(\mathbf{x}_{t+1},\mathbf{u}_{t+1})\big) - Q_\pi(\mathbf{x}_t,\mathbf{u}_t) \tag{2.13b}$$

Both updates exploit the recursive definition of the cost (e.g. see Eq. 2.11a). They form biased, bootstrapped estimates of the cost of a state and control pair, via the first two terms on the right-hand side of the equation. The interpretation of the update as an error is derived from negation of the state-action value estimate in the third term on the right-hand side. However, the difference between the rules arises in how the bootstrapped estimate of cost is formed. In the on policy expected SARSA update, $\delta^{SA}(Q_\pi)$, the bootstrap is formed under the state value function[11], $V_\pi(\mathbf{x}_{t+1})$, of the

simulation model.

[11]Given that we are only keeping estimates of the state-action values in memory, this can be estimated via Eq. 2.9b

policy used to collect the data transition, $\boldsymbol{\tau}_{t:t+1}$. This has similarities with the policy evaluation step in DP (see Eq. 2.11a). Whereas, in the off policy update, $\delta^{QL}(Q_\pi)$, the estimate of the cost is formed under the policy, which greedily minimises the state-action value function. This is not necessarily the same policy used to collect the data. Hence, the Q learning update evaluates the greedy policy, rather than evaluating the current policy and under some assumptions will converge to the optimal state-action value function (Mnih et al., 2014, 2015a).

For both learning rules to identify the optimal state-action value function efficiently, policy improvement (i.e. deterministically minimising the state-action value function in data collection) is required (Tsitsiklis, 1994). Typically, algorithms balance acting randomly and optimally to further aid efficiency. The intuition for this is that acting randomly enables the policy to observe a wider distribution of trajectories and therefore more about the underlying system. This gives rise to the use of stochastic policies. For discussion on the importance and the use of stochastic policies in RL, we direct the reader to (McFarlane, 2018; Sutton and Barto, 2018a; Pistikopoulos et al., 2021).

Despite the benefit of model-free learning rules and the sample efficiency gained from use of stochastic policies, in current form the space complexity of TD learning methods is at best linear in the cardinality of the control sets and the state set. As the state and control sets become very large, the TD learning problem becomes intractable if one holds point estimates for the state-action values in memory. To approach this RL uses the final key algorithmic component of function approximation. The general idea of function approximation is to instead approximate $Q(\mathbf{x}, \mathbf{u}) \approx Q(\mathbf{x}, \mathbf{u}; \theta)$, via a parametric function, with parameters $\theta \in \mathbb{R}^{n_\theta}$ (Melo et al., 2008; Sutton and Barto, 2018a). This removes the requirement for one to store point estimates of all state-action values in memory, but rather to identify a parameterized function instead. As a result, the problem of identifying the optimal state-action value function, is reformulated to find the best parameters, θ^* (Jin et al., 2020). Two popular parameteric function approximations are shown in Fig. 2.6. In the case of function approximation, the Q learning update is modified so that it is amenable to statistical parameter estimation practices. This is demonstrated through popular minimisation of the squared Q

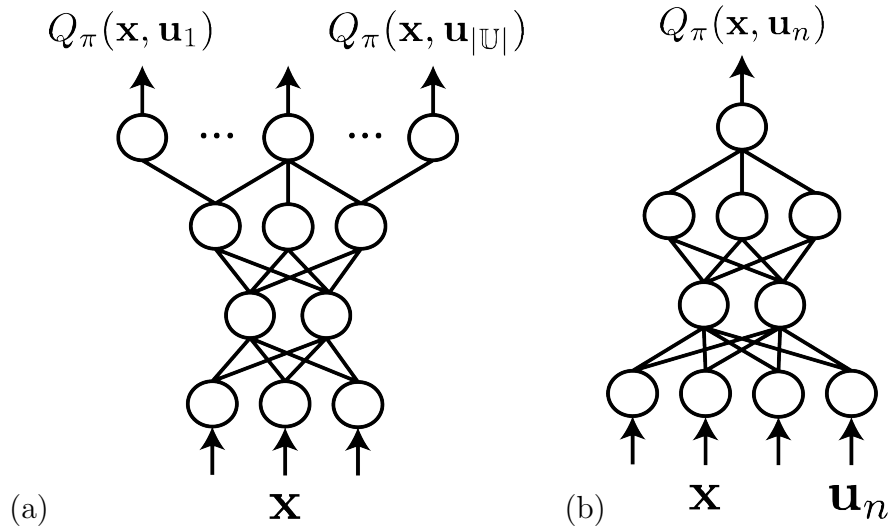


Figure 2.6: Possible structures of feedforward artificial neural network approximations to the state-action value function, Q_π. The structures differ in that a) defines the approximation as, $Q_\pi : \mathbb{X} \to \mathbb{R}^{n_{|\mathbb{U}|}}$, providing a map from states to the state-action values of all possible controls; whereas b) defines, the state-action value function, $Q_\pi : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$, as a map from state and controls, to a respective state-action value. As a result, a) is amenable to small discrete control spaces, whereas b) can handle continuous control spaces, but requires optimisation to conduct the policy improvement step.

learning error (Bas-Serrano et al., 2021), such that stochastic gradient descent steps[12] can be taken using batches of state transitions to improve the current parameters, θ, to better represent the policy, π:

$$\begin{aligned} \delta^{QL}(\theta) &= \varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \mathcal{T}_{\pi^*}\big(Q_\pi(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}; \theta)\big) - Q_\pi(\mathbf{x}_t, \mathbf{u}_t; \theta) \\ L(\theta) &= \frac{1}{2}\delta^{QL}(\theta)^2 \\ \theta &= \theta - \alpha \nabla_\theta L(\theta) \end{aligned} \tag{2.14}$$

where $\delta^{QL}(\theta)$ is the Q learning error and $L(\theta)$ is the loss function in the form of the mean square of the Q learning error. In practice, the full gradient is typically not utilised but instead a semi-gradient is used instead and formed as:

$$\nabla_\theta L(\theta) \approx \delta(\theta) \nabla_\theta Q(\mathbf{x}_t, \mathbf{u}_t; \theta) \tag{2.15}$$

where $\nabla_\theta Q$ is the gradient of state-action value prediction with respect to the parameter vector, and for general composite functions, such as neural networks, may be computed via backpropagation (Goodfellow et al., 2016).

[12]Note that SGD is not the only method that can be used. For example approximate second order methods such as ADAM (Kingma and Ba, 2014) are typically applied.

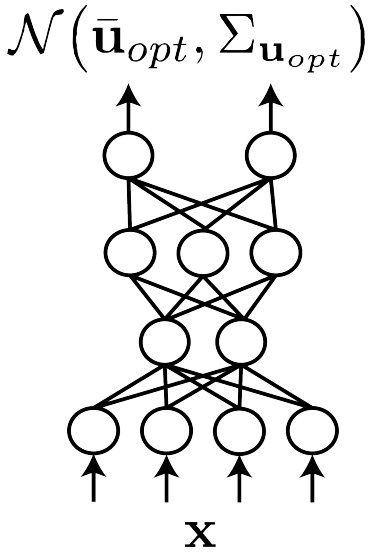


Figure 2.7: A typical Gaussian stochastic policy as constructed by policy gradient methods.

We have now seen that the major ideas in indirect RL methods, is to identify the optimal state-action value function, via three key components: model free learning rules, simulation and function approximation. Direct RL methods also use all of these key components. However, direct RL methods aim to directly parameterize a policy instead, such that $\pi^*(\mathbf{u} \mid \mathbf{x}) \approx \pi(\mathbf{u} \mid \mathbf{x}; \theta^*)$, where

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\tau_{t:T} \sim p(\tau_{t:T};\theta)}\big[G_t\big] \tag{2.16}$$

and $\boldsymbol{\tau}_{t:T} = (\mathbf{x}_t, \mathbf{u}_t, \ldots, \mathbf{u}_{T-1}, \mathbf{x}_T)$ is the system trajectory from time index t to the end of the horizon, and $p(\boldsymbol{\tau}_{t:T}; \theta)$ describes the associated distribution over trajectories subject to the policy parameters. There are two main methods that dominate direct RL: policy gradients (PG) and direct policy search. The former, PG methods, are a well established approach to policy identification, with the latter observing more recent interest. Firstly, we will explore the use of PG methods.

PG methods are underpinned by the policy gradient theorem (Sutton et al., 1999, 2000) and typically parameterise a Gaussian stochastic policy (if the control space is continuous) as demonstrated in Fig. 2.7. The PG theorem is explained extensively in (Sutton and Barto, 2018a) and in Appendix B.1. However the PG can be formed exactly as:

$$\nabla_\theta \mathbb{E}_{\tau_{t:T} \sim p(\tau_{t:T};\theta)}\big[G_t\big] = \mathbb{E}_{\tau_{t:T} \sim p(\tau_{t:T};\theta)}\big[\sum_{t'=t}^{T} \nabla_\theta \log \pi(\mathbf{u}_{t'} \mid \mathbf{x}_{t'}; \theta) Q_\pi(\mathbf{x}_{t'}, \mathbf{u}_{t'})\big]$$

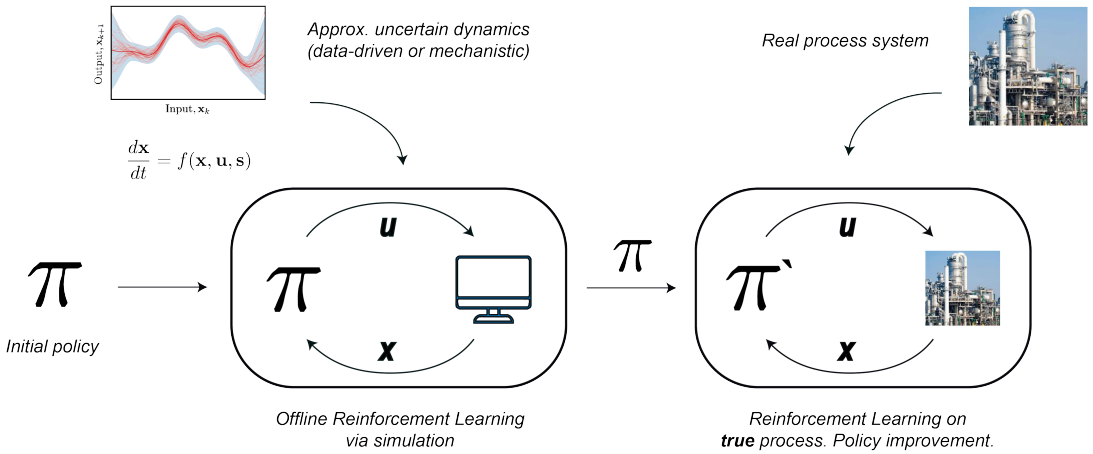


Figure 2.8: A framework for the identification and deployment of approximately optimal policies via RL. It consists of an initial policy learning step through simulation of an approximate process model. The policy is then transferred to the real system for the purposes of online optimisation and further learning.

This equation states that the policy gradient is the direction, which makes trajectories of lower cost more likely, or in other words, creates choosing those controls which have a lower state-action value more likely. Stochastic gradient ascent or approximate second-order methods can then be deployed to update the policy parameterization. Given that an expression for the policy gradient is generally unavailable in closed form (i.e. we do not know Q_π exactly), we can instead approximate it directly through a sample average approximation (SAA) and subsequent use of e.g. backpropagation if neural network policy function approximations are used. This leads to a number of algorithms (Schulman et al., 2017a,b; Haarnoja et al., 2018; Abdolmaleki et al., 2018), the most well known of which is the REINFORCE algorithm (Sutton et al., 1999).

Having outlined the major classes of RL algorithm, we can now first emphasise the major attraction for the use of RL in sequential decision making problems common to PSE, and where the major areas of research focus lie.

The promise of model-free Reinforcement Learning and open challenges

Utilising the three major components of RL algorithms (i.e. model-free learning rules, simulation and function approximation), a general framework for the use of RL in the process industries can be described. This is expressed by Fig. 2.8.

The major idea here is to first identify high quality, optimal policy function approximations offline through simulation of an approximate model together with its

associated uncertainties. Once this initial policy learning phase has been conducted, we can then transfer this policy to the real process for purposes of online optimisation (Levine et al., 2020). Additionally, model-free learning rules potentially enable one to further improve the policy online directly from process data once deployed. This holds promise for the development of learning based control and online optimisation schemes (Fisac et al., 2018), potentially mitigating the requirement for expertise heavy scheme re-calibration. Although, at this point, it is not completely clear if such potential can be fulfilled safely.

Further major benefits of the framework described by Fig. 2.8 are detailed in the following,:

- The identification of approximately optimal state-feedback policies independently of the assumptions that the state and control sets are discrete and small[13], and that knowledge of probabilistic discrete time dynamics exists in closed form, overcoming the limitations of DP (Sutton and Barto, 2018a; Nian et al., 2020). The former point is mitigated by the use of function approximation and the latter by model-free policy learning rules.

- When state-feedback policies are approximated via optimisation formulations, one can often fail to identify decisions online if the problem becomes infeasible under the optimisation model (Scokaert and Rawlings, 1999). RL policies generally identify decisions by prediction rather than optimisation meaning that a control will always be identified.

- Additionally, conducting optimisation online can often become infeasible if the system dynamics are faster than the time required to optimise (Wolf and Marquardt, 2016). Instead RL identifies a policy function approximation offline through simulation of an approximate process model and its associated uncertainties, enabling provision of fast online control decisions for the real process via prediction. As such, it can be applied to handle decision making problems at all time-scales of the PSE hierarchy (Glavic et al., 2017).

[13]In other words, that the cardinality of these sets is small.

- Finally, the identification of policies via simulation enables one to optimise systems independently of smoothness assumptions in the underlying model or assumptions on the form of the uncertainty. This enables ease in optimisation of for example, production scheduling environments (Wang et al., 2021) and biochemical systems that utilise a switch model (Brancato et al., 2022).

The benefits listed promise potential improvements in both operational performance, wide applicability and flexible use of complex systems models. However, major challenges also exist in the implementation of RL algorithms. These challenges primarily relate to safety (Brunke et al., 2022), robustness (Waubert de Puiseau et al., 2022) and stability guarantees for RL-based closed-loop decision making (Nian et al., 2020; Osinenko et al., 2022), as will be explored immediately.

Safety in sequential decision making scenarios relates primarily to satisfaction of operational state constraints, as desired, over the discrete-time horizon. This is particularly challenging because:

1. The MDP framework does not provide an explicit mechanism for handling state constraints (Altman, 1999; Shin et al., 2019).

2. In the case that the support of uncertain variables is unbounded constraints become difficult to satisfy absolutely, but may instead be handled with high probability via chance constraints (Li et al., 2008; Yang et al., 2022)

3. The approximate model used for offline policy learning is rarely a perfect representation of the true process, which means that focus should be in evaluating the generalisation capabilities of the RL controller, to the real process and not the model used for offline policy identification. If this is not well satisfied the policy could drive the process into high risk operational regions of the state space (Curi et al., 2020). This is because RL control predictions are derived from data-driven function approximations and are not based on any physical understanding of the system. This is a major benefit of model-based decision strategies and optimisation formulations that leverage some degree of mechanistic description of the system.

To handle 1) there have been many works proposed that approach the problem

by attempting to fuse the benefits of mathematical programming with respect to constraint handling with RL. Examples are provided by Amos et al. (2018); Zanon et al. (2020); Wabersich and Zeilinger (2021). However, a key observation made in Fisac et al. (2018) is that general additional modifications to the policy learning mechanism to ensure safety should ideally not compromise the ability of the scheme to utilise RL's model-free learning rules. Ultimately, the model-free nature of these learning rules is the basis for the potential benefits of RL. Specifically, the use of mathematical programming for constraint handling, imposes non-trivial learning rules, which require at the very least that the underlying system is smooth (Nian et al., 2020). Additionally, its use requires solving an optimisation problem online, which has downsides in terms of the required computation and problems associated to feasibility and failure to find a solution online. This reduces the applicability of RL to more general systems.

Similar ideas extend to 2). Although, RL and mathematical programming may be combined to good effect for chance constrained linear systems (Pfrommer et al., 2022); when considering nonlinear systems handling chance constraints via mathematical programming becomes challenging. This is primarily in view of the requirements to propagate the uncertainty associated with the state over the discrete time horizon (Heirung et al., 2018). More information on this is provided in Section 2.2. As a result, there is substantial incentive to develop RL methods for chance constraint satisfaction independently of mathematical programming to allow for probabilistic constraint satisfaction in nonlinear systems.

The final point 3) is a general focus in many communities interested in RL. This has led to the establishment of the offline RL community (Levine et al., 2020), who generally deal with learning a policy offline; either via a model (Yu et al., 2020, 2021b) or via a fixed stationary dataset (Kumar et al., 2020b). The idea behind these algorithms is to modify the model-free learning rule to ensure that the policy generalises beyond the offline learning framework with inspiration either from various multi-armed bandit strategies (Kuleshov and Precup, 2014; Rashidinejad et al., 2021) or via concepts resting within robust optimisation (Agarwal et al., 2020; Kumar et al., 2020b).

The aspect of robustness in RL essentially boils down to two main points. The first is related to the definition of the objective of decision making (Waubert de Puiseau et al., 2022) and the second is related to algorithm implementation (Engstrom et al.,

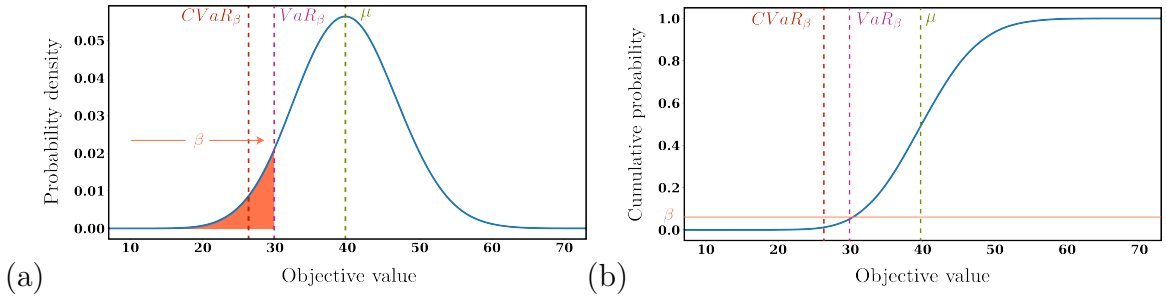


Figure 2.9: Description of the performance distribution associated with a policy and measures such as the mean, the conditional value-at-risk, CVaR, and the value-at-risk, VaR, for a given probability level β, as well as the expected value, μ under a) the probability density function and b) the cumulative distribution function.

2019). With respect to the former point, RL algorithms generally optimise for the expected performance of the policy from the initial state distribution over the discrete finite time horizon. However, in process operation considering other measures of the performance may have substantial benefit (Waubert de Puiseau et al., 2022). This provides motivation to consider risk-averse (Greenberg et al., 2022) or distributional RL formulations (Bellemare et al., 2022). The main idea in these two methods is to take information from the full distribution of policy performances to inform construction of the policy. For example, in risk-averse RL the objective is to optimise for or impose constraints on various risk-sensitive objectives (Ahmadi et al., 2021), such as the conditional value-at-risk (CVaR) (Rockafellar et al., 2000). Given that the CVaR defines the expected value in the worst returns that occur with at least a user defined probability (see Fig. 2.9); this enables one to construct policies, which ensure against high severity outcomes in the tails of the policy performance distribution (Urpí et al., 2021). For example, if a process objective is economic, then constraining the expected performance of the worst 10% of policy performances (the CVaR for probability level 0.9) to be less than a certain cost, could help to protect the operation from making a loss. Meanwhile, distributional RL can be thought of as encompassing risk-averse RL and it generally aims to identify state and control dependent approximations to specific discrete percentiles of the quantile function associated with a policy performance (Bellemare et al., 2017a), or to learn the full quantile function itself (Yang et al., 2019; Dabney et al., 2018a) via quantile regression (Dabney et al., 2018c). Although quantile regression has been applied previously in stochastic model predictive control to handle joint chance constraints (Yang et al., 2022), such a concept has not been

readily incorporated into optimisation based predictive control frameworks.

The final element of robustness regards implementation. Typically, RL practice utilises flexible data driven functions such as neural networks for policy function approximation. These models are all associated with hyperparamters that define their structure (Murphy, 2022), but also their initial parameters (Zhu et al., 2021a). Additionally, many of the model-free learning rules that comprise the field are composed by hyperparameters such as pre-defined step-sizes (Kingma and Ba, 2014), parameters that govern the degree of bias and variance within the updates (Schulman et al., 2018b) and the discount factor (Amit et al., 2020; François-Lavet et al., 2015). Generally, although there are rules of thumb for the selection of these hyperparameters, there is little theoretical understanding as to how these factors combine to effect a) the dynamics of policy learning, and b) the performance of the final policy (Engstrom et al., 2019). This results in the requirement for either some automated search algorithm (Claesen and De Moor, 2015) or the use of technical expertise and rules-of-thumb to solve an algorithm tuning problem (Engstrom et al., 2019). In reality, this adds an additional layer of complexity and technical expense in identifying an optimal RL policy. This is a generally expensive step, because high variance can be observed in the final policy performance when the algorithm is subject to relatively small changes in hyperparameters.

The final point regarding the lack of closed-loop stability guarantees of RL policies primarily arises due to a relative lack of research focus. Arguably, much of the theoretical research within RL has been directed towards developing algorithms with convergence guarantees (Melo and Ribeiro, 2007; Sutton and Barto, 2018a; Meyn, 2022) and much of the applied research has investigated its optimality (Joshi et al., 2021; Spielberg et al., 2019; Campos et al., 2022; Martínez et al., 2022). This has led to relative neglect in trying to ascertain closed-loop stability. Since the turn of the century (Perkins and Barto, 2002), and as indicated by the recent review provided by Osinenko et al. (2022), interest is building in this research direction. This is largely due to the recent engagement of the control community within RL research. This means that many of the approaches proposed so far to provide stability guarantees for RL have relied on the assumption of an available process model (Gros and Zanon, 2019a); given that this is generally the case for the guarantees derived from the control

community (Mayne et al., 2000). This further emphasises the notion that although, in its full potential RL is fully model-free, to be realised in industry there will likely be some dependence on the use of a model (Osinenko et al., 2022).

Having elucidated the major drivers for research and a framework for the identification of RL policies in PSE, we will now explore the means by which RL has been applied in more detail.

Indirect Reinforcement Learning methods in Process Systems Engineering

As nonlinear and non-smooth systems are of interest to this work, it is unlikely that a linear function approximation to the state-action value function, Q_π, is likely to well satisfy the Bellman optimality equation. This notion is in keeping with standard practice within the PSE and RL community, which generally exploits the advent of deep learning, where complex and highly flexible classes of neural network (functions) can be used as parameterisation. In fact, deep Q learning (DQN) (Mnih et al., 2013, 2015c) has become a well-known, benchmark indirect RL method[14]. For example, Hong et al. (2020) deployed a DQN algorithm to identify optimal evacuation routes for oil and gas processing and storage facilities in deep-water fields. They draw attention to the benefits of identifying an optimal evacuation route via prediction in a hypothetical accident situation. Decisions may be identified despite the uncertainty of, for example, emergency exits being blocked or out of use. Further, Singh and Kodamana (2020) demonstrate comparison between Q learning without function approximation (what is known as a tabular implementation) and with function approximation on a nonlinear fed-batch polymerisation problem. The tabular implementation uses point estimates for each $Q_\pi(\mathbf{x}, \mathbf{u})$ and requires discretisation of a continuous control space (which was the temperature of a heated jacket) and state space (which was the temperature of the reactor). The authors remark that as the discretisation becomes finer, the sample efficiency of the tabular implementation decreases quickly, whereas the DQN implementation is able to handle continuous states efficiently. The general intuition behind this is that tabular implementations are not able learn about multiple controls or states close in the respective spaces from a single update. This is however not the

[14]The method itself is comprised by algorithmic components, which are well worth discussing, but beyond the narrative of this section. Please see Mnih et al. (2013, 2015c); Van Hasselt et al. (2016); Zhang and Sutton (2017) for more information.

case for the function approximation, which shares a parameter vector across all states and controls (Mnih et al., 2015a). As a result, the parameter updates effect the policy across the control and state spaces. A similar work is provided by Nikita et al. (2021), who demonstrate a DQN approach for optimising the flowrate of a chromatography column. The authors use a rigorous, nonlinear simulation model to identify a policy that optimises the column flowrate profile over a discrete finite time horizon, to ensure a specific purity in the separation of components. Once the policy had been identified it was then used to optimise a real column with the authors reporting purity of the components as desired, together with a significant reduction in time spent over the conventional strategy of manual tuning the flowrate on the real system.

However, all the works mentioned above use a discretised control space and structures the state-value function approximation, $Q_\pi(\mathbf{x}, \mathbf{u}; \theta)$, according to Fig. 2.6a). This means the algorithm is highly dependent on the discretisation being well chosen. Continuous control spaces can be handled via indirect methods in one of two approaches. The first is provided by the state-action value function structure in Fig. 2.6b). For example, Pan et al. (2021) demonstrate such a structure for the online optimisation of two uncertain, nonlinear fed-batch processes. They use an evolutionary algorithm to identify the control, which greedily minimises the state-action value function approximation. In the first study, the authors optimise a Phycocyanin production process demonstrating 3% improvement in closed loop performance over a nonlinear model predictive control (NMPC) scheme, which represents the best-case deterministic model. Additionally, in the second study they investigate the performance of the approach on a semi-batch catalytic chemical reaction process. This time the NMPC scheme outperforms the RL implementation by approximately 8%, however, this does demonstrate the potential for RL to compete with current advanced predictive control schemes. It should be noted that in both cases, the authors assume that there is a simulation model with a perfect description of the process uncertainty available to identify the policy offline, with validation then conducted 'online' via Monte Carlo simulation of the same model. This is of course an ideal assumption.

The same structure (e.g. Fig. 2.6b)) has been optimised to find the optimal control via various other approaches. One example is to formulate the problem of identifying the optimal control input to a rectified linear unit (ReLU) neural network as MILP

(Ryu et al., 2019). The authors demonstrate their method on a classic physics based control problem, where the aim is to stabilise an inverted pendulum, which is described by nonlinear dynamics, over a discrete time horizon. This is a description in common with many fed-batch process systems in PSE. They demonstrate their approach to outperform many other stochastic search methods by 2-20%. This includes the use of genetic algorithms, and the cross entropy method (De Boer et al., 2005; Lim et al., 2018; Wen and Topcu, 2018). A possible explanation for this is because given enough computational budget, the MILP approach is able to enact a policy improvement step exactly, whereas this is not necessarily the case with stochastic search methods (which are heuristic approaches to optimisation).

Although, improved formulations for optimising ReLU networks via MILP have been proposed within the PSE community (Tsay et al., 2021), conducting policy improvement via optimisation is more computationally expensive than e.g. making a control prediction via a neural network policy. This limits the application of indirect methods to optimise systems with fast dynamics. The other more popular solution presents itself in the form of actor-critic methods, which roam a shared conceptual ground between indirect and direct methods. Although, the application of the latter will be explored in more detail in the subsequent section, the general idea of actor-critic methods is to parameterise both a state-action value function, $Q_\pi(\mathbf{x}, \mathbf{u})$, as well as policy, $\mathbf{u} = \pi(\mathbf{x}; \hat{\theta})$, where $\hat{\theta} \in \mathbb{R}^{n_{\hat{\theta}}}$ are the parameters of the policy. The problem then reduces to identifying optimal parameters, θ^* and $\hat{\theta}^*$, for both approximations. Fig. 2.10 represents the construction of the networks for an actor-critic method known as deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015).

For example, Ma et al. (2019) demonstrate the application of DDPG for the control of a semi-batch polymerisation process. Here, the kinetics are described by a free radical polymerisation reaction, and the aim of control is to manipulate the monomer and initiator inflow rates in order to track a reference trajectory for the weight-average molecular weight of the batch. This trajectory was determined offline via dynamic optimisation and is a good indicator of the molecular mass distribution of the product, which dictates the polymer quality and structure. The approach demonstrated robustness in the face of time delay and noisy observations of the system state. Other applications of actor-critic algorithms for control have been demonstrated in Martínez

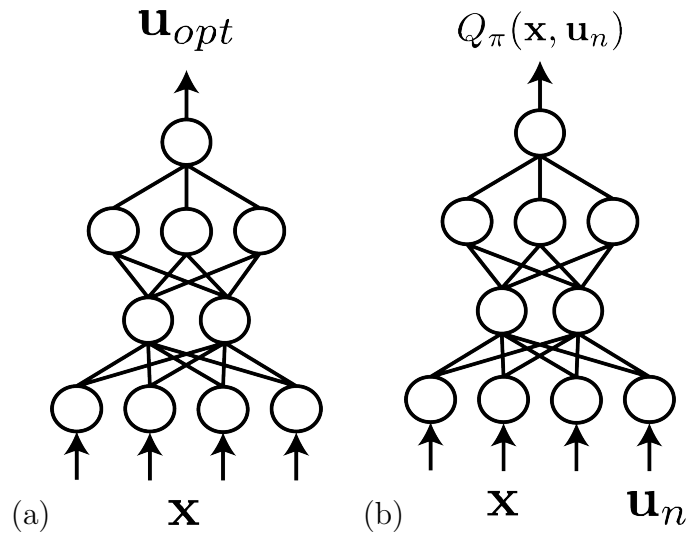


Figure 2.10: Network structures used in the deep deterministic policy gradient (DDPG) actor-critic method. Here a) a policy function approximation provides a map from states to optimal control predictions, $\pi : \mathbb{X} \to \mathbb{U}$, whereas b) defines the state-action value function, $Q_\pi : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$, as in Fig. 2.6.

et al. (2022), where multiple setpoint changes in the level of a continuous stirred tank reactor and concentration of a respective component are enacted without the requirement to re-tune the policy. This is typically posed as a meta-RL problem (Nagabandi et al., 2018), which is an area of RL that deals with learning a policy that generalises to different control tasks. This is the formulation posed by McClement et al. (2021), who demonstrate an actor-critic algorithm for multiple set point changes in a system characterised by first order dynamics. The general idea of this scheme is to identify a control-orientated, reduced order representation of a system state, that enables ease in generalisation of the control policy across tasks. The authors highlight the potential of RL to enact multiple setpoints indicating that it could be applied to a wide variety of process systems and potential ease in recalibrating control and online optimisation schemes. This idea has been successfully extended to second-order linear systems in McClement et al. (2022).

Yoo et al. (2021b) demonstrated the potential of RL to handle multiple dynamical phases within batch processes. Specifically, they highlighted DDPG on a polymerisation process with two dynamical regimes. This non-stationarity in dynamics is often an issue for other model-based schemes (Shin et al., 2019). The authors report that RL provides improvements over NMPC in closed-loop with respect to the satisfaction of operational constraints in this case. A demonstration of RL in the 'real-world' was

provided by Lawrence et al. (2022). The authors use a modified version of DDPG, to track level in a two-tank system. Specifically, although the authors parameterise the state-action value function via a neural network, they parameterise the policy as a PID controller. It is argued that this improves the stability properties of the closed-loop system. Additionally, the authors report the identification of a well tuned PID controller, learned 'from scratch' directly on the real system after 30-40 minutes of experimentation. This highlights the potential to realise the benefits of model-free learning rules in identifying a policy directly from the process. This is certainly a promising result. To the author's knowledge, there has not yet been such a demonstration on a real-world nonlinear system directly of concern to PSE. However, there have been impressive achievements from other fields, which are worth mentioning. Of particular note is the recent work in Degrave et al. (2022), where an actor-critic algorithm was able to stabilise the high-temperature plasma of a tokamak (which is the reactor used for nuclear fusion based energy generation). This is a control task that is fundamental to the maturation of fusion technology. The authors demonstrate a single neural network controller able to manipulate the reference voltage for 19 different active control coils that are used to generate a magnetic field to suspend the plasma. The work utilises initial policy learning in an offline rigorous simulation model and is then transferred to the real process to provide control. This has been an open challenge to control due to the nonlinearity of the system and the requirement for high frequency, multivariable control inputs. The system is able to stabilise the plasma, providing sustained energy generation. However, the authors do note that the RL controller has no guarantees for rejecting disturbances in the plasma and should likely be implemented together with a simpler fallback system in case instability arises.

There have also been implementations of DQN at the level of production scheduling. For example, it is common for the scheduling operation to utilise various heuristic rules to react to realisations of uncertainty (this is discussed in more detail later in Section 2.3). In the work provided by Han and Yang (2020), the authors utilise a DQN agent to select different heuristic rules upon realisations of plant uncertainty in a multi-product sequential production environment. Additionally, work by Kim et al. (2021a) demonstrates a DQN approach to the same production environment but with machine breakdown. Here the network directly assigns jobs to machines rather than selecting

heuristics. The authors report a 37% improvement over the use of a deterministic schedule determined offline.

The previous works mentioned certainly demonstrate the wide potential and advantages of model-free RL. A number of works have also considered handling the open challenges posed previously within this section. For example, Yoo et al. (2021c) recently considered the development of a dynamic penalty function for handling constraints in deterministic systems. The method gradually imposes constraints on the policy as learning proceeds using the constraint aggregation function (Bloss et al., 1999). This is to approach a problem commonly observed in offline policy learning; it is often the case that when constraints are imposed by setting the cost function to a penalty function, a policy can get 'stuck' in a poor local optima. This is generally due to non-smoothness in the true state-action value function derived from handling constraints via a penalty function. By dynamically imposing the constraints, the aim is to mitigate problems associated with this approach. It is demonstrated in the work by Yoo et al. (2021c) that their method identifies a constraint satisfying policy in 93 of 100 different computational runs, using the DQN algorithm in a relatively simple vehicle control problem. The problem is described by a linear dynamical system, with constraints on the vehicle position, velocity (i.e. state constraints) and acceleration (i.e. control input bounds). This work goes someway to improving the robustness and reliability of RL algorithms operating in constrained process systems. Slightly more theoretical work is provided in Bas-Serrano et al. (2021), who present a new algorithm that aims to mitigate some of the problems arising in the use of bootstrapping in the Q learning update (see Eq. 2.13b). The authors report drastically improved performance, compared to DQN, on the nonlinear dynamical system derived from the inverted pendulum problem mentioned previously. This contribution helps to develop the robustness of RL algorithms, given that the bias in bootstrapped updates can often destabilise indirect RL methods. More recently, it has been proposed to combine the benefits of model-free RL and mathematical programming approaches, by incorporating the state-action value function directly into the objective of a model predictive controller, as a terminal cost (Oh et al., 2022). The resultant algorithm, Q-MPC, was able to outperform differential dynamic programming (DDP), DQN and DDPG with a nonlinear, and uncertain penicillin fed-batch production system. Specifically,

improvements of approximately 3% were reported in the closed-loop performance. It is worth noting, that as the time horizon used to construct the Q-MPC formulation tends to one discrete step, the formulation tends to ordinary DQN, but with the use of e.g. IPM or SQP to identify the optimal greedy control. The authors report improved sample efficiency over conventional DQN and DDPG approaches, and is able to outperform DDP by accounting for process-model mismatch via the objective (i.e. the state-action value function approximation). The authors leave the satisfaction of state constraints for future work. Similarly, Hedrick et al. (2022) propose to integrate a deep SARSA implementation and MPC, but instead within a hierarchical framework, where the RL controller instead tunes the prediction and control horizon of an MPC controller at each discrete control interaction within the process operation. The first control identified by the MPC scheme is then implemented. The authors demonstrate their approach on a selective catalytic reduction (SCR) unit, which is a common approach for treatment of flue gas. The dynamics of this operation are inherently nonlinear and subject to time delay. The MPC implementation uses an approximate linear dynamical model and a tracking objective. The results show the proposed method is able to achieve a 30% reduction in the integral squared error over the MPC implementation on its own.

Having reviewed some of the major contributions to indirect methods within PSE, we now turn our attention to direct RL methods.

Direct Reinforcement Learning methods and Policy Search in Process Systems Engineering

As discussed previously, a major set of direct RL algorithms are known via PG methods, which have been widely applied within PSE. For example, Shim and Lee (2022) employed a PG method in the context of process design to identify an optimal sequence of distillation operations. This was opposed to the classical super-structure optimisation approach. The authors conclude that the approach could be particularly promising for more large-scale problems, where superstructure approaches become particularly expensive mixed integer nonlinear programs (MINLP) (Mencarelli et al., 2020). A more large-scale work was presented by Khan and Lapkin (2022), who have explored the use of hierarchical RL schemes (based on policy gradient methods) for the

purposes of process design and synthesis. Specifically, they design an intensified ethylene oxide production plant, reporting improved results compared to baselines within the literature. It is worth commenting that the application of RL to process design does not necessarily inherit benefit through the use of the MDP framework (as the parameters of these problems are often assumed deterministic); however, the use of RL can help to discover new heuristics for process synthesis and avoids use of expensive, exact optimisation solvers which necessitates restriction of the solution space based on knowledge. This is also noted in (Göttl et al., 2022; Stops et al., 2022).

The model-free nature of the policy gradient has also been explored within PSE, typically within the context of batch-to-batch control and optimisation. For example, Petsagkourakis et al. (2020b) explored ways in which to quickly update policy function parameters when deployed to a 'real' bioprocess and then subsequently improve that policy from batch-to-batch. Specifically, the authors froze a subset of model parameters, only updating the parameters in the final layer of the neural policy function approximation. This allows for only slight modification to the output of the policy and essentially retains the knowledge gained in initial offline policy learning via the frozen parameters. This has also been applied in the domain of image classification (Russakovsky et al., 2015). Similarly, Yu and Guo (2020) applied a policy gradient method to a chemical mechanical polishing process, which is an operation common to the semiconductor manufacturing industry. They explored the ability of RL to handle run-to-run variations in the material removal rate (from a wafer) due to process degradation, setpoint change and variations in noisy disturbances. The authors conclude that the policy gradient approach used, performs preferably to other common heuristic methods, including a neural network parameterisation of an MPC controller. The application of RL in the context of batch-to-batch is a essentially the same idea as directly improving the policy online as proposed in Fig. 2.8. A recent work has explored the application of the policy gradient online (Dogru et al., 2021). The authors demonstrate the transfer of a policy initially learned offline to 'real' (computational) process. The authors report that under application of the policy gradient online, the policy retains closed-loop stability. This is certainly a promising result, but one would assume that the hyperparameters associated with the update would heaviy influence such a conclusion. Further work should characterise, which hyperparameter settings

induce stable behaviour.

Zhu et al. (2021b) implemented a policy function approximation for online optimisation of a polyol process. They discuss and provide analysis for the results, which are of similar quality to that of mathematical programming. Additionally, they discuss the problems of policy function interpretability (or a lack of it) and highlight that although there is obscurity in the use of a neural network, one can analyse the control profiles generated in simulation of an uncertain process model and align this with knowledge of the physical system. Extensions to this analysis aim to investigate the sensitivity of the policy prediction to small changes in the state input. This idea is encompassed by the developing field of neural network verification, which aims to use optimisation to certify the robustness of neural networks to changes in the input (Albarghouthi, 2021). The maturation of these approaches is likely to be important if we are to see the deployment of neural policy functions to safety critical applications. Work provided by Mendiola-Rodriguez and Ricardez-Sandoval (2022) also explores the robustness of policy gradients to various disturbances for control and optimisation of an anaerobic digester. They highlight its benefit over traditional MPC approaches in terms of time to compute a decision online and accounting for process uncertainty in closed-loop. The algorithm was found to be robust to a variety of disturbances, and the authors indicate the potential to incorporate the policy function approximation into an integrated design and control under uncertainty framework (Sachio et al., 2022). Another work within PSE, has considered the use of RL for flexibility analysis in the design of engineering systems (Caputo and Cardin, 2022a). The authors demonstrate the advantage of policy gradient methods in identifying a policy that determines how best to expand the capacity of a waste-to-energy system at discrete intervals over a 15 year period. The aim of the policy is to increase the net-present value of the design subject to high degrees of uncertainty in demand over the period. The RL approach is benchmarked to common heuristic decision-rules and demonstrates marked performance improvements of approximately 30%. This highlights the ability of RL to account for uncertainty in closed-loop decision making. Similar motivations were demonstrated in Hubbs et al. (2020a), who deployed a policy gradient method for online closed-loop scheduling of a continuous chemical production system under demand uncertainty. The approach was benchmarked to nominal and stochastic MILP approaches, with

the policy gradient method demonstrating improved performance over the stochastic receding horizon MILP and comparative performance to the shrinking horizon nominal MILP implementation. The major benefit here is that the scheduling decisions of the RL, are orders of magnitude more efficient than the MILP implementations. Previous work presented in Park et al. (2021), has combined RL with graph neural networks (GNNs) (Bronstein et al., 2021), which are used to identify a control-oriented, reduced dimension latent representation of a deterministic sequential production environment. This is a promising approach for larger problem sizes and assumes that the production environment can be well modelled as a disjunctive graph (Błażewicz et al., 2000). A similar idea has also been presented in Zhang et al. (2020a). Both works compare the algorithm proposed to common heuristics priority dispatching rules, with only Zhang et al. (2020a) also benchmarking to an optimisation approach. Both works demonstrate improvements over the heuristics, for example Park et al. (2021) showed at least a 5% improvement in objective performance over the next best heuristic, and Zhang et al. (2020a) showed as much as 25% improvement. However the latter work also reported a gap of up to 40% compared to the optimisation implementation. It is well known that combining graph neural networks and RL is open challenge with these system characterised by complex 'learning' dynamics (Gupta et al., 2021; Munikoti et al., 2022). This is in part because the algorithmic challenges of identifying policy gradient RL policies are combined with those of GNNs (i.e. the model structure selection becomes even more complex). As a result, there remains question as to the optimality of such an approach.

Other applied policy gradient RL works within PSE research explore handling mixed integer control spaces, effectively handling constraints on control inputs and aiding the sample efficiency of policy gradient estimation. These key issues for application are explored. Firstly, Campos et al. (2022) explored various different constructions of the policy function and methods for control selection, to best handle mixed integer control sets within scheduling of energy systems. Specifically, the authors consider the operation of a district cooling plant in order to provide cooling utility. A case study of a large-scale district cooling plant employing real demand and price data is presented. It is shown in analysis that the algorithm can mitigate storage constraint violations, and the solution can be improved using the model-free learning rules once deployed.

Given that the demand and electricity price is uncertain and actually forecast within the system state, the work highlights that the algorithm can handle state uncertainty and incomplete state information.

Hubbs et al. (2020b) explored the use of action-masking, rather than penalising for control constraint violations through the cost function, in a number of operations research problems including the bin packing problem, which is a classic combinatorial optimisation problem. This is highlighted in the operational healthcare applications presented in Van Houdenhoven et al. (2007); Vijayakumar et al. (2013). Action-masking works specifically in discrete control spaces, by masking the controls in the output of the policy function if they are infeasible. This is a more efficient approach that using the model-free policy learning rule alone. There are approaches to action-masking that operate over continuous control spaces (Wabersich and Zeilinger, 2021; Chen et al., 2018b). These approaches allow one to convert state constraints into control constraints. The mechanism of these approaches is based on modifying the control prediction of the neural policy function via a dynamic optimisation problem with a least squares objective between the neural policy prediction and the modified control. Despite the safety benefits, there is reliance on the use of an underlying process description, which can impose non-trivial learning rules and restricts the underlying process model to be smooth and deterministic. This is likely to impose sub-optimality if the model is subject to high uncertainty.

Finally, the work provided in Deisenroth and Rasmussen (2011) differentiates a Gaussian process state space model to gain exact estimates of the policy gradient. The authors applied their algorithm to a robotics problem, which is essentially a nonlinear dynamical system. The authors demonstrated that they could control the system at hand using very few samples of data collected from the real world. However, this does rely on the explicit use of Gaussian process state space models to approximate the true system (Sternberg and Deisenroth, 2017) and so assumes that the dynamics are smooth. Similar ideas were also expressed in Chen et al. (2018b), but instead under the assumption of linear time invariant dynamics enable accurate estimation of the state-action value function (within the PG) using a reduced number of samples by exploting the structure of the problem, rather than using a function approximation. This leads to two orders of magnitude improvement in objective performance over the same number

of training samples as an actor-critic algorithm, which uses a function approximation, for an infinite horizon, linear quadratic regulator (LQR) problem (approximated as finite horizon with a terminal cost). However, this approach does explicitly require the structure of the problem to be LQR.

The use of direct first order methods to identify approximately optimal policy functions is an area worthy of discussion. As in general optimisation problems, if there exists non-convexity or non-smoothness in the mapping from policy function parameters to policy performance, first order methods are not guaranteed to perform well and may get stuck in poor local minima. This has led to the development of various zero-order methods, which aim to optimise policy parameters independently of any gradient information from the underlying decision process (Salimans et al., 2017a; Powell, 2021) by transferring the policy search from the control space, directly to the policy function parameter space. This is further motivated by the ambiguity regarding selection of other hyperparameters that are associated with first-order RL approaches and general lack of theoretical understanding as to how these algorithms work in practice (Ilyas et al., 2018; Nota and Thomas, 2019a; Chen et al., 2020; Kumar et al., 2021).

The first major empirical demonstration of zero-order methods in RL was provided by Salimans et al. (2017a), who proposed to use natural evolutionary strategies (NES) within the high dimensional parameter space ($n_\theta > 5000$) of neural policy functions in the context of game-based control benchmarks. This was an insightful paper given wide acknowledgement that zero-order optimisation tends decrease in performance as the dimensionality of the search space increases (Balasubramanian and Ghadimi, 2022). This is likely negated in the work provided, due to the flexibility of neural network functions and the presence of redundancy in the parameter space. NES was first proposed in Wierstra et al. (2014) and functions by maintaining a Gaussian distribution over the parameter space (rather than the control space as in the case of policy gradients). Neural network parameters are then sampled from the distribution and evaluated through simulation in the underlying environment. The evaluation results of the population are then utilised to form the 'natural gradient update' for the mean and standard deviation of the distribution (Amari and Douglas, 1998). Greater interest in these approaches is developing. For example, a company driving the use

of RL within industry, with machine learning applications in specialty glass production, additive manufacturing and wind generation (Nnaisense, 2022b); have recently released an open-source python package tailored for implementation of evolutionary approaches to RL, which provides basis for further research into use of these methods (Nnaisense, 2022a). Evolutionary approaches to RL have also been demonstrated in a more niche physics-based application of directing self-assembly of molecular structures (Whitelam and Tamblyn, 2020). The authors commented on the ease of application and training relative to first-order RL methods.

2.1.3 Extracting decision rules from process data

Process knowledge expressed by existing operational decision systems can be extracted from process data in two ways - both can be thought as under the conceptual umbrella of 'learning from demonstration' (LfD). The first approach is to identify a parametric function approximation to the action of the existing system. This parameterisation could then be used, for example, to help hot-start the RL process, or simply deployed to the process to remove the requirements for online optimisation (if, for example, the existing decision system is a model predictive controller). The second approach is to extract the objective function of the decision maker, based on the control and state trajectories expressed in data. In reality, this is never actually known. This is because we cannot construct a perfect model of the real world, and therefore, for example the objective function of a model predictive controller and that extracted from the data will be unlikely to align exactly. Instead, the objective that is extracted will contain information about the real system, or any changes made by operators. Optimizing for this objective, will then account for process knowledge.

These two approaches are well known by behavioural cloning and inverse reinforcement learning (IRL), respectively. There is a wealth of engineering focused literature demonstrating use of the two approaches. For example, early works in behavioural cloning considered industrial engineering applications (Michie et al., 1990), 'learning to fly' aircraft (Sammut et al., 1992) and motor control of a mobile robot (Esmaili et al., 1995). However, these early works noted the apparent 'brittleness' of behavioural cloning, when the identified parameterisation of the existing control strategy was deployed (Bratko et al., 1995). Specifically, the authors note that the parameterisation

often produced unstable responses when the initial conditions or dynamics were different to those corresponding to the available process data. This is an intuitive result given that these conditions represent off-distribution (i.e. extrapolative) predictive tasks - an area where ML systems are notoriously poor if due consideration is not provided in learning (Krueger et al., 2021).

There has also been a documented risk of over-fitting the parameterisation to the data available in behavioural cloning, meaning it can often produce unstable responses on the same system as the data was collected (i.e. in interpolative predictive tasks, where ML systems generally perform well). This notion is also supported by Yoo et al. (2021b), who demonstrated that imitation learning an NMPC controller, performs worse than directly learning an RL controller from 'scratch' on a fed-batch polymerisation process. A popular fix to this problem is to synthetically create extra data, by augmenting the data one has available with some level of noise. This essentially provides some form of implicit regularisation (Kanervisto et al., 2020a; Hernández-García and König, 2018). This has also been a popular approach to ensuring real-world generalisation in RL systems (Laskin et al., 2020; Park et al., 2022). These two perceived downsides of behavioural cloning has generally led to its deployment as a means of parameter initialisation in for example offline RL policy learning problems (i.e. within the simulation model). For example, Sachio et al. (2022) used behavioural cloning to initialise an RL policy for a control task from an MPC controller. However, it has also been used to abstract the mapping provided by an MPC controller (Lucia and Karg, 2018; Paulson and Mesbah, 2020) with the resultant parameterisation proposed to be transferred to the real-system. However, we argue that it would be ill-advised to deploy these parameterisations directly to the real system, unless very large amounts of data were available. This is re-affirmed by Yoo et al. (2021b); Kumar et al. (2022).

The other route provided by IRL is generally deemed a more robust approach. Instead of identifying a parametric model of the demonstrated behaviour in process data, IRL identifies models of the stage cost function, $\varphi : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}$, guiding the existing scheme (Arora and Doshi, 2018). This means that a parameterisation of the existing scheme can then be identified through RL under the cost function abstracted in an approximate process model offline. IRL generally considers that the behaviour of the existing scheme is optimal under that cost function. There are

a number of foundational works including Ng et al. (2000); Abbeel and Ng (2004), which first presented the idea of Apprenticeship learning. Here, the idea is to identify a cost function, $\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ as a linear combination of basis functions, $\boldsymbol{\phi} : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \rightarrow \mathbb{R}^{n_w}$, of the state and control variables. Note that a cost function with a given setting of $\mathbf{w} \in \mathbb{R}^{n_w}$ and $\boldsymbol{\phi}$ defines a specific MDP or control task. The selection of $\boldsymbol{\phi}$ is usually informed by expert knowledge of the control task at hand, although there is ambiguity regarding their selection. Once the basis features have been defined, the problem reduces to identifying the best cost weighting, $\mathbf{w}$, to recover the behaviour of the existing scheme as best as possible.

The motivation behind the Apprenticeship Learning algorithm (Abbeel and Ng, 2004) has the following concept at its core, which considers the state value function at the initial state of the existing scheme, π^E, should lower bound that of any other arbitrary policy, π:

$$\mathbb{E}_{\pi^E}\big[\sum_{t=0}^{T} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_t, \pi^E(\mathbf{x}_t), \mathbf{x}_{t+1})\big] \leq \mathbb{E}_{\pi}\big[\sum_{t=0}^{T} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{x}_{t+1})\big] \tag{2.17a}$$

$$\mathbf{w}^T \mathbb{E}_{\pi^E}\big[\boldsymbol{\upsilon}^E\big] \leq \mathbf{w}^T \mathbb{E}_{\pi}\big[\boldsymbol{\upsilon}\big] \tag{2.17b}$$

where Eq. 2.17b follows directly from rearrangement of Eq. 2.17a; the feature counts characterising the behaviour of the existing decision system are defined as $\boldsymbol{\upsilon}^E = \sum_{t=0}^{T} \boldsymbol{\phi}(\mathbf{x}_t, \pi^E(\mathbf{x}_t), \mathbf{x}_{t+1})$; the terms on the left hand-side correspond to the state value function of the expert policy, π^E, expressed in process data; π is any arbitrary policy learned under the cost function parameterised by $\mathbf{w} \in \mathbb{R}^{n_w}$, with associated feature counts, $\boldsymbol{\upsilon} \in \mathbb{R}^{n_w}$; and the term on the right hand-side of both equations corresponds to the state value function of the associated policy. As the state value function of the existing scheme lower bounds that of any other policy identified within the associated MDP, the action of the existing scheme is considered to be optimal. The method itself identifies a cost weighting, $\mathbf{w}$, to obtain an ϵ - optimal policy, π such that:

$$\|\mathbf{w}^T(\mathbb{E}_{\pi^E}\big[\boldsymbol{\upsilon}^E\big] - \mathbb{E}_{\pi}\big[\boldsymbol{\upsilon}\big])\|_2 \leq \epsilon \tag{2.18}$$

where $\epsilon \in \mathbb{R}$ is a user defined hyperparameter, which is a) chosen to provide termination criterion for the algorithm and b) provides a minimum separation required

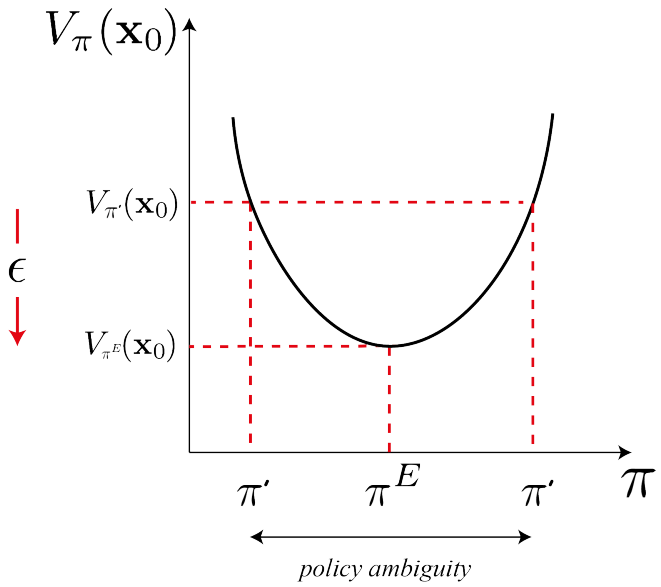


Figure 2.11: Intuition behind the Apprenticeship Learning algorithm. At convergence, the method identifies a policy, which is at most within a distance, ϵ, of the state value function of the existing scheme, $V_{\pi^E}(\mathbf{x}_0)$ in the initial state. There is ambiguity as to which policy to learn and how best to define the distance, ϵ.

between the existing scheme, π^E, and the policy identified, π in terms of the state value function at the start of the horizon, $V(\mathbf{x}_0)$. This is expressed by Fig. 2.11.

The use of ϵ as a termination criterion means that the larger the value chosen, the earlier the algorithm converges. However, even if ϵ is small, the policy identified by the method, may not well reflect the behaviour of the existing scheme (as expressed by the feature counts of the basis functions, $\boldsymbol{\upsilon}^E$) and many policies may also satisfy this condition. This provides considerable ambiguity and an ill-posed problem.

This work provided in Ziebart (2010) aims to resolve this ambiguity via the principle of maximum entropy (Jaynes, 1957) and maximum entropy optimisation. The high-level idea here is to identify a policy, $\pi(\mathbf{u}|\mathbf{x})$, to match the feature counts, $\boldsymbol{\upsilon} = [\upsilon_1(\boldsymbol{\tau}), \ldots, \upsilon_{n_w}(\boldsymbol{\tau})]$, associated with the available process data, $\mathcal{D} = \{\boldsymbol{\tau}_{(n)}, \boldsymbol{\upsilon}^E(\boldsymbol{\tau}_{(n)})\}_{n=1}^N$, in expectation. As a result, the algorithm instead identifies a policy, which recovers the characteristic behaviour of the existing scheme (as quantified via the feature counts rather than objective performance, i.e. the state-value function in the initial state). This major conceptual difference between the original AL algorithm (Abbeel and Ng, 2004) and the maximum entropy approach (Ziebart, 2010) is summarised by Fig. 2.12. Maximum entropy optimisation is a well-known idea, which has been applied more widely within science and engineering (D'Alessandro et al., 1999; Tanyimboh and

Sheahan, 2002; Heckelei and Wolff, 2003; He et al., 2019; Cheng et al., 2019; Thebelt et al., 2022). Given that this provides basis for one of the research objectives in this thesis, we shall briefly review maximum entropy optimisation and its application to sequential decision making problems.

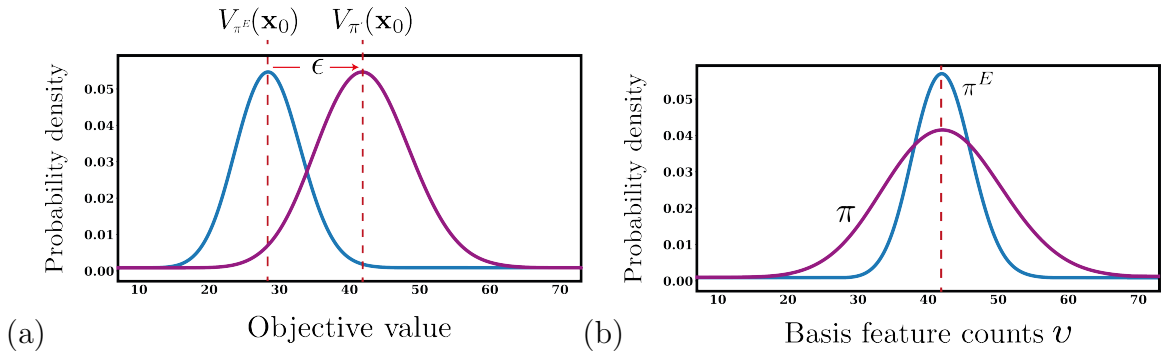


Figure 2.12: Informal intuition behind the differences between a) apprenticeship learning and b) maximum entropy inverse RL. In a) the problem is posed within 'value space' (i.e. objective performance), whereas in b) it is posed directly in the space of basis feature counts, which characterise the policy with respect to control objectives.

The principle of maximum entropy states that: given testable information, $\mathcal{D} = \{\mathbf{z}_n, \mathbf{r}(\mathbf{z}_n)\}_{n=1}^N$, where $\mathbf{r}(\mathbf{z}) = [r_1(\mathbf{z}), \ldots, r_{n_w}(\mathbf{z})]$, is a function of a discrete[15] random variable, $Z \sim p_{gt}(\mathbf{z})$; the best approximating probability mass function (pmf), $p(\mathbf{z}) \approx p_{gt}(\mathbf{z})$, is the one that has maximum information entropy, $H(Z)$, and satisfies the constraint of testable information (Murphy, 2022; Amos, 2022).

To provide further intuition, we first provide insight on information entropy. The information entropy of a random variable is defined as:

$$H(Z) = -\sum_{\mathbf{z}\in\mathbb{W}} p(\mathbf{z}) \log p(\mathbf{z}) \tag{2.19}$$

where $\mathbb{W} \in \mathbb{R}^{n_u}$ is the support of the random variable, Z, and $-\log p(\mathbf{z})$ is the Shannon information, which can be thought as the information content of observing a realisation of the random variable[16] (Lombardi et al., 2016). Information entropy can therefore be thought of as the expected information gain associated with observing

[15]This definition has been assumed for the purposes of notation; the ideas also extend to continuous random variables.

[16]The basic intuition follows from the range of a logarithm over the domain provided by the range of a valid probability mass function. As the probability of a realisation, $\mathbf{z}$, tends to zero, the Shannon information tends to infinity. Likewise as the probability mass of an event tends to one, the Shannon information tends to zero. Hence the less likely an event is, the more information it provides about the random variable.

an additional realisation of the random variable (Ronen and Karp, 1994; Muñoz-Cobo et al., 2017). Hence, the approximating pmf, $p(\mathbf{z})$, with greatest information entropy that satisfies the constraint of testable information can be thought as the least biased choice (Murphy, 2023).

The constraint of testable information has deliberately been left as a reasonably obscure term until now. Given that one has many observations within the dataset, $\mathcal{D}$, an intuitive way to describe this data concisely is by its first two moments (i.e. mean and variance) (Cover, 1999). In practice, we could constrain the approximating distribution to satisfy both, however, in RL we tend to be most interested in what happens in expectation. Under the assumption that we are interested just in expectation, we can define the constraint of testable information as:

$$\mathbb{E}_{p(\mathbf{z})}\big[r_i(\mathbf{z})\big] = \mathbb{E}_{z\sim\mathcal{D}}\big[r_i(\mathbf{z})\big], \quad \forall i \in \{1,\dots,n_w\} \tag{2.20}$$

which constrains the approximating distribution, $p(\mathbf{z})$, to match the testable information (provided by the existing scheme in $\mathcal{D}$) in expectation. With these two components, one can define a primal problem to maximise Eq. 2.19 as the objective, with constraints provided by Eq. 2.20. The decision variables are the values of the approximating distribution, $p(\mathbf{z})$, over the support defined by $\mathbb{W}$, and hence other constraints that impose non-negativity and normalisation of these values are also defined (Cover, 1999). Full definition of the formulation, together with identification of its dual are provided by Appendix A.5. It is worth noting here, however, that in the case that the support, $\mathbb{W}$, becomes continuous, the primal problem becomes infinite dimensional providing barrier to computation (Ziebart, 2010). However, the problem is convex and so the distribution can instead be identified by exploiting duality. The dual is a maximum log-likelihood problem defined under the exponential distribution as follows:

$$\max_{\boldsymbol{\lambda}} \ \log p(\mathcal{D}|\boldsymbol{\lambda}) \tag{2.21}$$

$$\log p(\mathcal{D}|\boldsymbol{\lambda}) = \sum_{n=1}^{N}\Big[\boldsymbol{\lambda}^T\boldsymbol{r}(\mathbf{z}_n) + \log A(\boldsymbol{\lambda})\Big] \tag{2.22}$$

where $A(\boldsymbol{\lambda}) = \big(\sum_{\mathbf{z}\sim\mathbb{W})}\exp\big(\boldsymbol{\lambda}^T\boldsymbol{r}(\mathbf{z})\big)\big)^{-1}$ is the partition function, which enforces normalisation of the exponential distribution; and $\boldsymbol{\lambda}\in\mathbb{R}^{n_w}$ are the Lagrange multipliers. The problem reduces to identifying the Lagrange multipliers for the constraint of testable information (note that generally n_w is much less than $|\mathbb{W}|$).

To approach the ambiguity of policy identification associated with the original work in Abbeel and Ng (2004), Ziebart (2010) extended maximum entropy optimisation to multi-stage decision processes. By identifying that the expert data is described by $\mathcal{D} = \{\boldsymbol{\tau}_{(n)}, \boldsymbol{\upsilon}^E(\boldsymbol{\tau}_{(n)})\}_{n=1}^N$, the methodology reformulates the IRL problem as maximising the log-likelihood of observing the demonstrated behaviour under the maximum entropy policy, which following from Eq. 2.21 takes the form:

$$\max_{\boldsymbol{\lambda}} \ \log p(\mathcal{D}|\boldsymbol{\lambda}) \tag{2.23}$$

$$\log p(\mathcal{D}|\boldsymbol{\lambda}) = \sum_{n=1}^{N} \Big[\boldsymbol{\lambda}^T \boldsymbol{\upsilon}^E(\boldsymbol{\tau}_{(n)}) + \log A(\boldsymbol{\lambda})\Big] \tag{2.24}$$

where the partition function, $A(\boldsymbol{\lambda}) = \big(\sum_{\boldsymbol{\tau}\sim p_{\pi_H^*}} \exp\big(\boldsymbol{\lambda}^T \boldsymbol{\upsilon}(\boldsymbol{\tau})\big)\big)^{-1}$, can be estimated from the expected feature counts associated with the optimal maximum-entropy policy, π_H^*, which induces some distribution over trajectories $\boldsymbol{\tau} \sim p_{\pi_H^*}(\boldsymbol{\tau})$. The policy, π_H^* is optimal for an MDP with cost function defined as $\varphi = \lambda^T \boldsymbol{\phi}(\mathbf{x}_t, \pi_H^*(\mathbf{x}_t), \mathbf{x}_{t+1})$. Given that we generally, assume the availability of a process model, the partition function can be estimated directly through sampling. The intuition behind this formulation is that the maximum entropy model of the existing scheme, places exponentially more probability mass on those trajectories of lower cost (Ziebart et al., 2008; Finn et al., 2016b).

A full derivation of Eq. 2.24 is shown in Appendix B.3 and its gradient with respect to $\boldsymbol{\lambda}$ is also derived. Importantly, from the perspective of application, the methodology enables one to recover the weights of a cost function that is either linear (Ziebart et al., 2008) or nonlinear (Wulfmeier et al., 2016) in the basis functions, $\boldsymbol{\phi}$. When the cost function proposed is linear, the estimation is also a convex problem, which ensures one identifies a global solution for the set of basis functions and process data. Additionally, the use of a cost function, which is linear in the parameters is particularly useful from the perspective of interpretability.

Since the work of Ziebart (2010), a number of different IRL methodologies have been presented. Particularly generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016), guided cost learning (Finn et al., 2016b) and adversarial IRL (AIRL) (Fu et al., 2017) leverage some of the major advances in generative modelling. Specifically, these methods exploit connections between generative adversarial

networks (Goodfellow et al., 2014), energy based models (Teh et al., 2003) and maximum entropy inverse reinforcement learning to identify an imitation, π, of an expert policy, π^E (Finn et al., 2016a). These methods in someway remove the ambiguity behind defining basis functions, $\boldsymbol{\phi}$, with expert knowledge. This is achieved by essentially forcing the policy to generate process trajectories, $\boldsymbol{\tau}$, (i.e. sequences of states and controls) which are from the same distribution as that expressed in the expert data, $\boldsymbol{\tau} \sim \mathcal{D}$. Another, main benefit of this approach is that it leverages more expressive cost functions (i.e. neural networks) than those that are linear or nonlinear combinations of hand designed basis functions. This is of course at the cost of interpretability, which is a key focus in engineering applications (Schweidtmann et al., 2021; Steurtewagen and Van den Poel, 2021). Additionally, in chemical engineering we often have knowledge of the key properties of the control task and so can design features, $\boldsymbol{\phi}$, which both characterise the behaviour of the existing decision system and provide control objectives.

Until now, the approach provided by inverse reinforcement learning has primarily been ignored within the PSE and chemical engineering communities, although it has been readily applied in robotics environments (Coates et al., 2009). Behavioural cloning techniques have been more widely applied. This is primarily due to their shared conceptual ground with supervised learning and model building practice. They are also cheaper to implement computationally. However, there has been more recent interest in the development of IRL approaches. For example, Anandan et al. (2022) implemented an IRL approach to learn the behaviour of an existing NMPC scheme for a crystallisation process. The results were compared to the behaviour of the NMPC scheme on a computational representation of the 'real' process. The authors reported that the IRL scheme was able to recover the behaviour of the NMPC approach and also able to provide robust and stable control on the real system; highlighting a benefit of the IRL approach over conventional behavioural cloning (Yoo et al., 2021b).

Having reviewed the key principles and ideas of (inverse) Reinforcement Learning in sequential decision making problems as deployed in the research objectives presented by this thesis, focus is directed towards the state-of-the-art works presented by the PSE community in process control, online optimisation and in production scheduling.

2.2 Process control and online optimisation of batch process systems

2.2.1 Batch and fed-batch process systems

In this section, the review will focus on the state-of-the-art methods used for data-driven process control and online optimisation of batch processes. Focus is directed here, primarily because these decision-making tasks occur over discrete and finite time horizons (i.e. they are not continuous processes and infinite-horizon decision processes), but also because the dynamics of these systems are subject to uncertainties. Further the dynamics are also often nonlinear, which means propagating the uncertainties associated with state evolution is an area of research focus, providing basis for one of the items in this thesis.

From a high level, batch and fed-batch process systems deal with the conversion of reactants input to a given equipment item to an output. The process output could consist of multiple intermediate or final products (Rippin, 1993). There is a wide diversity of batch and fed-batch processes ranging from biochemical fermentation processes (Todaro and Vogel, 2014), to crystallisation (Benyahia et al., 2021; Zheng et al., 2022) and emulsification (Stork et al., 2003; De Hert and Rodgers, 2017; Calvo et al., 2020), as well as to catalytic chemical reaction processes, such as polymerisation (Zhang, 2008; Özkan et al., 2006) and hydrogenation of biomass (Biradar et al., 2014; Wang et al., 2019a; Akhade et al., 2020). In the following, we will explore the major challenges and contributions to modelling and online optimisation of batch and fed-batch processes. There are other aspects of batch process operations, such as state estimation, soft-sensing and maintenance, which are well worth discussing, but out of this scope of this thesis. For more discussion please see Kadlec et al. (2009).

2.2.2 Modelling approaches

For many batch process systems, discrete time process evolution of batch process systems can be well described by global process models, which are smooth functions of the current system state and control, otherwise known as state space models. This is primarily because, for example, changes in system composition and process operating

conditions are primarily driven by the reaction kinetic rates, enthalpies, macro-scale volumetric flows, and energy losses from the system (i.e. they may be derived from mass and energy balances) (Jakobsen, 2008). Subject to classical simplifying assumptions such as spatial homogeneity in composition and temperature, this ultimately enables process description in terms of systems of ordinary differential equations and mechanistic models.

However, if the assumptions and reasoning behind the derivations of the mechanistic model do not well capture the real process, then predictions of process evolution may incur high errors and mismatch (Sharma and Liu, 2022). For example, in building a mechanistic model we often assume that the state evolution is a time invariant parametric function of the system state and control input (this gives rise to the term time-invariant state space model). However, for more complex reaction systems this is often not the case. An example of this arises in biochemical reaction systems using cells, where the kinetics can often switch based on the underlying biochemistry of each cell and the culture micro-environmental conditions (Almquist et al., 2014; Jing et al., 2018). This phenomenon is often termed as non-stationarity in the process dynamics (Yoo et al., 2021a,b). More widely speaking, there are a number of remedies to handling mismatch in mechanistic modelling.

The first modelling paradigm to discuss is pure data-driven modelling (Fuentes-Cortes et al., 2022; Sharma and Liu, 2022). Data-driven modelling has been an established approach within the systems identification community for a long time (Chiuso and Pillonetto, 2019). Broadly speaking the major methods used there can be subdivided into subspace and prediction error methods. In subspace methods, one can efficiently identify linear time-invariant (LTI) models of process dynamics by exploiting the singular value decomposition of the Hankel matrix, which (under some assumptions on the data-generation process) enables one to identify model parameters via the normal equations (i.e. the solution to parameter estimation is globally optimal and analytical) (Verhaegen, 2015). This enables one to avoid poor, low quality local optima, but requires the dynamics to be well approximated via LTI dynamics. A number of other subspace algorithms have been developed since to account for other assumptions on data-generation (Van Overschee and De Moor, 1993; Katayama et al., 2005; Ghosh et al., 2019; Cox and Tóth, 2021).

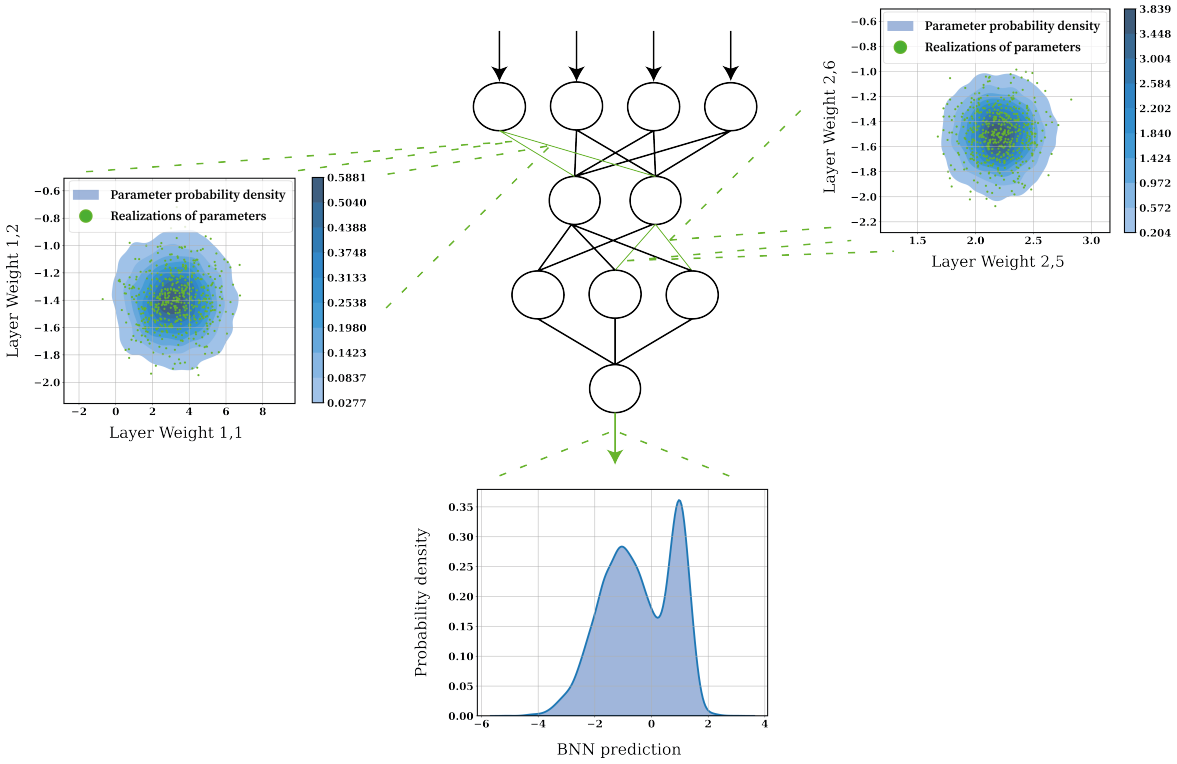


Figure 2.13: Bayesian neural networks have probability density functions over parameters and may be interpreted as an infinite ensemble of conventional neural networks, each with unique point estimates for parameters.

However, as mentioned the assumptions of LTI can often be restrictive (i.e. the evolution of state may not be well described as a linear combination of the current state and control). This has led to the development and use of other more flexible model classes such as the Hammerstein-Wiener (Wills et al., 2013; Wang and Georgakis, 2019), autoregressive models (Yang and Lam, 2019) and artificial neural networks (Lu and Tsai, 2008; Yan and Wang, 2012; del Rio-Chanona et al., 2016). Additionally, there are probabilistic data-driven models, which instead of providing a deterministic prediction about the evolution of state, forecast state evolution via a conditional probability density function (cpdf). This cpdf may express the aleatoric model uncertainty (i.e. that irreducible uncertainty that represents the natural variation of the data-generating process) alone or also the epistemic (i.e. that reducible part of model uncertainty, which could be decreased with more data). Examples of these models include heteroscedastic noise neural networks (Yang and Chen, 1998), and Bayesian neural networks (BNNs) (Hernández-Lobato and Adams, 2015) and Gaussian processes (GPs) (Williams and Rasmussen, 2006), respectively. It is important to note that GPs are non-parametric in that, although they do have hyperparameters, the

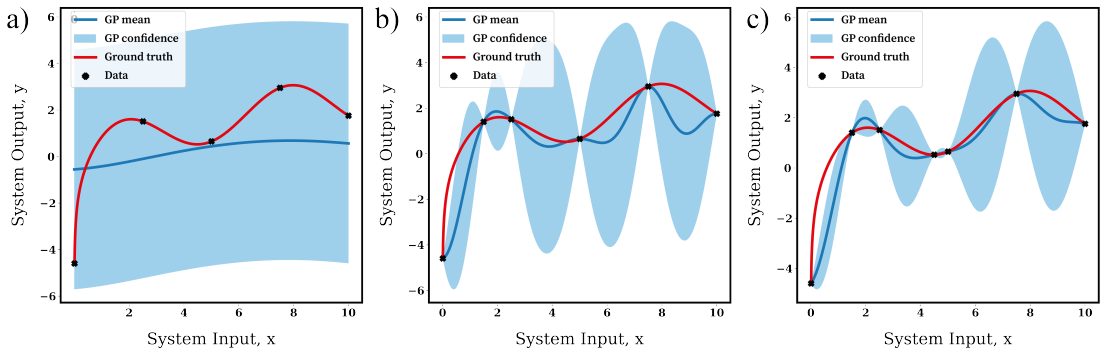


Figure 2.14: Expression of a Gaussian process posterior distribution for the modeling of a smooth noiseless function. The figure demonstrates the effects of an increasing number of data points in the model (a) 5, b) 6, c) 7 data points). In this instance, increasing the number of data points reduces the epistemic uncertainty estimate and the mean GP prediction becomes a better representation of the ground truth.

mechanism for inference exploits statistical relationships asserted to exist in the available data. This is opposed to generating predictions via the composition of parametric functions of the current state and control as in the case of neural networks. Further intuition behind BNNs and GPs is provided by Fig. 2.13 and 2.14. The parameters for all the models mentioned may be identified via the prediction error method, which is essentially a term descriptive of iterative, search-based Bayesian and Frequentist estimation routines (Astrom, 1979). Additionally, all of these models possess model structures uninformed by mechanistic knowledge. Rather the model structure typically enters as a hyperparameter, which must be optimised via methods such as, for example, random search (Bergstra and Bengio, 2012), grid search (Liashchynskyi and Liashchynskyi, 2019), Bayesian optimisation (Frazier, 2018), population based training (Jaderberg et al., 2017), as well as other stochastic search optimisation routines (Aszemi and Dominic, 2019; Lorenzo et al., 2017). However, the selection of these hyperparameter optimisation routines should consider a) the available computational budget and b) the expense of training and evaluating a model. Classical objectives in these hyperparameter optimisation problems are the k-fold mean squared error (Rodriguez et al., 2009; Fushiki, 2011) or other information criteria such as the Akaike (AIC) and Bayesian information criteria (BIC) (Konishi and Kitagawa, 1996). The use of information criteria has the particular benefit of incurring lower computational cost than k-fold, however, it is generally unclear which information criteria is actually best to use for a given modelling problem (Dziak et al., 2020).

Data-driven modelling has been widely used for modelling of dynamics in batch process systems. A recurrent neural network configuration was used in Zheng et al. (2022) to model the dynamics of a batch crystallisation process. Although the study was purely computational and leveraged a first-principles mechanistic model as a means of data generation; the proposed approach was able to accurately approximate the temporal evolution of temperature, solute concentration and crystal number produced by the mechanistic model. Whereas, Chen et al. (2022) utilised GPs to identify nonlinear batch process systems from an impulse response. Similarly, Bradford et al. (2018) demonstrated the use of GP state space models and artificial neural networks for modelling and dynamic optimisation of algal production processes. The GP approach was demonstrated to be particularly powerful for predicting the evolution of nitrate and the bioproduct (Lutein) concentration, however, the ANN was able to better predict biomass concentration. Further benefits of the GP were the ability to account for model uncertainty in the dynamic optimisation formulation, something that the ANN is not able to do naturally. Efforts have been made to account for the uncertainty of ANNs, primarily through the use of bootstrapping approaches. For example, Zhang (2004) presented a bootstrapped ANN approach to modelling polymerisation processes and this framework was also used for model identification and batch-to-batch optimisation in Zhang (2008).

Generally, all of the works detailed so far, which utilise data-driven models, have made approximations to state evolution between discrete time indices (i.e. via $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$). More recently a set of methods known under the umbrella of neural ordinary differential equations (NODEs) have been proposed to enable data-driven approximations to continuous-time dynamics (Chen et al., 2018a). Although, their application in the context of PSE has been relatively limited, they have been applied within a two-step framework, as proposed by Bradley and Boukouvala (2021), for the purpose of estimating mechanistic model parameters. The reasoning behind this is that NODEs can accurately estimate the time derivative of a system's state based on available data, hence their use for parameter estimation of more interpretable mechanistic models removes the requirements for schemes such as single and multiple shooting, and collocation to transcribe the ODE system into an NLP formulation (these schemes are described in more detail in the next section). Instead, the system is described purely

by the derivative predictions from the NODE and the interpretable mechanistic expressions derived from physical and chemical knowledge and is hence directly an NLP (i.e. not reliant on transcription via shooting). This allows for parameter estimation of multiple different models in one procedure, in the case that there is structural ambiguity. However, the estimation of NODEs is typically conducted via a single shooting procedure and they are known to be difficult to train (Turan and Jäschke, 2021). Hence, the potential benefits of this indirect approach are likely to be realised on a case-by-case basis. For more discussion regarding data-driven approaches to model identification, the interested reader is referred to Fuentes-Cortes et al. (2022); Thebelt et al. (2022); Pan et al. (2022).

The major differences in data-driven and mechanistic modelling essentially boil down to the following:

- Data-driven modelling is an excellent approach to making highly accurate interpolative predictions. However, the mechanistic model structure provided by first principles and semi-empirical approaches is amenable to extrapolating beyond the domain in which one has data.

- Additionally, mechanistic models have interpretability, which is largely lost when using a data-driven model such as a GP or BNN. There are methods that aim to provide interpretability for data-driven approaches, but they are comparatively limited relative to the clarity provided by mechanistic expressions (Doshi-Velez and Kim, 2017).

In order to combine the benefits of both approaches, there is interest in integration of both data-driven and mechanistic modelling in the form of hybrid modelling. An excellent overview of this area is provided by Duarte et al. (2004); Von Stosch et al. (2014); Sharma and Liu (2022); Fuentes-Cortes et al. (2022); Bradley et al. (2022). The major ideas used in construction of hybrid models is that: there can be a) a serial arrangement of mechanistic and data-driven models, with the data-driven model used to dynamically predict parameters in the mechanistic model; or b) the two can be deployed in a parallel arrangement where the data-driven model is used to predict the process-model mismatch. For example, Zhang et al. (2020b) utilised a serial approach to identify an approximation to a lutein photoproduction process. Whereas, Chen

et al. (2004) presented a serial approach to modelling a CSTR. In both cases, this essentially allows the data-driven model to 'mop' up what is not captured by the mechanistic model. However, in the arrangement provided by a) there is a degree of ambiguity regarding those parameters to hold constant and those that one should predict via the data-driven model. Conventionally, such ambiguities are handled either by using existing process knowledge, conducting further experiments and collecting more data (Von Stosch et al., 2014), or superstructure-based parameter estimation schemes (Wieland et al., 2021; Zhang et al., 2020b). However, given a) the use of hybrid models implies a lack of of process knowledge, b) there is expense associated with further data collection in process systems, and c) superstructure approaches for hybrid models can provide very large mixed-integer NLP; currently there is an open question as to how to resolve this ambiguity for hybrid models beyond manual and iterative model construction, especially when using neural networks (i.e. parametric models with many parameters).

Having outlined the major ideas in modelling of batch process systems, we now turn our attention to the solution approaches available for their use in process optimisation.

2.2.3 Solution approaches

In this section, we will review the major solution approaches to online optimisation of batch process systems when an approximate process model is available. Model predictive control (MPC) is the benchmark scheme in the domain of advanced process control and optimisation (APC) (Rawlings et al., 2017). The general idea of MPC follows: given the current state of the system, identify a discrete and finite sequence of control inputs that optimises the temporal evolution of a dynamical system over a discrete time horizon according to some objective function and operational constraints. Then input the first control in the sequence to the real process system and wait until the process transitions to the next time index, at which point observe the new state of the process and then subsequently re-optimise. This is essentially a way of incorporating state feedback into the decision-making process (Lee, 2011). The time horizon may either shrink or remain the same length at each iteration. The former is known via shrinking-horizon and the latter via receding-horizon MPC. The latter is summarised by Fig. 2.15.

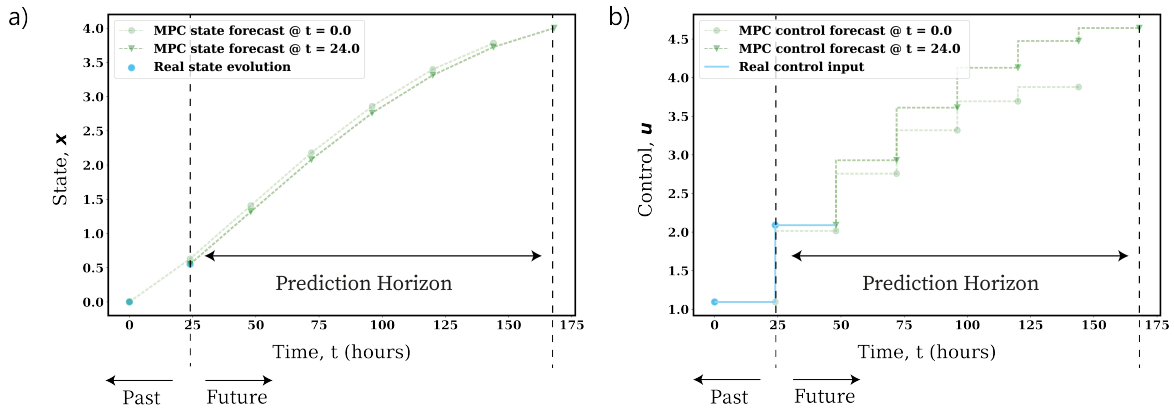


Figure 2.15: Demonstration of the use of state-feedback in receding horizon MPC for online optimisation of an uncertain, nonlinear fed-batch process. Optimised forecast and evolution of a) the state trajectory, and b) the control trajectory (composed of piecewise constant control inputs).

MPC is a direct approach to optimal control and is reliant upon the identification of some finite dimensional description of process evolution as a model. Various discretisation schemes, such as direct single-shooting, direct multiple shooting and direct collocation (Kelly, 2017), can be deployed to identify finite dimensional expressions when the underlying model is an ODE. These methods essentially transcribe the continuous time problem into a discrete-time problem, with a general NLP form. For example, single shooting discretises the control profile into such that the control input is parameterised locally over a number of finite time intervals (i.e. it is typically piecewise constant as in Fig. 2.15) (Ou et al., 2022). The state profile is then constructed by integrating the ODE system from the initial state over the time horizon, subject to changes in the control profile (Rawlings et al., 2017). This means that the state variables at each discrete time interval do not actually enter into the optimisation problem. Instead, the only decision variables are the parameters determining the local control parameterisations. Additionally operational constraints can be handled by enforcing they are satisfied at the discrete time indices that each local control parameterisation is updated.

However, single-shooting is known to run into problems for nonlinear systems with long time horizons and can produce non-physical state profiles due to a propagation of errors if the initial guess for the optimal control parameterisation is not close to optimality (Hussein et al., 2019). Hence, multiple shooting is often used to provide remedy (Baake et al., 1992). Multiple shooting follows the same intuition as single

shooting, but now the state variables at each time interval also enters as a decision variable. This is enabled by decomposing the original system into a number of separate dynamic systems (one for each discrete time interval). Each system is then integrated according to an initial state and the local control parameterisation in the same way as in single shooting but simply over the time interval rather than the entire horizon. To ensure the continuity of the overall state trajectory, the final state of a given subsystem is constrained to be equivalent to the initial state of the next (Beintema et al., 2021).

The final direct transcription method widely used is direct collocation. Direct collocation discretises both the state and control profiles in a similar way to multiple shooting, however the state evolution over each local time interval is instead approximated via orthogonal polynomials. The problem then reduces to identifying the state profile, the parameters of the polynomials and the local control parameterisations. Collocation is particularly advantageous for nonlinear systems with path constraints as it enables one to easily enforce constraints at discrete points (known as collocation points) within a finite time interval (Arellano-Garcia et al., 2020). For more information on these methods the reader is directed to Gautschi (1996); Tjoa and Biegler (1991); Kelly (2017); Rawlings et al. (2017).

The major benefit of MPC is that if operational constraints are imposed upon the problem and the underlying model is a perfect description of the system, the solution identified will be (at least locally) optimal under both the dynamical model and operational constraints, given that the control solution must satisfy the Karush-Kuhn-Tucker (KKT) conditions. Various feasibility and stability guarantees can also be provided under appropriate assumptions. However, if the dynamics are subject to general uncertain elements (e.g. disturbance and parametric uncertainties), then various formulations, which consider this uncertainty are required. Perhaps the most known formulations are robust and stochastic MPC. In the following, discussion will focus most on solution methods applicable to the use of nonlinear models.

Robust MPC typically comes in two flavours, in both cases one typically assumes

the uncertain parameters, $\mathbf{s} \in \mathbb{S} \subset \mathbb{R}^{n_\mathbf{s}}$,[17] belong to a compact set. The first is minimax MPC. To the author's knowledge no minimax formulations exist for systems with nonlinear dynamics, however, for linear systems it can optimise for the worst-case uncertainty. Beyond lacking the ability to handle nonlinear dynamics, the major trouble with minimax is that it can be conservative and generally becomes computationally intractable for anything other than short time horizons (Lofberg, 2003). However, in the case that one has an additive disturbance and linear dynamical system, then approximate problems may be identified through reformulation and efficiently solved instead (Löfberg, 2003).

A more popular approach to robust MPC is provided by tube-based MPC, which has been applied to both linear (Fleming et al., 2014; Rawlings et al., 2017; Langson et al., 2004) and nonlinear systems (Cannon et al., 2011; Mayne et al., 2011). The high-level idea here is that as the system transitions between discrete time indices, the system state will vary within some 'tube' around the nominal state evolution. The intuitive idea of tube based robust MPC is then to essentially make an adaptation to the nominal control input, in order to drive the system towards the reference trajectory provided by the nominal state profile. As a result, the system will vary some distance around the nominal profile, such that a 'tube' is constructed centred around the nominal profile as the system evolves through time (Zeilinger et al., 2014). In the case that the uncertain parameters are not described by compact sets, then operational constraints may be tightened to ensure their probabilistic satisfaction (Rawlings et al., 2017) through formulation of what is known as chance constraints (Nemirovski and Shapiro, 2006). However, if the uncertainty is described by compact sets, constraint tightening can enable constraint satisfaction absolutely (Köhler et al., 2020).

Stochastic MPC formulations instead attempt to optimise for the expected performance of the system and typically handle constraints probabilistically via chance constraints (but they can also be handled robustly if the uncertainties belong to compact sets). Excellent overviews are provided in Mesbah (2016); Heirung et al. (2018); Farina et al. (2016). Generally, this requires propagating the uncertainty associated with the state over the discrete time horizon. Chance constraints can be handled by backing off

[17] As before these uncertain parameters may be descriptive of parametric uncertainty or some, for example, additive disturbance.

the nominal state from the constraint boundary to allow for process variation. In nonlinear systems, this can become challenging, given that nonlinear transformations of distributions do not preserve their form (i.e. a nonlinear transformation of a Gaussian random variable is no longer Gaussian). As a result, in nonlinear systems a number of methods exist to propagate uncertainty to handle constraints. The first approach is to conduct closed-loop Monte-Carlo simulations offline to estimate the back off distance for the purposes of computation online. This has been demonstrated in both hybrid and purely data-driven Gaussian process models (Bradford et al., 2021b, 2020).

The second approach to mention is encompassed by the unscented Kalman filter and the concept of sigma points (García-Fernández et al., 2015), which essentially enables one to make Gaussian approximations to nonlinear transformations of Gaussian random variables (Bradford and Imsland, 2018). Chance constraints can then be handled through use of statistical inequalities, which enable identification of deterministic finite-dimensional surrogate expressions that are functions of the expected uncertain state and state covariance matrix at a given discrete time index, as well as the probability with which one would like to satisfy the constraints (Paulson et al., 2020). In the case of more than one operational constraint (i.e. joint chance constraints), this reformulation is encompassed by application Boole's inequality and the Cantelli-Chebyshev inequality (Lin and Bai, 2011). For example Bradford and Imsland (2018), used sigma points to propagate uncertainty in an NMPC scheme for a chemical catalytic fed-batch reactor. The results demonstrate that the higher the probability of constraint satisfaction defined, the more conservative the performance in the process objective. This represents a trade-off to be considered by the process operation. Meanwhile, Thangavel et al. (2020) demonstrated uncertainty propagation via sigma points and then subsequent tight box over-approximations to the uncertain state at each discrete time index in the horizon to ensure constraints robustly for a semi-batch polymerization reactor. This is enabled because box sets are compact. The tightness of these over-approximations reduces conservatism often observed in classic robust approaches.

Another approach to propagating uncertainty is provided by polynomial chaos expansions (PCEs), which again use orthogonal polynomials, but this time within the context of approximating the probability density functions associated with the state

discrete-time trajectory (Mesbah et al., 2014). PCEs allow for one to approximate both the mean and variance of the pdfs via the deterministic coefficients of the polynomials and hence to handle chance constraints. However, the approximation of these moments is known to be sensitive to specific hyperparameters within the scheme that are generally difficult to validate Paulson and Mesbah (2018).

Perhaps the final means of uncertainty propagation is the approach provided by stochastic programming. Essentially, this consists of generating multiple scenarios by realising uncertain parameters, incorporating them into a model and then enforcing constraints. Solution to this approach identifies a) a here and now decision (as an immediate control input) and b) recourse decisions, which are used to react to the uncertainty as realised in the future (de la Penad et al., 2005; Yu and Biegler, 2019). However, when the number of scenarios used in the model is low, constraint satisfaction in reality can become problematic, primarily because the empirical distribution sampled does not well approximate the underlying reality (Mesbah, 2016). Further given that the first two moments of the state trajectory are not approximated, classical deterministic reformulations of chance constraints cannot be readily used. However, there are means by which to determine the number of scenarios required to ensure constraints with a given probability, although generally these ideas are attributed to the robust optimisation community (Campi et al., 2009). Clearly, however, as the number of scenarios grows, the size of the resultant model can cause solution to become intractable[18]. This has led to interest in various decomposition schemes, which enable one to separate a large problem into relatively smaller sub-problems (Birge, 1997; Kang et al., 2015; Krishnamoorthy et al., 2019). Although some degree of centralised computation is retained, this enables one to exploit parallel computation for solution (Marti et al., 2015). Other works look to selectively generate the scenario tree by identifying worst-case uncertainties, essentially reducing the size of the model and retaining tractability, whilst handling constraints (Krishnamoorthy et al., 2018; Holtorf et al., 2019; Shang and You, 2019).

Having reviewed the major mathematical programming solution approaches[19] to

[18]In the domain of linear dynamics models, convex approximations to the original problem may be obtained as demonstrated in the scenario community (Campi et al., 2009).

[19]Other formulations exist including distributionally robust model predictive control. We direct the reader to Zhong et al. (2022, 2021) for more information.

online optimisation of batch processes, focus will now be directed towards a review of production scheduling within the process industries. These problems share similarity to both MDPs and batch process systems in that production schedules are typically generated over discrete, finite time horizons and plant dynamics are also subject to uncertainties.

2.3 Production scheduling

The process industries are ultimately concerned with identifying the most cost effective and efficient route to convert raw materials into products that can be sold at a price determined by the market. However, the most efficient route is not determined solely by an underlying physical process or chemical reaction mechanism. It is also determined by the operational decisions taken within a processing environment, which affect for example, the availability and inventory of materials, as well as the timing, sequencing and sizing of production operations within the plant (Castro et al., 2018). These decisions have influence not only the cost effectiveness and efficiency of production, but also relationships held between the producer and client (Payne and Frow, 2006). For example, timely and reliable delivery of products to clients has a positive effect on a company's reputation. Whereas unreliable and late deliveries have the potential to destroy a client's perception of a company. These considerations are present in almost all manufacturing industries, which in the UK contributed to over £402 billion[20] of domestic sales in 2019 (ONS, 2021).

The operational decisions described are ultimately captured by production scheduling and tactical supply chain management decisions. Generally, production scheduling decisions are identified within offline decision-making environments (Verderame et al., 2010). In the offline scheduling domain, for example, scheduling decisions are identified over a time horizon. The reasoning for this primarily arises in the requirement to provide staff with a degree of certainty over the future production operations, and less so for the purposes of optimality. Additionally, these problems are often described by large MILP, which means iteratively solving these problems in a fashion similar to MPC is often not feasible. As a result, as we shall see in Section 2.3.3, this leads to a

[20]This figure was reported as it reflects what one would deem 'normal' production levels, since given the COVID-19 pandemic these figures have fallen nearly 10% (ONS, 2021).

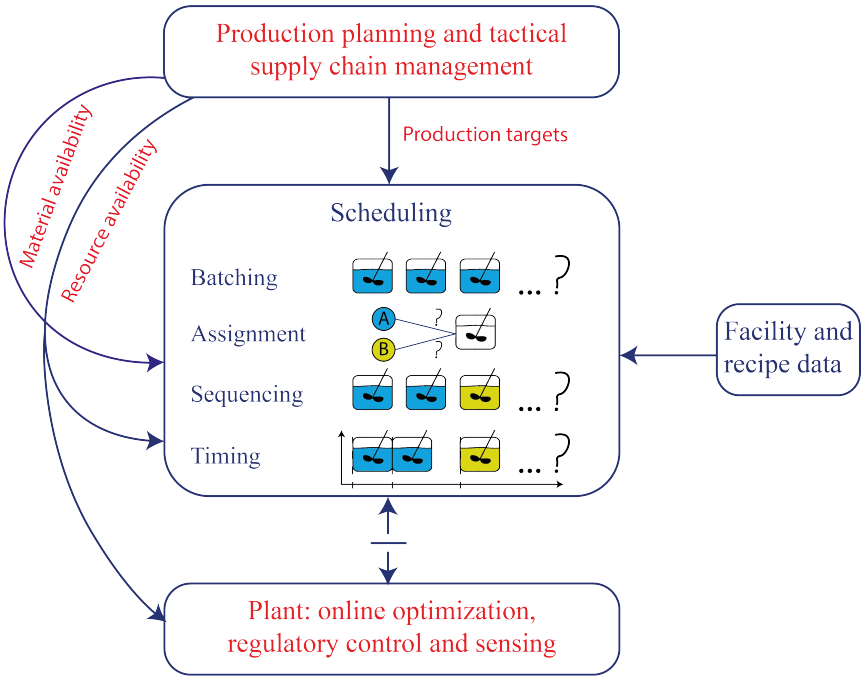


Figure 2.16: Intuition behind the decisions provided by the scheduling function and its interaction with the plant and upper-level supply chain management functions.

focus on the notion of robustness in generation of the schedule. This is primarily because the scheduling problem is in fact subject to elements of uncertainty. For example, client demand is rarely known with certainty over the time horizon, one would like to optimise for. Additionally, uncertainty arises within the plant via e.g. batch processing time uncertainties, and unit availability may be subject to uncertain unit breakdowns (Gupta et al., 2016). As we saw in Section 2.1.1, optimal decision-making is reactive to realisations of uncertainty. This has led to interest in rescheduling and online scheduling paradigms, where the schedule may be re-computed via a receding-horizon MPC type approach at discrete instances in time (Tang et al., 2010; Subramanian et al., 2012; Georgiadis et al., 2019). Alternatively, reactive scheduling decisions may be made through use of rule-based heuristics developed within the operations research (OR) community or knowledge held by operatives (Harjunkoski et al., 2014). Intuition behind the scheduling function is provided by Fig. 2.16.

In the following, we will explore the major classifications of production environments, as well as modelling and solution approaches for production scheduling problems.

2.3.1 Classification of batch production environments

Regardless of the decision-making environment, the ultimate modelling and therefore solution approach will largely be governed by the nature of the underlying production environment. Broadly speaking, the batch industry can be subdivided into sequential, network and hybrid processing environments. In all of these classifications given the available equipment items, the scheduling function may be required to assign, sequence and size production tasks in equipment items through time. Generally, this is under the assumption that production of each batch has a processing time and converts available raw materials to potentially multiple products, and that there is a finite number of resources within the plant. Even though the scheduling decisions required across these environments are similar, their classification has an effect on the nature of the decisions required by the scheduling function.

Sequential production environments

Sequential production environments are thought as more widely within the OR community as flexible jobshop or flexible flowshop problems, provided no sizing decisions are required (Maravelias, 2012; Méndez et al., 2006). In the following, we shall explore the major subdivisions of batch sequential production environments: multistage and multipurpose problems.

The multistage batch environment essentially requires the conversion of raw materials to final products via the a sequence of intermediate production stages. Intuition is provided by Fig. 2.17. Each equipment item is limited to processing one production job at a time, a finite subset of the equipment can process in parallel within a given stage, and no splitting or mixing of a batch (job) is allowed throughout the sequence of process stages. For example, this is common in manufacture of active pharmaceutical ingredients where batch identifiability needs to be retained inline with good manufacturing practice (GMP) (EMA, 2000). Additionally, all the products go through the stages in the same sequence (although products may skip a processing stage). Multistage multiproduct problems are generally known within the operations research community via the flexible flowshop description (Méndez et al., 2006).

Multipurpose production environments might also consist of multiple stages of production. However, the major difference here is that the products do not need to

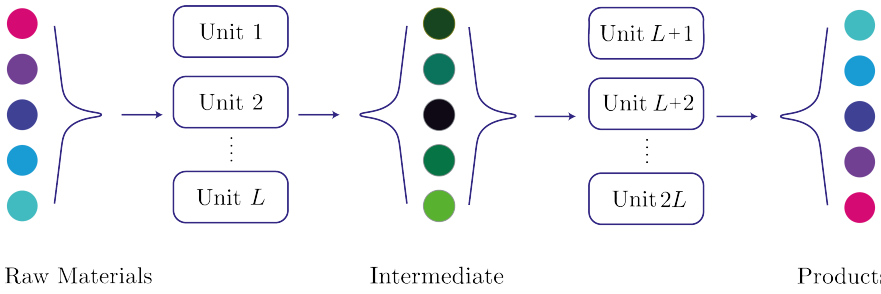


Figure 2.17: Multistage batch production environments. Here, a two stage production environment is shown. Reactants are converted to intermediates and then products through multiple processing steps.

be processed in the same sequence of stages (Maravelias, 2012). Additionally, mixing and splitting of batches is not allowed, however, a product might observe the same stage twice within the sequence of operations. Multipurpose problems are known via the flexible jobshop description. General batch environments that allow for mixing and splitting are also common within industry. These processes are considered more widely under the network classification, which will be discussed subsequently.

Network production environments

In network processing environments, there is free mixing and splitting of batches, and material can flow from storage vessels to reactors and to subsequent storage (provided appropriate assumptions on connectivity are made). Given that material can be mixed freely the concept of a batch essentially becomes redundant. Similarly, the concept of a production stage becomes ill defined as the route for a product is generally flexible. To handle this the frameworks provided by the state-task network (STN) and resource-task network (RTN) have been deployed as process modelling frameworks (Maravelias, 2012).

Generally, however, many production environments consist of both sequential and network structures and so hybrid structures also exist. This is discussed at length in Maravelias (2012) and we direct the reader for more information.

2.3.2 Modelling approaches

The major modelling considerations within production scheduling problems are the framework chosen for a) representation of time and b) representation of production

environment. We briefly discuss different approaches in the following.

Representations of time

The representation of time within scheduling models is a key consideration. Broadly speaking there are two approaches for the explicit handling of time: discrete-time models and continuous-time models. Both approaches essentially operate by defining a time-grid. Discussion is focused towards the modelling of time in sequential environments.

In discrete-time models, the time horizon is discretised into a finite number of time intervals (in a similar manner to which control trajectories are discretised within MPC approaches) either specific to a unit or globally for all units (Velez and Maravelias, 2013). A production task is then assigned to a unit at a given time index, with the operation terminating at the boundary of a given time interval. In general, units can process one production task or job at a time, and so the scheduling model will then require that no other production jobs be assigned to the unit for the process time of the task allocated, which may span multiple time intervals (Floudas and Lin, 2004). Generally, discrete-time formulations lead to relatively large MILP model sizes. This is because for inherent sub-optimality not to be introduced into the model, the discrete-time interval should be the greatest common factor of all processing times (Maravelias, 2021b). If this is not the case, production tasks may finish in the middle of a time interval, which can then lead to under-utilisation of the equipment (given that a production task may only be assigned to a unit at a given time index). In order to handle these larger model sizes, interest has been placed in enhancing the efficiency of optimisation. This has been approached from the perspective of adding additional 'valid inequalities' or inequality constraints to reduce the number of infeasible subproblems identified through the iterative search of branch and bound-based solvers (Dedopoulos and Shah, 1995; Sundaramoorthy and Maravelias, 2008). Other approaches have explored the use of decomposition techniques, exploiting the temporal structure of the problem (Bassett et al., 1996).

Whereas, in continuous-time models the time horizon is not discretised but rather events are defined according to the sequencing of production tasks. Continuous-time formulations provide much smaller MILP models than discrete-time models, and as

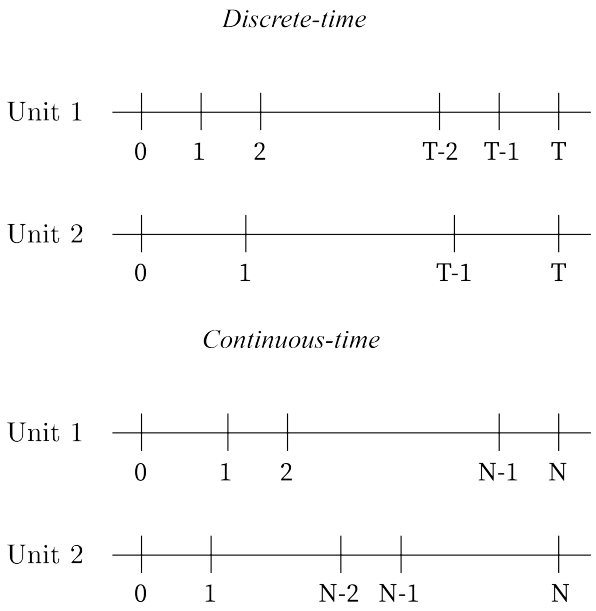


Figure 2.18: Discrete-time and continuous-time modelling of time. In both cases unit-specific representations are shown (i.e. the handling of time is not global).

a result the majority of sequential production environments utilise continuous-time formulations (Harjunkoski et al., 2014). The time at which events take place is essentially dictated by the sequence of tasks in equipment items and the processing times of those events. However, the manner in which event points are determined depends on approach. Firstly, there is a precedence based approach, which builds the time-grid implicitly by determining the sequence of tasks in units. It is relatively intuitive (Cerda et al., 1997; Liu and Karimi, 2008), but suffers from worse relaxations due to e.g. Big-M reformulations of bilinear terms (Harjunkoski et al., 2014; Castro et al., 2018). Alternatively, there are slot based representations, which require one to predetermine the number of event points (which is not a trivial exercise) but result in tighter models. As a result, event points may be specific to a unit or global (Mouret et al., 2011; Maravelias, 2021b). For example, Liu and Karimi (2008) present a continuous-time precedence based MILP model for a multistage sequential environment. Whereas, Mouret et al. (2011) explore use of slot-based representations. Intuition is provided by Fig. 2.18.

Representations of production environment

In general sequential production environments, there is no mixing and splitting of batches. This allows for one to model the relationship between the objects at hand,

which are production tasks, $i \in \mathbb{I}$, and units, $j \in \mathbb{J}_k$, which belong to respective stages, $k \in \mathbb{K}$ (Georgiadis et al., 2019). The aim of the scheduling function is then to identify the assignment and sequencing of those operations over the time horizon, subject to the constraints imposed on job route.

There are multiple process representations in sequential environments, however an example is the disjunctive graph (which applies more specifically to multipurpose environments, i.e. job shops) (Šeda, 2007). The graph itself is defined by a set of vertices, which define production tasks and each vertex is associated with features that identify the job (batch) and the respective stage/task. The graph is also composed of edges which connect any two vertices. There are two types of edges: disjunctive, which are undirected and whose features define the unit capable of processing the tasks the edge connects; and conjunctive, which are directed and defines the viable sequence of tasks that a job (batch) can be processed by. All initiating tasks associated with a job are connected to a dummy vertex, as are all terminating tasks. Disjunctive graphs have utility in providing visual definition of the allowable operations within a jobshop and hence can be used to inform definition of an optimisation model (Błażewicz et al., 2000). Additionally, a schedule may be identified by turning each disjunctive edge into a directed edge. This enables one to define the sequence of production tasks within units and allows one to define the state of the job shop at any given time (Zhang et al., 2020a; Park et al., 2021). For more discussion of process representation in sequential environments please see Šeda (2007); Panwalkar and Koulamas (2019).

There have been a number of different MILP models defined for sequential environments. There is large heterogeneity between these models primarily due to the diversity in different production environments. For example, in Cerda et al. (1997), the authors presented a continuous-time MILP formulation for a single-stage multiproduct facility with multiple units operating in parallel. The basis of the model is the identification of a viable sequence of jobs in equipment items, and as consequence the timing at which these operations take place. However, the model does not consider basic restrictions on resources or materials. This leads to for example, works such as Méndez et al. (2000), which extend the model (of Cerda et al. (1997)) to account for storage constraints. Similarly, work by Sundaramoorthy and Maravelias

(2008) demonstrates a continuous-time MILP model for a multi-stage sequential production environment, which also allows for batching decisions to be made; whereas Sundaramoorthy et al. (2009) present a discrete-time MILP model for a multi-stage sequential production environment where batching decisions are not required, but utility constraints are imposed.

Meanwhile, in general network production, mixing and splitting of batches may be allowed and basic restrictions on resources, such as storage should be considered to ensure feasibility. Therefore, a minimum requirement for the development of an efficient and general mathematical model for a network production environment, is the ability to define the relationships between resources (such as raw materials, intermediates, final products, utilities, equipment items, etc.) and production tasks (e.g. operations which transform materials). This has led to the development of various process modelling frameworks including: the state-task network (STN), as proposed in Kondili et al. (1993); Shah et al. (1993) for discrete-time and in Maravelias and Grossmann (2003) for continuous-time representations; and the resource-task network (RTN) (Pantelides, 1994; Castro et al., 2004). These frameworks enable one to graphically and mathematically represent the relationships of material transformation and resource utilisation within the plant and as a result to consider general production scheduling restrictions. The STN represents the relations between states (i.e. the consuming and produced materials) and production tasks. Whereas, the RTN is slightly more general and considers not just the material - production task relationship, but the relationship between general resources (as mentioned above) and production tasks. The STN and RTN are detailed by Fig. 2.19a) and b) respectively. The STN and RTN provide basis for many optimisation models and frameworks proposed for network production scheduling and integrated problems, as demonstrated in for example McAllister et al. (2022); Perez et al. (2022); Castro et al. (2018).

Having formalised the major modelling approaches, we now explore the major solution approaches.

2.3.3 Solution approaches

Solution approaches can be broadly separated into exact methods and heuristic methods. Both will be explored in the following with focus on handling uncertainty.

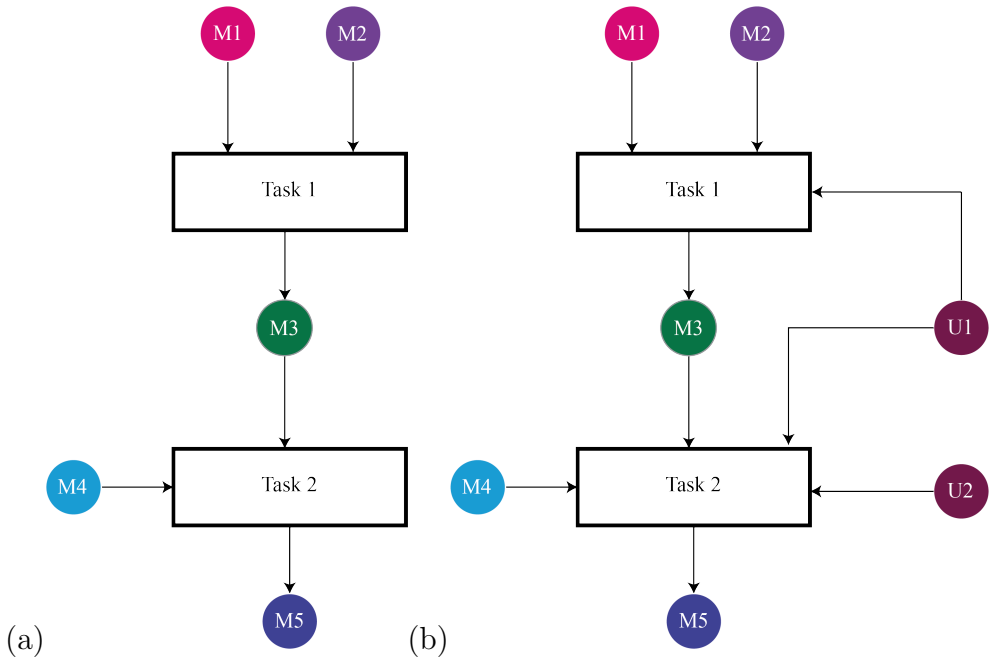


Figure 2.19: a) The state-task network (STN) process representation and b) the resource-task network (RTN) process representation. The states in a) and resources in b) are represented by circles, whereas the production tasks are represented by rectangles.

Exact solution methods

Exact solution methods to MILP problems are essentially encompassed by branch and bound methods and other MILP solvers. These primarily rely on use of nominal data within the model. There are stochastic and robust formulations, however, which we will explore in the following.

Stochastic approaches to scheduling generally rely on describing uncertain parameters from known distributions and then constructing deterministic MIP models and solving them within the framework provided by stochastic programming. An excellent review is provided by Li and Grossmann (2021). In cases where the support of uncertain parameters is continuous[21] or the uncertainty is realised at numerous time indices within the horizon, the size of the model generally becomes intractably large if exact description is used. This leads to the use of approximate formulations, such as two-stage and multi-stage stochastic programming. The general idea here is that instead of describing the uncertainty exactly, one can instead solve a smaller,

[21]In the case of multiple uncertain parameters, this point applies even to incidences where the support for a variables is small and discrete. See Shapiro and Philpott (2007) for more discussion in this vein.

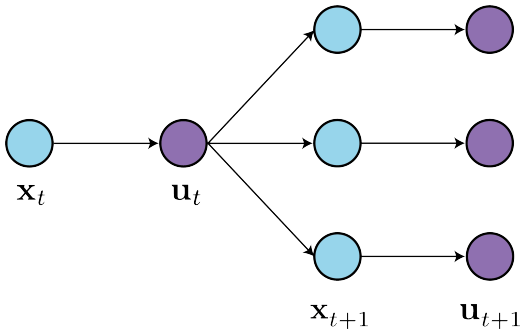


Figure 2.20: An example of two-stage stochastic programming. Uncertainty is generally assumed to be realised once within the time horizon.

finite-dimensional deterministic model with approximate description of the uncertainty provided by Monte Carlo sampling of uncertain parameters.

In two-stage stochastic programming, the uncertainty is assumed to be realised once within the time horizon. In the first stage, decisions are identified with certainty, and in the second stage recourse decisions are identified to react to the different realisations of uncertainty (Ruszczyński and Shapiro, 2003). As a result, the number of recourse decisions essentially grows linearly with the number of scenarios. Only the decisions known with certainty are then implemented, with recourse decisions re-identified later in the decision process. This is described by Fig. 2.20. This means that stochastic programming is ideally suited to online receding horizon implementations given that the recourse decisions are not known with certainty. Multi-stage stochastic programming follows exactly the same idea, but now uncertainty is realised at multiple points within the decision horizon (Shapiro et al., 2021). As a result, the number of recourse decisions grows linearly in the number of stages and number of realisations at each stage. The use of stochastic formulations is arguably much more prevalent within production planning (tactical supply chain operations), primarily because the problem class observed there tends to be LP, rather than MILP (Li and Grossmann, 2021) primarily because these problems tend to deal with material flow rather than assignment decisions as in production scheduling. However, application of stochastic programming to scheduling has been demonstrated in a number of works. For example, Chu and You (2013) demonstrate an integrated production scheduling and dynamic optimisation approach for batch process systems with two-stage stochastic programming. They take advantage of the structure of the problem via decomposition methods enabling identification of a solution in a number of hours of computation time. Similarly, Sand

and Engell (2004) propose a stochastic programming model for online scheduling of a batch polystyrene network production environment. Again, they utilise decomposition methods to enhance computation. Zhang et al. (2016) present an integrated production scheduling and electricity procurement stochastic programming model to handle uncertainty in demand and electricity price. They demonstrate the ability to pose risk-sensitive formulations such as the conditional value-at-risk. With the use of relatively few scenarios and advanced decomposition techniques the model could be solved to an optimality gap of a few percent within two hours. Finally, Ye et al. (2014) demonstrated a stochastic programming approach to production scheduling in a large-scale steelmaking continuous casting process under demand uncertainties. The model solution time was on the order of hours. The computational times reported for both Zhang et al. (2016); Ye et al. (2014) were reliant on scenario reduction techniques that aim to pick a subset of the scenarios in the full space model that best represent the underlying uncertainties (Li and Floudas, 2014, 2016). Solutions to the full space models defined took significantly longer. This generally limits the applicability of stochastic programming based solution approaches in the context of online decision processes.

The other uncertainty aware formulation for production scheduling is provided by robust optimisation approaches. As mentioned in previous sections a key idea in robust optimisation is to optimise for the worst-case scenario within an uncertainty set. This enables use of a much smaller model than in the case of stochastic programming. However, it tends to produce more conservative solutions because it generally does not consider the option for recourse decisions to be made later in the time horizon. For example, Li and Ierapetritou (2008a) examine different robust formulations to production scheduling within a network production environment (the model is based on the STN representation) with uncertain processing times and demands. They discuss the benefits of satisfying constraints with some probability rather than robustly and highlight that this results in a less conservative solution. Zhang et al. (2018) present a method that is able to consider the correlation between parameters, which together with a cutting plane method is able to reduce the size of classical uncertainty sets (e.g. ellipsoid, box etc., see Ben-Tal et al. (2009) for more information) estimated on parameters. This has the effect of reducing conservatism in the solution. This was demonstrated via scheduling of an ethylene production operation with uncertainty in

the consumption rate of fuel gas. Finally, more recent work via McAllister et al. (2022) has extended previous ideas presented in Gupta et al. (2016); Subramanian et al. (2012) that drew connections between the online scheduling problem and model predictive control. Specifically, McAllister et al. (2022) utilised ideas in tube-based robust non-linear model predictive control to ensure robustness in rescheduling policies derived from a discrete-time MILP formulation. This is achieved by essentially constraining the solution to satisfy some terminal set. This also enables one to consider shorter time horizons, but requires the availability of a nominal reference solution (which is used to construct the terminal set). Clearly, this is a very strong contribution, but it does require iteratively solving an MILP problem, which may provide computational barrier in practice.

Heuristic solution methods

In order to approach the computational barrier associated with use of exact approaches online, various heuristic solution methods have been demonstrated.

Perhaps one of the most popular approaches is the use of metaheuristic optimisation methods. The general idea here is to use some directed, but stochastic search strategy to permute the schedule and then evaluate it in an underlying simulation model via sample approximations to some measure of the objective (Juan et al., 2015; Khan, 2018). This primarily relies on the fact that a high degree of parallelism can be achieved with simulation and that this is generally cheaper than solving an MILP. Although, it should be noted that often metaheurisitc methods are also used when the problem itself becomes large scale (i.e. has a large number of resources, tasks and units). Some examples of stochastic search methods include genetic algorithms (GA) (Banzhaf et al., 1998; Laroque et al., 2012), particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995a; Lin and Huang, 2014), simulated annealing (SA) (Kirkpatrick et al., 1983a; Altiparmak et al., 2002), artificial bee colony (ABC) (Gao and Liu, 2012; Li et al., 2020) and tabu search (Glover, 1989; Vela et al., 2020). Each method has its own properties and hence selection is generally tailored to the problem at hand. For example, simulated annealing is a relatively exploitative algorithm (i.e. it generally does not search the space widely), whereas ABC is generally more explorative (i.e. searches the decision space widely). For more information on these methods, please

see the papers referenced above, as well as the work provided by Sörensen et al. (2018); Camacho-Villalón et al. (2022).

The use of metaheuristic optimisation methods has been well documented in production scheduling problems, particularly within the operations research community. For example, Stützle et al. (1998) used an ant colony optimisation method, in order to optimise a large-scale multipurpose sequential production environment. The ant colony method maintains a population of ants and coordinates mutations to each of their positions based on update rules inspired by the collective behaviour of ant colonies when searching for food. Zhang et al. (2019b) hybridised the ant colony method with a local search algorithm to improve the exploitative nature of the algorithm and demonstrated their method on a steel manufacturing environment with uncertainty in the processing times. Almeder and Hartl (2013) proposed a variable neighbourhood search method (Hansen et al., 2010) for optimisation of a multipurpose sequential production environment with uncertain processing times. They reported improvements of 3-10% in the case studies examined over various rule-based strategies. More recently, Cao et al. (2021) demonstrated a gravitational search algorithm for scheduling of a hybrid flowshop. Major focus in these methods is directed towards what is the best representation of the rule and how best to search over it (Nguyen et al., 2019).

One major comment to be made here is that although, metaheuristic search methods are very capable at flexibly handling uncertain production environments, typically identification of a schedule does not consider the ability to identify recourse decisions. This is primarily because metaheuristics are generally most effective when the dimensionality of the decision variables is in the region of $10^1 - 10^3$ (Hussain et al., 2017). Based on Bellman's principle of optimality, this is likely to render the solution suboptimal. Additionally, if they were to be implemented online, their implementation does require re-running a search process. Although this is likely to be more computationally efficient than online MILP implementations, it is by no means cheap.

The requirement for an online solution method to be cheap has led to the wide use of rule-based strategies within production scheduling. Many of these heuristics have been developed specifically within the operations research community and can be thought as the PID controllers of production scheduling. For example, priority-rules in the context of multiproduct sequential production environments, may be used to determine the

feasible sequence of jobs on machines, when uncertainty is realised (Sweeney et al., 2019). Typically, the computation of job priorities accounts for nominal processing time, job due date etc. and considers each unit to be operating alone. This is discussed extensively in (Haupt, 1989). Common rules include prioritising those jobs with the minimum release time, shortest or longest processing time, earliest due date, and longest due date (Singer, 2001; Zhang et al., 2020a). Due to the diversity of potential priority rules, there has been interested in developing hyper-heuristic methods, which search over an optimal sequence of heuristic priority rules (Ochoa et al., 2009; Yska et al., 2018; Shady et al., 2020) or conditional to the current 'state' of the system (Riedmiller and Riedmiller, 1999). There has also historically been interest in the development of expert-rule based scheduling systems, which are derived from operator knowledge (Suresh and Chaudhuri, 1993). A good example of this is described by the ISIS system developed at Carnegie Mellon (Fox, 1994).

Having provided extensive background to decision-making in process systems engineering and the main methods in this thesis, we now direct attention to the main research objectives provided by this thesis.

Chapter 3

Using process data to generate an optimal control policy via apprenticeship and reinforcement learning

This research item is published in the American Institute of Chemical Engineers (AIChE) Journal, and is accessible via the following reference:

Mowbray, M., Smith, R., Del Rio-Chanona, E.A. and Zhang, D., 2021. Using process data to generate an optimal control policy via apprenticeship and reinforcement learning. *AIChE Journal*, 67(9), p.e17306.

3.1 Introduction

Recent initiatives for efficiency improvements in industrial process operation has driven interest in the development of high performance, advanced process control (APC) schemes. Reinforcement learning (RL) has achieved impressive results on benchmark game-based control tasks (Mnih et al., 2015b; Heess et al., 2015), providing an avenue for research in translation to APC. In spite of its high potential, RL has yet to produce any meaningful impact in the (bio)chemical process industry. This work presents a two-step approach to RL-based policy learning, which leverages process data to parameterise an existing control law and then improves the performance of closed loop policy, and reduces technical investment, as well as data demand.

RL constitutes a subfield of machine learning (ML), which aims to learn optimal control policies. Here, the control problem is formulated as a Markov decision process (MDP), which describes decision making as a value maximisation problem. MDPs construct a probabilistic framework for the discrete-time evolution of a stochastic decision process, with the cost (or value) associated with a control policy, and ultimately process trajectory, evaluated by a reward function. Explicitly, MDPs provide a mathematical basis for sequential decision-making in stochasticuch control further. Additionally, the approach promises to increase the learning efficiency of RL-based control policies, reducing computa environments, which is a description common to process control (Kirk, 1998). Fig. 3.1 details the interpretation of process control as an MDP. The structure of MDPs provide natural closed loop feedback control.

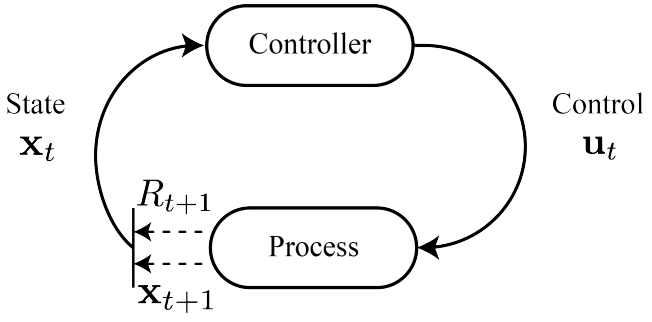


Figure 3.1: Translation of the framework provided by MDPs to process control, where the process is analogous to an environment, and the controller to an agent. $\mathbf{x}_t$ is representative of the true system state at discrete time t; $\mathbf{u}_t$ is the control action computed by the control law at discrete time t; and R_{t+1} is the scalar feedback signal (reward) indicative of the quality of process evolution at time *t+1*.

Solution to an MDP provides a policy, $\pi(\cdot)$, which minimises the expected cost or equivalently maximises the expected value associated with the evolution of process state. Such a policy satisfies the Bellman optimality equation, which is a discrete-time analogue to the continuous-time Hamilton-Jacobi-Bellman (HJB) equation (Kirk, 1998). Dynamic programming (DP) methods provide exact solution to the Bellman optimality equation. However, such an approach assumes knowledge of the exact process dynamics. DP becomes additionally impractical in the highly dimensional continuous state and action spaces often observed in the process industries (Liu et al., 2017). In contrast, RL methods do not require knowledge of the exact process dynamics to learn a solution policy. Instead, RL learns from experience of the process, allowing for $\pi(\cdot)$ to be recalibrated as the process evolves through time via process data (Petsagkourakis et al., 2020b). Furthermore, RL has shown significant industrial potential as demonstrated in a number of research works, which have explored application to the calibration of PID controllers (Lawrence et al., 2020); set point tracking (Spielberg et al., 2019); dynamic optimisation of nonlinear, stochastic systems (Petsagkourakis et al., 2020b; Kim et al., 2020a; Kim and Lee, 2020); de novo drug (Gottipati et al., 2020) and protein design (Angermueller et al., 2020); and in augmentation of the performance of various model predictive control (MPC) approaches (Gros and Zanon, 2019b; Zanon et al., 2020). Indeed, the potential use of RL draws discussion of its relation to MPC in the development of APC schemes. MPC schemes require periodic recalibration, which demands expense in technical expertise and often process down-time. The data-driven nature of RL could well mitigate this. Further, the framework provided by MDPs accounts for process stochasticity in a closed loop manner, converse to MPC where decisions are based on open-loop simulation of the process model, with the loop only 'closed' upon observation of the system state at the next discrete time index. Hence, inputs from an RL controller will account for disturbance whereas MPC may not. This provides a theoretical basis for the benefit of RL over MPC controllers.

One set of RL algorithms are known generally as policy optimisation methods. Policy optimisation methods aim to learn a policy by implicitly learning the value or cost over the decision space (Schulman et al., 2017b,a, 2018a) and directly parameterising a policy. There are a number of approaches to policy optimisation as underpinned by evolutionary strategies, finite difference and policy gradient methods

(Lehman et al., 2017; Sutton et al., 1999). Policy optimisation methods have been deployed for tasks including dynamic optimisation of nonlinear stochastic processes (Petsagkourakis et al., 2020a) and tracking problems (Lawrence et al., 2020). For further review of RL methods and their application within the process industries, we direct the reader to the following works (Spielberg et al., 2019; Shin et al., 2019).

The learning process encapsulated by RL demands both time and technical investment in policy training. This is highlighted further given that RL-based controllers are currently unable to generalise well across control tasks e.g. different changes of set point, meaning policy training is typically undertaken for each task (Beaulieu et al., 2020). As a result, implementation of RL control policies is computation and expertise expensive. To solve this problem, this work proposes a method to reduce the time and resource investment demanded by RL, through leverage of process data to learn from demonstration provided by an existing (but unknown) control policy. Then, the initialised RL is improved by learning from the real process over a short time period, thus outperforming the existing control policy. This two-step strategy has been recently deployed in domains including autonomous helicopter flight (Coates et al., 2009) and self-driving cars (Wu et al., 2020; Silver et al., 2010). To demonstrate this approach, Section 3.2 will introduce the preliminaries and motivation, Section 3.3 will outline the methodology, with Section 3.4 exhibiting different case studies.

3.2 Preliminaries

3.2.1 Policy gradients and Reinforce

Policy gradient methods directly learn a policy. Through the use of artificial neural networks as parameterisation, the policy may be deployed naturally in either discrete or continuous action spaces through appropriate network construction (Sutton and Barto, 2018a). Policy gradient methods do not explicitly learn the value of the policy. Instead, under the policy gradient theorem, acting with respect to the policy and gaining experience of the process dynamics provides approximation of the direction in which value increases fastest in parameter space. Hence, learning proceeds through gradient ascent to update the parameters of the policy to ensure control policies of high value (or low cost) are more probable(Sutton et al., 1999).

One policy gradient algorithm, Reinforce with baseline, approximates the direction in which the policy observes increased performance through Monte Carlo realisations of the process dynamics under the current policy parameterisation. This algorithm has several advantages such as convergence to locally optimal solutions in policy (Zhang et al., 2020d) and efficient exploration of the decision space without requirement for a bandit strategy or further optimisation routine for action selection – as is the case in many pure action-value methods(Simmons-Edler et al., 2019). Demonstration of the method is also available(Petsagkourakis et al., 2020a). Therefore, it is used in this work to learn an RL parameterisation of an existing control policy from process data. Despite use of the Reinforce with baseline algorithm in this work, other RL methods capable of operating in continuous control and state spaces (i.e. that identify a policy function approximation for a continuous control space) could be implemented. These include entropy regularised policy optimisation methods (Schulman et al., 2018a), trust region policy optimisation (TRPO) (Schulman et al., 2017a), and proximal policy optimisation (PPO) methods (Schulman et al., 2017b).

3.2.2 Learning from demonstrations via apprenticeship

Learning from demonstrations encompasses an increasingly prevalent and established group of methods, which leverage data generated from an existing but unknown control policy to aid learning-based control systems. This concept is generally termed as apprenticeship learning (AL). AL has been adopted in a number of complex control domains (Coates et al., 2009; Silver et al., 2010), but to our knowledge, this work is the first to propose use of the method to leverage plant data directly, and this is one of the primary contributions of this work. The concepts of AL are expressed in three main subfields including behavioural cloning (i.e. supervised learning), inverse optimal control, and inverse reinforcement learning (IRL).

This study exploited IRL built upon the framework provided by MDPs (Coates et al., 2009; Silver et al., 2010). MDPs express process objectives mathematically as a reward function. The reward function provides a scalar feedback signal indicative of the optimality of process evolution. IRL is concerned with the task of mathematically abstracting the reward function given process knowledge and demonstrations from an existing control policy. The IRL problem is formalised as: given observations of an

existing policy over time, sensory inputs available for determination of the originally demonstrated control law and a model of the process; determine the reward function that can mostly justify the demonstrated behaviour (Silver et al., 2010; Abbeel and Ng, 2004; Ziebart et al., 2008). IRL proceeds on the assumption that demonstrated control action is noisily optimal under the reward function derived (Ziebart et al., 2008; Wulfmeier et al., 2016). However, it should be noted that this does not necessarily imply that the policy is optimal in view of the true objectives for process control and optimisation.

As such, IRL leverages process data to learn a reward function, which encodes the control objectives of an existing scheme into a feedback signal. A control policy that maximises the utility of this reward function within the MDP framework, provides a parameterisation of the existing control scheme. Hence the pairing of IRL with RL as an MDP solver, allows for synchronously learning the parameterisation of an existing but unknown control policy as described in process data. The generated reward function can be used to compare against the process objective (if known) and suggest if the extracted control policy is suitable for online learning. Moreover, manual modifications are always implemented during process control even if the process objective is known. These manual modifications cannot be quantified by human operators, but can be retrieved from historical data by IRL. Therefore, using IRL to generate a reward function is advantageous for parameterisation of the optimal control policy.

3.2.3 Motivation

In the following work, we demonstrate a framework for learning and optimisation of chemical processes. The framework consists of two steps: offline learning, and online learning and improvement. Here, the use of terminology is converse to that common in the machine learning community. In this work, offline learning indicates a process of AL (via IRL) to infer control objectives from process data and the learning of a corresponding parameterisation of the control policy described by data; online improvement then indicates the transfer of the learned parameterisation to the real system for the purpose of further policy improvement under the true process objective. The framework enables the learning of an RL-based control policy, by leveraging process data from existing control schemes (offline) and subsequently improves the learned policy

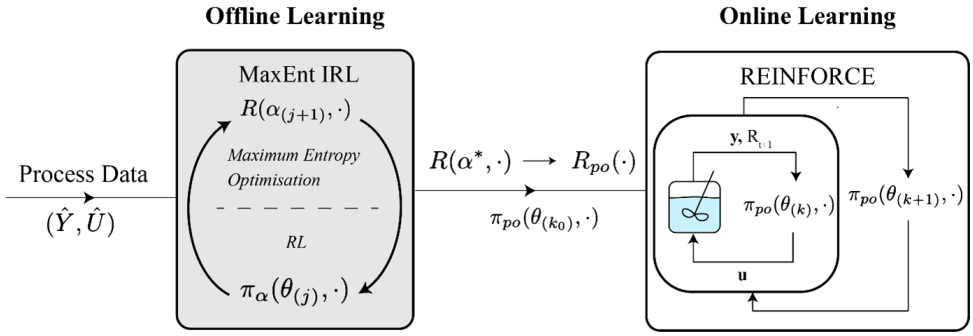


Figure 3.2: The offline–online framework proposed for the learning and optimisation of processes. Offline learning utilises process data, $(\hat{\mathbf{Y}}, \hat{\mathbf{U}})$, to learn a reward function, $R(\boldsymbol{\alpha}^*, \cdot)$, and a parameterisation of the demonstrated policy, $\pi_{po}(\theta_{(k_0)}, \cdot)$. Online learning utilises the learned parameterisation as initialisation for further policy optimisation under a reward function, $R_{po}(\cdot)$, descriptive of the true process objective.

parameterisation via further RL (online). The automation of offline learning and the policy tuning process that is associated, provides a significant contribution given the technical, computational and data demands of RL-based policy learning.

Offline learning produces a parameterisation of the existing control policy, which could be deployed directly for control. The parameterisation will achieve similar performance to that expressed by the original control scheme. If necessary, the parameterisation may then be transferred to the second stage of online learning for further policy improvement. It should be emphasised that the leveraging of process data is significant given the practical difficulties in learning an RL based policy 'from scratch' (Petsagkourakis et al., 2020a; Karg et al., 2019). The framework also lends itself to the improvement and recalibration of the control scheme temporally. Fig. 3.2 provides further description of the framework proposed.

3.3 Methodology

3.3.1 Problem statement

The following work proceeds on the formulation of the underlying problem of process control as a Markov decision process (MDP). The true dynamics of an MDP are

described as follows

$$\mathbf{x}_{t+1} \sim p\left(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t\right) \tag{3.1}$$

$$\mathbf{y}_{t+1} \sim p\left(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}\right) \tag{3.2}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is a vector of continuous variables representative of the true system state, $\mathbf{u} \in \mathbb{R}^{n_u}$, the manipulated variables (MVs), $\mathbf{y} \in \mathbb{R}^{n_y}$, the observed control variables (CVs) and t is indicative of the discrete time index (Rohani, 2017). The process evolution between discrete time indices t and $t+1$ is governed by the conditional density function, $p\left(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t\right)$. Similarly, the observation, $\mathbf{y}_t$, of the true state of the system, $\mathbf{x}_t$, is governed by the conditional density, $p\left(\mathbf{y}_t|\,\mathbf{x}_t\right)$. To facilitate learning of a policy prior to transfer to the real system, approximation of the true dynamics proceeds via construction of state space models and assumptions regarding process stochasticity, hence:

$$\mathbf{x}_{t+1} \;= f\left(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t\right) \tag{3.3}$$

$$\mathbf{y}_{t+1} = g(\mathbf{x}_{t+1}) \tag{3.4}$$

where $f : \mathbb{R}^{n_x \times n_u \times n_s} \to \mathbb{R}^{n_x}$ is representative of the process dynamics and $\mathbf{s}_t \in \mathbb{R}^{n_s}$ is representative of the process disturbance. The mapping $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ is the state observation associated with measurement noise(Heess et al., 2015).

The following work deploys RL to learn a control policy from process data. The objective of RL is to minimise the expected cost of a dynamic process (or equivalently to maximise its value). A process trajectory, $\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{y}_0, \mathbf{u}_0, \ldots, \mathbf{u}_{T-1}, \mathbf{x}_T, \mathbf{y}_T)$, describes the manner in which a process evolves over a given discrete time horizon of length T. The cost or value, $G\left(\boldsymbol{\tau}\right)$, of the process trajectory over a finite horizon is denoted:

$$G\left(\boldsymbol{\tau}\right) = \sum_{t=1}^{T} \gamma^{t-1} R_t \tag{3.5}$$

where $\gamma \in (0, 1]$ is a discount factor, which provides a net present value interpretation of future value; and R_t is the reward assigned to the process' evolution between time indices $t-1$ and t. However, in view of process stochasticity, the probability of observing $\boldsymbol{\tau}$ adheres to a conditional density $p(\boldsymbol{\tau}|\theta)$ based on the control policy and process dynamics:

$$p\left(\boldsymbol{\tau}|\theta\right) = \bar{\rho}\left(\mathbf{x}_0\right) p\left(\mathbf{y}_0|\,\mathbf{x}_0\right) \prod_{t=0}^{T-1} \pi\left(\mathbf{u}_t|\mathbf{y}_t, \theta\right) p\left(\mathbf{x}_{t+1}|\mathbf{x}_t,\ \mathbf{u}_t\right) p\left(\mathbf{y}_{t+1}|\,\mathbf{x}_{t+1}\right) \tag{3.6}$$

where $\bar{\rho}(\mathbf{x}_0)$ is the probability density of the initial system state; $\pi(\mathbf{u}_t|\mathbf{y}_t;\theta,\cdot)$ is the conditional density function descriptive of the learned policy, which is parameterised by $\theta \in \mathbb{R}^{n_\theta}$; and $p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)$ is the conditional density function representative of the process dynamics.

Note that the definition of a policy as a conditional density function implies it is stochastic. This is important in the scope of the learning process associated with RL but does not necessarily assert the use of a stochastic policy upon deployment for control of the real system (only the mode might be used in practice). The objective of the RL problem and learning process is to find a policy $\pi(\cdot,\theta^*)$ that maximises the objective $J(\boldsymbol{\tau})$, such that:

$$\pi(\theta^*,\cdot) = \underset{\pi(\theta,\cdot)}{\arg\min} -J(\boldsymbol{\tau}) \tag{3.7}$$

$$J(\boldsymbol{\tau}) = \int p(\boldsymbol{\tau}|\theta)\, G(\boldsymbol{\tau})\ d\boldsymbol{\tau} \tag{3.8}$$

Eq. 3.7 describes the probability-weighted average of trajectory value and hence reformulation may utilise equivalence of $J(\boldsymbol{\tau})$ as the expectation of trajectory value under the policy parameters θ, such that

$$J(\boldsymbol{\tau}) = \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\theta)}\left[G(\boldsymbol{\tau})\right] \tag{3.9}$$

The description provided in this section formalises the problem of optimal control under the framework provided by MDPs. One approach to finding approximate solution to the problem described by Eqs. 3.7-3.9 is encompassed by policy optimisation RL methods.

3.3.2 Policy gradients and Reinforce

Policy gradient methods are a subset of policy optimisation methods, which estimate the gradient of the objective detailed by Eq. 3.7, with respect to the parameters of the current policy. Mathematically, this is described by the policy gradient theorem (Sutton et al., 1999). The Appendix B.1 provides full derivation and explanation of the policy gradient theorem. Given an estimate of the true policy gradient, gradient ascent methods facilitate policy improvement to make trajectories of higher reward more probable. In this manner, the policy parameterisation is updated in the direction

provided by the policy gradient, which is described by Eq. 3.10:

$$\nabla_{\theta_{(j)}} J(\boldsymbol{\tau}) = \nabla_{\theta} \int p(\boldsymbol{\tau}|\theta)\, G(\boldsymbol{\tau})\, d\boldsymbol{\tau} \tag{3.10}$$

$$= \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\theta)}[G(\boldsymbol{\tau})\, \nabla_{\theta} \log p(\boldsymbol{\tau}|\theta)] \tag{3.11}$$

$$\theta_{(j+1)} = \theta_{(j)} + \omega \nabla_{\theta_{(j)}} J(\boldsymbol{\tau}) \tag{3.12}$$

where j is the iteration of policy optimisation and $\omega \in \mathbb{R}_+$ is the step size in the direction of the policy gradient $\nabla_{\theta_{(j)}} J(\boldsymbol{\tau})$. The derivation of Eq. 3.10, leverages the use of a logarithmic identity (see Appendix B.1). This enables mathematical separation of the conditional probability functions descriptive of the process dynamics and policy (see Eq. 3.6). Given the process dynamics are independent of the parameterisation θ of the policy $\pi(\theta, \cdot)$, examination of Eq. 3.6 provides:

$$\nabla_{\theta(j)} \log p(\boldsymbol{\tau}|\theta) = \sum_{t=0}^{T-1} \nabla_{\theta(j)} \log \pi\left(\mathbf{u}_t|\mathbf{y}_t, \theta_{(j)}\right) \tag{3.13}$$

Consequently, the policy gradient described by Eq. 3.10 is reformulated as:

$$\nabla_{\theta(j)} J(\boldsymbol{\tau}) = \mathbb{E}_{\boldsymbol{\tau}}\left[G(\boldsymbol{\tau}) \sum_{t=0}^{T-1} \nabla_{\theta(j)} \log \pi\left(\mathbf{u}_t|\mathbf{y}_t,\ \theta_{(j)}\right)\right] \tag{3.14}$$

Exact computation of the true policy gradient requires full knowledge of the conditional density functions descriptive of process dynamics. Given such knowledge of the process dynamics are unavailable, the policy gradient is approximated by directly sampling the process under the current policy parameterisation over a given time horizon via a Monte Carlo method (Petsagkourakis et al., 2020b). This is encapsulated by the Reinforce with baseline algorithm, which is detailed by Algorithm 3.1.

Algorithm 3.1 Reinforce with Baseline

Input: A policy π with initial parameters θ_0; learning rate ω; episode length T; K episodes for Monte Carlo rollouts of the policy; and, N training epochs. Early stopping conditions may also be implemented.

for $j = 1, \ldots, N$ **do**

a. Perform Monte Carlo realizations of the policy for T timesteps and K trajectories. Store all state action pairs observed $\left(\mathbf{u}_t^k,\ \mathbf{y}_t^k\right)$, as well as the total return from the episode G_t^k (see Eq. 3.5 and Algorithm B.1)

b. Estimate the policy gradient and update the parameters of the policy such that:

$\theta_{(j+1)} = \theta_{(j)} + \omega_{(j)}\ \frac{1}{K}\sum_{k=1}^{K}\left[\left(G^k - b\right)\nabla_\theta \sum_{t=0}^{T-1} \ln \pi\left(\mathbf{u}_t^k|\mathbf{y}_t^k,\ \theta_{(j)}\right)\right]$, where $b = \frac{1}{K}\sum_{k=1}^{K} G^k$

end for

Output: An approximately optimal policy $\pi(\mathbf{u}|\mathbf{y},\ \theta^*)$

Through utilisation of the Monte Carlo method, an unbiased approximation of the true policy gradient is obtained. However, due to the stochastic nature of both the policy and process dynamics, the gradient may observe high variance. In order to reduce the variance of approximation, a baseline b is introduced (Petsagkourakis et al., 2020b). This baseline is formulated directly as the expectation of cost associated with the realisations of the policy. In this manner, the update balances the cost of an action against the expected cost from the current policy.

It is of important note that the parameterisation of the policy must be continuously differentiable as prescribed by the policy gradient theorem. Naturally, this lends to application of artificial neural networks (ANN) for function approximation in this work. Specifically, a recurrent long-short term memory (LSTM) neural network was used for parameterisation of the control policy. Recurrent LSTM neural networks have demonstrated utility in dynamic stochastic control problems with extension to systems characterised by partial observability (Heess et al., 2015). General detail of the mathematical operations specific to LSTMs can be found in the following works (Hochreiter and Schmidhuber, 1997; Colah, 2015), with figurative description of the network used in this application provided by Appendix B.2. The investigation utilised

the Pytorch 1.3.1 framework and first order gradient ascent method Adam to train the LSTM network proposed. The network structure was composed of two hidden layers, each with 20 LSTM cells. A leaky rectified linear unit (ReLU) activation function was applied across both hidden layers and a ReLU6 activation function was applied across the output layer, naturally bounding the output prediction. For a random variable z, the ReLU6 transformation is described as:

$$\text{ReLU6}\left(z\right) = \min\left(\max\left(0, z\right), 6\right)$$

The network designed in the context of this work, predicts the mean (μ_t) and standard deviation (σ_t) of a unimodal multivariate normal distribution. This distribution describes the conditional density function representative of the control policy, such that: $\mathbf{u}_t \sim \pi\left(\mathbf{u}_t|\mathbf{y}_t, \mathbf{H}_t,\ \theta\right) = \mathcal{N}\left(\mu_t, \sigma_t^2\right)$, where $\mathbf{H}_t$ is a learned parameterisation of the history of process states provided by the LSTM cells, and σ_t^2 is the variance. Here, we formally construct the control policy as stochastic. However, upon deployment of the policy to the real system, the policy may be assumed deterministic through selection of the actions corresponding to the mode (equivalently, the mean) of the multivariate normal distribution, such that $\mathbf{u}_t = \mu_t$.

In this section, we have presented an approach to solving the MDP characteristic of a control problem through use of the policy gradient method, Reinforce with baseline, in combination with an LSTM network for parameterisation of the learned policy. In the following, we introduce an approach to policy learning, namely maximum entropy inverse reinforcement learning (MaxEnt IRL), which utilises existing process data to learn from demonstration. Conceptually, this approach is commonly known as apprenticeship learning (AL).

3.3.3 Apprenticeship learning via inverse reinforcement learning

Apprenticeship learning (AL) via inverse reinforcement learning (IRL) is a general approach to policy learning from demonstration (i.e. process data). The benefits to such an approach are two-fold. Firstly, AL via IRL provides a parameterisation of the existing control policy expressed in the process data. Secondly, it facilitates RL-based policy learning under the 'real' process objective as it provides an initial policy to

hot-start the RL procedure. Otherwise, initially, the agent (or controller) will explore the control action space randomly, which results in a data hungry and time-consuming approach. These benefits are exploited by the framework proposed in Section 3.2.3 as detailed by Fig. 3.2.

The foundational IRL algorithms construct the reward function $R : \mathbb{Y} \rightarrow \mathbb{R}$ as a linear combination of state features representative of the system state $\varphi \in \mathbb{R}^{d\times 1}$, such that:

$$R = \alpha_1\varphi_1 + \alpha_2\varphi_2 + \ldots + \alpha_d\varphi_d \tag{3.15}$$

where $\alpha_i \in \mathbb{R}$ are feature weightings and $\varphi_i : \mathbb{Y} \rightarrow \mathbb{R}$ explicitly represent the system state ($\mathbf{y}$), but also implicitly encode control objectives. Typically, φ are hand designed based on process and control task knowledge(Abbeel and Ng, 2004). Knowledge of process objectives can also be applied to place bounds on the weights α in the reward function, however, this may not always be desired as one could assert technical bias on the problem and reduce the feasible region. From this definition of the reward function $R(\boldsymbol{\alpha}, \mathbf{y})$, consequent reformulation of the policy optimisation objective $J(\boldsymbol{\tau})$ in Eq. 3.9 yields

$$J(\boldsymbol{\tau}) = \mathbb{E}_{\tau\sim p(\boldsymbol{\tau}|\theta)}\left[\sum_{t=1}^{T}\gamma^{t-1}R(\alpha, \mathbf{y}_t)\right] \tag{3.16}$$

$$J(\boldsymbol{\tau}) = \sum_{i=1}^{d}\alpha_i\mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\theta)}\left[\sum_{t=1}^{T}\gamma^{t-1}\varphi_i(\mathbf{y}_t)\right] \tag{3.17}$$

This may be further decomposed through definition of trajectory features υ_i, such that for the discounted case:

$$\upsilon_i^{\gamma} = \sum_{t=1}^{T}\gamma^{t-1}\varphi_i(\mathbf{y}_t) \tag{3.18}$$

$$J(\boldsymbol{\tau}) = \sum_{i=1}^{d}\alpha_i\mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\theta)}[\upsilon_i^{\gamma}] \tag{3.19}$$

$$= \boldsymbol{\alpha}^T\mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\theta)}[\boldsymbol{\upsilon}^{\gamma}] \tag{3.20}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{d\times 1}$ and $\boldsymbol{\upsilon}^{\gamma} \in \mathbb{R}^{d\times 1}$. Equivalently, undiscounted trajectory features $\boldsymbol{\upsilon}$ may be recovered by setting $\gamma = 1$. The characterisation of a policy and process trajectory in terms of $\boldsymbol{\upsilon}$ enables RL to learn from multiple, distributed trajectories

and reduces the problem to learning feature weights $\boldsymbol{\alpha}^*$(Abbeel and Ng, 2004; Ziebart et al., 2008). Conceivably, a number of different reward functions exist that recover the desired behaviour. The current study uses the MaxEnt IRL framework proposed by Ziebart et al. (2008); Ziebart (2010), which proceeds in identification of $\boldsymbol{\alpha}$ via a probabilistic approach as underpinned by the principle of maximum entropy.

3.3.4 Maximum entropy inverse reinforcement learning (MaxEnt IRL)

In AL we are interested in learning a policy as described by a conditional probability density function $\pi(\mathbf{u}_t|\mathbf{y}_t, \cdot)$, such that upon deployment of the policy to the real system, the process observes the same evolution as that described by process data (see Eq 3.6). Explicitly, the investigation learns the expert's policy expressed by process trajectories $\mathbf{T} = [\boldsymbol{\tau}_1^E,\ ..,\ \boldsymbol{\tau}_K^E]$ as characterised by trajectory features, $\left\{\boldsymbol{\upsilon}_k^E\right\}_{k=1,..,K}$. MaxEnt IRL (Ziebart et al., 2008) is an established method and poses solution to the problem of learning such an approximate policy. It learns a reward function that maximises the likelihood of observing the demonstrated trajectories, $\mathbf{T}$, given an accurate model of the process dynamics. Further discussion is provided in Appendix B.3. It follows that the log-probability of observing a given trajectory $\boldsymbol{\tau}$ is proportional to the cumulative undiscounted reward observed between a start and terminal state (Ziebart, 2010), such that:

$$p\left(\boldsymbol{\tau}|\boldsymbol{\alpha}\right) = \frac{\exp\left\{\boldsymbol{\alpha}^T\boldsymbol{\upsilon}(\boldsymbol{\tau})\right\}}{Z(\boldsymbol{\alpha}, \cdot)} \tag{3.21}$$

where $\boldsymbol{\upsilon} = [\upsilon_1, \ldots, \upsilon_d]$, and $Z(\boldsymbol{\alpha}, \cdot) = \sum_{\boldsymbol{\tau}} \exp\left\{\boldsymbol{\alpha}^T\boldsymbol{\upsilon}(\boldsymbol{\tau})\right\}$ is the partition function, which enforces normalisation of the distribution. Formally, the approach prescribes that each of the demonstrations, $\boldsymbol{\tau}^E \in \mathbf{T}$, are independently and identically distributed such that the likelihood of observing the set of trajectories, $\mathbf{T}$, expressed in process data is:

$$p(\mathbf{T}|\boldsymbol{\alpha}) = \prod_{k=1}^{K} p\left(\boldsymbol{\tau}_k^E\middle|\boldsymbol{\alpha}\right) = \prod_{k=1}^{K} \frac{1}{Z\left(\boldsymbol{\alpha},\ \cdot\right)} \exp\left\{\boldsymbol{\alpha}^T\boldsymbol{\upsilon}_k^E\right\} \tag{3.22}$$

where $Z\left(\boldsymbol{\alpha}, \cdot\right)$ is assumed constant for all $\boldsymbol{\tau}^E \in \mathbf{T}$ (Ziebart et al., 2008); and $p(\mathbf{T} \mid \boldsymbol{\alpha})$ is the likelihood of observing the set of demonstrations. Under the maximum entropy formulation (Ziebart et al., 2008; Wulfmeier et al., 2016; Ziebart, 2010),

optimal solution of the feature weights $\boldsymbol{\alpha}^*$ is:

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} p(\mathbf{T}|\boldsymbol{\alpha}) = \arg\max_{\boldsymbol{\alpha}} \prod_{k=1}^{K} p\left(\boldsymbol{\tau}_k^E|\boldsymbol{\alpha}\right) \tag{3.23}$$

The gradient of the log-likelihood objective (Eq. 3.23) with respect to feature weights $\boldsymbol{\alpha}$ is formulated as:

$$\nabla_{\boldsymbol{\alpha}_{(i)}} \sum_{k=1}^{K} \log p(\boldsymbol{\tau}_k^E|\boldsymbol{\alpha}_{(\mathbf{i})}) \ = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\upsilon}_k^E - \nabla_{\boldsymbol{\alpha}_{(i)}} \log Z(\boldsymbol{\alpha}_{(i)},\ \cdot) \tag{3.24}$$

$$\nabla_{\boldsymbol{\alpha}_{(i)}} \log Z(\boldsymbol{\alpha}_{(i)},\ \cdot) = \mathbb{E}_{\boldsymbol{\tau}^\pi \sim p(\boldsymbol{\tau}^\pi|\boldsymbol{\alpha}_{(\mathbf{i})},\ \theta^*)}\ [\boldsymbol{\upsilon}^\pi] \tag{3.25}$$

where $\nabla_{\boldsymbol{\alpha}_{(\mathbf{i})}} \log Z(\boldsymbol{\alpha}_{(i)}, \cdot)$ is estimated via policy optimisation in the underlying MDP to find a policy $\pi(\cdot, \theta^*)$**,** which maximises the following modified objective and then subsequently performing Monte Carlo realisations of the solution policy under the process dynamics to provide sample trajectories $\xi = [\boldsymbol{\tau}_1^\pi, \ldots, \boldsymbol{\tau}_N^\pi]$ characterised by $\{\boldsymbol{\upsilon}_n^\pi\}$, where $n = 1,\ ..., N$. This is also discussed further in Appendix B.3. Eqs. 3.24 suggest that the MaxEnt IRL problem finds a weight vector $\boldsymbol{\alpha}^*$, which minimises the differences between the expected trajectory features of the learned policy and that which is demonstrated. Gradient based optimisation methods may be deployed to find solution $\boldsymbol{\alpha}^*$ by stepping parameter values $\boldsymbol{\alpha}$ in the direction of the gradient (Ziebart et al., 2008; Ziebart, 2010). This work utilises the first order gradient ascent method (Eq. 3.26):

$$\boldsymbol{\alpha}_{(i+1)} = \boldsymbol{\alpha}_{(i)} + \kappa \nabla_{\boldsymbol{\alpha}_{(i)}} \log p\left(\mathbf{T} \mid \boldsymbol{\alpha}_{(i)}\right) \tag{3.26}$$

where κ is a learning rate. The problem formulated here constitutes a bi-level optimisation, with the upper level task approached by MaxEnt IRL and the lower level task handled by the policy gradient method Reinforce. In each iteration i of the upper MaxEnt IRL problem a new reward function $R(\boldsymbol{\alpha}_{(i)},\ \cdot)$ is abstracted. The underlying MDP is subsequently solved by policy optimisation and estimation of the partition function and $\mathbb{E}[\boldsymbol{\upsilon}^\pi]$ provided. The Reinforce method and the approach to solving the lower level optimisation task is detailed by Algorithm 3.1. It should be noted that the approaches to policy optimisation provided by PPO and entropy regularisation could provide further stability in learning and accuracy in estimation of the partition function, respectively. In view of the length of the horizon specific to many control tasks,

discounted trajectory features $\boldsymbol{\upsilon}^{\gamma}$, as described by Eq. 3.18, should be used rather than the undiscounted features. This establishes the upper MaxEnt IRL task as a nonconvex optimisation (Zhou et al., 2018) but provides performance improvements in the lower level policy optimisation task. Algorithm 3.2 details the MaxEnt IRL algorithm further.

Algorithm 3.2 MaxEnt Inverse Reinforcement Learning

Input: A policy $\pi^{A}_{(0)}$ with initial parameters $\theta_{(0)}$; a weight vector $\boldsymbol{\alpha}$; state feature functions $\boldsymbol{\varphi}(\mathbf{x})$; trajectory features representative of the demonstrated trajectories $\boldsymbol{\upsilon}^{E}$; maximum iterations $N_{\max}$; learning rate κ;

for $n = 1, \ldots, N_{\max}$ **do**

1. Perform policy optimisation of $\pi^{A}_{(n-1)}$ under the current reward function $R(\boldsymbol{\alpha}_{(n)})$ via Algorithm 3.1. Return $\pi^{A}_{(n)}$ as solution to the MDP defined.

2. Perform Monte Carlo simulation of $\pi^{A}_{(n)}$ (via Algorithm B.1) to evaluate the policy. Return the trajectory features characteristic of the expected process evolution under the policy $\mathbb{E}\left[\boldsymbol{\upsilon}^{\pi_{(n)}}\right]$

3. Approximate the gradient of the likelihood of observing the demonstrated trajectories with respect to the weights:

$$\nabla_{\alpha} \log p\left(\mathbf{T} \mid \boldsymbol{\alpha}\right) = \tfrac{1}{K} \textstyle\sum_{k=1}^{K} \boldsymbol{\upsilon}_{k}^{E} - \mathbb{E}\left[\boldsymbol{\upsilon}^{\pi_{(n)}}\right]$$

4. Perform gradient ascent such that: $\boldsymbol{\alpha}_{(n+1)} = \boldsymbol{\alpha}_{(n)} + \kappa \nabla_{\alpha} \log p\left(\mathbf{T} \mid \boldsymbol{\alpha}\right)$

end for

Output: Optimal weights $\boldsymbol{\alpha}^{*}$ and parameterisation of the demonstrated policy $\pi_{\mathrm{po}}(\boldsymbol{\theta}_{(k_0)}, \cdot)$ for further policy improvement in online learning.

3.3.5 Overview of proposed methodology

The methodology proposed leverages the large amount of process control data available to industry to learn an RL-based parameterisation of a previously implemented control scheme through AL via IRL. This parameterisation should express the existing control law as well as the process knowledge of operators provided the available data is sufficiently rich. Once a parameterisation is constructed offline, it is deployed as initialisation for further RL-based policy improvement (online). This online learning proceeds under a reward function descriptive of the real process objectives. Through

this approach, we significantly reduce the computational and technical investment associated with training an RL-based control policy. Specifically, the improvements noted are drawn from the offline section of the framework. Here, we combine simulation with the use of IRL to automate analysis of historical process data. This enables us to directly abstract a reward function, which provides clear preference (discrimination) over controls from: (i) knowledge of the process control task we are concerned with (represented by the basis features $\boldsymbol{\varphi}$ in the reward function); and (ii) empirical observations of the system and its behaviour in response to controls (by optimizing the feature weight $\boldsymbol{\alpha}$). Learning under this reward function provides a parameterisation of the existing control scheme expressed in process data. Section 3.4 presents a number of computational case studies for empirical demonstration of the framework described.

3.4 Computational case studies

3.4.1 Introduction to the case studies

The optimisation objective of the following studies is set point tracking in a multiple-input, multiple output (MIMO) control scheme. Specifically, the process is a non-isothermal continuous stirred tank reactor (CSTR) under operation of an endothermic isomerism reaction of the form $A \rightarrow B$. The reaction rate temperature dependence is described by the Arrhenius kinetics. Demonstration is provided in the form of process data generated by the action of a PID control scheme, produced via a discrete time Python 3.7.3 implementation. The controlled variables ($\mathbf{y}$) are concentration of reagent C_A^{obs} and temperature of the reactor T^{obs}. The manipulated variables ($\mathbf{u}$) are the temperature of a heating jacket T_E and concentration of the reagent in the input stream C_{A0}. Bounds are placed upon the absolute values of the action space. Definition of process variables follows:

$$\mathbf{y} = \left[C_A^{\text{obs}},\ T^{\text{obs}}\right]^T$$

$$\mathbf{x} = [C_A,\ T]^T \qquad \mathbf{u} = [C_{A0},\ T_E]^T$$

In the case studies presented, the process model is of deviation variable form and was derived from first principles. The deviation variable z^* of random variable z is

expressed as:

$$z^* = z - z_{\text{ss}}$$

where z_{ss} is the previous steady state value of z. Process stochasticity (disturbance) is assumed zero mean Gaussian, as is the nature of system observation. Therefore, approximation of the true underlying process dynamics takes the form of a system of stochastic differential equations (SDE), such that:

$$\mathbf{x}^*_{t+1} = \mathbf{x}^*_t + h\left(\mathbf{x}^*_t,\, \mathbf{u}^*_t\right) d\text{t} + \delta\left(\mathbf{x}^*_t\right) dW_t \tag{3.27}$$

$$\mathbf{y}^*_{t+1} = g(\mathbf{x}^*_{t+\mathbf{1}}) \tag{3.28}$$

where the function $h\left(\cdot\right)$ is descriptive of the underlying process dynamics; $\delta\left(\cdot\right)$ the magnitude of disturbance, as described by the Wiener process, W_t, (Mao, 2015); and $g(\cdot)$ the nature of system observation. In the following studies:

$$h\left(\mathbf{x}^*_t,\, \mathbf{u}^*_t\right) = \begin{bmatrix} -3.997 & -0.446 \\ -6.092 & -1.581 \end{bmatrix} \mathbf{x}^*_t + \begin{bmatrix} 0.500 & 0 \\ 0 & 0.305 \end{bmatrix} \mathbf{u}^*_t \tag{3.29}$$

$$\delta\left(\mathbf{x}^*_t\right) = \begin{bmatrix} 0.500 & 0 \\ 0 & 0.300 \end{bmatrix} \mathbf{x}^*_t \tag{3.30}$$

$$g\left(\mathbf{x}^*_{t+\mathbf{1}}\right) = \begin{bmatrix} 1 + \mathcal{N}\left(0,\ 0.025\right) & 0 \\ 0 & 1 + \mathcal{N}\left(0,\ 0.025\right) \end{bmatrix} \mathbf{x}^*_{t+\mathbf{1}} \tag{3.31}$$

and the Euler Maryuama method was utilised for system integration (Mao, 2015). The Appendix B.5 provides formal derivation and parameter values. Given the formulation of the MIMO problem, the investigation is concerned with controlling the evolution of error ε within both the temperature T^{obs} and reagent concentration C_A^{obs} control loops.

3.4.2 Design of state features for apprenticeship learning

The introduction provided in Section 3.3.4 outlines a framework for learning the weight vector $\boldsymbol{\alpha}^*$, which provides a linear mapping from state representations $\boldsymbol{\varphi}$ to scalar cost. Further, for a given representation, a set of possible process trajectories exist, which match the counts of state features (trajectory features) of the existing policy.

Therefore, design of $\boldsymbol{\varphi}$ should consider both the process, optimisation objectives and restriction of the possible set of trajectories. As a result, this work proposes the use of three types of state features, all of which provide consistent control objectives temporally and utilise knowledge of the underlying process control task.

Type I

The first state feature proposed is encapsulated by the radial basis function (RBF). The RBF provides a similarity measure and allocates exponentially lower cost or greater value for those control policies which achieve set point tracking. The feature is formulated as:

$$\widehat{\varepsilon} = \frac{y_{\text{sp}} - y}{y_{\text{sp}} - y_{\text{ss}}} \tag{3.32}$$

$$\varphi_I\left(\widehat{\varepsilon}\right) = \exp^{-(\beta\widehat{\varepsilon})^2} \tag{3.33}$$

where y_{ss} is the previous observed steady state of the system, y_{sp} is the desired set point, β is the shape parameter and $\varphi_I\left(\widehat{\varepsilon}\right) = [0, 1]$. The closer the value of β to zero, the greater the offset tolerated and the denser the reward landscape. Conversely, higher values of β provide exponentially greater rewards for trajectories closer to the set point, but a sparser reward landscape. In the following case studies, the investigation utilised $\beta = 10$.

Type II

Although the Type I feature is an absolute measure of control performance, alone it does not fully characterise the evolution of system response. Furthermore, the set of possible process trajectories, which could match the representation of the demonstrated policy $\boldsymbol{v}^E$ is large. To restrict the possible set, Type II and III features take inspiration from the PID control law, which at a given time is a linear combination of the error $\varepsilon = y_{\text{sp}} - y$ in the control loop at the current time point (proportional), the manner in which the error has evolved over time (integral) and the projected evolution of error in the future (derivative). Hence, the Type II state feature proposed intends to quantify how the absolute error in a control loop evolves temporally. As such, Type

II state features are described as:

$$\varphi_{\mathrm{II}}\left(\widehat{\varepsilon}\right) = \int_0^t |\widehat{\varepsilon}| d\mathrm{t} \approx \sum_{j=1}^{\mathrm{tc}} |\widehat{\varepsilon}| \Delta t \tag{3.34}$$

where Δt is equivalent to the sampling time or times at which control is provided (in this work the two are synonymous), $|\cdot|$ refers to the absolute value; j the discrete time index and tc the current time point. The absolute magnitude of the error provides clear control objective regardless of whether the error $\widehat{\varepsilon}$ is positive or negative in value. If this was not taken, actions that decrease error in the control loop may be penalised or rewarded in an RL setting depending upon whether the integral of the error becomes positive or negative as a result.

Type III

The design of Type III state features aim to quantify how the error in the control loop may evolve into the future. As a result, the feature approximates the derivative of the error in the control loop at the sampled time:

$$\varphi_{\mathrm{III}}\left(\widehat{\varepsilon}\right) = \frac{d|\widehat{\varepsilon}|}{d\mathrm{t}} \approx \frac{|\widehat{\varepsilon}_{\mathrm{tc}}| - |\widehat{\varepsilon}_{tc-1}|}{\Delta t} \tag{3.35}$$

where $tc-1$ is the previous discrete time index. In view of the proposed state features, the investigation is able to characterise control trajectories and provide direct and consistent control objective. As a result, the reward function R of the MDP described is specified as:

$$\begin{aligned} R = \alpha_1 \varphi_I\left(\widehat{\varepsilon}_{C_A^*}\right) + \alpha_2 \varphi_I\left(\widehat{\varepsilon}_{T^*}\right) + \alpha_3 \varphi_{\mathrm{II}}\left(\widehat{\varepsilon}_{C_A^*}\right) \\ + \alpha_4 \varphi_{\mathrm{II}}\left(\widehat{\varepsilon}_{T^*}\right) + \alpha_5 \varphi_{\mathrm{III}}\left(\widehat{\varepsilon}_{C_A^*}\right) + \alpha_6 \varphi_{\mathrm{III}}\left(\widehat{\varepsilon}_{T^*}\right) \end{aligned} \tag{3.36}$$

3.4.3 Case study definitions

Three case studies demonstrate the use of the framework in different contexts and control tasks. Table 3.1 details the specific experimental setup. Case study I demonstrates the framework proposed for deployment when subjectively near optimal control is provided by an existing control scheme. Case study II demonstrates the framework is still effective when the control demonstrated by an existing scheme is subjectively suboptimal. Case study III explores the potential to transfer knowledge within the framework in order to aid efficiency in learning on different control tasks.

Table 3.1: Conditions of design for the case studies detailed. The real initial state of the controlled variables $\mathbf{x_0}$ is drawn from the respective distributions. The set point y^*_{sp} details the new setpoint of the respective control variables as set at $t = 0$.

Case Study	System Parameter	Concentration ($\mathbf{C^*_A}$) Control Loop	Temperature ($\mathbf{T}^*$) Control Loop
I	Initial state distribution $\overline{\rho}(\mathbf{x}_0)$ Set point Y^*_{sp}	$\mathcal{N}(0,\ 0.25)$ -1	$\mathcal{N}(0,\ 0.75)$ 4
II	Initial state distribution $\overline{\rho}(\mathbf{x}_0)$ Set point Y^*_{sp}	$\mathcal{N}(0,\ 0.25)$ 1	$\mathcal{N}(0,\ 0.75)$ 4
III	Initial state distribution $\overline{\rho}(\mathbf{x}_0)$ Set point Y^*_{sp}	$\mathcal{N}(0,\ 0.25)$ -2.5	$\mathcal{N}(0,\ 0.75)$ 3

3.5 Results and discussion

3.5.1 Case study I – Learning from near optimal demonstrations

The purpose of this case study is to construct an RL controller which learns from demonstration provided by a near optimal control policy and then to improve it further. As such, we demonstrate the full utility of the offline-online framework proposed. Firstly, offline learning under MaxEnt IRL is deployed to find a linear combination $\boldsymbol{\alpha}^*$ of state features, which infers and encodes control objectives into a feedback signal or reward function. Under this reward function a parameterisation of the control policy expressed in process data is learned in order to match the demonstrated process behaviour as characterised through expected trajectory features. The learned parameterisation is then improved under the real process objective, which in this case is pure tracking. Here the demonstrated control policy is that of a well-tuned PID controller (PID1 as detailed by the Table B.3).

Results of apprenticeship learning via MaxEnt IRL

Utilising 500 Monte Carlo realisations of the PID1 policy, the methodology was able to generate an informative dataset and subsequently characterise the policy using the six basis features presented in Eq. 3.36, with $\gamma = 0.99$ and $T = 50$ indicates the length of the discrete-time finite horizon. The trajectory feature expectations of PID1 are outlined in Table 3.2.

Table 3.2: The expected discounted trajectory features of PID1 ($\boldsymbol{\upsilon}^{\gamma,E}$) and the policy learned through AL ($\boldsymbol{\upsilon}^{\gamma,\,\pi}$), and IRL's feature weight ($\boldsymbol{\alpha}^*$) generated in CS I. Y^*-Type indicates the type of trajectory feature and the respective control loop error.

	Trajectory Features					
	$C_A^* - I$	$T^* - I$	$C_A^* - II$	$T^* - II$	$C_A^* - III$	$T^* - III$
$\mathbb{E}\left[\boldsymbol{\upsilon}^{\gamma,E}\right]$	21.63	20.68	4.08	7.93	-22.87	-22.43
$\mathbb{E}[\boldsymbol{\upsilon}^{\gamma,\pi}]$	21.41	20.76	4.31	7.03	-22.28	-22.71
$\boldsymbol{\alpha}^*$	0.137	0.652	-0.067	-0.630	-0.194	-0.343

From Table 3.2, it is concluded that under the characterisation of the PID1 policy $\boldsymbol{\upsilon}^{\gamma,E}$, Algorithm 3.2 was able to learn an agent parameterisation of the demonstrated policy (i.e. PID controller). This was achieved after just four iterations of the algorithm. Each iteration is composed of solving an MDP via RL (detailed by Algorithm 3.1) and then updating the weight vector $\boldsymbol{\alpha}$ via Eq. 3.24. The hyperparameters for Algorithm 3.2 and each iteration are detailed by the Appendix B.7. It is worth reiterating that there is a set of possible policies, which observe the same expected trajectory feature counts $\mathbb{E}\left[\boldsymbol{\upsilon}^{\gamma,E}\right]$ as that of the demonstrated policy. In the context of this work, further restricting the possible set is not necessary, however, introduction of further state features $\boldsymbol{\varphi}$ would facilitate such. Given that $\boldsymbol{\varphi}$ compose the reward function and all express inherent set point tracking objectives, intuitively, any of the policies from the possible set, which match the trajectory features of the demonstrated policy should provide good initialisation for further policy improvement. The learned weight vector $\boldsymbol{\alpha}^*$ may also be interpreted and provide insight into the dynamics of the respective control loops.

The state features which are specific to the temperature control loop receive a greater weight than the concentration control loop. This is likely reflective of the endothermic nature of reaction and the relative changes of set point in the temperature loop and concentration loop. Compared to changing reactant concentration, an increase in reactor temperature T will likely shift reaction equilibrium more significantly in a manner to increase consumption of reagent. As a result, the system dynamics act in a way to aid the set point change in the concentration control loop. Hence, greater weighting is allocated to control of the temperature control loop.

In this section, we show the utility of the offline learning method proposed in the context of learning by demonstration (or AL). Subsequently, we demonstrate how

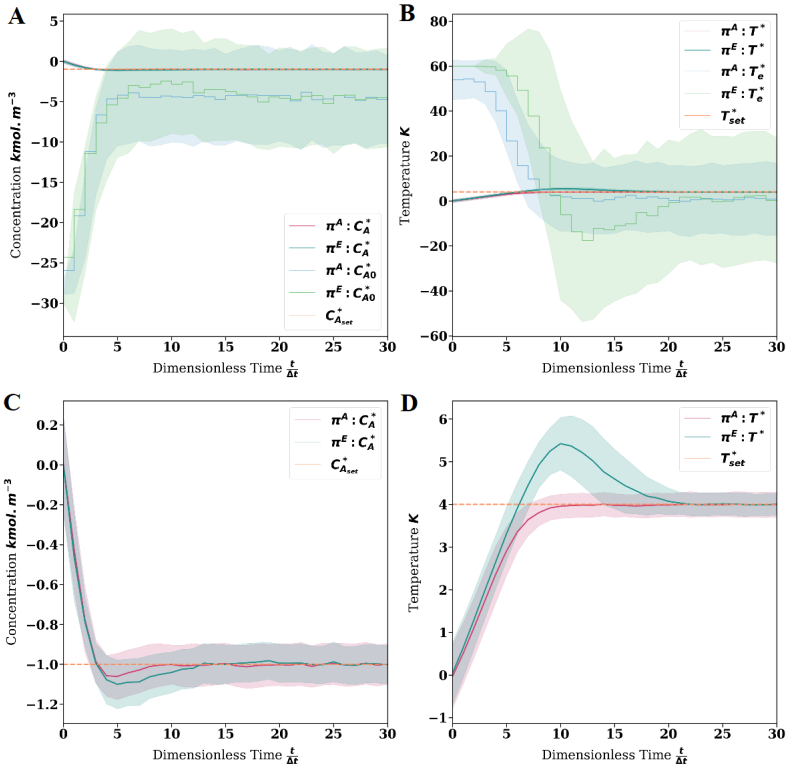


Figure 3.3: Optimal policy of the agent in Case study I. A and B: Control and system response of the concentration control loop and of the temperature control loop, respectively. C and D: Zoomed system response in the concentration control loop and in the temperature control loop, respectively. π^A and π^E indicate the policy of the agent (after online learning) and the PID, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation. Line colours of manipulated variables: blue - π^A; light green - π^E. Line colours of control variables: red - π^A; dark green - π^E. Line colour of set points: orange.

online learning may be deployed for further policy improvement.

Online learning and optimal control

Further improvement of the initial policy (Section 3.5.1) utilises Algorithm 3.1 and a real process reward function shown as Eq. 3.37, which expresses pure set point tracking objective 5

$$R = \varphi_I \left(\widehat{\varepsilon}_{C_A^*}\right) + \varphi_I \left(\widehat{\varepsilon}_{T^*}\right) \tag{3.37}$$

5 Here, the parameter β in φ_I (Eq. 3.32) is re-tuned to ensure that high performance set-point tracking is achieved ($\beta = 30$). The final result of the policy obtained is displayed in Fig. 3.3.

Examination of Fig. 3.3A describes the control policies of the agent and PID1 within the concentration control loop. Given the initialisation provided by IRL, further online RL based policy improvement learns a control observably similar but relatively smoother, to that demonstrated by the PID controller. Explicitly, the policy improvement was provided by two rounds of online learning, with 10 training iterations (epochs) per round. As a result, the agent is able to facilitate a system response, which meets set point faster with less overshoot observed than using the PID controller (shown in Fig. 3.3C). Similar observations are made in analysis of Fig. 3.3B and Fig. 3.3D, which demonstrate the response of the temperature control loop. In this case, the online updated RL yields a better temperature response characterised by a fast rise time with no observable overshoot.

3.5.2 Case Study II - Learning from suboptimal demonstrations

In Case study II, the demonstrations (process data) are derived from a second PID controller (PID2 detailed by the Table B.3). Compared to Case study I, the demonstrations provided by the PID controller here are of an overdamped control response, which subjectively appears suboptimal.

Results of apprenticeship learning via MaxEnt IRL

In similar fashion to Section 3.5.1, Algorithm S1 was used to characterise the demonstrations from PID2. Table 3.3 details the resultant trajectory feature expectations $\mathbb{E}[\boldsymbol{\upsilon}^{\gamma,E}]$.

Table 3.3: The expected discounted trajectory features of the PID2 ($\boldsymbol{\upsilon}^{\gamma,E}$) and the policy learned through AL ($\boldsymbol{\upsilon}^{\gamma,\,\pi}$), and IRL's feature weight ($\boldsymbol{\alpha}^*$) generated in CS II. $Y^* - Type$ indicates the type of trajectory feature and the respective control loop error.

	Trajectory Features					
	$C_A^* - I$	$T^* - I$	$C_A^* - II$	$T^* - II$	$C_A^* - III$	$T^* - III$
$\mathbb{E}\left[\boldsymbol{\upsilon}^{\gamma,E}\right]$	13.76	8.52	8.02	15.53	- 22.49	- 20.71
$\mathbb{E}[\boldsymbol{\upsilon}^{\gamma,\pi}]$	16.41	7.10	6.46	13.29	-21.82	-18.79
$\boldsymbol{\alpha}^*$	-0.259	-0.182	-0.545	-0.093	-0.545	-0.545

Once again, Algorithm 3.2 facilitates the learning of an agent parameterisation of the demonstrated policy in three iterations. It is of note, however, that the methodology was unable to match the trajectory features exactly. Instead, a good approximation of the demonstrated policy was produced. There are two points of discussion here. Firstly, it is likely that the reward function itself is underspecified and further state features $\boldsymbol{\varphi}$ should be proposed. Secondly, it is possible that the objectives of the demonstrated control policy cannot be described purely as a linear combination of the state features (Wulfmeier et al., 2016) – although the linear approximation in this case is reasonable, given the similarity of the trajectory features.

In this case study, state features relevant to the concentration control loop are allocated the greatest weighting. This is because the set points are changed in the same direction (as detailed by Table 3.1). Naturally, a rise in reagent concentration will cause a decrease in temperature (endothermic reaction), whilst a rise in temperature will facilitate the conversion of reagent concentration. As the reaction equilibrium is more sensitive to the temperature change, greater weightings must be added to the concentration control loop to reach the new set point.

Furthermore, Type I state features are allocated negative weights, which is unusual. Intuitively, Type I features represent a similarity measure between the current state of the system and the desired set point. Given that the feature value is non-negative ($\varphi_I = [0, 1]$), a negative reward weighting means that the IRL learnt objective function will prevent the process from reaching the new set point. This is the primarily attributed to the fact that a large proportion of the demonstrations never reached the new set point (Fig. 3.4 and 3.5) due to the overdamped control response. As AL considers the expert's (i.e. PID controller) actions as a noisily optimal control policy, it will find the optimal solution of weight vector $\boldsymbol{\alpha}^*$ to reproduce this overdamped control response. Therefore, the current result indicates that if the demonstration data does not contain a good control policy, it is essential to further improve the AL generated policy through online learning.

Online learning and optimal control

As in Section 3.5.1, online learning is performed to improve the AL policy (initialised for RL). Given that a degree of offset was present in both control loops as detailed by

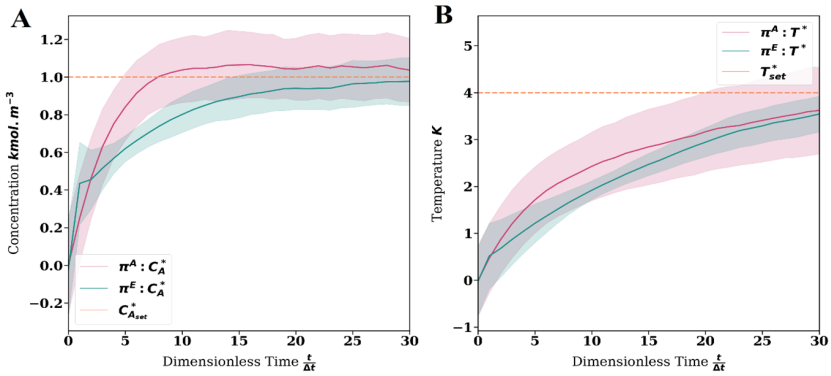


Figure 3.4: System response over the first 30 control interactions from the policy learned from demonstration during AL in Case study II. A and B: System response in the concentration control loop and the temperature control loop, respectively. π^A and π^E indicate the response associated with the policy of the agent (after offline learning) and that demonstrated, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation. Line colours of control variables: red - π^A; dark green - π^E. Line colour of set points: orange.

Fig. 3.4, two short rounds of RL policy improvement, again consisting of 10 training epochs, proceeded with hand tuning of the parameter β in each round. Fig. 3.5 details the final results of the update RL model.

From Fig. 3.5, it is found that the improved policy of the agent π^A, observes a faster rise time, no overshoot and subjectively better set point tracking than the demonstrated policy (PID). In this way, the methodology shows ability to learn from suboptimal demonstrations and then efficiently improve the learned parameterisation of the demonstrated policy through online learning (in this work, 24 minutes spent online to update the RL).

3.5.3 Case study III - Knowledge transfer in learning from demonstration

Finally, Case study III demonstrates how knowledge transfer from one task improves the efficiency of offline apprenticeship learning for further set points. Here, we again assume the availability of existing demonstrations as described by process data. The control task (set point change) in this study is described by Table 3.1 and is different to both tasks examined in Case study I and II. Again, we would like to learn a parameterisation of the control policy (offline) expressed in the process data and then improve

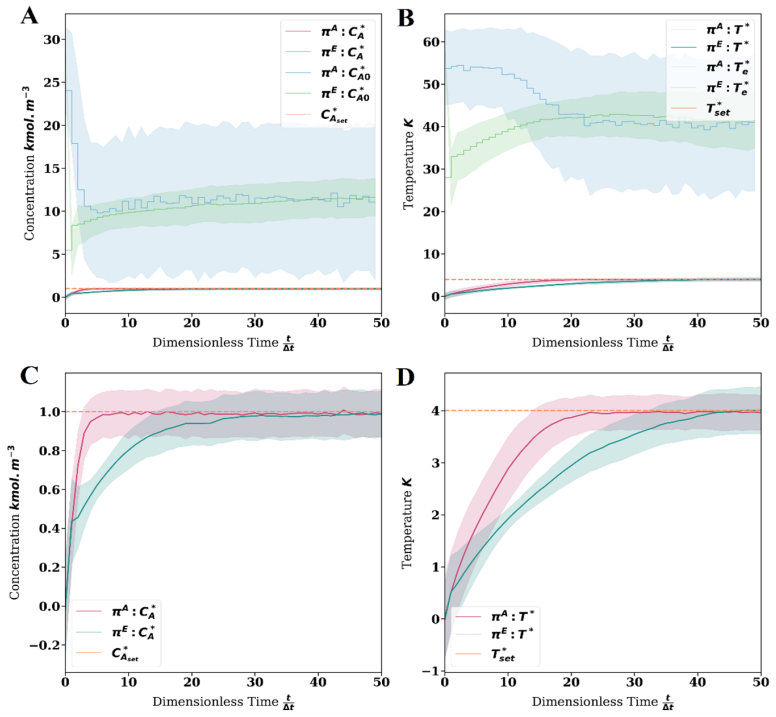


Figure 3.5: Optimal policy of the agent in CS II over the full simulated horizon. A and B: Control and system response of the concentration control loop and the temperature control loop, respectively. C and D: Zoom of the system response in the concentration control loop and in the temperature control loop, respectively. π^A and π^E indicate the policy of the agent (after online learning) and the PID, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation. Line colours of manipulated variables: blue - π^A; light green - π^E. Line colours of control variables: red - π^A; dark green - π^E. Line colour of set points: orange.

it further (online), but we wish to reduce the computational budget associated with offline AL. Thus, we propose to transfer knowledge from a previous study to improve computational and learning efficiency.

Knowledge transfer is in the form of the offline learned policy parameterisation $\pi_{\text{po}}(\theta_{(k_0)},\ \cdot)$ and weight vector $\boldsymbol{\alpha}^*$ from a previous task. Here, knowledge is transferred from Case Study I, given its better PID performance than Case Study II. Both $\boldsymbol{\alpha}^*$ and $\pi_{\text{po}}(\theta_{(k_0)},\ \cdot)$ from Case Study I are provided as initialisation for AL of the new task in Case Study III. Update of this initialisation only takes 80 epochs. Previously, the two studies recovered demonstrated behaviour within a total of 300 and 250 epochs of policy optimisation, respectively. This reduction in the computational intensity of policy learning demonstrates that the computational burden of AL via IRL – under the current methodology – may be significantly reduced through knowledge transfer. In this study, process data was generated using PID1. Table 3.4 details the corresponding trajectory feature expectations $\boldsymbol{\upsilon}^{\gamma,E}$.

Table 3.4: The expected discounted trajectory features of the PID1 generated in CS III. Y^*- Type indicates the type of trajectory feature and the respective control loop error.

	Trajectory Features					
	$C_A^* - I$	$T^* - I$	$C_A^* - II$	$T^* - II$	$C_A^* - III$	$T^* - III$
$\mathbb{E}\left[\boldsymbol{\upsilon}^{\gamma,E}\right]$	16.07	18.36	8.08	8.35	- 21.83	- 22.78
$\mathbb{E}[\boldsymbol{\upsilon}^{\gamma,\pi}]$	14.00	18.04	9.37	6.50	-19.94	-21.06
$\boldsymbol{\alpha}$	0.664	0.052	-0.223	-0.226	-0.403	-0.541

Given the parameterisation as learned via IRL, a further two rounds of 10 epochs of RL enabled further policy improvement online. The results are presented in Fig 3.6. Fig. 3.6A and B highlight how the policy learned under knowledge transfer achieves pure set point tracking with a smoother control policy than that demonstrated by PID1. Once again, Fig 3.6C and D show that this control policy successfully facilitates a system response with fast rise time, but no overshoot or oscillatory behaviour around the set point, as is present in the demonstrations.

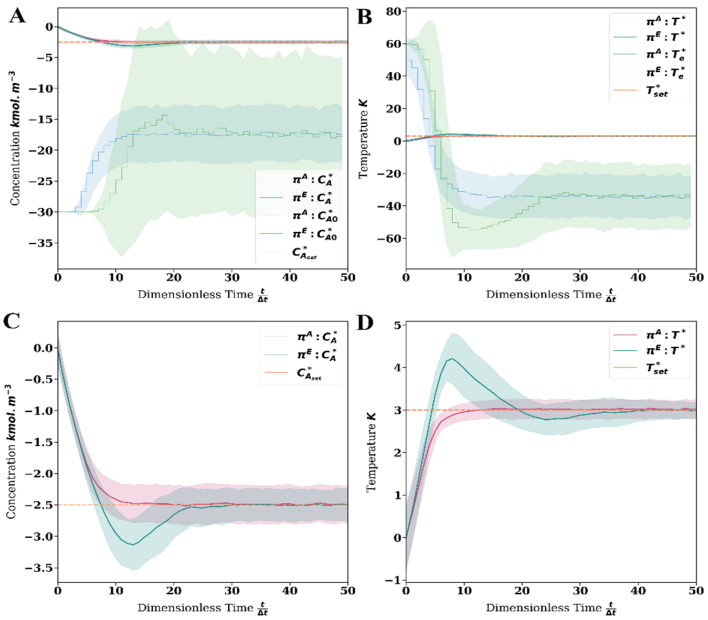


Figure 3.6: Policy π^A generated as a result of knowledge transfer through AL and online policy optimisation. A and B: Control and system response of the concentration control loop and the temperature control loop, respectively. C and D: Zoom of the system response in the concentration control loop and the temperature control loop, respectively. π^A and π^E indicate the policy of the agent (after online learning) and the PID, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation.

3.6 Conclusions

In this paper, we propose a framework based on apprenticeship learning (AL) to learn a control law based on process data, this approach allows us to synthesise a neural network control policy from a previous controller (e.g. PID, MPC or human controllers) more robustly than with supervised learning. Having learned a parameterisation of the control law, subsequent deployment of RL enables further policy improvement by directly interacting with the real process, thus outperforming the existing control law. Here, AL is implemented through inverse reinforcement learning (IRL). Given the data-driven nature of IRL, the RL-based policy parameterisation promises to express the action of the control scheme and process knowledge of the operators. RL is constructed using a policy optimisation algorithm, although other methods could be also applied in the future. Based on the case studies, it is concluded that the proposed framework can effectively extract control information from available process data, transfer knowledge between different cases, and can result in a better optimal control policy efficiently. It should be noted that we assume the availability of rich informative datasets. If the data is not informative, the framework is unlikely to be effective. Future work will explore implementation of various data augmentation strategies, based on physical knowledge or statistical analyses, to artificially synthesise informative datasets.

Chapter 4

Safe chance constrained reinforcement learning for batch process control

This research item is published in the Computers & Chemical Engineering (CACE) Journal, and is accessible via the following reference:

Mowbray, M., Petsagkourakis, P., del Rio-Chanona, E.A. and Zhang, D., 2022. Safe chance constrained reinforcement learning for batch process control. *Computers & Chemical Engineering*, 157, p.107630.

4.1 Introduction

Recently, there has been growing interest amongst the research community and industry in the development of reinforcement learning (RL) based control schemes (Shin et al., 2019). This is underpinned by the ability of RL to naturally account for process stochasticity and handle nonlinear dynamics, and reflected by a growing literature that demonstrates application empirically in applications ranging from set point control (Mowbray et al., 2021; Spielberg et al., 2019), online optimisation and control of batch processes (Kim et al., 2020b; Joshi et al., 2021), real time optimisation (Powell et al., 2020) and production scheduling (Hubbs et al., 2020a). All of these works rely on offline simulation of a process model, with results often validated on the same model that the RL policy was trained. This implicitly considers that the model used offline is in fact a perfect description of the real process and, in the context of control, provokes the question: *"if a model is available, why not use model predictive control (MPC)?"*. In practice, the real system is never perfectly described by the available model. In the presence of uncertainty, the predictions from a model may not have closed-form expression, e.g. propagation of uncertainty using Bayesian inference. Here lies the real attraction of RL controllers - the ability to find an optimal control policy (Kirk, 2004; Bertsekas et al., 1995) independently of closed-form expressions of the uncertain process dynamics, as is required by conventional finite dimensional optimisation approaches such as stochastic, tube and distributionally robust MPC (Kouvaritakis and Cannon, 2016; Langson et al., 2004; Lu et al., 2020). Additionally, the use of RL allows for a greater diversity of models i.e. they are not required to be smooth.

However, there is a dualism implicit to RL. RL is very data expensive because knowledge about the uncertain dynamics and the quality of a control policy is instead gained by sampling (Sutton and Barto, 2018a). Offline learning (simulation) is absolutely required due to the cost of real world data and the operational and safety risk associated with conducting the RL process online. As a result, there remains a dependence on the availability of a description of the physical system for offline simulation, which provides means to conduct preliminary learning before deployment to the real system. Despite this, few works consider the transfer of the policy (Petsagkourakis et al., 2020b) to the real online system, which promotes concerns for

operational safety[1]. For example, if model-process mismatch exists, constraints may be violated or the process driven to unsafe operating regimes. Given the acknowledgement that no model is a perfect description of the real process - the development of methods should consider that RL exploits the mathematical nature of the offline model. Similar concerns are addressed in Hüllen et al. (2020).

Broadly, there are two approaches to synthesising the type of safe controller required: modifications could be made to the reinforcement learning process (Kumar et al., 2020b; Agarwal et al., 2020; Yu et al., 2021b), or modifications made to the offline model (Kidambi et al., 2021; Yu et al., 2020), which can then be integrated into the RL objective. Recent works are discussed in the following with consideration directed to both operational and safety concerns.

4.1.1 Safe Reinforcement Learning

One of the earliest works in process systems engineering (PSE), which considers the online operational safety of reinforcement learning is provided by Lee and Lee (2005). Here, the authors present an action-value method, with integration of a Parzen probability density estimator (Parzen, 1962) to bias the action-value function approximation based on the local data density. In this case, the data is used to construct the action-value function and hence the data density helps quantify epistemic uncertainty (i.e. the reducible part of model uncertainty arising from a lack of information - data or knowledge - about the underlying functional (Hüllermeier and Waegeman, 2019)). This concept is shared in more recent work (Clements et al., 2020), and enables the implementation to produce conservative controls and restricts optimisation from exploiting the mathematical nature of the approximate action-value function. However, this approach does not consider operational constraints or the accuracy of the underlying model. For RL to be deployed to real process systems, operational constraints should be satisfied with high probability (if soft). One approach to achieve this is underpinned by modification of the control selected by the RL agent, in order to ensure the system remains within some safe set via direct optimal control (DOC) (Li et al., 2021; Wabersich and Zeilinger, 2021). However, the use of DOC retains explicit

[1]This is also placed in the scope of a wider concern regarding the *interpretability* of machine learning systems

dependence upon a process model and imposes non-trivial learning rules that could affect the optimality of the policy produced.

Other methods directly leverage the Markov decision process (MDP) formulation, upon which the reinforcement learning problem is built. This approach tends to avoid DOC and promotes use of 'model-free' methods. A reasonably popular approach to address constraints in the RL setting is provided by the constrained MDP (CMDP) formulation. In Achiam et al. (2017), the authors approach the need for satisfaction of operational constraints via CMDP, but do so in expectation and simultaneously negate process-model mismatch. In Huh and Yang (2020), the authors propose the identification of a lyapunov function (this time model-free and outside of the CMDP framework) to ensure the process stays within some safe set with a given probability. However, potential issues arising from plant-model mismatch are similarly ignored in offline simulations. In Leurent et al. (2020), an approach to robust control is presented (i.e. the method optimises for the worst case event), and the presence of process-model mismatch is considered. However, the framework is limited to linear systems with additive uncertainty. Recently, in Peng et al. (2021) the authors present an approach to address high probability constraint satisfaction based on the augmented lagrangian. However, the penalty term presented does not provide information about the quality of control selection (i.e. essentially ignoring the RL problem) and is likely to lead to conservative control policies. There have been two methods proposed recently, by Petsagkourakis et al. (2020a); Pan et al. (2020), which integrate a similar penalty method into the RL problem properly, and achieve high probability constraint satisfaction. This is achieved through deployment of the concept of constraint tightening, which is common to the stochastic MPC (sMPC) community (Mesbah et al., 2019; Valdez-Navarro and Ricardez-Sandoval, 2019; Rafiei and Ricardez-Sandoval, 2020). A further method has been proposed by Yoo et al. (2021c) for the case of hard constraints, which constructs a slow non-stationary MDP to promote stability of learning via the implementation of a dynamic penalty method. However, the aforementioned works negate the presence of offline model-process mismatch.

Most of the previous works ignore issues arising from process-model mismatch. The domain of batch RL (otherwise known as offline RL) has drawn a lot of recent research interest (Kumar et al., 2020a). The promise of this field lies in the synthesis

of real-world control policies from existing datasets (offline). The key idea in batch RL is to learn with awareness of the limitations of the available data. Many of the works set in this domain focus on action-value methods and look to bias (or regularise) the action-value function approximation (Agarwal et al., 2020) by considering the data density (Kumar et al., 2020b) in a manner not dissimilar to Lee and Lee (2005). More recently, attention has been directed towards considerate construction of an offline model, based on the available data and this directs attention in the following analysis.

4.1.2 Uncertainty aware modelling and control

A key consideration in the development of model-based RL approaches is the relationship between model construction and policy learning. For example, in Rajeswaran et al. (2020), the problem of learning under the limitations of a local model and improving policy performance on the real process is considered within a game theoretic framework (similar to model-based design of experiments). However, it is not clear as to whether this approach would ensure real-process safety unless modifications were made to the reward function. This problem is approached by the work presented in Petsagkourakis and Galvanin (2020) and more recently in Kidambi et al. (2021). In Zanon and Gros (2020), the authors integrate RL into a robust, linear MPC scheme, by using an RL policy to parameterise an uncertainty set. This allows for ensurance of *optimality* under the scheme, but is traded at the price of restrictive modelling assumptions. In Lütjens et al. (2019), model uncertainty is incorporated into a penalty function for RL, however, the uncertainty estimate is gained through approximate methods such as bootstrapping and MC dropout, which provides computational cost. In Kidambi et al. (2021), the epistemic uncertainty associated with offline prediction is quantified via the variance of a model ensemble. The epistemic uncertainty is used to modify the reward function of the MDP to synthesise a safe control policy without further interaction with the real system. A type of model, which achieves this more naturally than an ensemble, is the Gaussian process (GP). GPs are data-driven models and their use is well documented in PSE applications (Sternberg and Deisenroth, 2017; Frigola, 2015; Bradford et al., 2020, 2018; del Rio Chanona et al., 2021). In part, this is due to their compatibility with small datasets, but primarily for their natural quantification of epistemic and aleatoric uncertainty. In a number of previous works,

(realisations of) GPs have been used to inform control decisions. Most of these works lie in the domain of sMPC (Bradford et al., 2020; Umlauft et al., 2018), however, a few hail from the field of RL-based policy optimisation (Deisenroth and Rasmussen, 2011; Curi et al., 2020; Berkenkamp et al., 2017). In Deisenroth and Rasmussen (2011), the authors compute gradients for policy improvement analytically, resulting in a highly efficient algorithm for unconstrained problems. In Curi et al. (2020), the authors utilise GPs and the variance of the posterior distribution to produce a controller-directed exploration strategy, but negate propagation of model uncertainty and, again, process constraints. Whereas Berkenkamp et al. (2017) present an algorithm that simultaneously balances exploration and exploitation of a GP model, considers constrained problems and provides stability guarantees for the policy identified. In the following, we draw from works closer to sMPC (Umlauft et al., 2018; Bradford et al., 2020), to synthesise a safe RL-based control policy, which considers both operational constraints and process-model mismatch.

4.1.3 Contribution

A number of RL-based methodologies have been proposed to ensure operational constraints are satisfied with high probability (Pan et al., 2020; Petsagkourakis et al., 2020a; Peng et al., 2021). Other works have been proposed to consider the process-model mismatch that exists when learning an RL policy offline (Kumar et al., 2020b; Kidambi et al., 2021; Yu et al., 2021b). However, as far as the authors are aware, there are no RL methods, which achieve both. In this work, we propose a method that synchronously satisfies operational constraints with high probability, whilst respecting the limitations of a process model. Specifically, we deploy the use of GPs to construct a data-driven state space model. The variance of the posterior predictive distribution of the GP is used in two different ways: firstly, it provides a constraint tightening mechanism to back the nominal (or expected) process away from the constraint boundary (to provide constraint satisfaction with high probability); and, secondly, it is used to penalise exploration of regions of the GP model with high epistemic uncertainties. The full method as proposed also implements a Bayesian optimisation strategy in order to tune the degree of constraint tightening - balancing operational risk with performance. Here, we draw analogue to reward shaping, except in this case, we identify a policy

variant mechanism for constraint satisfaction as desired (Ng et al., 1999). Importantly, the dimensionality of the shaping problem is equivalent to the number of operational constraints imposed on the system, which provides means to scale the method to larger problems. Further advantages include the inheritance of the MDP framework - which theoretically enables us to account for uncertainty in a proper closed loop manner - as well as the mitigation of resolving an optimisation problem online (as is required by conventional methods). Instead controls are selected via inference, which lends itself naturally to handling systems of both fast and slow dynamics. Additionally, the approach is completely data-driven and synchronously accounts for model uncertainty, removing demands for assumption of mechanistic process knowledge.

The following is structured as follows: in Section 4.2, we outline the problem statement and implicitly define the processes of interest; in Section 4.3, the methodology is presented; in Section 4.4 a fed-batch bioprocess case study is presented with a view to demonstrate the methodology; in Section 4.5 and 4.6 the results and discussion, and conclusion are presented, respectively.

4.2 Problem statement

This work is concerned with the synthesis of an optimal control strategy for nonlinear, uncertain systems of the form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t) \tag{4.1}$$

where $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ denotes the system state; $\mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ the control inputs to the system; $t = [1, \ldots, T]$ denotes the discrete time index; $\mathbf{s} \in \mathbb{S} \subseteq \mathbb{R}^{n_s}$, where $\mathbb{S}$ represents a set of realisations of process stochasticity; and, $f : \mathbb{X} \times \mathbb{U} \times \mathbb{S} \to \mathbb{X}$. Here, no formal assumption is made regarding the source of stochasticity $\mathbb{S}$, but it could be introduced via parametric uncertainty or disturbances. In either case, given the presence of stochasticity within system description, Eq. 4.1 may be expressed equivalently via the following conditional probability density function:

$$\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \tag{4.2}$$

Specifically, it is assumed that the process dynamics adhere to description as a Markov process, and therefore that the associated decision-making problem may be formalized

as a Markov decision process (MDP). MDPs provide a probabilistic value framework for decision making in uncertain systems, which display the Markov property. Under the MDP framework, the probability of observing a given process trajectory $p(\boldsymbol{\tau})$, under a control policy π is described:

$$p(\boldsymbol{\tau}) = p(\mathbf{x}_0) \prod_{t=0}^{T-1} \pi(\mathbf{u}_t|\mathbf{x}_t) p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \tag{4.3}$$

where $\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{u}_0, \ldots, \mathbf{x}_T)$ denotes the process trajectory; $p(\mathbf{x}_0)$ denotes the initial state distribution; $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ the process dynamics; and the policy $\pi(\mathbf{u}_t|\mathbf{x}_t)$ is explicitly defined as a conditional probability function over control inputs. Provided process evolution is subject to a stochastic policy and process dynamics, the performance of a policy is evaluated via the expected discounted sum of rewards $R_{t+1} \in \mathbb{R}$ accumulated from the initial state:

$$\begin{aligned} G(\boldsymbol{\tau}) &= \sum_{t=0}^{T-1} \gamma^t R_{t+1} \\ J &= \int p(\boldsymbol{\tau}) G(\boldsymbol{\tau}) d\boldsymbol{\tau} \end{aligned} \tag{4.4}$$

where the reward is allocated by a reward function $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to R_{t+1}$ and $\gamma = [0, 1]$ is the discount factor. Therefore, the optimal policy π^*:

$$\pi^* = \arg\max_{\pi} J \tag{4.5}$$

One approach to learning such a controller is via Reinforcement Learning (RL). However, under the framework provided by MDPs, the optimal policy π^* (and, hence RL) implicitly neglects the satisfaction of both safety and operational constraints. In applications related to this work (i.e. industrial batch process systems), the satisfaction of both operational and safety constraints is of concern. As such, it is of interest to develop an RL-based methodology for the synthesis of an optimal control policy π^*_C, which respects constraints. The problem statement follows that common to works set

in the domain of stochastic optimal control:

$$\mathcal{P}(\pi_C) := \begin{cases} \max\limits_{\pi} J \\ \text{s.t.} \\ \mathbf{x}_0 \sim p(\mathbf{x}_0) \\ \mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{u}_t \sim \pi(\mathbf{u}_t|\mathbf{x}_t) \\ \mathbf{u}_t \in \hat{\mathbb{U}} \\ \mathbb{P}(\bigcap\limits_{i=0}^{T}\{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}) \geq 1-\alpha \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{4.6}$$

where $\hat{\mathbb{U}} \subset \mathbb{U}$ represents the set of control inputs, which satisfy hard constraints on the control space; and, $\hat{\mathbb{X}} \subset \mathbb{X}$ denotes the set of states, which satisfy operational and safety constraints imposed on the state space. Under the assumption that the problem definition may have n_g constraints, $\hat{\mathbb{X}}$ may be expanded more generally as the *joint* chance constraint set, such that:

$$\hat{\mathbb{X}}_t = \{\mathbf{x}_t \in \mathbb{G}_{j,t}, \forall j \in \{1, \ldots, n_g\}\} \tag{4.7}$$

where $\mathbb{G}_{j,t} \subset \mathbb{R}^{n_x}$ defines the set of states, which ensure satisfaction of the j^{th} constraint at time step t. Specifically, in the following analysis, we assume that:

$$\mathbb{G}_{j,t} = \{\mathbf{x}_t \in \mathbb{R}^{n_x} : A_j^T \mathbf{x}_t - b_j \leq 0\} \tag{4.8}$$

where $A_j \in \mathbb{R}^{n_x}$ and $b_j \in \mathbb{R}$ define the j^{th} constraint. The general principles discussed subsequently extend to problems with nonlinear constraints. However, in that case, the constraints should be represented by lower order power series expansions of the nonlinear functions (Rafiei and Ricardez-Sandoval, 2018) i.e. the nonlinear expressions should be linearized. Given that the process is stochastic, the constraints are 'softened' such that satisfaction is guaranteed for all time $t = \{0, \ldots, T\}$ with a desired probability, denoted $1 - \alpha$.

Theoretically, solution to Eq. 4.6 may be realised via exact dynamic programming (DP), which requires exact descriptions of the probabilistic process dynamics. In process systems, these are typically unavailable. Further, DP is known to suffer from

the *the curse of dimensionality*, which implies that high dimensional problems, or those that operate over continuous state and control spaces, are computationally intractable. In the domain of sMPC, works generally leverage reformulation of the problem via deterministic expressions for the joint chance constraints and modelling assumptions regarding the nature of process stochasticity (Mesbah et al., 2019; Subramanian et al., 2021). This work similarly forms a deterministic surrogate of Eq. 4.6 in combination with Gaussian process (GP) data-driven modelling, and identifies a reinforcement learning (RL) based control policy, which naturally accounts for process stochasticity in a closed-loop manner. These benefits are complementary to those noted in Section 4.1.3. In the following section, a methodology is proposed for synthesis of the controller π_C.

4.3 Methodology

4.3.1 Gaussian processes for data-driven dynamic modelling

Model-free RL-based policies are learned through Monte Carlo (MC) sampling of the process dynamics and iteratively improved based on the collected data. This is otherwise known as policy iteration. For real world applications, the synthesis of RL-policies is dependent upon an accurate description (model) of the process dynamics. For nonlinear, uncertain processes, construction of mechanistic dynamical models can be problematic, even if understanding of the fundamental mechanisms driving process behaviour exists. Hence, the construction of a purely data-driven model is proposed to represent the discrete time, evolution of the nonlinear, uncertain dynamical system described by $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t)$, i.e. Eq 4.1. In order to construct a representation of the system dynamics, it is assumed that: a) f is a smooth function and b) there is an available dataset $\mathcal{D}$, which is composed as follows:

$$\mathcal{D} = [\boldsymbol{\Upsilon}^T \ \mathbf{Y}^T], \qquad \mathbf{Y} = [\mathbf{y}_i, \ldots, \mathbf{y}_N], \qquad \boldsymbol{\Upsilon} = [\boldsymbol{\upsilon}_i, \ldots, \boldsymbol{\upsilon}_N], \qquad \boldsymbol{\upsilon}_i = \left[\mathbf{x}_i^T \ \mathbf{u}_i^T\right]^T, \tag{4.9}$$

where $\boldsymbol{\upsilon} \in \mathbb{V} \subseteq \mathbb{R}^{n_\upsilon}$, $n_\upsilon = n_x + n_u$ are input measurements and $\mathbf{y} \in \mathbb{Y} \subseteq \mathbb{R}^{n_x}$ are output measurements of the system, which are gathered subject to some noisy process $\boldsymbol{\omega} \in \mathbb{W} \subseteq \mathbb{R}^{n_x}$ (Rasmussen, 2006). Here, $\mathbb{W}$ is assumed to be an infinite set

representative of possible realisations of system noise, such that:

$$\begin{aligned} \mathbf{y}_i &= f(\boldsymbol{\upsilon}_i) + \boldsymbol{\omega}_i \\ \boldsymbol{\omega}_i &\sim \mathcal{N}(0, \Sigma_n) \end{aligned} \tag{4.10}$$

where $\Sigma_n = diag([\sigma^2_{n,1}, \ldots, \sigma^2_{n,n_x}]) \in \mathbb{R}^{n_x \times n_x}$ defines a diagonal matrix, where each element on the diagonal denotes a state dependent variance. Further, as usual, it is assumed that all datapoints $d_i = [\boldsymbol{\upsilon}_i, \mathbf{y}_i]$ (equivalent to rows of $\mathcal{D}$) are independently and identically distributed (i.i.d.). Parallel can be drawn between Eq. 4.2, such that Eq. 4.10 is equivalently described as a conditional probability function $\mathbf{y} \sim p(\mathbf{y}|\boldsymbol{\upsilon})$. This description of data generation shares similarities to assumptions made in Section 4.2 and directs attention to a branch of probability theory known as stochastic processes (SPs), and in particular Gaussian processes (GPs).

Gaussian processes

SPs define a probability model over an infinite collection of random variables, any finite subset of which have a joint distribution (Lindgren, 2012). This definition leads to the interpretation of SPs as probability distributions over functions (Rasmussen, 2006), such that one realisation of an SP can be thought of as obtaining a sample from a function space. When the distribution over the function space is assumed Gaussian, the resultant model is termed a GP.

A GP is fully specified by a mean function, $\mathbf{m} : \mathbb{V} \to \mathbb{R}$, and covariance function, $k : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$, such that:

$$f_{GP}(\boldsymbol{\upsilon}) \sim \mathcal{GP}\big(\mathbf{m}(\boldsymbol{\upsilon}), k(\boldsymbol{\upsilon}, \boldsymbol{\upsilon}')\big) \tag{4.11}$$

A number of covariance functions exist within the GP toolbox. Selection of both the function and the associated hyperparameters, $\boldsymbol{\lambda} \in \mathbb{R}^{n_\lambda}$, define the properties of the GP in function space. As such, the decision as to appropriate covariance function is often informed by domain knowledge and understanding of the modelling problem at hand. The definition of hyperparameters is handled by maximisation of the marginal log-likelihood (this is discussed in C.1.1 and referred to as GP training). Popular choices include the Matern 5/2 and radial basis function (RBF) covariance functions (Rasmussen, 2006). Definition of the mean function is also important. Often, a zero

mean ($\mathbf{m}(\boldsymbol{\upsilon}) = 0$) is assumed, which is not unreasonable given standardisation of the output data, $\mathbf{Y}$.

GP model inference takes place within the framework provided by Bayesian reasoning. The assertion of a modelling decision regarding the mean and covariance function therefore represents a prior belief about the possible properties of the hidden, functional relationship expressed in the dataset $\mathcal{D}$. When presented with a new test input $\boldsymbol{\upsilon}^* \in \mathbb{R}^{n_\upsilon}$, the construction of a single GP model for the j^{th} state leads to the generation of an associated prediction $y_j^* \in \mathbb{R}$ via the following joint prior distribution:

$$\begin{bmatrix} \mathbf{Y}_j^T \\ y_j^* \end{bmatrix} = \mathcal{N}\left(0, \begin{bmatrix} K + \sigma_n^2 I_N & K_* \\ K_*^T & k(\boldsymbol{\upsilon}^*, \boldsymbol{\upsilon}^*) \end{bmatrix}\right) \tag{4.12}$$

where $\mathbf{Y}_j \in \mathbb{R}^{1\times N}$ denotes the j^{th} row of the output of the training dataset $\mathbf{Y}$; $K \in \mathbb{R}^{N\times N}$ denotes the Gram matrix, such that provided with training input measurements (see Eq. 4.9), element $k_{m,n} = k(\boldsymbol{\upsilon}_m, \boldsymbol{\upsilon}_n)$, where $m = [1, \ldots, N]$ and $n = [1, \ldots, N]$; σ_n^2 denotes the variance of the noise associated with observation of state $y_j \in \mathbb{R}$ (see Eq. 4.10); $K_* \in \mathbb{R}^N$ denotes the covariance of the test datapoint $\boldsymbol{\upsilon}^*$ with the existing (training) input measurements; and, lastly, $k(\boldsymbol{\upsilon}^*, \boldsymbol{\upsilon}^*) \in \mathbb{R}$ represents the variance of the test datapoint.

Furthermore, as GPs operate through Bayesian reasoning, by *conditioning* the joint prior distribution (Eq. 4.12) upon the observed dataset $\mathcal{D}$ and the test point $\boldsymbol{\upsilon}^*$, we obtain a predictive posterior Gaussian distribution, with mean μ_j and variance σ_j^2 as follows:

$$\begin{aligned} \mu_j(\boldsymbol{\upsilon}^*) &= K_*^T (K + \sigma_n^2 I_N)^{-1} \mathbf{Y}_j^T \\ \sigma_j^2(\boldsymbol{\upsilon}^*) &= k(\boldsymbol{\upsilon}^*, \boldsymbol{\upsilon}^*) - K_*^T (K + \sigma_n^2 I_N)^{-1} K_* \end{aligned} \tag{4.13}$$

In the context of dynamical systems modelling, Eq. 4.13 represents a probability model over the next state of the dynamical system at the next discrete time index. The construction of a posterior probability function is particularly useful in engineering applications, given that it expresses elements of both aleotoric and epsitemic model uncertainty. Typically, the mean is taken as the model's prediction, however, prediction may also be directly sampled from posterior distribution (Bradford et al., 2018). This will be discussed further in section 4.3.1.

Thus far, the methodology has formalised the construction of GPs, and defined

them as multiple-input, single-output models. Hence a single GP provides a functional mapping descriptive of the future discrete time evolution of a single state, given observation of the full system state and control inputs at the current time index. It is of interest to this work to construct a multiple-input, multiple-output state space model. This is discussed subsequently in Section 4.3.1 and has been presented previously by other related works (Bradford et al., 2020; Umlauft et al., 2018; Deisenroth and Rasmussen, 2011). We direct the interested reader for more information.

Gaussian processes for state space modelling

In this study, state space models are constructed by training n_x GP models separately and combining them to simultaneously predict the state vector $\mathbf{x} \in \mathbb{R}^{n_x}$ at the next discrete time interval, $t+1$. Specifically, under the assumption that each of the n_x models has been constructed and trained according to Section 4.3.1 and C.1.1, this implies that the the posterior prediction from the GP state space model, when presented with $\boldsymbol{\upsilon}_t$ follows:

$$\begin{aligned} \boldsymbol{\mu}(\boldsymbol{\upsilon}_t; \mathcal{D}) &= \big[\mu_1(\boldsymbol{\upsilon}_t), \ldots, \mu_{n_x}(\boldsymbol{\upsilon}_t)\big] \\ \boldsymbol{\Sigma}(\boldsymbol{\upsilon}_t; \mathcal{D}) &= diag(\sigma_1^2(\boldsymbol{\upsilon}_t), \ldots, \sigma_{n_x}^2(\boldsymbol{\upsilon}_t)) \\ \mathbf{x}_{t+1} &\sim \mathcal{N}\big(\boldsymbol{\mu}, \boldsymbol{\Sigma}\big) \end{aligned} \tag{4.14}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{n_x}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n_x \times n_x}$ amd $\mathbf{x}_{t+1} \in \mathbb{R}^{n_x}$ is the next state . In the following section, we discuss how the GP state space model is used to generate realisations of underlying process stochasticity, and relate discussion directly to the decision making process.

Gaussian process realisations and decision making

For effective and safe control and optimisation of process systems, a control policy must consider worst-case realisations of process stochasticity. In GP models, function realisations are sampled from the GP. Each function realisation represents a specific instance of model uncertainty across process evolution - including the worst case. In order to achieve this, model uncertainties must be propagated correctly. This work implements the method detailed in Umlauft et al. (2018), which recursively updates the dataset $\mathcal{D}$ as the process evolves between discrete time indices. This process is

detailed by Algorithm C.1 in C.1.2 and relies upon linear algebra to account for the effects of conditioning the GP models on the updated dataset. See Bradford et al. (2020); Strassen (1969) for more details.

In the following section, an approach that synchronously combines concepts from sMPC and RL to produce a self-optimizing, policy varying reward shaping mechanism is presented, which provides probabilistic constraint satisfaction. Specifically, a penalty function method is combined with the concept of backoffs.

4.3.2 Safe chance constrained policy optimisation with Gaussian processes

In this section, we provide details of the methodology, which enables combination of GP state space models with RL-based policy optimisation for high probability constraint satisfaction. To achieve this, the methodology is organised as follows and the full algorithm is detailed by Algorithm 4.2:

1. In Section 4.3.2, the general stochastic optimal control problem defined by Eq. 4.6 is modified to consider the nominal evolution of the states and obtain a deterministic expression for the probabilistic joint constraints. To facilitate this, we implement an approach similar to Petsagkourakis et al. (2020a) in combination with a GP state space model.

2. In Section 4.3.2 the deterministic surrogate constraints are incorporated into a reformulation of the RL objective (see Eq. 4.4) via an l_p penalty function[2] (Nocedal and Wright, 2006; Larson et al., 2019) and detail of a general constrained policy optimisation algorithm is provided (Schulman et al., 2017b).

3. Then, in Section 4.3.2, an 'efficient' global optimisation strategy (Jones et al., 1998) is presented to iteratively tune the penalty function enabling satisfaction of the original joint chance constraints with the desired probability $1 - \alpha$ establishing a strong connection between this work and reward shaping (Ng et al., 1999).

[2]The subscript p of l_p denotes the norm incorporated into the penalty function

The methodology is formalized with a view to the use of policy optimisation methods, however, the concepts discussed can also be integrated into actor-critic and action-value methods (Sutton and Barto, 2018a).

Probabilistic joint chance constraints

In this section, reformulation of the probabilistic joint chance constraint detailed by Eq. 4.6 is presented. The joint chance constraints are restated here for ease:

$$\mathbb{P}(\bigcap_{i=0}^{T}\{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}) \geq 1-\alpha \tag{4.15}$$

The following analysis proceeds to obtain a set of deterministic surrogate constraints, which can then be integrated into a revised objective for RL-based policy optimisation. In particular, we leverage Boole's inequality and the Cantelli-Chebyshev inequality to obtain a deterministic constraint for each of those that comprise the original joint constraint. The analysis follows Paulson et al. (2020); Farina et al. (2014).

Lemma 1 ***Boole's Inequality*** *(Boole, 1847): Consider a finite set of countable events $\{Z_1, Z_2, \ldots, Z_{n_g}\}$, the probability that one of these events occurs is no greater than the sum of the probabilities of the individual events:*

$$\mathbb{P}\Big(\bigcup_{i=1}^{n_g} Z_i\Big) \leq \sum_{i=1}^{n_g} \mathbb{P}(Z_i) \tag{4.16}$$

Now, considering n_g constraints comprise the joint chance constraint, then applying this result enables decomposition of Eq. 4.15, into n_g individual chance constraints. As in Petsagkourakis et al. (2020a), for ease of notation, we define the following:

$$X = \max_{(t,j)\in\{0,\ldots,T\}\times\{1,\ldots,n_g\}} A_j\mathbf{x}_t - b_j, \qquad g = \{\mathbf{x}\in\mathbb{R}^{n_x} : X\}, \qquad \mathbb{G}_j^{'} = \bigcap_{i=0}^{T}\{\mathbf{x}_i \notin \mathbb{G}_{j,i}\}$$

where $\mathbb{G}_j^{'}$ defines the set of states, which do not satisfy constraint j for all time indices and $X \in \mathbb{R}^{n_x}$ defines a random variable. From Lemma 1:

$$\mathbb{P}\Big(\bigcup_{j=1}^{n_g}\{g \subset \mathbb{G}_j^{'}\}\Big) \leq \sum_{i=1}^{n_g} \mathbb{P}(g \subset \mathbb{G}_j^{'}) \tag{4.17}$$

Explicitly, Eq. 4.17, dictates that the probability of achieving joint constraint satisfaction under a given policy π is lower bounded by the probability of satisfying each of

the respective constraints individually. Therefore, guaranteeing satisfaction of chance constraints individually can be considered a robust approximation to joint satisfaction:

$$\iota_j = \mathbb{P}(g \subset \mathbb{G}_j^{'}) \implies \alpha \leq \sum_{j=1}^{n_g} \iota_j$$

where $\iota_j \in \mathbb{R}$, subject to satisfying Eq. 4.17. This enables approximation of Eq. 4.15 via the following:

$$\sum_{j=1}^{n_g} \mathbb{P}\big(\bigcap_{i=0}^{T} \{\mathbf{x}_i \in \mathbb{G}_{j,i}\}\big) = 1 - \sum_{j=1}^{n_g} \iota_j \tag{4.18}$$

In this work, we define $\iota_j = \alpha/n_g,\ j = [1, \ldots, n_g]$. Having decomposed the original joint chance constraint into a set of individual chance constraints, the methodology looks to express a set of deterministic surrogate expressions (of the original probabilistic chance constraints), which can then be incorporated into the method presented.

To proceed, we deploy the concept of *constraint tightening*, which is an approach commonly deployed within the domain of sMPC. The intuition behind constraint tightening is described as follows. The process of concern is subject to unbounded uncertainties. We consider that under a given policy π, the process will vary probabilistically within a given region of $\hat{\mathbb{X}}_t$. Specifically, one can assume that the process will vary within some euclidean distance from the nominal or expected behaviour with a given probability. If we *back* the nominal process *off* from the constraint boundary then we will be able to achieve chance constraint satisfaction with the desired probability. This is underpinned by the Cantelli-Chebyshev inequality, which is described by Lemma 2

Lemma 2 ***Cantelli-Chebyshev Inequality*** *(Ogasawara, 2019): Consider a random variable Z, with expected value $\mathbb{E}\big[Z\big]$ and finite variance $\Sigma[Z]$, then:*

$$\mathbb{P}\big(Z - \mathbb{E}\big[Z\big] \geq \delta\big) \leq \frac{\Sigma\big[Z\big]}{\Sigma[Z] + \delta^2}$$

The mechanism of constraint tightening takes the form of a set of *backoffs* $\boldsymbol{\varepsilon}_j = [\varepsilon_{j,0}, \ldots, \varepsilon_{j,T}]$, which can be conceptualised as the necessary euclidean distance from the expected or nominal state $\bar{\mathbf{x}}_t \in \mathbb{R}^{n_x}$ to the constraint boundary to guarantee chance satisfaction with a given probability (note, backoff values are specific to both the constraint and time index). As stated in Section 4.2, the analysis provided in this work assumes affine constraints. Therefore, the tightened constraint sets follow:

$$\begin{aligned} \bar{\mathbb{G}}_{j,t} &= \{\bar{\mathbf{x}}_t \in \mathbb{R}^{n_x} : A_j^T \bar{\mathbf{x}}_t + \varepsilon_{j,t} - b_j \leq 0\} \\ \bar{\mathbb{X}}_t &= \{\bar{\mathbf{x}}_t \in \bar{\mathbb{G}}_{j,t}, \forall j = \{1, \ldots, n_g\}\} \end{aligned} \tag{4.19}$$

The determination of the backoff values $\varepsilon_{j,t}$ is handled via the following analysis. Specifically, we work from the developments made in Farina et al. (2014); Magni et al. (2009), which (via Lemma 2) show that the Cantelli-Chebyshev approximation of the backoff set is equivalent to:

$$\varepsilon_{j,t} = \sqrt{\frac{1-\iota_j}{\iota_j}}\sqrt{A_j^T \Sigma[\mathbf{x}_t] A_j} \tag{4.20}$$

where $\boldsymbol{\varepsilon}_{j,t}$ represents a robust approximation of the backoff required for individual chance constraint satisfaction with the desired probability ι_j. In this work, we deploy a GP state space model to estimate both the nominal state $\bar{\mathbf{x}} \in \mathbb{R}^{n_x}$ and the variance of the state $\Sigma[\mathbf{x}_t]$, as described by Eq. 4.14, enabling construction of a deterministic expression for each of the individual chance constraints. In practice, it is well documented that use of the Cantelli-Chebyshev approximation leads to overly-conservative control policies, which operate far from the constraint boundary. In order to balance the performance of the control trajectory, with constraint satisfaction, we propose to tune $\varepsilon_{j,t}$ via a multiplying factor $\xi_j = [0, 1]$ for each constraint. As such, the deterministic surrogate for each of the individual chance constraints, detailed by Eq. 4.18, are described:

$$A_j^T \bar{\mathbf{x}}_t + \xi_j \sqrt{\frac{1-\iota_j}{\iota_j}}\sqrt{A_j^T \Sigma[\mathbf{x}_t] A_j} - b_j \leq 0 \tag{4.21}$$

The approach to the tuning of the multiplying factors, $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_{n_g}]$, could be handled via Bayesian optimisation (BO) or bisection method (Petsagkourakis et al., 2020a; Bradford et al., 2020; Pan et al., 2020). This work employs a BO strategy, which is detailed by Section 4.3.2. The computational implications for this are small given the efficiency of BO.

The use of a GP to parameterise the backoff values $\boldsymbol{\varepsilon}_j$ is more efficient than the set of methods proposed previously by Petsagkourakis et al. (2020a); Pan et al. (2020). In those works, initial backoff values were estimated via MC sampling and then tuned. Here, we provide a method to analytically express the backoff values via the posterior predictive distribution of the GP state space model, removing the requirement for sampling and the potential inaccuracies it brings in initialisation. This is the primary novelty of this work.

The methodology has now obtained a set of deterministic surrogate constraints for joint chance constraint satisfaction. Identification of these expressions enables

reformulation of the original problem statement $\mathcal{P}(\cdot)$ described by Eq. 4.6 as follows:

$$\hat{\mathcal{P}}(\pi_C) := \begin{cases} \max\limits_{\pi} J \\ \text{s.t.} \\ \mathbf{x}_0 \sim p(\mathbf{x}_0) \\ \mathbf{x}_{t+1} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\upsilon}_t), \Sigma(\boldsymbol{\upsilon}_t)) \\ \mathbf{u}_t \sim \pi(\mathbf{u}_t|\mathbf{x}_t) \\ \mathbf{u}_t \in \hat{\mathbb{U}} \\ \mathbf{x}_t \in \bar{\mathbb{X}}_t \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{4.22}$$

where $\boldsymbol{\upsilon}_t = \begin{bmatrix} \mathbf{x}_t^T & \mathbf{u}_t^T \end{bmatrix}^T$ and solution to $\hat{\mathcal{P}}(\cdot)$ is equivalent to that of the original $\mathcal{P}(\cdot)$. Due to the presence of a GP state space model within the problem description, $\hat{\mathcal{P}}$ is a function space optimisation problem. Previous works have solved this problem via nonlinear MPC with precalculation of the backoff values, and description of the discrete time state evolution according to the mean of the GP (equivalent to the nominal process) (Bradford et al., 2020). In this work, we use RL to solve $\hat{\mathcal{P}}$ (hence the use of function realisations and Algorithm C.1, detailed by C.1.2) with incorporation of the deterministic surrogate constraints into a modified RL objective. This is achieved via an l_p penalty function, under a given value of the backoff multipliers, $\boldsymbol{\xi}$. Solution to this problem under the optimal backoff multipliers $\boldsymbol{\xi}^*$ is deemed equivalent to finding solution to Eq. 4.22 as discussed subsequently.

Safe constrained policy optimisation with fixed backoffs

As GPs express process uncertainties, they present an avenue to synthesise policies, which only exploit regions of the state space in which the model is confident of the true process behaviour i.e. where epistemic uncertainties are low. By incorporating the variance prediction, $\Sigma(\boldsymbol{\upsilon}_t)$, of the GP state space model posterior directly in the RL performance index, we force the ultimate RL policy to avoid the areas that the GP is uncertain and provide explicit mechanism to mitigate exploitation of the mathematical nature of the GP model. Hence the policy *pessimistically accounts for the limitations of the data-driven model* when deployed to the real process.

Use of the l_1 or l_2 penalty functions is particularly appealing because of the exactness (under certain conditions) to the solution of $\hat{\mathcal{P}}$ (Nocedal and Wright, 2006). This would further preserve the approximation provided by Eq. 4.22 to Eq. 4.6. The general penalty function, $\boldsymbol{\varphi}_p : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}$ is detailed as follows:

$$\boldsymbol{\varphi}_p(\mathbf{x}, \mathbf{u}, t) = R_{t+1} - \mathbf{tr}\big(\zeta \Sigma[\boldsymbol{\upsilon}_t]\big) - \kappa \left\| [A^T \mathbf{x}_{t+1} + \boldsymbol{\varepsilon}_t - b]^- \right\|_p \tag{4.23}$$

where $A \in \mathbb{R}^{n_x \times n_g}$ and $b \in \mathbb{R}^{n_g}$ define the set of inequality constraints; $\boldsymbol{\varepsilon}_t \in \mathbb{R}^{n_g}$ the set of backoff values relevant to the set of constraints at a given time index (see Eq. 4.21); $[\mathbf{z}]^- = \max(0, \mathbf{z})$ defines an element wise operation over $\mathbf{z} \in \mathbb{R}^{n_g}$; $\|\cdot\|_p$ the general p-norm; $R_{t+1} \in \mathbb{R}$ the rewards accumulated under the original process objective e.g. productivity maximisation in a (bio)chemical process; and, $\kappa \in \mathbb{R}$ and $\zeta \in \mathbb{R}^{n_x \times n_x}$ (a diagonal matrix) weight the penalty for constraint violation and model uncertainty, respectively - relative to R_{t+1}. The incorporation of model uncertainty, therefore, is represented by the term $\mathbf{tr}\big(\zeta \Sigma[\boldsymbol{\upsilon}_t]\big)$. It is expected that in some cases there is likely to be a dependence between the uncertainty and constraint penalty terms, which may lead to over-penalisation of constraint violations. This may favour the identification of conservative policies, although this is likely to be case dependent and may be mitigated by the tuning process discussed in Section 4.3.2. Expression of the penalty function, enables redefinition of the RL objective $J(\boldsymbol{\tau})$ via $\bar{J}_C(\boldsymbol{\tau})$:

$$\begin{aligned} \bar{G}_C(\boldsymbol{\tau}) &= \sum_{t=0}^{T-1} \gamma^t \boldsymbol{\varphi}_p(\mathbf{x}, \mathbf{u}, t) \\ \bar{J}_C &= \int p(\boldsymbol{\tau}) \bar{G}_C(\boldsymbol{\tau}) d\boldsymbol{\tau} \end{aligned} \tag{4.24}$$

It is hypothesised that RL-based optimisation of this new objective will synthesise a policy, which provides chance constraint satisfaction and exploits regions of the state space well characterised by the model - encouraging the learning of inherently safe control policies. Further, because the modifications are made directly to the reward function itself, the approach is compatible with any RL method. Given the GP state space model is constructed over continuous state and control variables, as usual, the RL can learn a parameterisation of the optimal constrained policy:

$$\begin{aligned} \pi_C^*(\mathbf{u}|\mathbf{x}; \theta, \cdot) &\approx \pi_C^*(\mathbf{u}|\mathbf{x}) \\ \pi_C^*(\cdot, \theta) &= \arg\max_{\theta} \bar{J}_C \end{aligned} \tag{4.25}$$

where $\theta \in \mathbb{R}^{n_\theta}$ denotes a vector representation of the policy parameters (typically the weights and bias of a neural network). A general algorithm for constrained policy optimisation under a fixed set of backoff values is provided by Algorithm 4.1. These backoffs are adjusted via BO - details are presented later in the manuscript in Section 4.3.2 and Algorithm 4.2.

Algorithm 4.1 Safe Policy Optimisation for Fixed Backoffs

Input: Experimental dataset $\mathcal{D}$; GP state space model $f_{GPSS} = [f_{GP}^1(\boldsymbol{\upsilon}), \ldots, f_{GP}^{n_x}(\boldsymbol{\upsilon})]$ with hyperparameters $\hat{\Lambda} = [\hat{\boldsymbol{\lambda}}_1, \ldots, \hat{\boldsymbol{\lambda}}_{n_x}]$ trained on $\mathcal{D}$; Initial control policy $\pi(\mathbf{u}|\mathbf{x}; \theta_0)$; Policy optimisation algorithm $f_{PO}(\cdot)$; backoff multipliers $\boldsymbol{\xi}$; Finite horizon length T; initial state distribution $p(\mathbf{x}_0)$; Memory $\mathcal{B}_{info}$ for information required for $f_{PO}(\cdot)$; K episodes; tolerance criterion;

1. $i = 0$

while not converged **do**

2a. Obtain a batch of K rollouts over a horizon of T discrete intervals according to Algorithm C.1, via $\pi(\mathbf{u}|\mathbf{x}; \theta_i)$, f_{GPSS}, and $p(\mathbf{x}_0)$. Return the trajectory information[3] of each rollout and any further necessary information for $f_{PO}(\cdot)$ and store in $\mathcal{B}_{info}$.

2b. Perform policy optimisation $\theta_{i+1} = f_{PO}(\mathcal{B}_{info}, \theta_i)$

2c. Reset memory $\mathcal{B}_{info}$

2d. i += 1

2e. Assess tolerance criterion

end while

3. Assess final policy performance $J(\theta_i)$ under the unconstrained reward function R and approximate the probability of joint constraint violation (Eq. 4.15) denoted $F_{LB}(0)$ via the method detailed in Appendix C.4

Output: Optimal constrained policy $\pi_C^*(\mathbf{u}|\mathbf{x}; \theta_i)$ under backoff multipliers $\boldsymbol{\xi}$ and associated performance indices $J(\theta_i)$ and $F_{LB}(0)$

The description provided by Algorithm 4.1 considers all *on-policy* policy optimisation approaches, denoted generally as $f_{PO}(\cdot)$, although there is no reason the approach could not utilise an off-policy method too (Pan et al., 2020). The detail provided formalises the process of obtaining function space realisations from the GP state space model, f_{GPSS}, each of which represents a potential instance of the uncertain process

[3]This includes the rewards $\varphi_{0:T-1}^{(k)} = [\varphi_1^{(k)}, \ldots, \varphi_{T-1}^{(k)}]$ under Eq. 4.23 and the current backoff multipliers $\boldsymbol{\xi}$, for the sequence of controls $\mathbf{u}_{0:T-1}^{(k)} = [\mathbf{u}_1^{(k)}, \ldots, \mathbf{u}_{T-1}^{(k)}]$ and states $\mathbf{x}_{0:T}^{(k)} = [\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_T^{(k)}]$

detailed by Eq. 4.1. Every process trajectory is ranked according to Eq. 4.23 and the current iterate of backoff multiplier, $\boldsymbol{\xi}$, values. Using the collected experience (including relevant information that describes decision making), stored in the memory, $\mathcal{B}_{info}$, the weights of the policy are updated by $f_{PO}(\cdot)$. This is repeated until a convergence criterion is satisfied. In the following computational experiments detailed by this work, *the methodology was integrated with the proximal policy optimisation (PPO) algorithm* (Schulman et al., 2017b). This is an attractive option given: a) the ability to directly parameterise a policy as a conditional probability distribution over a *continuous* control input space; b) compatibility with recurrent neural networks (Schulman et al., 2017b); c) sample efficiency relative to conventional policy optimisation methods i.e. *reinforce*; and, d) ease of implementation. Full detail of the PPO algorithm is provided by C.3.

In this section, the methodology has provided a mechanism to incorporate information about the constrained problem into the reward signal characteristic of the MDP. In doing so, a strong connection to reward shaping is established. In reward shaping, policy invariant modifications of the reward function are identified to aid learning of the optimal policy π^* (Ng et al., 1999; Dong et al., 2020). In this work, we construct a policy varying modification of the reward function in order to satisfy operational constraints. The resultant penalty function (Eq. 4.23) contains a number of free parameters. In the subsequent section, it is proposed to tune the backoffs, $\boldsymbol{\varepsilon}$, via the multipliers, $\boldsymbol{\xi}$, (see Eq. 4.21) and a BO scheme. This leaves decision as to the parameters $\kappa \in \mathbb{R}$ and $\zeta \in \mathbb{R}$ open to the implementation, although it is recommended that they are large real values (Nocedal and Wright, 2006). Ultimately this provides mechanism for the implementation to balance operational risk and performance manually.

Optimisation of backoff multipliers

The primary objective of this section is to identify a mechanism which facilitates synthesis of a policy that:

(a) achieves high probability constraint satisfaction as desired, and

(b) performs with respect to the original process objective as specified by $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}$.

'Efficient' global optimisation of the backoff multipliers $\boldsymbol{\xi}$ is proposed, and so the methodology explores definition of an objective to evaluate candidate values of $\boldsymbol{\xi}$ as follows.

Firstly, discussion is directed in how best to evaluate a) from the policy generated by Algorithm 4.1. Specifically, with reference to Eq. 4.17 and the following works (Paulson et al., 2020; Petsagkourakis et al., 2020a):

$$F_X(0) = \mathbb{P}(X \leq 0) = \mathbb{P}(\bigcap_{i=0}^{T} \{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}) \tag{4.26}$$

where $F_X(\cdot)$ indicates the cumulative distribution function (cdf), which in this case is analytically intractable. In order to assess a), it is proposed to validate the probability of constraint satisfaction empirically via MC sampling under the GP state space model i.e. via the sample approximation of $F_X(0)$, denoted $F_{SA}(0)$. The specific approach is detailed in Petsagkourakis et al. (2020a) and repeated in C.4 for completeness. Ultimately, through this sampling-based method, a lower bound for Eq. 4.26 and $F_{SA}(0)$ is obtained and denoted $F_{LB}(0)$. This accounts for potential inexactness introduced through finite samples. The evaluation of b) is more simple and directed via the definition of the process objective in the form of the reward function R. As such, the investigation may evaluate the performance of the policy under the original, unconstrained objective provided by Eq. 4.4. This work therefore proposes the use of the following objective function in evaluation of candidate multiplier values $\boldsymbol{\xi} \in \mathbb{R}^{n_g}$:

$$\begin{aligned} \boldsymbol{U} &= (F_{LB}(0) - (1-\alpha))^2 \\ J_{BO} &= -(J(\boldsymbol{\tau}) - \beta\sigma_J)\exp(-c\boldsymbol{U}) \end{aligned} \tag{4.27}$$

where $\beta = [0, 1]$, $c \in \mathbb{R}^+$ and σ_J denotes the standard deviation of the policy with respect to the unconstrained process objective. This is a modification to the objective function proposed previously in Petsagkourakis et al. (2020a), which equated J_{BO} to $\boldsymbol{U}$. Here, Eq. 4.27, provides a smoother latent function and naturally balances the objectives a) and b). The inclusion of the term σ_J also incentivises those policies, which exploit regions of the state space well characterised by the model. The factor c provides a shape parameter for the RBF part of the objective, with higher values providing greater incentive to obtain joint constraint satisfaction as desired. Care should be taken in selection as the higher the value, the sparser the mapping provided by the objective. This is likely to have consequences for the efficacy of optimisation.

In the following case studies, $\beta = 0.1, c = 1$ and a BO scheme was deployed via GP surrogate models with RBF covariance functions and zero mean priors to optimise the backoff multipliers $\boldsymbol{\xi}$:

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} J_{BO} \tag{4.28}$$

Due to the expensive black box optimisation proposed BO is deemed the most appropriate approach. BO proceeds to construct and exploit a GP surrogate model to sample new candidate points. Construction of the GP surrogate demands a small initial dataset, describing a set of inputs, $\boldsymbol{\Xi}$, and their corresponding fulfilment of the objective function, $J_{\boldsymbol{\Xi}}$. New sampling points (or in this case, candidate backoff multipliers) are sampled to maximise an acquisition function (AF), which is a function of the posterior distribution of the GP surrogate. The AF, denoted $f_{AF}(\cdot)$, used in this work was the expected improvement (EI) function (Frazier, 2018; Jones et al., 1998). It was found that the EI AF, $f_{AF}^{EI}(\cdot)$, was most efficient in this case, balancing exploration and exploitation of the GP surrogate model to find the optimal solution. The expected improvement function is detailed as follows (Jones et al., 1998; Brochu et al., 2010):

$$\varrho = \frac{\mu(\boldsymbol{\xi}) - J_{BO}^{+}}{\sigma(\boldsymbol{\xi})}$$

$$f_{AF}^{EI}(\boldsymbol{\xi}) = \begin{cases} \sigma(\boldsymbol{\xi})\psi(\varrho) + (\mu(\boldsymbol{\xi}) - J_{BO}^{+})\phi(\varrho), & \text{if } \sigma(\boldsymbol{\xi}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.29}$$

where $\phi(\cdot)$ is the Gaussian cumulative distribution function, $\psi(\cdot)$ is the Gaussian probability density function, J_{BO}^{+} is the objective value of the current best backoff multiplier values $\boldsymbol{\xi}^{+}$ (Jones et al., 1998; Frazier, 2018), and $\mu(\cdot)$ and $\sigma(\cdot)$ are detailed by Eq. 4.13. Dissecting Eq. 4.29, the first term on the right hand side incentivises exploring regions of the input space associated with high uncertainty in the posterior distribution, and the second term provides basis to exploit regions of the input space corresponding to high mean predictions in the posterior (Jones et al., 1998). As such, Eq. 4.29 provides explicit mechanism to balance exploration and exploitation of the GP surrogate model, in a fashion not dissimilar to the exploration-exploitation paradigm in RL. For more detail on BO in this context, we direct the interested reader to previous work (Petsagkourakis et al., 2020a; del Rio Chanona et al., 2021) and a comprehensive review (Frazier, 2018).

Algorithm 4.2 formalises the approach to reward shaping detailed by this Section. In **Step 1**, an optimal policy parameterisation is learned for the unconstrained problem. This is used as an initialisation for learning of the optimal constrained policy thereafter. In **Step 2.a**, a number of policies are learned for the constrained problem each utilising different values of the backoff multipliers. In **2.b**, each of the policies is assessed with respect to J_{BO}, providing an input-output dataset, where the inputs are backoff multipliers and the outputs are corresponding performances under the objective (J_{BO}). In **Step 3**, a surrogate GP model is built via this input-output dataset for subsequent BO. Pseudocode for BO is provided by **Step 4**, with **Step 5** documents the return of the solution policy from memory. In the following section, the method is demonstrated on a microalgal lutein photo-production dynamic process and benchmarked against dynamic optimisation and NMPC strategies.

Algorithm 4.2 Safe Chance Constrained Policy Optimisation

Input: Desired probability of joint chance constraint satisfaction, α; GP prior for Bayesian Optimisation f_{BO}; Acquisition function f_{AF}; Objective function J_{BO}; maximum number of acquisitions for BO, M; Initial set of B backoff multiplier values $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_B]$ generated via sobol sequence (Sobol', 1967);

1. Perform policy optimisation for unconstrained problem to maximise Eq. 4.4 via modification to Algorithm 2. Return policy $\pi^*(\cdot, \theta)$.

2i. Train a set of B constrained policies $\pi^*_{init} = [\pi^*_C(\cdot, \theta_1), \ldots, \pi^*_C(\cdot, \theta_B)]$ to maximise Eq. 4.24 under the respective backoff values, $\boldsymbol{\Xi}$, via Algorithm 2 with $\pi^*(\theta)$ as initialisation.

2ii. Return performance indices $F_{LB}(0)$ and $J(\boldsymbol{\tau}, \theta)\ \forall\ \pi^*_C(\cdot, \theta) \in \pi^*_{init}$ and assess J_{BO}, such that $J_{\Xi} = [J_{BO}(\boldsymbol{\xi}_1), \ldots, J_{BO}(\boldsymbol{\xi}_B)]$.

3. Train a GP model given input-output pairs representative of (backoff multiplier values and policy performance under J_{BO}) $\boldsymbol{\Xi}$ and $J_{\boldsymbol{\Xi}}$ according to C.1.1 and condition to obtain updated predictive posterior distribution, $p(J_{BO}|\boldsymbol{\xi}, \boldsymbol{\Xi})$.

for $m \in \{1, \ldots, M\}$: **do**

4i. According to $p(J_{BO}|\boldsymbol{\xi}, \boldsymbol{\Xi})$ find $\boldsymbol{\xi}_{B+m} = \arg\max_{\boldsymbol{\xi}} f_{AF}(\cdot)$ and update $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{B+m}]$

4ii. Train constrained policy $\pi^*_C(\cdot, \theta_{B+m})$ via Algorithm 2, $\pi^*(\cdot, \theta)$ for initialisation under the backoff values $\boldsymbol{\xi}_{B+m}$. Return performance indices $F_{LB}(0)$ and $J(\boldsymbol{\tau}, \theta_{B+m})$ for $\pi^*_C(\cdot, \theta_{B+m})$, assess $J_{BO}(\boldsymbol{\xi}_{B+m})$ and append to dataset, $J_{\Xi} = [J_{BO}(\boldsymbol{\xi}_1), \ldots, J_{BO}(\boldsymbol{\xi}_{B+m})]$

4iii. if $m < M$: repeat step **3.**

end for

5. Return $\pi^*_C(\theta)$ corresponding to $\boldsymbol{\xi}^* = \arg\max_{\boldsymbol{\xi}} J_{\boldsymbol{\Xi}}$

Output: Optimal Constrained Policy $\pi^*_C(\theta)$

4.4 Case Study

To demonstrate the methodology, a case study was selected from previous work conducted by Zhang et al. (2019a); del Rio-Chanona et al. (2017), which is underpinned by a set of ordinary differential equations (ODEs). The problem and standard benchmarks are detailed via the following subsections.

4.4.1 A microalgal lutein photo-production dynamic process

Fed-batch fermentation processes are thought to be ideal systems for RL-based controllers and particularly suited to data-driven approaches to control and optimisation. This is due to characteristics of predominantly batch mode operation and complex physical phenomena driven by the metabolic reaction network. The complexity of the process physics often provides impediment to structural and practical model identification, with large parametric uncertainties common across bioprocess systems. To demonstrate the method proposed here, we consider an *in-silico* microalgal lutein photo-production process described as follows:

$$\begin{aligned}
\dot{c}_X &= u_0 \frac{c_N}{c_N + K_N} c_X - u_d c_X \\
\dot{c}_N &= -Y_{N/X} u_0 \frac{c_N}{c_N + K_N} c_X + F_{N,in} \\
\dot{c}_L &= k_0 \frac{c_N}{c_N + K_{NL}} c_X - k_d c_L c_X
\end{aligned} \tag{4.30}$$

where c_X $(g\ L^{-1})$ defines the biomass concentration; c_N $(mg\ L^{-1})$ defines the nitrate concentration; c_L $(mg\ L^{-1})$ defines the lutein (product) concentration; $F_{N,in}(mg\ h^{-1})$ is the nitrate inflow to the system (a control input); $u_0 \in \mathbb{R}$ (h^{-1}) is the specific biomass growth rate, which is a function of the incident light intensity $I_0 \in \mathbb{R}$ $(\mu mol\ m^{-2}\ s^{-1})$ to the reactor and the maximum theoretical growth rate $u_m \in \mathbb{R}$ (h^{-1}); $k_0 \in \mathbb{R}$ $(mg\ g^{-1}\ h^{-1})$ is the specific lutein production rate, which is a function of I_0 and the maximum theoretical production rate $k_m \in \mathbb{R}$ $(mg\ g^{-1}\ h^{-1})$; $k_d \in \mathbb{R}$ $(L\ g^{-1}\ h^{-1})$ is the lutein consumption rate; $u_d \in \mathbb{R}$ (h^{-1}) is the biomass specific decay rate; $Y_{N/X} \in \mathbb{R}$ $(mg\ g^{-1})$ is the nitrate yield coefficient; and, $K_N \in \mathbb{R}$ $(mg\ L^{-1})$ and $K_{NL} \in \mathbb{R}$ $(mg\ L^{-1})$ are the nitrate half-velocity constant for cell growth and lutein synthesis, respectively. The growth rates of biomass and lutein are constituted by the terms u_0 and k_0. These are both functions of the incident light intensity to the reactor and are detailed as follows:

$$\begin{aligned}
u_0 &= \frac{u_m}{20} \sum_{n=1}^{9} \left(\frac{I_0}{I_0 + k_s + \frac{I_0^2}{k_i}} + 2 \frac{I_{\frac{nL}{10}}}{I_{\frac{nL}{10}} + k_s + \frac{I_{\frac{nL}{10}}^2}{k_i}} + \frac{I_L}{I_L + k_s + \frac{I_L^2}{k_i}} \right) \\
k_0 &= \frac{k_m}{20} \sum_{n=1}^{9} \left(\frac{I_0}{I_0 + k_{sL} + \frac{I_0^2}{k_{iL}}} + 2 \frac{I_{\frac{nL}{10}}}{I_{\frac{nL}{10}} + k_{sL} + \frac{I_{\frac{nL}{10}}^2}{k_{iL}}} + \frac{I_L}{I_L + k_{sL}} + \frac{I_L^2}{k_{iL}} \right)
\end{aligned} \tag{4.31}$$

where $k_i \in \mathbb{R}$ and $k_{iL} \in \mathbb{R}$ are light inhibition terms for biomass growth and lutein synthesis, respectively. Similarly, $k_s \in \mathbb{R}$ and $k_{sL} \in \mathbb{R}$ are light saturation terms for biomass growth and lutein synthesis, respectively. More information on parameter definitions and values is provided by del Rio-Chanona et al. (2017); Zhang et al. (2019a). The states $\mathbf{x} = [c_X, c_N, c_L]$ of the system are absolute and hence $c_i \geq 0 \; \forall \; i \in \{X, N, L\}$. Further to $F_{N,in}$, the control input is also constituted by I_0, such that $n_u = 2$ and $\mathbf{u}(t) = [F_{N,in}, I_0]^T$, with bounds $0.1 \leq F_{N,in} \leq 100 \; mg \; h^{-1}$ and $100 \leq I_0 \leq 1000 \; \mu mol \; m^{-2} \; s^{-1}$. It is assumed that the process is subject to stochasticity in the form of 5 % parametric uncertainty. This and the initial state distribution is detailed by Table 4.1. Parametric values not detailed are assumed constant following the original works (del Rio-Chanona et al., 2017; Zhang et al., 2019a). Additionally, the initial state distribution, $p(\mathbf{x}_0)$, is defined in keeping with the work (Zhang et al., 2019a). However, the state constraints imposed are specific to this work and not the previous.

Table 4.1: Case Study: List of parametric and initial state distributions imposed to describe uncertainty in the real underlying bioprocess.

Variable	Uncertainty Distribution
u_m	$\mathcal{N}(0.152, 0.0038)$
K_N	$\mathcal{N}(30, 0.75)$
u_d	$\mathcal{N}(5.93 \times 10^{-3}, 1.483 \times 10^{-4})$
$Y_{N/X}$	$\mathcal{N}(305, 7.625)$
k_m	$\mathcal{N}(0.35, 0.00875)$
K_d	$\mathcal{N}(3.71 \times 10^{-3}, 9.275 \times 10^{-5})$
$\mathbf{x}_0$	$\left[\mathcal{N}(0.27, 3.125 \times 10^{-3}), \mathcal{N}(765.0, 9.5625), \mathcal{N}(0.0, 0.0)\right]$

Here, affine constraints are defined using notation from Section 4.3.2:

$$A = \begin{bmatrix} 1 & 0 & -1.67 \\ 0 & -1 \times 10^{-3} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 2.6 \\ 0.15 \\ 0 \end{bmatrix} \tag{4.32}$$

The constraints were constructed to represent common operational concerns in bioprocessing. The first column of A considers the potential raw material to product conversion via constraint of the maximum biomass concentration (as biomass is a 'by-product'). The second column considers the protection of cell growth (via a minimum nitrate constraint) and the third ensures continued productivity (via constraint of the maximum ratio of secondary metabolite to biomass). The process objective reward

function $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \rightarrow R_{t+1}$ is as follows:

$$R_{t+1} = \begin{cases} \mathbf{d}^T \mathbf{x}_{t+1} - \Delta \mathbf{u}_t^T C \Delta \mathbf{u}_t & \text{if } t = T-1 \\ -\Delta \mathbf{u}_t^T C \Delta \mathbf{u}_t, & \text{otherwise} \end{cases} \tag{4.33}$$

where $t = [0, \ldots, T]$ and the length of the finite horizon is defined $T = 6$; $\Delta \mathbf{u}_t = \mathbf{u}_t - \mathbf{u}_{t-1} \in \mathbb{R}^{n_u}$ defines the change of controls between discrete time steps; $C = diag([0.16, 8.1 \times 10^{-5}]) \in \mathbb{R}^{n_u \times n_u}$ provides a penalty for changing the controls and promotes the learning of 'stable' control profiles; and, $d = [0, -0.001, 4]^T \in \mathbb{R}^{n_x}$ provides an overall objective for process operation i.e. to maximise the production of lutein and minimise waste of nitrate. The problem definition is common to both Section 4.4.2 and the benchmark described in Section 4.4.3, except the benchmark does not consider any form of parametric uncertainty. A formalisation of the control problem follows:

$$\mathcal{P}(\pi_C) := \begin{cases} \max\limits_{\pi_C} \mathbb{E}_{\pi_C} \left[\sum\limits_{t=0}^{T-1} R_{t+1} \right] & \text{(see Eq. 4.33)} \\ \text{s.t.} & \\ \mathbf{x}_0 \sim p(\mathbf{x}_0) & \\ \mathbf{s}_t \sim p(\mathbf{s}) & \text{(see Table 4.1)} \\ \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t) & \text{(see Eqs. 4.30 and 4.31)} \\ \mathbf{u}_t = \pi_C(\mathbf{x}_t) & \\ \mathbf{u}_t \in \hat{\mathbb{U}} & \\ \mathbf{x}_t \in \hat{\mathbb{X}}_t & \text{(see Eqs. 4.7 and 4.32)} \\ \forall t \in \{0, ..., T-1\} & \end{cases} \tag{4.34}$$

4.4.2 Safe chance constrained policy optimisation

To demonstrate the methodology, this work deploys the PPO algorithm with both actor and critic recurrent long-short term memory (LSTM) neural network parameterisations. The actor network expresses a mapping between observed states and controls (i.e. a control policy) and the critic provides a mapping between a state and the value of that state under the policy (this is known as the value function). The use of a critic provides means to deploy the general advantage estimate (GAE) form of the

policy gradient (PG) within the PPO framework. The GAE enables the implementation to manually balance the bias and variance of the advantage PG. This provides means to synchronously ensure stable learning, improve the sample efficiency of the algorithm and find a clear direction (in weight space) for policy improvement. For more information on PPO and the GAE, the reader is directed to C.3 and Schulman et al. (2017a,b). The implementation utilised Pytorch 1.7.1. Information about the structure of the actor, critic and all hyperparameters defining the PPO algorithm as used in this work, may be found in C.3.3. See Table 4.2 for definition of general case study parameters.

Table 4.2: Case Study: List of key algorithm parameters

Variable	Value
Penalty weight, κ	34
Uncertainty penalty weight, ζ	$300 \times diag([1/\sigma^2_{1\Upsilon}, \ldots, 1/\sigma^2_{n_x\Upsilon}])^4$
Tolerance criterion	$\lvert\bar{J}_C(\boldsymbol{\tau}, \theta_i) - \bar{J}_C(\boldsymbol{\tau}, \theta_{i-1})\rvert \leq 10^{-3}$
Joint Probability of constraint violation, α	0.001
Probability of individual constraint violation, ι_j	0.00033

In order to train the desired policy $\pi^*_C(\cdot, \theta)$ via Algorithm 4.2, a GP state space model is required. In this work, the model was built using an initial dataset $\mathcal{D}$[5], generated by simulation of the uncertain process' response to 32 different control sequences, $\mathbf{u}^{(j)}_{0:T},\ j = [1, \ldots, 32]$ (hence the dataset contains information from 32 separate batch experiments). Each control sequence was generated via transformation of a Sobol sequence (of length T) to the bounded controls space (as detailed in Section 4.4.1). In practice, this dataset could be generated via an initial design of experiments (Petsagkourakis and Galvanin, 2020). Having generated $\mathcal{D}$, ($n_x = 3$) individual GP models were constructed to form a state space model (for the prediction of each state) via the methodology outlined in Section 4.3.1. A prior distribution with mean function $\mathbf{m}(\boldsymbol{\upsilon}) = 0$ and a matern 5/2 covariance function was specified for each of the constituent models. The covariance function was selected according to preliminary experiments, which examined the model's predictive accuracy. All GPs were constructed with the GPy 1.9.9 python package and subsequent BO utilised GPyOpt 1.2.6. Details

[4] $\sigma^2_{n_x\Upsilon}$ represents the variance of the distribution of state x_{n_x} in the dataset $\mathcal{D}$

[5] The dataset used for model construction may be found at https://github.com/mawbray/Lutein-Dataset

of the data used for model construction, as well as metrics relating to the predictive accuracy of the model are detailed in C.2.

In the presentation of results for this work, the investigation is concerned with two main questions. Firstly, does Algorithm 4.2 enable identification of a reward function, which provides policy performance with respect to the process objective and probabilistic constraint satisfaction? And, secondly, does the incorporation of the posterior variance prediction of the GP state space model (into the reward function (Eq. 4.23)) provide means to minimise the risk of policy deployment (to the real uncertain process), by ensuring the policy exploits regions of the model with small model-process mismatch? These two questions will direct discussion in Section 4.5. All results were generated under view of the policy as deterministic i.e. $\mathbf{u}_t = \pi(\mathbf{x}_t)$. This was achieved through selection of the control corresponding to the mode of the conditional distribution $\pi(\mathbf{u}|\mathbf{x})$.

4.4.3 Benchmark for process optimisation

The results from the proposed methodology were benchmarked relative to the control profiles generated from a) dynamic optimisation (DO) strategies, and b) nonlinear model predictive control (NMPC). Both a) and b) use the process model detailed by Eq. 4.30. The deterministic form of this model (i.e. with no parametric uncertainty) represents the most accurate deterministic model, which may be built for process prediction and optimisation. Therefore, the controls generated from a) and b) assume that the underlying process is deterministic, and are subsequently validated on the stochastic analogue of the process concerned. As both a) and b) neglect the existence of uncertainty over the parameter values assumed from del Rio-Chanona et al. (2017), validation of the strategies on the stochastic variant of the process (detailed by Section 4.4.1) directly investigates the effects of uncertainty (model-plant mismatch) on performance with respect to the objective and constraint satisfaction. It should be noted that this benchmark is not reflective of the existing state-of-the-art optimisation methods, such as sMPC that similarly consider model uncertainty. The control strategy for a) was generated offline through optimisation of the control inputs to the model detailed in Eq. 4.30. Hence the control policy generated is deterministic and unconditional to online state observation. Conversely. the control strategy for

b) was generated online through perfect state observation as in the RL case. Both benchmarks utilised the orthogonal collocation method and one finite element per control interval (Biegler, 2007; Kelly, 2017) and the IPOPT solver (Wächter and Biegler, 2006). This was facilitated by the Casadi 3.5.1 Python package (Andersson et al., 2012). In the case that a feasible solution could not be found online, the MPC scheme was tuned further with an approximate problem solved to minimise constraint violation. From empirical analysis, this tuning increased the performance of the NMPC scheme. Further information on the approximate problem is available in C.5.

4.4.4 Key performance indicators

In the following section, this work will investigate the utility of the algorithm, and presentation of the results will focus on the ability of the proposed method to find a safe constrained policy $\pi_C^*(\cdot, \theta)$. Explicitly, the policy should exploit regions of the real process state space (i.e. Eq. 4.30), well characterised by the approximating process model (i.e. Eq. 4.14), therefore minimising mismatch between the state distributions simulated under the offline process model and observed under the real uncertain process. This will be demonstrated in two ways. First, via visual comparison as presented figuratively, and secondly via the quantitative metrics (key performance indicators) available to the investigation. Primarily, these metrics are the performance of the policy with respect to the unconstrained process objective $J(\tau)$ (see Eqs. 4.4 and 4.33) and the probability of joint chance satisfaction as evaluated by $F_{SA}(0)$ and $F_{LB}(0)$. The same metrics will be used to evaluate the performance of the benchmarks of DO and NMPC.

4.5 Results and discussion

4.5.1 Results of safe chance constrained policy optimisation

Firstly, the results of Algorithm 4.2 with respect to the approximate offline state space model are displayed by Fig. 4.1. Explicitly, here, we demonstrate the performance of the final policy $\pi_C^*(\cdot, \theta)$ on the *GP state space model*. The results were obtained according to 500 function realisations of $\pi_C^*(\cdot, \theta)$ via Algorithm C.1. Fig. 4.1 a)

expresses a representation of the state evolution $\mathbf{x}_{0:T}$ and Fig. 4.1 b) provides a visualisation of the performance of the policy with respect to the constraints. In Fig. 4.1 a), the average state evolution and an associated confidence interval of one standard deviation for the validation trajectories is represented by a solid line and a shaded region, respectively. It can be seen that the agent learns to maximise the productivity objective - balancing maximisation of the lutein product at the end of the batch with a decrease in the concentration of nitrate left in the system. This is achieved in a manner that accounts for worst case process stochasticity by backing the nominal or expected state trajectory away from the constraint boundary. This is highlighted by Fig. 4.1 b). In particular, the shaded regions indicate 99% confidence intervals for process deviation and the dark blue solid line plot indicates the nominal process. Further, the utility of tuning the backoff multipliers via Algorithm 4.2 is highlighted given that the worst case realisations of process stochasticity do not violate, but approach the constraint boundary very closely. The performance of the policy $\pi^*_C(\cdot, \theta)$ with respect to both process objective and constraint satisfaction on the GP process model is detailed by Table 4.3. The performance of the policy on the process model is however, not the

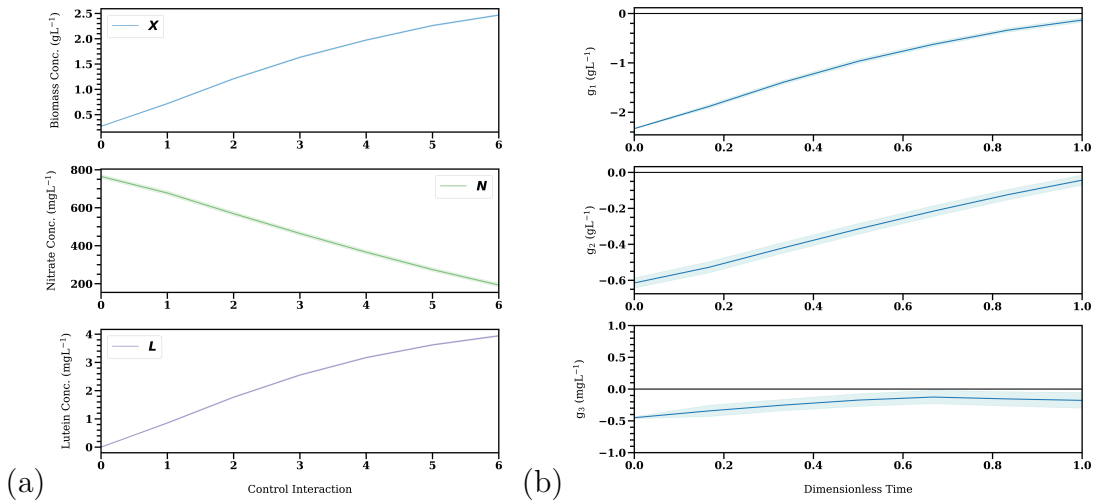


Figure 4.1: Results from Case Study. (a) The state profile produced from the final policy learned on the Gaussian Process model plotted against control interactions (as a proxy for time). Control interactions are provided every 24 hours of process operation. (b) The corresponding distribution of trajectories with respect to the operational constraints. The i^{th} constraint is denoted $g_i := A_i^T \mathbf{x} - b_i$. The light blue shaded areas represent the 99th to 1st percentiles and solid blue line represents the expected trajectory. The black line plot represents the threshold of constraint violation i.e. when $g_i = 0$

primary contribution of this work. Rather, it is of interest to validate the safety of the

policy on deployment to the real stochastic process and highlight the particular use of the training approach detailed. To achieve this, results were obtained by sampling the real process described by Eq. 4.30, with the parametric uncertainty detailed in Section 4.4. The results of this are expressed by Fig. 4.2. Similar to Fig. 4.1 b),

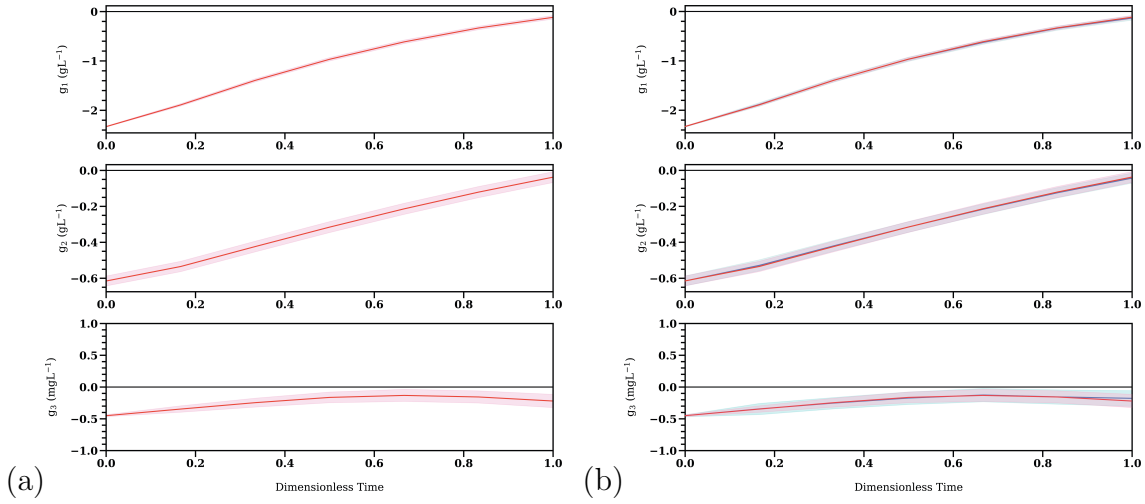


Figure 4.2: Results from Case Study. (a) The distribution of trajectories with respect to the operational constraints as sampled from the real uncertain process. (b) An overlay of the distributions observed when the policy is deployed on the real uncertain process (red) and the GP state space model (blue) as plotted in Fig. 4.1. The i^{th} constraint is denoted $g_i := A_i^T \mathbf{x} - b_i$. The shaded areas represent the 99th to 1st percentiles and solid line represents the expected trajectory. The black line plot represents the threshold of constraint violation i.e. when $g_i = 0$

Fig. 4.2 a) details the performance of the policy with respect to the constraints, but upon deployment to the real uncertain process. Again, the shaded regions indicate 99% confidence intervals of process deviation and the dark blue solid line indicates the nominal process. Fig. 4.2 b) provides comparative detail of the distribution of the trajectories with respect to the constraints when the policy is deployed on the GP state space model (blue) and when deployed to the real uncertain process (red). As previously, the shaded regions indicate 99% confidence intervals for process deviation and the solid line indicates the nominal process. It is observed that there is very little mismatch between the model and real process in this region of the state space and as a result the distributions of the first and second constraint (g_1 and g_2) are almost indistinguishable. Notably, however, there is indeed clear, but small amounts of mismatch between the nominal process trajectories on the GP state space model and the real process as demonstrated via the third constraint plot of g_3. Interestingly,

this plot shows that 99% of the real process trajectories (the red region) are contained within the (blue) region described by the samples from the GP state space model. This indicates the potential that the offline GP model (epistemic) uncertainty, expressed via the variance of the posterior, could be able to provide constraint satisfaction and ensure safe RL policies. Table 4.3 demonstrates the utility of the algorithm in achieving

Table 4.3: Case Study: Comparison of probabilities of joint constraint satisfaction $F_{LB}(0)$ and $F_{SA}(0)$ and objective values of $\pi_C^*(\cdot, \theta)$ as learned via the methodology on the real process and GP state space model. The objective performance is quantified via the mean and variance due to process stochasticity. See Eq. 4.33 for detail of the process objective.

Process	$F_{LB}(0)$	$F_{SA}(0)$	Process Objective (Eq. 4.33)
Offline Gaussian process model	1.0	1.0	15.29 +/- 0.11
Online real uncertain process	1.0	1.0	15.23 +/- 0.096

constraint satisfaction as desired in both the offline model and real uncertain process. There is a small discrepancy between the performances of the two validations. This could be explained either due to the number of finite samples (500) used in assessment of policy performance, or via small amounts of nominal process mismatch between the GP model and the real process. If the latter view is taken and it is assumed the biomass and nitrate states are perfectly predicted (biomass is not included in the objective directly and nitrate is, but weakly), then this difference corresponds to a 0.375% prediction error of the nominal lutein trajectory. In the following sections, the work detailed here is benchmarked against results observed from implementing control policies on the uncertain process determined via a) offline DO and b) NMPC. The approach to generation of these results is discussed in Section 4.4.3.

4.5.2 Comparison to benchmark methods

The benchmark for this case study is provided by DO and NMPC, both of which are common approaches to process control. In the following sections the investigation provides comparative analysis to demonstrate the utility and limitations of the methodology.

Comparison to dynamic optimisation

In order to demonstrate the effects of process stochasticity for dynamic optimisation (DO), control profiles were generated for the system (Eq. 4.30) from four different initial conditions, all of which are probable to be drawn from the initial state distribution detailed in Section 4.4. As previously, all results are derived from 500 realisations of the real uncertain process model. The comparative performance of the DO benchmark is detailed by Table 4.4. From Table 4.4 it is clear that the effects of small amounts of

Table 4.4: Case Study: Comparison of probabilities of joint constraint satisfaction $F_{LB}(0)$ and $F_{SA}(0)$ and objective values of $\pi_C^*(\cdot, \theta)$ under the proposed dynamic optimisation (DO) benchmark. Four different results are reported for DO, corresponding to the four different initial conditions used to generate the control profile offline. The objective performance is quantified via the mean and variance due to process stochasticity. See Eq. 4.33 for detail of the process objective.

Algorithm	Initial Conditions $\mathbf{x}_0$	$F_{LB}(0)$	$F_{SA}(0)$	Process Objective $J(\tau)$
DO I	[0.276, 784, 0.0]	0.036	0.056	16.68 +/- 0.24
DO II	[0.273, 774, 0.0]	0.046	0.068	16.68 +/- 0.25
DO III	[0.270, 765, 0.0]	0.030	0.048	16.65 +/- 0.25
DO IV	[0.267, 755, 0.0]	0.043	0.064	16.61 +/- 0.25
Proposed	$\mathbf{x}_0 \sim p(\mathbf{x}_0)$	1.0	1.0	15.23 +/- 0.096

stochasticity have dramatic implications for the probability of joint chance constraint satisfaction for DO. Both the statistically robust $F_{LB}(0)$ and the sample approximate $F_{SA}(0)$ are less than 0.06 for all DO control profiles. This highlights the utility of the method proposed in accounting for process stochasticity. It is also necessary to comment on the standard deviation of the performance with respect to the process objective as reported. The RL policy trained by the method achieves a lower variance in performance than that of the DO scheme. This is worth discussion as it highlights the ability of RL policies to naturally account for process stochasicity in a closed loop feedback control manner. Whereas, the variance of performance reported for the DO strategies is similar across all results and expresses the effects of process stochasticity on an open loop nominal (and deterministic) control policy. However, it is also important to note that although the RL method proposed performs with respect to constraint satisfaction, it does not achieve as well as DO with respect to the expected unconstrained process objective $J(\tau)$. This is mainly because backing the nominal process away from the constraint boundaries in order to account for variability, will

naturally incur a decrease in the nominal performance of the policy. However, as the process objective function is only reduced by 8% and the constraints are satisfied with high probability, the current approach is still advantageous.

Despite the comparative benefits of RL, it is worth highlighting that the performance is sensitive to correct specification of initial state distribution, $p(\mathbf{x}_0)$, in offline training. Initialising the system in an initial state, $\mathbf{x}_0$, not well described by $p(\mathbf{x}_0)$ will likely lead to deterioration in the performance of the RL policy. Compared to the traditional NMPC approach in which process model can be continuously re-calibrated using online data, other advanced techniques (Wang et al., 2019b) could conceivably be applied given the slow dynamics under consideration in this case study. Although updating RL online is out of current study's scope, it is worth investigating in future work.

Comparison to nonlinear model predictive control

The generation of the NMPC trajectory similarly assumes use of the deterministic variant of Eq. 4.30, as the process model. Here, however, the control policy is updated online via complete observation of the real uncertain process state (as is typical). The initial state is drawn from the initial state distribution detailed in Section 4.4.1, which was also used to train and validate the RL policy $\pi_C^*(\cdot, \theta)$. Table 4.5 reports the respective KPIs for the method proposed and the NMPC scheme. Interestingly,

Table 4.5: Case Study: Comparison of probabilities of joint constraint satisfaction $F_{LB}(0)$ and $F_{SA}(0)$ and objective values of $\pi_C^*(\cdot, \theta)$ under the proposed benchmark of nonlinear model predictive control (NMPC). The objective performance is quantified via the mean and variance due to process stochasticity. See Eq. 4.33 for detail of the process objective.

Algorithm	$F_{LB}(0)$	$F_{SA}(0)$	Process Objective $J(\tau)$
NMPC	0.12	0.148	11.58 +/- 4.07
Proposed	1.0	1.0	15.23 +/- 0.096

with reference to Table 4.5, the method proposed performs better than NMPC with respect to the process objective. In this case, this is primarily due to the destabilisation of NMPC by process stochasitity, which was evidenced by the frequent inability to find control solutions online. This is common when stochastic systems are driven close to constraint boundaries with deterministic methods. The inability of NMPC

to find control solutions online is the primary reason for the difference in objective performance as detailed by Table 4.5 (Note: if solution could not be found, an approximate problem was solved to minimise constraint violation, and this was found to considerably improve performance - see Section 4.4.3 for information). In combination with worst cases of process stochasticity, this provides a skewing of the nominal process performance as reported. Demonstration of the sensitivity of the NMPC control scheme to process stochasticity is best expressed in analysis of the control trajectories generated in validation on the real uncertain process. This is reported by Fig. 4.3.

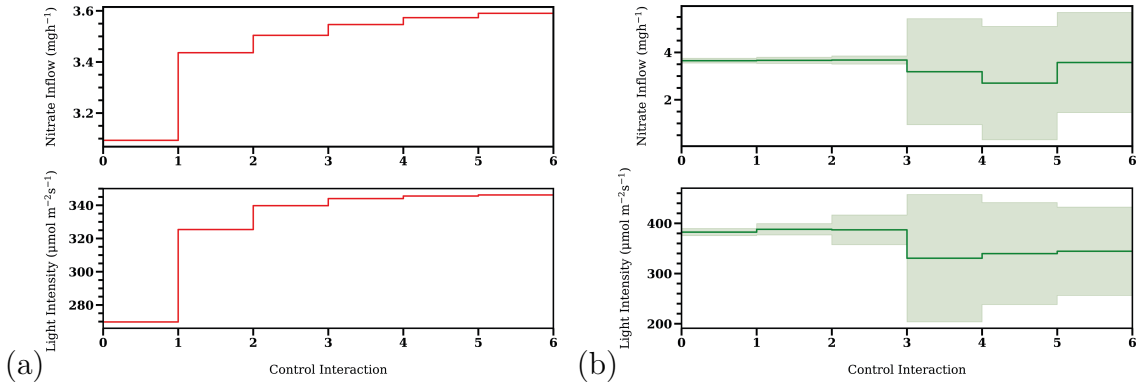


Figure 4.3: Results from Case Study. (a) The distribution of controls selected by the RL policy, $\pi^*_C(\cdot, \theta)$, upon validation on the real uncertain process. Red solid line represents the average control trajectory and the light red shaded region represents a 1 standard deviation confidence interval (which is essentially non-existent), (b) The distribution of controls selected by the NMPC policy upon validation on the real uncertain process. Green solid line represents the average control trajectory and the light red shaded region represents a 1 standard deviation confidence interval

From Fig. 4.3 the relative effect of stochasticity on the NMPC scheme is apparent. Fig. 4.3 a) displays the distribution of controls selected under the RL policy, $\pi^*_C(\cdot, \theta)$, on the real uncertain process. The red solid line represents the average control trajectory and the red shaded region, which is essentially indistinguishable, represents a confidence interval of one standard deviation. Fig. 4.3 b) represents the distribution of controls selected by the benchmark NMPC control policy when validated under the real uncertain process. Here, the green solid line represents the average control trajectory and the green shaded region, which is relatively large, represents a confidence interval of one standard deviation. It is likely that the average control trajectory plotted is not representative of actual control behaviour, i.e. the distribution of controls at each time interval is not best described by a unimodal Gaussian. However, the figure

plotted well expresses the relative variance of controls selected.

From comparison of 4.3 a) and b), it is clear that the RL method proposed naturally accounts for process stochasticity in a closed loop manner, with little variance in the distribution of controls shown. This is characteristic of a control strategy, which is robust to process uncertainty. This is especially beneficial in the context of cell cultivation or fermentation processes, where cell metabolism is sensitive to variation in the environmental conditions. As a result, the oscillatory control behaviour demonstrated by the NMPC control scheme would likely have a detrimental effect on the efficacy of operation/cell metabolism. It should be noted that the detrimental effects of process stochasticity on deterministic control strategies are demonstrated here, with small amounts of uncertainty. In the types of processes of concern to this work, (parametric) uncertainties can be much larger. This further contextualises the benefits provided by the strategy proposed i.e. the ability to simultaneously account for process-model mismatch and constraints.

The results provided by the work provokes the following question: is the primary benefit of the RL method proposed (relative to the NMPC result) derived due to the benefits of accounting for uncertainty in closed loop (i.e. operating within the MDP framework), or due to the description of process uncertainty provided by the GP model? Admittedly, it is difficult to answer this question certainly without further computational experiments and more thorough comparisons; however, through the current study it is believed that both elements are likely to be at play in separating the performance of the proposed method and NMPC. This question provides basis for future empirical studies.

4.6 Conclusion

In this work, an efficient, purely data-driven method has been proposed, which considers the safe deployment of RL policies from the offline training environment (process model) to the real uncertain process. The method also provides approach to ensuring joint chance constraint satisfaction with a set probability. This is facilitated through use of the aleatoric and epistemic uncertainties expressed naturally by Gaussian process models, as well as the concept of constraint tightening. The method was analysed

empirically and benchmarked against two commonly used, deterministic approaches to control and optimisation of fed-batch process systems. It was demonstrated that the presence of even small amounts of process stochasticity may have a destabilising effect on the performance of deterministic methods and their relative probabilities of achieving joint constraint satisfaction. It should be highlighted that the level of parametric uncertainty (5%) expressed in this case study is a common lower-bound to that typically observed in the processes of concern to this work. It is likely that the benefits of this method would be even more apparent in cases where higher uncertainties were present. Therefore, it is thought that the scheme proposed is likely to be competitive with state-of-the-art sMPC approaches that similarly account for model uncertainties. The benefit (or drawback, depending on the context) of RL being that it shifts the computational effort offline, and is therefore much faster online (although slower offline). Further, the formalisation of this approach and the link drawn to reward shaping, enables combination of the method with any RL algorithm - policy optimisation, action-value methods and all that lies inbetween. We hypothesise that once deployed, the policy could be continuously improved offline as the local model is iteratively improved and updated between batches (Rajeswaran et al., 2020). Further, it is possible the method could be adapted to the multi-agent setting for distributed control of fed-batch processes or into the domain of continuous processing (McClement et al., 2021). We do however, assume the availability of an existing dataset, which provides information about the operational region of interest, however this could be developed using available mechanistic models and Petsagkourakis and Galvanin (2020) in a model-based design of experiments. Future work should consider the quantification of uncertainties in the parameterisation of the control function.

Chapter 5

Distributional reinforcement learning for scheduling of chemical production processes

This research item has been submitted to the American Institute of Chemical Engineers (AIChE) Journal. A revised version has been submitted for final decision. The paper is accessible on arXiv via the following reference:

Mowbray, M., Zhang, D. and Chanona, E.A.D.R., 2022. Distributional Reinforcement Learning for Scheduling of Chemical Production Processes. arXiv preprint arXiv:2203.00636.

5.1 Introduction

5.1.1 Online production scheduling: optimisation and simulation

The development of methods for efficient production scheduling of batch processes is an area of significant interest within the domain of process systems engineering and operations research (Sarkis et al., 2021; Kis et al., 2020). There are three main drivers for research within online production scheduling: a) identifying modelling approaches that integrate with the practicalities of online scheduling, b) considering plant uncertainties, and c) handling nonlinearities (Harjunkoski et al., 2014). There is a diverse array of modelling and solution methods for production scheduling problems (Maravelias, 2012). On one hand, modelling approaches can be broadly classified as discrete or continuous time (Floudas and Lin, 2004); on the other hand, optimisation approaches can be characterized as simulation-based or optimisation. The former is generally underpinned by stochastic search algorithms, whereas the latter is dominated via the use of mixed integer programming (MIP). However, industrial practice is generally dominated by the use of heuristics given the challenges outlined in a-c.

Recent works have argued that formulation of the underlying scheduling problem in terms of a discrete-time, state space model alleviates many of the problems associated with the former challenge (i.e. a)) (Gupta and Maravelias, 2017; Subramanian et al., 2012). The benefits of this approach primarily relate to the ease of incorporating state-feedback into the scheduling MIP model and mitigate the requirement for various heuristics to update the model when uncertainty is observed (as may be the case in other approaches e.g. continuous-time models). However, the intuitive benefits inherited from state-space modelling produces scheduling models of a much larger size than continuous-time formulations. This has implications for the solution time and suboptimality can be introduced if the discretization of the time domain is too coarse. Further, considering uncertainty within the model formulation is essentially intractable via MIP solution methods, at least for problems of industrial size. For example, in the case of stochastic MIP, considerable approximations to the scenario tree are required to solve the models (even offline) (Li and Grossmann, 2021). This means the most practical way to consider uncertainty is often via robust formulations. This leads to

smaller model sizes but requires approximation of plant uncertainties via a few finite dimensional, deterministic expressions. Many of these formulations are conservative (Li and Ierapetritou, 2008b), however a more recent work provided in McAllister et al. (2022) has ensured both robustness and performance by exploiting some of the ideas that exist in nonlinear model predictive control. This is clearly a strong contribution to the field, however its implementation requires repeatedly solving an MILP model over the time-horizon. This is something one can actively avoid through use of heuristic methods.

Considering uncertainty is widely acknowledged as an important facet of solution approaches (Li and Ierapetritou, 2008a; Beykal et al., 2022; Tian and Pistikopoulos, 2018). In the context of scheduling, the simulation community has demonstrated strong benefits in handling uncertainty (Oyebolu et al., 2019; Fu et al., 2019b). The high level idea here is generally to compute approximately optimal schedules by evaluating their performance via a Monte Carlo method and a stochastic search algorithm. These approaches have also proven useful in integrating decision-making functions where nonlinearities often arise in the underlying model (Dias et al., 2018). This is primarily because one avoids the formulation of MINLP. However, it should be noted that optimisation approaches fare well with regard to integrated problems when the resultant formulation is mixed integer linear programming (MILP) (Santos et al., 2021a; Charitopoulos et al., 2019a). In addition, simulation based approaches are generally expensive to conduct online because of the sample inefficiency associated with Monte Carlo methods and stochastic search algorithms.

The expense of conducting simulations online has led to the development of Reinforcement Learning (RL) based approaches to online production scheduling (Hubbs et al., 2020a). The primary idea here is to exploit the Markov decision process (MDP) framework to model production scheduling systems as a stochastic decision process. One can then use RL to identify a function approximation of an optimal decision policy for the underlying environment through offline simulations of the model. The policy function can then be deployed online to provide scheduling decisions, instead of online optimisation. This allows the scheduling element to a) consider nonlinearities and uncertainties within the process dynamics, b) use arbitrary descriptions of the uncertainty, and c) identify scheduling decisions in a very short time frame (i.e. on

the order of micro to milliseconds). Despite the additional work required in offline, simulation-based policy learning, these facets are appealing in the context of modern production environments.

5.1.2 Online production scheduling and Reinforcement Learning

Although RL has been demonstrated for sequential decision making in a number of case studies (Caputo and Cardin, 2022b; Lawrence et al., 2022; Yoo et al., 2021a), its application to physical production systems has been relatively limited. This is in part due to the fairly recent establishment of the online production scheduling paradigm (Gupta et al., 2016). For example, Waschneck et al. (2018) applied deep Q networks to optimise a flexible jobshop (i.e. a multipurpose multi-stage production facility), however, the authors provide little information as to how the method proposed accounts for constraints. Further, the approach is benchmarked to common heuristics rather than optimisation formulations. In Palombarini et al. (2018), an RL rescheduling approach is presented to repair and ensure the feasibility of the schedule when subject to realizations of plant uncertainty. However, the method does not consider the optimality of the original schedule and is not benchmarked to an existing method. More recently, Hubbs et al. (2020a) provided an extensive analysis of a Reinforcement Learning approach compared to deterministic and stochastic MIP for a single-stage, continuous production scheduling problem. The RL method is demonstrated to be an appealing solution approach. Despite their achievements, as this is a first proof-of-concept, the case study considered is relatively simple and does not involve precedence constraints or requirements for setup time between the end and start of subsequent operations within a given unit.

In addition, considering constraints is an important step in the development of RL algorithms, given the MDP framework does not provide an explicit mechanism to handle them. There has been much research in other decision-making problems regarding this (Petsagkourakis et al., 2022; Achiam et al., 2017). The presence of precedence and disjunctive constraints in production scheduling provides a challenge of a different nature. Detailed examples of these constraints are explored in Section 5.3.2.

Further, the use of RL poses challenges such as robustness and reliability (Waubert de Puiseau et al., 2022).

Recently, there has been a lot of interest in the development of distributional RL algorithms (Bellemare et al., 2023). Instead of formalizing the objective via expected performance, distributional RL algorithms try to find the optimal policy for other measures of the performance distribution (Bellemare et al., 2023). This enables identification of more risk-sensitive policies and consideration of the tails of the policy performance (Tang et al., 2019a). The proposition of risk-sensitive formulations has been common to mathematical programming for some time (Rockafellar and Uryasev, 2002), however, its use in scheduling problems has been limited (Najjarbashi and Lim, 2019; Chang et al., 2017). In this work, we utilise distributional RL to consider risk-sensitive formulations and low probability, worst-case events. This is particularly important in engineering and business applications, where decision-making is generally risk-averse and catastrophic events are highly disfavoured.

5.1.3 Contribution

In this work, we develop the methodology of Reinforcement Learning to production scheduling by proposing a novel and efficient policy optimisation method that a) handles precedence and disjunctive constraints, b) provides means to identify risk-sensitive policies, and c) is practical and observes stability in learning. In doing so, we improve the reliability and robustness of RL algorithms, but also inherit the advantages of identifying online scheduling decisions in real time from a function, and ease in handling nonlinearity and uncertainty.

To handle a), we present a logic based framework that implements transformations of the RL scheduling decisions to ensure they satisfy those constraints derived from propositional logic (i.e. precedence and disjunctive constraints). This is discussed in detail in Section 5.3.2 and represents a novel application of a concept known as action masking (Kanervisto et al., 2020b). The optimisation of the policy in view of this framework is handled by a stochastic search optimisation algorithm (PSO-SA) which combines particle swarm optimisation (PSO) and simulated annealing (SA) to balance exploitation and exploration of the policy parameters. The use of stochastic search optimisation approaches lend themselves naturally to distributional RL and facilitate b)

and c). Specifically, for b) one can gain unbiased estimates of risk-sensitive measures such as the conditional value-at-risk via a Monte Carlo method. This has particular advantage over other policy gradient and action-value based approaches to distributional RL, which generally make biased approximations to the desired risk-measure to optimise (Tang et al., 2019a) and the quantile function (Dabney et al., 2018b), respectively. Further, the use of stochastic search methods removes the dependence on first-order gradient-based policy learning methods, which are known to lack robustness (Tran et al., 2022). This is partly due to the use of noisy directions for policy improvement, whose evaluation is known to be expensive (i.e. policy gradients and deep Q learning) (Riedmiller et al., 2007a; Nota and Thomas, 2020). However, this is amplified in scheduling problems, primarily because the state value function is not a smooth function of the system state, implying that these approaches are unlikely to converge or observe stable learning dynamics (Zhang et al., 2021). It is worth highlighting that the methodological components proposed have been applied elsewhere (as is the norm in practically all operations research or mathematical programming solution methods), however their combination and deployment has been tailored specifically for production scheduling problems in this work

The proposed method is benchmarked on a classical uncertain production scheduling environment against an MILP approach. The aim of the study is to evaluate the performance of the method in terms of optimality over the production horizon considered and demonstrate the flexibility of the framework proposed. The problem is a multiproduct batch plant with parallel production lines. The case study is a modified version of the original study provided in Cerda et al. (1997) to include processing time and due date uncertainty. Extensive analysis of the results is provided. The rest of this paper is organized as follows: in Section 5.2, we present the problem statement; in Section 5.3 we outline a novel and efficient RL approach to scheduling; in Section 4.4, we outline the details of a case study, the results and discussion of which are presented subsequently in Section 4.5; and, in Section 5.6 we finish with concluding thoughts and plans for future work.

5.2 Problem statement

The scheduling problem is generally subject to both endogenous and exogenous sources of uncertainty. In this work, we focus on the online scheduling of parallel, sequential batch operations in a chemical production plant (Maravelias, 2012). We assume that the state of the plant at a given time index, $t \in \{0, \ldots, T\}$, within a discrete finite time horizon (of length T), is represented by a state, $\mathbf{x}_t \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$, where $\mathbf{x}_t$ can be thought as (but not limited to) current product and raw material inventory, unit availability, and tasks currently being processed. At discrete time steps within the scheduling process of the plant, the scheduler (agent or algorithm who decides the scheduling actions) is able to observe the state of the plant, and select a control action, $\mathbf{u}_t \in \mathbb{U} \subseteq \mathbb{Z}^{n_u}$, which represents an appropriate scheduling decision on the available equipment. The state of the plant then evolves according to the following difference equation:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t) \tag{5.1}$$

where $\mathbf{s}_t \in \mathbb{S} \subseteq \mathbb{R}^{n_s}$ represents a realization of some uncertain plant parameters or disturbance. Eq. 5.1 describes the stochastic evolution of the plant and could be equivalently expressed as a conditional probability density function (CPDF), $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. Here, we identify that the system has the Markov property (i.e. the future state only depends on the current state and control actions) and hence the system may be described as a Markov decision process (MDP).

The aim of the scheduler is to minimise objectives such as makespan (which defines the time to complete all of the required tasks on the available equipment) and the tardiness of product completion. Given these objectives, one can define a reward function, $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}$, which describes the performance of the decisions taken (e.g. the higher the reward, the more profit achieved by the scheduler). Solution methods for MDPs aim to identify a control policy, $\pi : \mathbb{X} \to \mathbb{U}$, whose aim is to maximize the reward:

$$Z = \sum_{t=0}^{T-1} R_{t+1} \tag{5.2a}$$

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\big[Z|X_0 \sim p(\mathbf{x}_0)\big] \tag{5.2b}$$

where $X_0 \in \mathbb{X}$ is the initial state of the plant, which is treated as a random

variable and described by an initial state distribution, $p(\mathbf{x}_0)$. The *return*, $Z \sim p_\pi(z)$, is a random variable that is described according to a probability density function, $p_\pi(z)$, under the current policy, π, because the plant dynamics are subject uncertainty. Generally, exact expressions of $p_\pi(z)$ in closed form are unavailable. Therefore, the solution policy, π^*, (i.e. Eq. 5.2) is evaluated via the sample average approximation.

Operationally, there is a constraint set, $\hat{\mathbb{U}}(\mathbf{x}_t) = \hat{\mathbb{U}}_t^{(1)}(\mathbf{x}_t) \times \hat{\mathbb{U}}_t^{(2)}(\mathbf{x}_t) \ldots \times \hat{\mathbb{U}}_t^{(n_u)}(\mathbf{x}_t)$, where $\hat{\mathbb{U}}^{(l)}(\mathbf{x}_t) \subset \mathbb{Z}$, that defines the available tasks or jobs that may be scheduled in the n_u units at any given time index. This may be defined by the viable sequencing of operations in units; requirements for unit cleaning and maintenance periods; requirements for orders to be processed in campaigns; and that processing of batches must be finished before another task is assigned to a given unit, to name a few. General intuition behind these constraints is provided in Section 5.3.2. These constraints may or may not be functions of uncertain process variables (e.g. if processing times are subject to uncertainties). In essence, we are trying to solve a discrete time finite horizon stochastic optimal control problem (SOCP) of the form:

$$\mathcal{P}(\pi) := \begin{cases} \max\limits_{\pi} \mathbb{E}_\pi\big[Z\big] \\ \text{s.t.} \\ X_0 \sim p(\mathbf{x}_0) \\ \mathbf{s}_t \in \mathbb{S} \subseteq \mathbb{R}^{n_s} \\ \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t) \\ \mathbf{u}_t = \pi(\mathbf{x}_t) \\ \mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t) \subset \mathbb{Z}^{n_u} \\ \forall t \in \{0, ..., T\} \end{cases} \tag{5.3}$$

where $\pi : \mathbb{X} \to \mathbb{U}$ is a control policy, which takes the state as input and outputs a control action, $\mathbf{u}_t$. In practice, mixed integer approaches to the scheduling problem either make large approximations to the SOCP formed (i.e. via stochastic programming) or assume description of a nominal model, neglecting the presence of uncertain variables entirely. The latter approach is especially common when considering problem sizes of industrial relevance. In the following section, we present an efficient and novel methodology to identify a policy π, which provides an approximately optimal solution to the finite horizon SOCP detailed by Eq. 5.3, via RL. Specifically, $\pi(\theta, \cdot)$ is defined by a neural network with parameters, $\theta \in \mathbb{R}^{n_\theta}$. The problem (i.e. Eq. 5.3) is then reduced to identifying the optimal policy parameters, θ^*.

5.3 Methodology

The approach that follows is proposed given the following characteristics of the RL production scheduling problem: a) Eq. 5.3 formulates control inputs (decisions) as discrete integer values, $\mathbb{U} \subset \mathbb{Z}^{n_u}$, that identify the allocation of a task (or job) in a unit at a given time index, and b) one must handle the hard constraints imposed by $\mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t)$.

5.3.1 Identifying discrete control decisions

In this work, we are concerned with identifying a policy, which is suitable for optimisation via stochastic search methods and is able to handle relatively large[1] control spaces. The reasoning behind the use of stochastic search is discussed extensively in Section 5.3.3. However, as stochastic search methods are known not to perform well when the effective dimension of a problem is high[2], there is a requirement for careful construction of the policy function.

We propose to make predictions via the policy function in a continuous latent space, $\mathbf{w}_t \in \mathbb{W} \subseteq \mathbb{R}^{n_u}$, and then transform that prediction to a corresponding discrete control decision, $\mathbf{u}_t$. As a result, the policy function requires an output dimensionality equivalent to that of the control space (i.e. n_u the number of units or equipment items in the context of production scheduling). The transformation could be defined by a stochastic or deterministic rounding policy. For example, the nearest integer function (i.e. round up or down to the nearest integer), denoted $f_r : \mathbb{W} \rightarrow \mathbb{U}$, is a deterministic rounding policy demonstrated implicitly by many of the studies provided in Hubbs et al. (2020b). Both of these transformations are non-smooth and generally assume that the latent space, $\mathbf{w} \in \mathbb{W}$, is a relaxed, continuous equivalent of the original control space. In this work, we implement a nearest integer function approach, $f_r(\mathbf{w}) = nint(\mathbf{w})$.

In the context of this work, this approach is appealing because it minimises the number of output nodes required in the policy function, and as a result, the number of function parameters. This lowers the effective dimensionality of the problem compared

[1]The term "large" is used to indicate a control space with high cardinality, $|\mathbb{U}|$.

[2]Stochastic search is known to be less effective when the number of decision variables is in the order of 1000s, as is common in the neural network function approximations often used in RL.

to other means of control selection (e.g. directly predicting the probability mass of a control in the output of the function, as is common in policy gradients with discrete control spaces[3]). This is key for the use of stochastic search optimisation methods. In the subsequent section, we explore the development of an approach to handling constraints imposed on the scheduling problem.

5.3.2 Constraint handling

Handling the constraints imposed on RL control selection (input constraints) is generally done explicitly (Mowbray et al., 2022a). In the class of problem of concern to this work, the structure of the constraints on the control space arises from standard operating procedures (SOPs). SOPs can generally be condensed into propositional logic. For example, consider the special case of a production environment with one processing unit available and two tasks i and m that require processing. We may define a sequencing constraint through the following statement: "task i can only be processed after task m". In the language of propositional logic, this statement is associated with a *True* or *False* boolean. If task m has just been completed and the statement is *True* (i.e. task i can succeed task m), then task i belongs to the constraint set at the current time index t, i.e. $i \in \hat{\mathbb{U}}_t$. This is known as a precedence constraint. More complex expressions can be derived by relating two or more propositional statements. For example, consider the following expression that relates requirements for starting, T^s, and end times, T^f, of tasks i and m in a production environment with one unit:

$$\{T_i^s \geq T_m^f\} \vee \{T_m^s \geq T_i^f\}$$

where $\vee$ is a disjunction operator and can be interpreted as an OR relation. Essentially, this statement says that no two tasks can be processed in a given unit at the same time, and is otherwise known as a disjunctive constraint (Maravelias, 2012). Such scheduling rules are conventionally transcribed into either generalized disjunctive programming or mixed integer programming formulations. Both solution methods enable constraint satisfaction.

In this work, we hypothesise that a set of controls, $\mathbf{u}_t \in \bar{\mathbb{U}}_t \subseteq \mathbb{Z}^{n_u}$, may be identified that adhere to the logic provided by the SOPs at each discrete control interaction,

[3]This approach requires $|\mathbb{U}|$ nodes in the output layer of the policy function, where $|\mathbb{U}| >> n_u$.

based on the current state of the plant, $\mathbf{x}_t \in \mathbb{X}_t$. This functional transformation is denoted, $f_{SOP} : \mathbb{U} \times \mathbb{X} \to \bar{\mathbb{U}}$, where $\bar{\mathbb{U}} \subseteq \mathbb{Z}^{n_u}$, and is assumed non-smooth.

If we are able to define and satisfy all constraints via propositional logic (i.e. f_{SOP}), we proceed to use the rounding policy, f_r, defined in Section 5.3.1. Specifically, by redefining $f_r : \mathbb{W} \to \bar{\mathbb{U}}$, the implementation provides means to identify a mapping from a continuous latent space to discrete controls that are feasible. This mechanism is widely known via action shaping, action masking or domain restriction, and has been shown to significantly improve the efficiency of RL learning process (Kanervisto et al., 2020b; Hubbs et al., 2020b).

In the case one is unable to identify (and satisfy) all input constraints, $\hat{\mathbb{U}}(\mathbf{x})$, via f_{SOP} we penalise the violation of those constraints that cannot be handled, and incorporate a penalty function for the constraint violation, $\phi : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \varphi_{t+1} \in \mathbb{R}$. For example, using discrete-time state space models, one cannot impose the constraint "no two units can start processing the same task, at the same time" via this logical transformation, without imposing considerable bias on the control selection of the policy. This is discussed further in D.2.5. Given $\mathbf{g}(\mathbf{u}) : \mathbb{U} \to \mathbb{R}^{n_g}$, represents the n_g constraints that cannot be handled innately via propositional logic, we define the *penalised return* from an episode as:

$$\begin{aligned} \phi &= R - \kappa_g \left\| [\mathbf{g}(\mathbf{u})]^+ \right\|_p \\ Z^\phi &= \sum_{t=0}^{T-1} \varphi_{t+1} \end{aligned} \tag{5.4}$$

where $\kappa_g \in \mathbb{R}$ is the penalty weight; $[\mathbf{y}]^+ = \max(0, \mathbf{y})$ defines an element wise operation over $\mathbf{y} \in \mathbb{R}^{n_y}$; and, $\|\cdot\|_p$ defines the general l_p-norm. From here we can identify a solution policy to Eq. 5.3, π^* as follows:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_\pi \left[Z^\phi \right] \tag{5.5}$$

A figurative description of the approach proposed is provided by Fig. 1, and formalized by Algorithm 5.1.

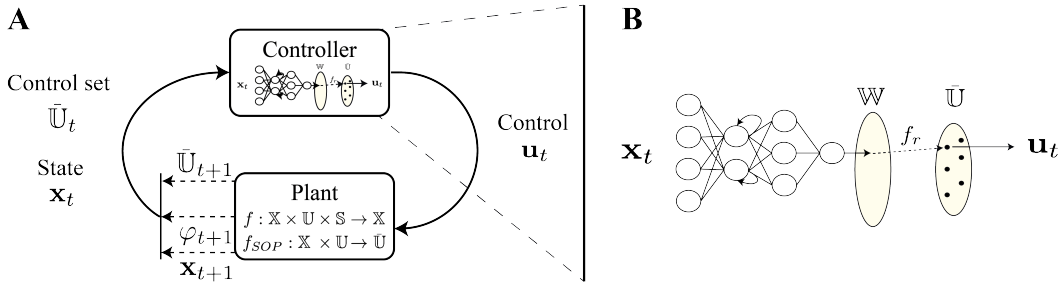


Figure 5.1: Figurative description of the feedback control framework used to formulate the scheduling problem. A) A feedback control framework that utilises logic to identify feasible scheduling decisions. B) Control selection via a deterministic rounding policy.

Algorithm 5.1 Control selection for production scheduling of an uncertain plant

Input: Policy function, $\pi(\theta_0, \cdot)$; functional transformation derived from plant standard operating procedure, $f_{SOP} : \mathbb{X} \times \mathbb{U} \to \bar{\mathbb{U}}$; description of uncertain plant dynamics, $f : \mathbb{X} \times \mathbb{U} \times \mathbb{S} \to \mathbb{X}$ (see Eq. 5.1); rounding policy, $f_r : \mathbb{W} \to \bar{\mathbb{U}}$; initial state distribution, $p(\mathbf{x}_0)$; the number of units (equipment) available, n_u; the number of tasks to be processed, n_T; the control set, $\mathbb{U}$; penalty function, $\phi : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}$; and, empty memory buffer, $\mathcal{B}_{info}$

1. Draw initial state, $\mathbf{x}_0 \sim p(\mathbf{x}_0)$

for $t = 0, \ldots, T-1$ **do**

2a. Identify control set, $\bar{\mathbb{U}}_t = \{f_{SOP}(\mathbf{x}_t, \mathbf{u}) \in \mathbb{Z}^{n_u}, \forall \mathbf{u} \in \mathbb{U}\} \subset \mathbb{Z}^{n_u}$

2b. Predict latent coordinate conditional to current state, $\mathbf{w}_t = \pi(\mathbf{x}_t; \theta)$

2c. Implement rounding policy, $\mathbf{u}_t = f_r(\mathbf{w}_t)$

2d. Implement scheduling decision and simulate, $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{s}_t)$, where $\mathbf{s}_t \in \mathbb{S}$.

2e. Observe feedback from penalty function, $\varphi_{t+1} = \phi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

end for

3. Assess Z^{ϕ} (see Eq. 5.4) and store, together with information required for policy optimisation, in $\mathcal{B}_{info}$

Output: $\mathcal{B}_{info}$

5.3.3 Stochastic search policy optimisation

Due to the nature of the problem at hand, this work proposes the use of stochastic search policy optimisation algorithms. In general, any stochastic search algorithm

can be used. A high level description of the algorithm used in this work is provided by Algorithm 5.3.4. Summarizing this approach, we use a hybrid algorithm, which combines particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995a) with simulated annealing (SA) (Kirkpatrick et al., 1983a) and a search space reduction strategy (Park et al., 2005a). For clarity, we refer to this algorithm simply as PSO-SA. The hybridization of the two algorithms helps to efficiently balance exploration and exploitation of the policy function's parameter space. For more details on this algorithm, please refer to D.1.

The use of stochastic search policy optimisation inherits three major benefits. The first being that stochastic search algorithms are easily parallelizable, which means offline computational time in policy learning can be reduced. The second benefit is that one is freed from reliance on accurate estimation of first order gradients indicative of directions for policy improvement, as in policy gradients and deep Q learning. Estimation of these directions is known to be computationally expensive (Riedmiller et al., 2007a), and there is potential for policies to become stuck in local optima, as well as instability in policy learning. This is particularly likely because the loss landscape with respect to the policy parameters is thought to be non-smooth and rough in scheduling problems[4] (Merchant et al., 2021; Amos, 2022). The final bonus is that one can easily optimise for a variety of measures of the *distribution of penalised return*, $p_\pi(z^\phi)$, (i.e. go beyond optimisation in expectation as is declared in Eq. 5.2) provided one is able to obtain a sufficient number of samples. This is discussed further in the next section.

5.3.4 Optimizing for the distribution of returns

Recent developments within RL have enabled the field to move beyond optimisation in expectation. This subfield is known as *distributional* RL. One of the advantages provided by distributional RL is that one can incentivize optimisation of the tails of the distribution of returns, providing a more risk-sensitive formulation. The concept of distributional RL was first introduced by (Bellemare et al., 2017b) in 2017.

[4]This is due to the structure of the control space and the nature of the scheduling task, i.e. controls that are close in the control space (have small euclidean distance between them) do not necessarily induce similar process dynamics.

However, optimisation of risk-sensitive criteria has been well established in the portfolio optimisation and management community for some time, with the advent of e.g. value-at-risk (VaR), conditional value-at-risk (CVaR) (Rockafellar et al., 2000), and Sharpe's ratio. Conventionally, (stochastic gradient-based) distributional RL algorithms are dependent upon making approximations of the probability density function of returns $p_\pi(z^\phi)$, so as to gain tractable algorithms (Bellemare et al., 2017b; Tang et al., 2019a). The major benefit of stochastic search approaches is that one is freed from such approximations by simply estimating the interested measure (e.g. mean, variance, (C)VaR) directly from samples (i.e. a Monte Carlo method). In the following, we outline the CVaR in the context of stochastic search policy optimisation, and reasons for its use as metric to optimise $p_\pi(z^\phi)$.

The conditional value-at-risk (CVaR)

The CVaR is closely related to the value-at-risk (VaR). The VaR defines the value, z^ϕ_β, of the random variable, Z^ϕ, that occurs with probability less than or equal to a certain pre-defined level, $\beta = [0, 1]$, under the cumulative distribution function (CDF), $F_\pi(z^\phi) = \mathbb{P}(Z^\phi \leq z^\phi|\pi)$. Informally, VaR (for some β) gives us a value, which is the best possible value we can get with a probability of at most β. For example, conceptualize that Z^ϕ represents the mass produced in a process plant per day (kg/day). If the plant has $z^\phi_\beta = 1000$ kg/day, and $\beta = 0.05$, this means that there is a 0.05 probability that the production will be of 1000 kg/day or less.

In the context of sequential decision-making problems, it is important to note that the CDF, and hence the VaR, is dependent on the policy, π. The following definitions are provided in terms of reward maximization, rather than loss (given the context of RL and MDPs). Specifically, the VaR is defined:

$$z^\phi_\beta = \max\{z^\phi : F_\pi(z^\phi) \leq \beta\} \quad \iff \quad z^\phi_\beta = F^{-1}_\pi(\beta) \tag{5.6}$$

It follows then that the CVaR is the expected value, $\mu^\phi_\beta \in \mathbb{R}$, of the random variable, Z^ϕ, with a probability less than or equal to β under $F_\pi(z^\phi)$:

$$\mu^\phi_\beta = z^\phi_\beta + \frac{1}{\beta} \int_{z^\phi < z^\phi_\beta} p_\pi(z^\phi)(z^\phi - z^\phi_\beta) dz^\phi \tag{5.7}$$

Eq. 5.7 expresses the CVaR (for a given probability β). This can be interpreted as the VaR minus the expected difference between the VaR and the returns, z^ϕ, which

are realized with probability $\leq \beta$, i.e. such that $F_\pi(z^\phi) \leq \beta$. This is further reinforced by Fig. 5.2, which provides a visualization of the VaR and CVaR.

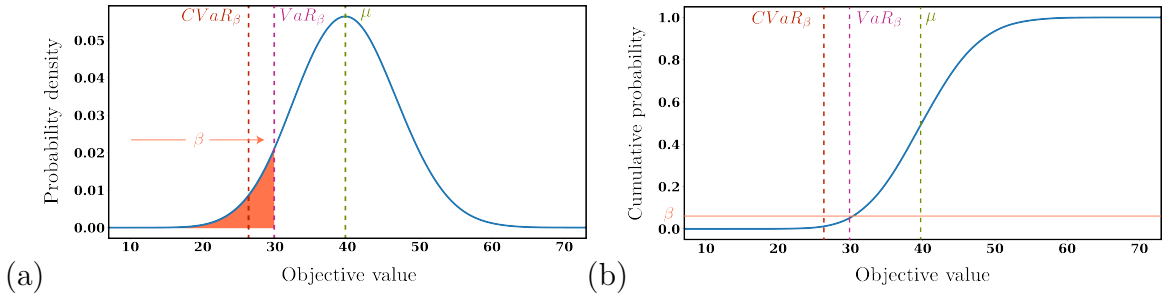


Figure 5.2: Description of the conditional value-at-risk, $CVaR_\beta$, and the value-at-risk, VaR_β, for a given probability level β, as well as the expected value, μ under a) the probability density function, $p_\pi(z^\phi)$, and b) the cumulative distribution function, $F_\pi(z^\phi)$.

The optimisation of the CVaR has advantage over the VaR in engineering applications because it provides more information about the performance of the policy within the tails of the distribution, $p_\pi(z^\phi)$. This is particularly beneficial if the distribution is characterised by heavy tails, which is often the case, given the possibility for rare events within the plant. Further, Z_β^ϕ possesses many undesirable functional properties that makes its optimisation more difficult than μ_β^ϕ (Rockafellar and Uryasev, 2002). Therefore, in this work we favor the CVaR and present means to estimate from samples in the proposed framework.

Optimisation of CVaR via sampling

We seek means by which to gain approximations of Eqs. 5.6 and 5.7 via sampling. Specifically, assuming one has N realizations, $Z_{MC}^\phi = [z_1^\phi, \ldots, z_N^\phi]$, then Eq. 5.6 may be approximated via the $\lfloor \beta N \rfloor^{\text{th}}$ order statistic, which we denote $\bar{z}_\beta^\phi$. The CVaR, μ_β^ϕ, may then be approximated via $\bar{\mu}_\beta^\phi$ as follows:

$$\bar{\mu}_\beta^\phi = \bar{z}_\beta^\phi + \frac{1}{\beta N}\left[\sum_{i=1}^{N} \min(0, z_i^\phi - \bar{z}_\beta^\phi)\right] \tag{5.8}$$

Here, we simply replace the integral term in Eq. 5.7 via its sample average approximation. The *minimum* operator enforces the bounds of integration detailed, such that only realizations less than the VaR are considered. The statistical properties associated with estimation of $\bar{\mu}_\beta^\phi$ have been well researched. Under some assumptions, the

Monte Carlo estimate of the CVaR converges with increasing samples at the same rate as the sample average approximation (Hong and Liu, 2009). We direct the interested reader to the cited works for more information.

How one incorporates this distributional perspective into the RL problem is highly flexible when applying stochastic search policy optimisation. The CVaR may be either optimised as a sole objective or it may be enforced as a constraint subject to appropriate definition of β. In other words, one may formulate either of the two following optimisation problems:

$$\pi^*_\beta = \arg\max_\pi \bar{\mu}^\phi_\beta \tag{5.9a}$$

$$\pi^*_\beta = \arg\max_\pi \mathbb{E}_\pi\left[Z^\phi\right] \quad s.t. \quad \bar{\mu}^\phi_\beta \geq b \tag{5.9b}$$

where $b \in \mathbb{R}$ is some minimum desired value. For the use of RL, we are dependent upon obtaining a closed form expression as an optimisation objective. Optimisation of Eq. 5.9a is trivial. Whereas Eq. 5.9b may be handled via a penalty method, or simply by discarding infeasible policies in the case of stochastic search algorithms. Please see Rockafellar and Uryasev (2002) for more information on CVaR optimisation formulations. A general approach to optimisation of either of the formulations (Eqs. 5.9a and 5.9b) is provided by Algorithm 5.3.4. The description is kept general to ensure clarity of the methodology. For further information on stochastic search optimisation and the algorithm used in this work, PSO-SA, please see D.1.

To provide demonstration of the ideas discussed, we now investigate the application of the methodology on a case study, which has been adopted from early work provided in Cerda et al. (1997). We add plant uncertainties to the case study and construct a discrete-time simulation of the underlying plant. The case study is discussed further in the next Section.

Algorithm 5.2 Distributional stochastic search policy optimisation

Input: Policy function (a neural network in this study), $\pi(\hat{\theta}, \cdot)$, parameterized by $\hat{\theta}$; a number of samples to evaluate each candidate policy, n_I; sample approximation of objective function to optimise, $f_{SA}(\cdot)$; a general stochastic search optimisation algorithm, $f_{SSO}(\cdot)$; population size, P; upper, θ_{UB}, and lower, θ_{LB}, bounds on the search space; population initialization method, $f_{init}(\hat{\theta}, P, \theta_{UB}, \theta_{LB})$; a number of optimisation iterations, K; a memory buffer, $\mathcal{B}_{SSO}$; memory for metrics of optimal policy, $\mathcal{B}_{\pi^*} = \{J_{\pi^*}, \pi(\theta^*, \cdot)\}$

1. Generate initial (neural network) population parameters, $\Theta^1 = f_{init}(\hat{\theta}, P, \theta_{UB}, \theta_{LB})$, where $\Theta^1 = \{\theta_1, \ldots, \theta_P\}$

for $k = 1, \ldots, K$ **do**

2a. Construct policy population, $\Pi^k = \{\pi(\theta_i, \cdot), \forall \theta_i \in \Theta^k\}$

for each (neural network) policy $\pi_{c,i} \in \Pi^k$ **do**

2b i. Evaluate *penalized return* of the policy, Z^{ϕ}, of $\pi_{c,i}$ via Algorithm 5.1 for n_I samples

2b ii. Return distribution information of the policy, $\mathcal{B}_{\pi_{c,i}} = \{\mathcal{B}^1_{info}, \ldots, \mathcal{B}^{n_I}_{info}\}$ from Algorithm 5.1

2b iii. Assess sample approximate objective, $J_i = f_{SA}(\mathcal{B}_{\pi_{c,i}})$

2b iv. Collect information for policy $\pi_{c,i}$ and append $[J_i, \mathcal{B}_{\pi_{c,i}}]$ to $\mathcal{B}_{SSO}$

2b v. if $J_i > J_{\pi^*}$ **then** update best known policy $\mathcal{B}_{\pi^*} = \{J_i, \pi_{c,i}\}$

end for

2c. Generate new parameters $\Theta^{k+1} = f_{SSO}(\mathcal{B}_{SSO})$, where $\Theta^{k+1} = \{\theta_{c,1}, \ldots, \theta_{c,P}\}$

end for

Output: $\pi(\theta^*, \cdot) \in \mathcal{B}_{\pi^*}$

5.4 Case studies

5.4.1 Problem definition

We consider a multi-product plant where the conversion of raw material to product only requires one processing stage. *We assume* there is an unlimited amount of raw material, resources, storage and wait time (of units) available to the scheduling element. Further, the plant is completely reactive to the scheduling decisions of the policy, π, although this assumption can be relaxed (with appropriate modification to the method stated here) if decision-making is formulated within an appropriate

framework as shown in Hubbs et al. (2020a). The scheduling element *must decide* the sequencing of tasks (which correspond uniquely to client orders) on the equipment (units) available to the plant. The requirement for sizing decisions is removed from the problem, based on the requirement for the production of tasks to be organised in campaigns (i.e. multiple sequential batches if required), as well as the fact that the yield of a batch is known with certainty. Full detail of the properties and requirements of the production environment is detailed in D.2.1.

The objective is to minimise the makespan and the tardiness of task (order) completion. Once all the tasks have been successfully processed according to the operational rules defined, then the decision making problem can be terminated. It should be noted that, although it is a valid objective in this case study, it is unlikely that makespan would provide a suitable objective in reality, primarily because the production continues beyond the end of the horizon. The problem is modelled as an MDP with a discrete-time formulation. Full details of the MDP construction, uncertain state space model and model data used in this work is provided by D.2, however the definition of system state and control spaces are provided subsequently.

The representation of system state in discrete-time scheduling MILP models is discussed at length in Subramanian et al. (2012); McAllister et al. (2022), where a state-task network representation is assumed. The state in that work is actually a matrix based on the use of lifting variables to account for the continued processing of production tasks over multiple time intervals. Instead, we pose a problem specific representation as follows:

$$\mathbf{x}_t = \Big[I_{1t}, \ldots, I_{Nt}, w_{1t}, \ldots, w_{n_u t}, \delta_{1t}, \ldots, \delta_{n_u t}, \rho_{1t}, \ldots, \rho_{Nt}, t\Big]^T \in \mathbb{R}^{2N+2n_u+1} \tag{5.10}$$

where $I_{it}\ \forall i \in \mathbb{I}$ quantifies the current inventory of client orders in the plant; the tasks processed within units in the plant over the previous time interval, $w_{lt}\ \forall l \in \mathbb{L}$; a forecast, based on nominal processing times, of the discrete time indices remaining until completion of the task campaigns being processed in all units, $\delta_{lt}\ \forall l \in \mathbb{L}$; a forecast of the the discrete time indices remaining until orders (tasks) are due, $\rho_{it}\ \forall i \in \mathbb{I}$; and, the current discrete time index, t. The major difference here compared to the state representation provided in McAllister et al. (2022) is that we desire a vector, rather than matrix, representation of state.

Similarly, we define the control space, $\mathbf{u} = [u_1, \ldots, u_{n_u}]^T \in \mathbb{U}$, where $\mathbb{U}^{(l)} = \mathbb{I} \cup \{N+1\}$, and $N+1$ is an integer value, which represents the decision to idle the unit for one time period. Given that the time grid is global across units, some logic is required to ensure that operations can span multiple time intervals and are not rescheduled at every discrete time index. This, together with means of forecasting state evolution and identification of control sets, is discussed at length in the supporting information (see D.2.2). However, it is worth noting that there is one input constraint, which cannot be handled explicitly through identification of control sets. This is the requirement for no production task to be repeated over the course of the time horizon. As discussed in section 5.3.2, this cannot be asserted via action masking without biasing the policy and so it is instead handled through a penalty method.

5.4.2 Benchmark

The benchmark for the following experiments as presented in Section 5.4.3 is provided by both offline and online implementations of the continuous time, mixed integer linear programming (MILP) model first detailed in Cerda et al. (1997).

To ensure that the solutions from the two time transcriptions (i.e. the continuous time formulation of the benchmark and the discrete-time formulation of the methodology proposed) are comparable, the data which defines the sequence dependent cleaning times, task-unit dependent processing times and release times are redefined from the original study to ensure that their greatest common factor is equal to the length of a time interval in the discrete-time transcription.

Online implementation was dependent upon updating the model data by receiving feedback from the state of the plant at a given discrete time index and fixing variables appropriately. The MILP model was resolved at each discrete time index in the horizon to allow fair comparison to the RL approach. Given that there are uncertainties in the plant dynamics, the MILP model utilises the expected values of the uncertain data. Similarly, in RL expected values of uncertain variables are maintained in the state representation, until the uncertainty is realized at which point the state is updated appropriately (Hubbs et al., 2020a). Please see Cerda et al. (1997) and D.3 for more information on the model and data used for the following experiments, respectively. All MILP results reported were generated via the Gurobi v9.1.2 solver together with

the Pyomo v6.0.1 modelling framework. The proposed method utilised the PyTorch v1.9.0 python package and Anaconda v4.10.3.

5.4.3 Experiments

Problem instances and sizes

In the following, we present the formulation of a number of different experiments across two different problem instances. The first problem instance is defined by a small problem size. Specifically, we are concerned with the sequencing of 8 client orders (tasks) on four different units. This corresponds to 304 binary decision variables and 25 continuous decision variables within the benchmark continuous-time MILP formulation. The second problem instance investigates the ability of the framework to handle larger problems with 15 orders and 4 units. In this case, the MILP formulation consists of 990 binary decision variables and 46 continuous decision variables.

Study designs and assessment of performance

Both of the problem instances are investigated thoroughly. We demonstrate the ability of the method to handle: a) uncertainty in processing times; b) uncertainty in the due date; and, c) the presence of finite release times. These three considerations, a)-c), are used as factors to construct a full factorial design of experiments. Each factor has two levels, either it is present in the underlying problem, or it is not. As a result, the majority of analysis focuses on the investigation of the method's ability to optimise in expectation. This is captured by experiments E1-E8 in Table 5.1.

A number of the experimental conditions were used to demonstrate the ability of the method to optimise for the CVaR of the distribution also. This was investigated exclusively within problem instance 1, as detailed by experiments D3-D8 in Table 5.1.

In all experiments, the processing time uncertainty is defined via a uniform probability density function (see Eq. D.14a) and the due date uncertainty is described by a Poisson distribution (see Eq. D.14b).

Further to the experiments proposed in Table 5.1, the proposed and benchmark methods are compared with respect to online and offline computational burden. The robustness of the proposed RL method to misspecification of the plant uncertainties

Table 5.1: Table of experimental conditions investigated. Details of the exact descriptions of uncertain variables are provided by D.2.

Optimizing in Expectation (Eq. 5.5)			
Reference	Processing time uncertainty	Due date uncertainty	Finite release times
E1	✗	✗	✗
E2	✗	✗	✓
E3	✗	✓	✗
E4	✗	✓	✓
E5	✓	✗	✗
E6	✓	✗	✓
E7	✓	✓	✗
E8	✓	✓	✓
Optimizing for Conditional Value at Risk (CVaR) (Eq. 5.9a)			
Reference	Processing time uncertainty	Due date uncertainty	Finite release times
D3	✗	✓	✗
D4	✗	✓	✓
D5	✓	✗	✗
D6	✓	✗	✓
D7	✓	✓	✗
D8	✓	✓	✓

is also investigated. The results are detailed in Section 5.5.4 and 5.5.5, respectively.

The performance indicators used to evaluate the respective methods include the expected performance of the scheduling policy, $\mu_z = \mathbb{E}_\pi\big[Z\big]$, the standard deviation of the performance, $\sigma_Z = \Sigma_\pi\big[Z\big]$ and the conditional-value-at-risk. Specifically, we use a version of the CVaR, $\bar{\mu}_\beta$, which considers the non-penalised returns[5] with $\beta = 0.2$ (i.e. $\bar{\mu}_\beta$ represents the expected value of the worst case policy performance observed with probability less than or equal to 0.2). Finally, we utilise a statistically robust approximation to the probability of constraint satisfaction, F_{LB}, to evaluate the constraint handling abilities of both solution algorithms. Formal details of evaluation of F_{LB} are provided by D.6. In the following section, the results for the different experiments are presented for the method proposed and benchmarked relative to a continuous-time MILP formulation.

[5]This is assessed via appropriate modification to Eq. 5.8, such that we consider the sum of rewards, rather than the return under the penalty function.

5.5 Results and discussion

In this section, we turn our attention to analysing the policy training process and ultimate performance of the framework proposed. In all cases, results were generated by the hybrid PSO-SA stochastic search algorithm with space reduction. In *learning* (or stochastic search), all candidate policies were evaluated over $n_I = 50$ samples (when the plant investigated was subject to uncertainty, if deterministic $n_I = 1$), with a population size of $P = 60$ and maximum optimisation iterations of $K = 150$. The structure of the network utilised for policy paramaterization is detailed by D.1.

5.5.1 Policy training

Demonstration of the training profiles for experiment E8, problem instance 1 are provided by Fig. 5.3. Fig. 5.5.1 details the formulation provided by Eq. 5.5 and Fig. 5.5.1 details Eq. 5.9a (i.e. the expected and CVaR objective, respectively).

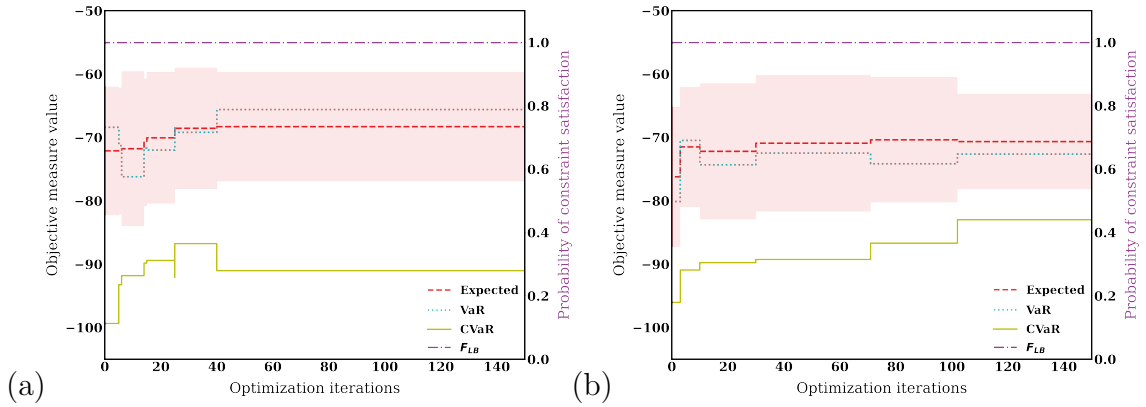


Figure 5.3: The training profile of the RL agent on experiment E8, problem instance 1. Metrics of the best known policy are tracked as the population is iterated. Plot a) shows the mean, standard deviation (shaded region around the expected profile), value-at-risk ($\beta = 0.2$), the corresponding conditional-value-at-risk and probability of constraint satisfaction, F_{LB}, for formulation Eq. 5.5. Plot b) displays the same information for formulation Eq. 5.9a.

The plots detail how the methodology steadily makes progress with respect to the measure of the objective posed by the respective problem formulation. The VaR represents the maximum objective performance observed with probability less than or equal to 0.2. Hence, the similarity between the VaR and the expected performance, as expressed by Fig. 5.3, indicates that the returns are not well described by a Gaussian distribution. This is an approximation required by many distributional RL algorithms,

but not stochastic search optimisation approaches as used in this work. Optimisation iterations proceed after population initialization. Therefore, the performance of the initial best known policy is dependent upon the initialization. Thereafter, each iteration consists of evaluating each candidate policy in the population over $n_I = 50$ samples, such that 150 iterations is equivalent to obtaining 450,000 simulations of the uncertain model. The computational time cost of this is discussed in Section 5.5.4. All current best known policies identified throughout training satisfy the constraints imposed on the problem, indicating the efficacy of framework proposed.

5.5.2 Problem instance 1

In this section, the results of investigations for experiments corresponding to problem instance 1 are presented. In this problem instance there are 8 customer orders and 4 units available to the production plant. Firstly, we present results confirming the ability of the method to identify an optimal scheduling solution for a deterministic plant, which corresponds to the generation of an offline production schedule. We then turn our attention to scheduling of the plant subject to uncertainties.

Optimisation of the deterministic plant (offline production scheduling)

There are two experiments that investigate the optimality of the method proposed in generation of an offline production schedule: E1 and E2 (see Table 5.1). Investigation E1 defines a deterministic plant without finite release times, whereas E2 includes finite release times.

Fig. 5.5.2 and 5.5.2 provides a comparative plot of the schedule generated via the policy identified via the method proposed and the MILP formulation for experiment E1, respectively. As reinforced by Table 5.2, the policy obtained from the method proposed achieves the same objective value as the benchmark MILP method, despite the slight difference in the structure of the solutions identified. The difference in the RL solution arises in the sequencing of tasks in unit 3 and 4, but does not affect the overall quality of schedule identified. Importantly, this result highlights the ability of the method proposed to account for e.g. sequence dependent cleaning times, sequencing constraints, and requirements to complete production in campaigns in the appropriate units as imposed in this case study

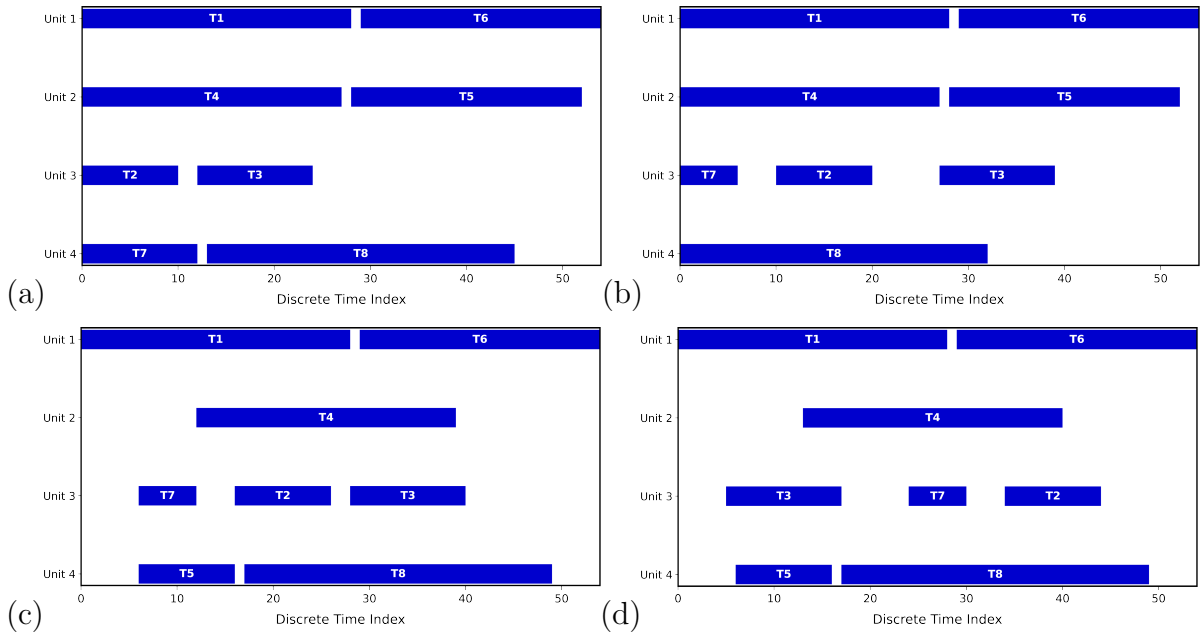


Figure 5.4: Investigating the offline schedule generated for the deterministic plant (problem instance 1). The results for experiment E1 generated by the a) RL and b) MILP methods and for experiment E2 generated by the c) RL and d) MILP methods. The label Ti details the scheduling of task i in a given unit.

Next, we turn our attention to the handling of finite release time within the framework provided. This is probed by experiment E2. Again, a comparative plot of the relative production schedules identified by the MILP formulation and proposed method is provided by Fig. 5.5.2 and 5.5.2. Despite a slight difference in the structure of the solution, arising from the sequencing of tasks in unit 3; as detailed from Table 5.2, the policy obtained from the method proposed achieves the same objective value as the benchmark MILP method on experiment E2. This further supports the capacity of the framework to handle the constraints imposed on the problem. In the following section, we explore the ability of the framework to handle plant uncertainties.

Optimisation of the uncertain plant (reactive production scheduling)

In this section, we turn our attention to demonstrating the ability of the framework to handle plant uncertainty. Specifically, we incrementally add elements of plant uncertainty to the problem via investigation of experiments E3-E8. Again, the MILP formulation is used to benchmark the work. For both approaches, policies were validated by obtaining 500 Monte Carlo (MC) simulations of the policy under the plant dynamics. This enables us to accurately estimate measures of the distribution of

return, $p_\pi(z)$.

Table 5.2: Table of results for the proposed method from investigation of experimental conditions detailed by Table 5.1 for Problem Instance 1. The policies synthesised were optimised under the objective provided by Eq. 5.5.

Reference	Method	μ_Z	σ_Z	$\bar{\mu}_\beta$	F_{LB}	Method	μ_Z	σ_Z	$\bar{\mu}_\beta$	F_{LB}
E1	Proposed	**-62.0**	0.0	-62.0	1.0	MILP	-62.0	0.0	-62.0	1.0
E2		**-65.0**	0.0	-65.0	1.0		-65.0	0.0	-65.0	1.0
E3		**-61.9**	4.4	-72.5	1.0		-63.3	4.4	-72.2	1.0
E4		**-66.0**	4.9	-75.6	1.0		-66.3	4.9	-76.5	1.0
E5		**-66.8**	8.7	-86.0	1.0		-70.1	9.6	-90.2	1.0
E6		-73.8	10.7	-97.5	1.0		**-73.6**	10.3	-94.0	1.0
E7		**-67.4**	10.9	-86.8	1.0		-71.6	11.3	-93.5	1.0
E8		-75.3	11.5	-101.13	0.99		**-75.1**	11.7	-97.7	1.0

Table 5.2 presents the results for the method proposed and the MILP benchmark for experiments E3-8. Here, the proposed method aims to optimise μ_Z (the expected performance), whereas the MILP formulation assumes the plant is deterministic, and only accounts for plant uncertainty via state-feedback. In 4 out of 6 experiments (E3, E4, E5 and E7), the RL method outperforms the MILP approach and these are statistically significant results based on two-sided Student's t-tests. The most significant of these is experiment E7, where the proposed approach (-67.4) outperforms the MILP (-71.6) by 5.8% in the objective. The MILP approach only marginally outperforms the proposed method (by $\leq 0.2\%$) in 2 of 6 of the experiments. This is observed in the results of experiment E6 and E8.

Further, in all but one of the experiments, constraints are respected absolutely across all 500 MC simulations by both the methods. In experiment E8, however, the method proposed violated the constraints in what equates to 1 out of the 500 MC simulations. This is primarily because in the optimisation procedure associated with the method candidate policies are evaluated over 50 MC simulations. This means that a realization of uncertainty, which caused the policy to violate the constraint expressed by the penalty function in validation, was not observed in the *learning phase* (see D.2.5 for more information regarding the specific constraint). This can be mitigated practically by increasing the number of samples, n_I, that a policy is evaluated for, at the cost of increased computation.

Most of the analysis so far has concentrated on the expected performance, μ_Z, of the policy and the probability of constraint satisfaction, F_{LB}. Generally, all standard

deviation, σ_Z, of $p_\pi(z)$ across the experiments are similar for both the RL approach and the MILP approach. There is some discrepancy between the CVaR, $\bar{\mu}_\beta$, of the RL and MILP which considers the worst 20% of returns associated with $p_\pi(z)$, however, this should be interpreted with care given that there is no justification for either method's formulation to be better than the other. However, this can be incentivized in the RL approach and this is discussed in the subsequent section.

Optimizing risk-sensitive measures

In this section, we present the results from investigation of experiments D3-8. The difference between these experiments and E3-8, is that now we are interested in optimizing for a different measure of the distribution, $p_\pi(z)$. In E3-8, the objective was optimisation in expectation, μ_Z. In D3-8, the objective is optimisation for the expected cost of the worst 20% of the penalised returns, i.e. Eq. 5.9a, with $\beta = 0.2$. Again, the optimisation procedure associated with the proposed method utilises a sample size of $n_I = 50$. All policies were then evaluated for 500 MCs.

Table 5.3: Results for distributional RL from experimental conditions detailed by Table 5.1. Results that are emboldened detail those policies that show improved CVaR over the MILP approach (as detailed in Table 5.2).

Method	Reference	μ_Z	σ_Z	$\bar{\mu}_\beta$	F_{LB}
Proposed	D3	-62.1	4.1	**-69.8**	0.99
	D4	-65.5	4.9	**-75.5**	0.99
	D5	-67.2	8.2	**-83.9**	1.0
	D6	-74.9	11.2	-99.0	1.0
	D7	-67.9	9.7	**-87.1**	0.99
	D8	-73.2	11.6	**-96.7**	1.0

Table 5.3 details the results of the investigations D3-8. As before, the results corresponding to $\bar{\mu}_\beta$ quantify the CVaR of the sum of rewards, Z, i.e. not including penalty for violation of constraints. The results that are emboldened show improvements in the CVaR over the MILP results as detailed in Table 5.2. To reiterate, the only difference here is that the MILP uses the expected values of data in its formulation, whereas the distributional RL approach (the proposed method) accounts for uncertainty in the data and optimises for the expected value of the worst 20% of penalized returns. This leads to improvements in the CVaR in 5 out of 6 of all experiments (D3-8), with an average improvement of 2.29% in the expected value of the worst 20% of the returns.

This figure should be interpreted with care, given that there is likely to be some statistical inaccuracy from the sampling process. However, the weight of the result supports that there is an improvement in performance gained from the RL formulation.

The expected performance, μ_Z, of the policy identified for experiments D3-8 is also competitive with the MILP formulation. However, this is not promised to hold generally, given improvements in the expected performance are not incentivized within the formulation (i.e. Eq. 5.9a, although this result is also noted in Sarin et al. (2014)). To balance both objectives the formulation provided by Eq. 5.9b could be investigated, if it is of interest to a given production setting.

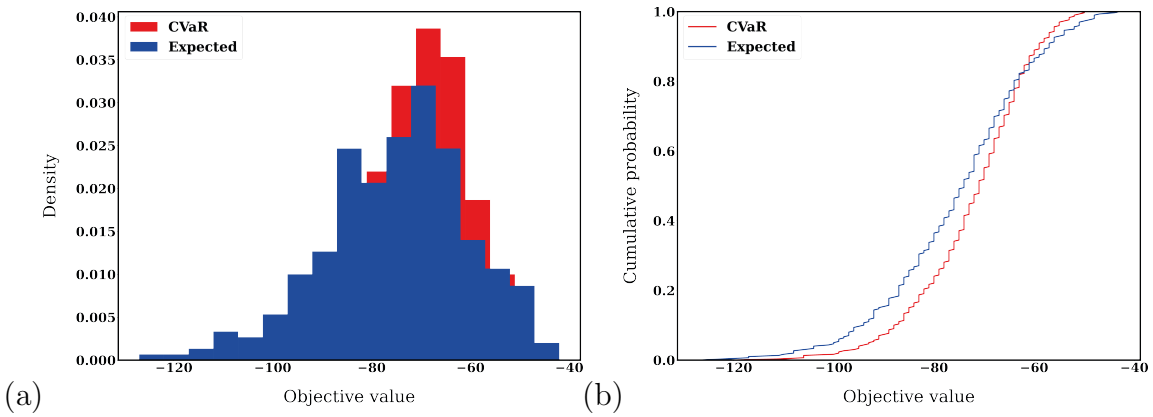


Figure 5.5: The distributions of returns observed in validation of the RL policy obtained from optimizing expectation (i.e. E8) and conditional value-at-risk (i.e. D8) within the same production environment. Plot a) a histogram of the objective performances, and b) the empirical cumulative distribution function associated with each policy.

Similar observations can be made with respect to the probability of constraint satisfaction as in Section 5.5.2. In 3 of the 6 experiments, the constraint imposed in the penalty function is violated in approximately 1 of the 500 MC simulations. Again, this could be mitigated by increasing the number of samples, n_I, in policy evaluation during the optimisation procedure. However, even at the current setting of $n_I = 50$, the constraints are respected with high probability, and this was deemed sufficient for this work.

Fig. 5.5 expresses the distribution of returns under the RL policies optimizing expectation and CVaR, as obtained in experiment E8 and D8, respectively. Both policies are learned in production environments defined by the same dynamics and

uncertainties. Their quantitative metrics can be found in Table 5.2 and 5.3, respectively. Particularly, Fig. 5.5.2 highlights how the CVaR policy observes improved performance in the tail of the distribution over the policy optimizing for expectation (i.e. the CVaR plot is shifted along the x-axis to the right). Although, the policy does not observe as good a best-case performance, this eloquently highlights the utility of distributional formulations to identify risk-sensitive policies.

5.5.3 Problem instance 2

Optimisation of the deterministic plant (offline production scheduling)

Here, the results of investigation of experiments E1 and E2 are presented for problem instance 2. The purpose of the investigation is to determine whether the method is able to scale with larger problem instances efficiently and retain a zero optimality-gap as in Section 5.5.2. Both experiments consider the plant to be deterministic, with E2 additionally considering the presence of finite release times.

Fig. 5.5.3 and 5.5.3 presents the comparative results of the schedule identified for experiment E1 of the RL and MILP approach. There are clear differences between the schedules. These arise in the sequencing of task 6 and 9 in unit 1, and in the sequencing of unit 3. However, for this case the RL approach is able to retain a zero optimality-gap. This provides promise for the method proposed in scaling to larger problem sizes. This is reinforced by the value of the objective as presented in Table 5.4, where both methods achieve an objective score of -107.

Fig. 5.5.3 and 5.5.3 presents comparative results of the schedule generated for experiment E2. Here, we see that again there is slight difference between the schedule generated for the RL relative to the MILP. However in this case, there is an optimality gap of 4.4% in the RL schedule generated. This is confirmed by Table 5.4, which details the RL method achieved an objective score of -143.0, whereas the MILP method achieved a score of -137.0. The difference between the schedules, which causes the optimality-gap, is observed in the scheduling of task 10 in unit 2 (in the case of RL). This leads to a slight increase in the overall makespan of production. This could be attributed to the complex and non-smooth mapping between the state and optimal control, which makes model structure selection a difficult task when using parametric

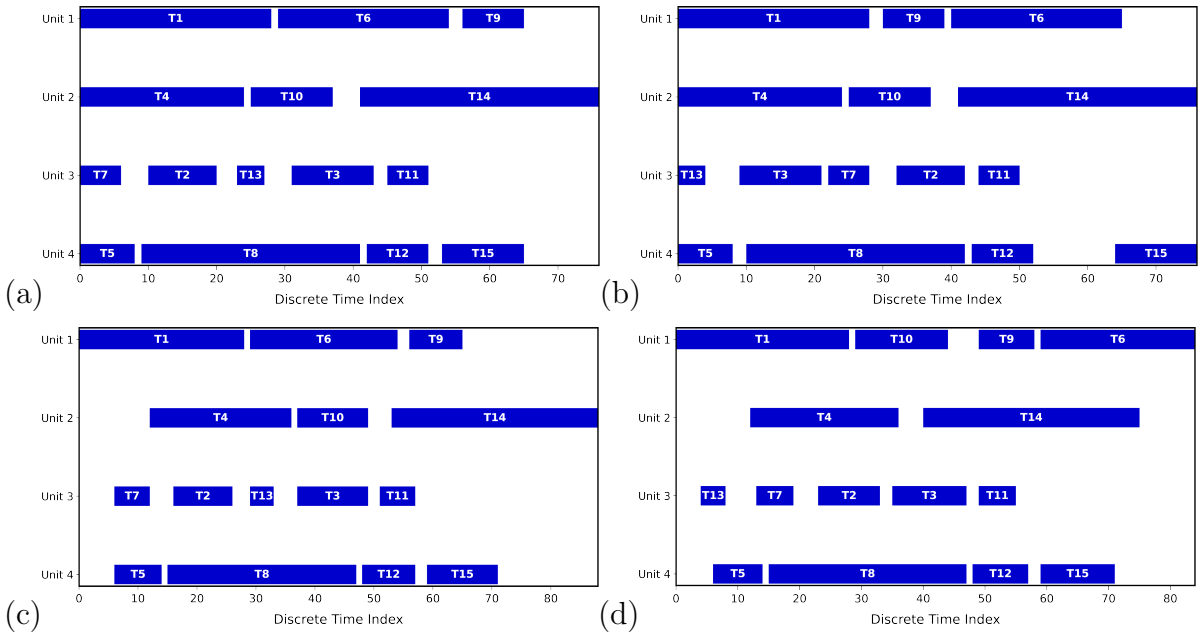


Figure 5.6: Investigating the offline schedule generated for the deterministic plant. The results for experiment E1, problem instance 2 generated by the a) RL and b) MILP methods. The results for experiment E2, problem instance 2 generated by the c) RL and d) MILP methods. The label Ti details the scheduling of task i in a given unit.

models, such as neural networks. An alternate hypothesis exists in the parameter space, rather than the model space. The mapping between policy parameters and objective is again likely to be highly non-smooth, which could pose difficulties for stochastic search optimisation. It should be noted, however, that the latter proposition should hold true for the experiment without finite release times (E1), but in this case an optimality gap is not observed. One could therefore consider devising algorithms to: a) enable use of function approximators more suited to expressing non-smooth relationships between the state and optimal control e.g. decision trees; b) automate the identification of the appropriate parametric model structure; or c) improve on the existing hybrid stochastic search algorithm used in this work. Despite the presence of the optimality gap in experiment E2, in both experiments, the framework is able to efficiently handle the constraints imposed on the plant. In the following section, we consider the addition of process uncertainties via investigation of experiments E3-8.

Optimisation of the uncertain plant (reactive production scheduling)

From Table 5.4, it is clear that generally the MILP outperforms the RL formulation proposed. On average, the performance gap is 2.2%. This can be decomposed, such that if we consider those experiments with finite release time present, the average gap is 2.95%; for those experiments that did not include finite release times, the average gap is just 1.4%. This further supports the argument that the optimality gap is dependent upon the underlying complexity of the optimal control mapping (i.e. the optimal scheduling policy becomes more difficult to identify when finite release times are included).All but one of these differences are statistically significant, with only experiment E5, showing no differences between the two algorithms (based on two-sided, Student's t-tests).

Table 5.4: Table of results for the proposed method from investigation of experimental conditions detailed by Table 5.1 for Problem Instance 2. The policies synthesised were optimised under the objective provided by Eq. 5.5. The * indicates the differences between the two groups are not statistically significant based on two-sided t-test.

Reference	Method	μ_Z	σ_Z	$\bar{\mu}_\beta$	F_{LB}	Method	μ_Z	σ_Z	$\bar{\mu}_\beta$	F_{LB}
E1	Proposed	**-107.0**	0.0	-107.0	1.0	MILP	-107.0	0.0	-107.0	1.0
E2		-143.0	0.0	-143.0	1.0		**-137.0**	0.0	-137.0	1.0
E3		-116.3	8.06	128.1	0.98		**-109.2**	8.11	121.2	1.0
E4		-150.5	14.6	175.4	0.97		**-142.7**	12.3	159.8	1.0
E5*		-123.4	18.4	150.2	1.0		-123.4	16.7	148.6	1.0
E6		-169.3	21.9	199.8	0.98		**-166.6**	22.1	193.5	1.0
E7		**-127.2**	19.6	150.1	0.95		-127.9	20.4	155.8	1.0
E8		-168.9	24.1	203.2	0.99		**-167.5**	24.6	200.1	1.0

It is also of note that the probability of constraint satisfaction decreases in this case study relative to problem instance 1, across experiments E3-8 (see Table 5.2). However, these changes are generally small, with the largest change equivalent to a 4% decrease. This is likely due to realizations of uncertainty unseen in the training having greater impact in validation, given the increased complexity of the decision making problem. Additionally, this change is in the robust statistic, rather than the empirical probability of constraint violation (see D.6) and hence the ability of RL in satisfying constraints is still satisfactory. This is because we are explicitly trading off a small risk of constraint violation with improved computational time to identify a scheduling decision. However, as previously stated, the potential to handle constraints could be improved by increasing n_I in training, at the cost of added computational complexity.

This is reinforced by the binomial proportion confidence interval literature (Wilcox, 2011). Similar ideas exist within the scenario optimisation community, although in this case, there are no such guarantees (Calafiore and Campi, 2005).

Additionally, the distributional RL formulation was investigated for problem instance 2. Due to space limitations the results are fully detailed by Table D.7 in D.4. On average across the experiments, the distributional formulation observes an improvement of 0.62% in the CVaR for probability level $\beta = 0.8$, with a maximum improvement of 5.28% in experiment D3. In 3 of 6 experiments improvements were observed for the CVaR and the largest gap (1.91%) across all experiments was observed in experiment D2. In this case, the true benefits of the distributional formulation seem inconclusive and it is hypothesised this is either due to structure of the problem, or the stochastic nature of RL.

5.5.4 Computational time cost in policy identification and decision-making

In this section, we are interested in the comparative time intensity of the RL approach proposed relative to the MILP approach. Having identified a model of the plant and the associated uncertainties, we are interested in how long it takes to identify: a) a control policy, π, (this is conducted offline) and b) a control decision, $\mathbf{u}_t$, conditional to observation of the plant state, $\mathbf{x}_t$, at discrete time index, t (this is conducted online). The code was parallelized to minimise the time complexity of computation. In practice, the computational metrics reported in the following could be bettered considerably, with respect to policy identification, if the appropriate hardware were available.

In the case of a), in RL we consider the amount of time to conduct stochastic search optimisation associated with the proposed method when the sample size $n_I = 50$; whereas, the MILP incurs no time cost in identification of the policy, given the optimisation formulation is essentially the policy, π. In b), we consider the time taken: for RL to conduct a forward pass of the neural network parameterization of the control policy; and, to solve an MILP problem. Table 5.5 details the respective results for problem instance 1 and 2.

From Table 5.5, it is clear that the RL method demands a much greater investment

Table 5.5: Normalized times for a) offline identification of a control policy, π, and b) identification of online scheduling decisions for problem instances 1 and 2.

	Time for identification		
Object of identification	Problem instance	MILP	RL
Control decision (online)	1	1	0.0067
	2	1	0.002
Control policy (offline)	1	0	1
	2	0	1

in the computational synthesis of a scheduling policy, π, offline. Generally, policy synthesis required between 1-2 hours for problem instance 1 and 2 under the sample size, n_I, detailed by this work. The computational expense of the simulation essentially scales linearly with the length of discrete time horizon. The differences in time for offline policy identification between problems 1 and 2 arises due to an increased number of orders (leading to a longer makespan and hence simulation time in problem instance 2). The effects of increased makespan on simulation time could be partially weakened through the use of a discrete-event simulation, as the methodology proposed here can also be applied to such models.

However, the work required online to update the control schedule, given a realization of plant uncertainty, is far lower in the case of RL than in the MILP formulation. For problem instance 1, the MILP is characterized by 304 binary decision variables and 25 continuous decision variables. In this case, inference of a control decision is 150 times cheaper via the RL method proposed. In problem instance 2, the MILP formulation consists of 990 binary decision variables and 46 continuous decision variables. For a problem of this size, the RL approach is 500 times cheaper than the MILP. The online computational benefits of RL are likely to be considerable for larger case studies, which is of significant practical benefit to the scheduling operation.

It is worth briefly mentioning here two main points. The first being that for many production operations, it is not possible to model demand as well as assumed in this work. Often, the number of orders and level of demand is highly variable from week-to-week and month-to-month, which means demand forecasting can be subject to high mismatch. In these cases, it is likely that the plant dynamics are no longer well represented by the original simulation model over a long horizon (Maravelias, 2012).

This is an issue that also effects MILP approaches. However, because offline policy identification only requires a short period of time, one could re-identify the policy in the offline simulation model with updated data in the same way that MILP models are also updated. Additionally, in such instances, interaction between the scheduling and upper level planning and business functions can help inform the definition of appropriate length of the time horizon to optimise over, which may also reduce the time required for offline simulation (as discussed earlier in this section).

The second point to discuss is that the notion of generating scheduling decisions from a function implies that at each discrete time index, the entire schedule is not generated for the time horizon, which may lead to a lack of certainty over future plant operations. This could be overcome by forecasting the system state via the nominal process data or developing a framework to identify the most likely schedule. This may then provide some certainty for the operation.

5.5.5 The effects of inaccurate estimation of plant uncertainties

To further consider the practicalities of RL, in this section, we consider the effects of inaccurately estimating the real plant uncertainties for the policy synthesised via experiment E8, problem instance 1. Specifically, we assume that in the offline model of the plant dynamics and plant uncertainties are as previous. While in the following experiments M1-8, we assume that actual plant uncertainties are different from that of the offline model. The processing time uncertainty has the same form (a discrete uniform probability distribution), but with the upper and lower bounds misspecified by an additive constant, $k_{pt} \in \mathbb{Z}_+$. Similarly, the rate of the Poisson distribution descriptive of the due date uncertainty is misspecified. Here, however, the misspecification is treated probabilistically, such that the rate is either unperturbed or misspecified by a constant, $k_{dd} \in \mathbb{Z}_+$ (this is either added or subtracted), with uniform probability. See D.5 for further details. In the respective experiments M1-8, k_{pt} and k_{dd} are defined to investigate the effects of different degrees of plant-model mismatch. In all cases, the policy identified from experiment E8 in problem instance 1 was rolled out in a plant where the true uncertainties are misspecified for 500 MC simulations. The experiments

and their results are detailed by Table 5.6.

Table 5.6: Table of experimental conditions investigated. In each experiment, we take the trained policy from experimental condition E8, problem instance 1 and evaluate its performance in a plant defined by different uncertainties. The degree of misspecification increases with experiment number.

Reference	Processing time, k_{pt}	Due date, k_{dd}	μ_Z	σ_Z	$\bar{\mu}_\beta^\phi$	F_{LB}
Model (E8)	0	0	-75.3	11.5	-101.1	0.99
M1	0	1	-74.8	12.3	-101.3	0.98
M2	0	2	-73.8	11.8	-99.9	0.99
M3	1	0	-76.0	13.3	-106.8	0.95
M4	1	1	-77.3	13.2	-104.4	0.95
M5	1	2	-78.0	14.8	-105.6	0.95
M6	2	0	-83.9	17.6	-121.7	0.78
M7	2	1	-81.5	17.5	-116.2	0.82
M8	2	2	-83.4	18.7	-118.5	0.82

From Table 5.6, it is clear that misspecification of plant uncertainties generally impacts the performance of the policy identified. The misspecifications identified are relatively mild in the case of due date uncertainty, however, they are more pronounced in the case of processing times. This is reflected in the results of the experiments, where little variation is observed in the policy performance when $k_{pt} < 2$. For example, the maximum deviation in terms of expected performance, μ_z, when this condition is not imposed, is observed in experiment M5 where the corresponding decrease in μ_z from the performance observed in the offline model (experiment E8) is just 3.6%. Similarly, the probability of constraint satisfaction, F_{LB}, remains high (slightly decreases to 0.95). Notably in experiments M1 and M2, the policy performance increases. As the estimated values for due dates are maintained in the state (fed to the policy), the misspecification is unlikely to induce a significantly different state distribution than the model and hence the result should be interpreted with caution (i.e. it likely arises from the statistical inaccuracies of sampling).

The effects on policy performance become notable when the processing time misspecification, $k_{pt} = 2$. In all experiments M6-8, where this condition is imposed, the decrease in expected performance, μ_Z, is in the region of 10%. Additionally, the magnitude of the CVaR for $\beta = 0.2$ (i.e. the expected value of the worst 20% of policy performances) decreases by 15-20%, and this is mirrored further by a considerable decrease in the probability of constraint satisfaction, such that $F_{LB} \approx 0.8$.

From the analysis above, it is clear that accurate estimation of the true plant uncertainties is important if RL is to be applied to a real plant. However, it does appear that RL is likely to be robust to small amounts of error, which may be unavoidable given the limitations of finite plant data.

5.6 Conclusions

Overall, in this work, we have presented an efficient RL-based methodology to address production scheduling problems, including typical constraints imposed derived from standard operating procedures and propositional logic. The methodology is demonstrated on a classical single stage, parallel batch scheduling production problem, which is modified to enable comparison between discrete-time and continuous-time formulations as well as to consider the effects of uncertainty. The RL methodology is benchmarked against a continuous-time MILP formulation. The results demonstrate that the RL methodology is able to handle constraints and plant uncertainties effectively. Specifically, on a small problem instance RL shows particular promise with performance improvements over the MILP approach by up to 4.7%. In the larger problem instances, the RL approach had practically the same performance as MILP. Further, the RL methodology is able to optimise for various risk-aware formulations and we demonstrate the approach is able to improve the 20% worst case performances by 2.3% on average. The RL is orders of magnitude (150-500 times) cheaper to evaluate online than the MILP approach for the problem sizes considered. This will become even greater when larger problem sizes are considered or if compared to discrete-time scheduling formulations. Finally, the RL approach demonstrated reasonable robustness to misspecification of plant uncertainties, showing promise for application to real plants. It should be emphasised however, that there is no notion of robustness guarantee associated with RL as may be seen in (McAllister et al., 2022). This is clearly an area worthy of future investigation.

In future work, we will consider the translation of this methodology to larger problem instances, multi-agent problems, and integration with other PSE decision-making functions, such as maintenance and process control. Additionally, we will look to investigate efficient frameworks for updating the policy function in the case of highly

variable demand patterns and generating the schedule for the entire time horizon.

Chapter 6

Distributional reinforcement learning for optimisation of multi-echelon supply chains

This research item has been submitted to the Industrial Engineering and Chemistry Research (IECR) Journal:

Wu, G., de Carvalho Servia, M. A., Petsagkourakis, P., Zhang, D., Del Rio Chanona, E.A., Mowbray, M., 2022. Distributional reinforcement learning for inventory management in multi-echelon supply chains, Submitted to Journal, 2022:

The paper was produced through supervision of a student (Guoquan Wu) who interned with the research group after completing his MSc at the Department of Chemical Engineering, Imperial College London. Max Mowbray is the corresponding author of this paper and responsible for the original draft.

6.1 Introduction

Supply chain management and optimisation are key features of the operation and development of sustainable industrial production systems. The outbreak and spread of the COVID-19 pandemic has brought great challenges to governments, enterprises, medical institutions and citizens. Throughout the pandemic, the global supply chain has been greatly impacted, and there have been sudden shortages of various commodities, ranging from toilet paper (Fisher, 2020), medical materials, computer chips, cars and other products (Swanson, 2020). As a result, the pandemic has emphasized the importance and relevance of supply chain operations, which has inspired the academic community to develop methods for supply chain management and robust optimisation under uncertainty. For example, Remko (2020) made a preliminary empirical exploration of the potential effects of COVID-19 on the supply chain, and proposed practical methods to enhance its resilience. Paul and Chowdhury (2020) built a supply chain model that can adapt to supply and demand disruptions caused by the pandemic. Ibrahim et al. (2021) developed a new optimisation-based vaccine supply chain model and embedded the model into a system framework to support the planning and delivery of vaccination campaigns for pandemics.

The traditional research methods of operational research are usually divided into two steps. The first is to, establish a mathematical model of the problem to be solved, including system dynamics, constraints and objective function; and the second is to design algorithms to solve the model, such as branch and bound (Karimi and Davoudpour, 2015), tabu search (Melo et al., 2012), genetic algorithms (Kannan et al., 2010) and linear programming solvers (Peidro et al., 2010). These approaches have been widely used in supply chain management and achieved state-of-the-art results.

With the development of new frameworks for manufacturing and the decreasing prices of information storage devices, various industries have accumulated a large amount of historical data. These valuable data contain realizations of uncertain supply chain dynamics, which explicitly describe the operational problem at hand. It is important to utilise these data to find rules and learn strategies to improve supply chain optimisation solutions. For example, Gao et al. (2019) proposed a two-stage

distributionally robust optimisation model, which takes advantage of the data availability in industry to address the optimal design and operations of shale gas supply chains under uncertainty and ambiguity.

In recent years, many researchers have proposed methods to deal with uncertainty in supply chain and scheduling problems. Santos et al. (2021b) developed a rolling horizon framework for the integrated personnel allocation and machine scheduling in large-scale industrial facilities. Tsay et al. (2019) used a data-driven approach to study scheduling-related dynamics from historical data of industrial processes, which is capable of describing the dynamic characteristics of industrial air separation units and its model predictive control system with accuracy. In addition, to dealing with long-term generation expansion planning under uncertainty, Lara et al. (2020) decomposed the problem via stochastic dual dynamic integer programming and then used parallelization to enhance the computation. By using a novel multi-parametric nonlinear programming algorithm, Charitopoulos et al. (2019b) developed a strict rescheduling mechanism to reduce the impact of dynamic interruptions on planning and scheduling operation decisions. However, for large-scale and complex supply chain systems, the traditional operations research methods still face many difficulties in practical application. Firstly, the solution space is often large and often supply chain scenarios involve many nodes with complex network relationships. As a result, the solution time is often large as the constructed model is typically subject to the phenomenon of combinatorial explosion with increasing problem size causing the time and space complexity of computation to increase exponentially (Lee et al., 2018b). The second challenge is that of large operational uncertainty. There are various uncertain factors in the operation of the supply chain, such as demand, price fluctuations and production uncertainty (Emenike and Falcone, 2020). Coping with uncertainty is one of the most important issues in supply chain systems. As the evolution of the system state is uncertain, deterministic (or nominal) supply chain models can often be associated with large predictive errors. This means many current approaches based on mathematical programming are often suboptimal because they explicitly require deterministic, finite-dimensional expressions in the underlying model, and those that explicitly account for uncertainty are intractable for large problems or online solutions.

Reinforcement learning (RL) is a subfield of machine learning that computes an

optimal decision-making policy by interacting with the underlying system via an iterative approach. The RL problem is formalized as a Markov Decision Process (MDP), which provides an effective framework for modelling uncertain, sequential decision-making problems (Puterman, 2014b). As RL has demonstrated impressive results in sequential decision-making tasks such as Go (Silver et al., 2017a), ATARI games (Mnih et al., 2015a) and engineering (Li, 2017), it has also attracted attention in the field of supply chain optimisation. RL can overcome the shortcomings of traditional operations research methods: it allows for flexible use of different classes of model and forms of uncertainty. As the process system state transitions, the controller (or policy) can make use of state feedback to produce reactive control solutions (Caputo and Cardin, 2022a). Another benefit is that RL does not need to explicitly identify the conditional probability density function descriptive of discrete time system dynamics, but instead uses simulation methods (i.e., Monte Carlo) to approximate it. This is especially beneficial, because such explicit knowledge of discrete time dynamics is typically unavailable in OR applications.

In view of the above characteristics, RL has been applied to supply chain management in recent years, and a series of excellent results have been demonstrated. For example, Oroojlooyjadid et al. (2017) introduced the latest deep Q-network (DQN) into a multi-agent cooperative supply chain problem and developed a cooperation framework with transfer learning to achieve knowledge transfer between controllers. The simulation results show that the DQN algorithm can obtain near-optimal results and with favorable performance relative to a base-stock policy. Gijsbrechts et al. (2021) applied the Asynchronous Advantage Actor Critic (A3C) algorithm to lost sales, dual-sourcing, and multi-echelon inventory models, and introduced a real data set for algorithm validation. The results show that the inventory cost calculated by the A3C algorithm is close to the optimal solution, and matches the performance of state-of-the-art heuristic methods. Kara and Dogan (2018) implemented RL to solve a supply chain system of perishable products, and compared SARSA and Q-learning algorithms with genetic algorithms. The results show that a better solution can be obtained by using an RL algorithm despite great changes in customer demand and short product life.

Although RL has made great achievements in the research of supply chain optimisation problems, most literature uses Q-learning and policy gradient algorithms to identify a solution policy. When combined with nonlinear function approximation (as is standard practice), these algorithms are generally unstable in training, get easily trapped in low-laying local optima and in most cases, are unable to provide guarantees of convergence to local solutions. Furthermore, problems related to training stability often arise, such as the vanishing or exploding gradient problem, leading to unsatisfactory results. More effective and robust policy learning methods are required to resolve these perceived shortcomings of gradient-based RL approaches. A potential means of achieving this is provided by derivative-free RL methods.

Derivative-free RL methods provide another way of policy learning for RL. A direct way of applying derivative-free optimisation methods to RL is to define the search space as the parameter space of the policy, and the objective function as the expected cumulative reward. For policy learning, derivative-free RL approaches have their own advantages. For example, they are invariant to control frequency and delayed rewards, tolerant of extremely long horizons, and do not need to identify search directions via backpropagation (Salimans et al., 2017b). However, it is also well known that 'vanilla' stochastic search algorithms are inefficient in high dimensions. This has led to the development of advanced algorithms tailored to address this problem (Memmel et al., 2022).

In general, evaluating the average cumulative reward and standard deviation of the reward over multiple episodes is sufficient to determine how well the RL algorithm performs a specific task. However, RL algorithms often show great differences in performance in different episodes, which reduces their reliability in terms of performance consistency. In addition, in some cases, minimum performance thresholds may always be required to avoid risk (Waubert de Puiseau et al., 2022). Therefore, a key challenge is how to ensure that the learned policy is safe, robust and performance consistent, which requires different measures of the performance distribution under RL policies.

Recent research on risk-sensitive policy learning draws inspiration from risk-measures common in finance, such as using value-at-risk (VaR) or conditional value-at-risk (CVaR) as the criterion to optimise. These approaches are part of a subfield of RL research, known as distributional RL (Bellemare et al., 2022). The VaR and CVaR

can be efficiently optimised using different mathematical programming formulations (Rockafellar and Uryasev, 2000). The utility of CVaR in non-finance applications has been fully discussed by Filippi et al. (2020). Recently, Ma et al. (2021) proposed conservative offline distributional actor critic (CODAC), an offline RL algorithm suitable for both risk-neutral and risk-averse domains by considering CVaR. Tang et al. (2019b) developed worst case policy gradient (WCPG) to learn risk-sensitive policies by maximizing CVaR. The learned policy was demonstrated to be risk averse and robust in an autonomous driving simulation. Furthermore, Mowbray et al. (2022b) presented a distributional RL framework to address precedence and disjunctive constraints on production scheduling problems, which is able to optimise for various risk-aware formulations. The results demonstrate that the method can improve the 20% worst case performances by up to 4.5% over an expected formulation, whilst providing comparative expected performance to a nominal, online MILP approach.

Drawing inspiration from the above works and to address current shortcomings in supply chain optimisation, in this work we propose a hybrid stochastic search algorithm for policy optimisation and risk sensitive formulations for distributional RL. Hybridization of stochastic search is well-known and has demonstrated performance improvements in many problems (Estrada-Wiese et al., 2018). Our method combines four stochastic optimisation algorithms: evolution strategy (ES) (Hansen et al., 2015), artificial bee colony (ABC) algorithm (Karaboga, 2005), particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995b) and simulated annealing (SA) (Kirkpatrick et al., 1983b). The composition of these methods allows one to balance exploitation and exploration in the search process. An additional search space reduction strategy is implemented to accelerate the optimisation process. Furthermore, we propose a novel distributional RL method based on this hybrid search algorithm to identify a risk-sensitive policy. Finally, we use a multi-echelon supply chain inventory management problem and two common operations research studies as benchmark test cases to demonstrate the effectiveness of our proposed method. The performance of the method proposed is compared with state-of-the-art algorithms like receding horizon MIP and proximal policy optimisation (PPO).

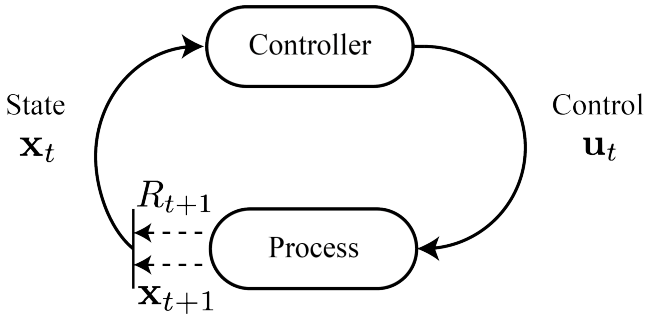


Figure 6.1: Interaction between the controller and stochastic process in a Markov decision process.

6.2 Preliminaries

6.2.1 Introduction to Reinforcement Learning

RL encompasses a set of methods, which approximately solve a Markov decision process (MDP), which is shown in Fig. 6.1 (Sutton and Barto, 2018a). MDPs, can be expressed as a 5-tuple $< \mathbb{U}, \mathbb{X}, P, R, \gamma >$, which includes the control space, $\mathbb{U} \subseteq \mathbb{R}^{n_u}$, state space $\mathbb{X} \subseteq \mathbb{R}^{n_x}$, discrete time state transition probability, $P : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \rightarrow [0, \infty]$, reward $R : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \rightarrow \mathbb{R}$ and discount factor $\gamma = (0, 1]$. The optimal policy in an MDP maximizes the expected sum of rewards over a finite, discrete time horizon. The MDP framework is outlined by Fig. 6.1.

From a high level, the implementation process of RL is as follows: a controller observes the system and its current state $\mathbf{x}_t \in \mathbb{X}$ at the time index t, and then takes an action according to $\mathbf{u}_t = \pi(\mathbf{x}_t)$. At the next time index $t+1$, the system then transitions to a new state, $\mathbf{x}_{t+1} \in \mathbb{X}$, and discrete-time system evolution is evaluated by a reward, $R_{t+1} \in \mathbb{R}$. The decision-making process trajectory, $\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, R_1, \ldots, \mathbf{x}_T, R_T)$ within the MDP.

Through a learning process, the controller finds the optimal policy π^* that maximizes the long-term cumulative reward. After time t, the long-term cumulative reward G_t with discount factor $\gamma = (0, 1]$ is defined as follows:

$$\begin{aligned} G_t &= R_{t+1} + \gamma \mathrm{R}_{t+2} + \ldots + \gamma^{T-1} R_T \\ &= R_{t+1} + \gamma \mathrm{G}_{t+1} \end{aligned} \tag{6.1}$$

The state value function $V^\pi(\mathbf{x})$ and the state-action value function $Q^\pi(\mathbf{x}, \mathbf{u})$ can

be expressed as follows:

$$V^{\pi}(\mathbf{x}) = \mathbb{E}\left[G_t | X_t = \mathbf{x}\right] \tag{6.2}$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u}) = \mathbb{E}\left[G_t | X_t = \mathbf{x}, \mathbf{u}_t = \mathbf{u}\right] \tag{6.3}$$

Traditional first order RL relies on the accurate estimation of gradient-based directions for policy improvement. However, evaluation of directions for policy improvement is generally noisy and computationally expensive (Riedmiller et al., 2007b). In addition, the update directions tend to be approximations of the gradients, which can have implications for convergence, as explored in Nota and Thomas (2019b); Tran et al. (2022); Ziemann et al. (2022). Through the advent of deep learning, classes of flexible and powerful models (e.g., neural networks) are available for use in RL. It is worth mentioning that although the gradient for policy improvement can be estimated, the RL objective is often a complex (i.e., nonlinear or non-smooth) function of the policy parameters. This is especially true when using a deep neural network model (that is not extremely large). For this type of optimisation problem, gradient-based RL methods may get stuck in local optimal solutions and face the gradient vanishing problem in the optimisation process (Qian and Yu, 2021), leading to unsatisfactory final results. A promising approach that circumvents these issues is the use of stochastic search optimisation algorithms within the RL framework.

6.2.2 Reinforcement Learning and stochastic search optimisation

Stochastic search optimisation for RL is a form of derivative-free RL. For policy learning, stochastic search algorithms have several advantages, for example, they often have stronger exploration ability and policy learning is easily parallelized (computationally).

The fitness function is the core of stochastic search algorithms, and in the RL case it represents the individual policy's fitness when evaluated in the uncertain process system. Here, the fitness represents the quality of each individual policy with respect to the process objective. We use the value of the long-term return in RL as the fitness function of the stochastic search algorithm, which is shown in Eq. 6.4:

$$\begin{aligned} F_{\text{fitness}}(\theta) &= \mathbb{E}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau})}[G_0] \\ &= \mathbb{E}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau})}[R_1 + \gamma R_2 + \ldots + \gamma^{T-1} R_T] \end{aligned} \tag{6.4}$$

where $\boldsymbol{\tau} \sim p(\boldsymbol{\tau})$ represents the distribution of trajectories induced under the current policy π; $\underline{\mathrm{T}}$ is the length of episode; and G_0 represents the total returns of an episode. Instead of calculating the state value function corresponding to each state, we directly calculate the overall evaluation value G_0 of each policy via the sample average approximation. Through this method, the effect of sparse immediate rewards – a common issue of state-action value and policy gradient methods – is avoided. Sparse reward problems are characterized by non-informative feedback from the reward function for most state transitions.

Despite these advantages, few researchers have applied stochastic search algorithms to policy learning. The OpenAI team has used evolutionary strategies to directly optimise the weights of the policy network (Salimans et al., 2017b). The experimental results on Atari games and MuJoCo control tasks show that ES algorithm has better exploration behavior than other policy gradient methods such as trust region policy optimisation. Choromanski et al. (2018) enhanced the parameter update of the ES algorithm through structural evolution and the use of intensive policy networks. The proposed algorithm uses fewer policy network parameters, so it can speed up training and inference. Such et al. (2017) used the genetic algorithm to solve the RL task. The algorithm iteratively maintains a policy population, updates the individuals in the population by a mutation operator, and then uses elitism to select the next generation of the population. The experimental results on the control tasks of Atari game and MuJuCo show that genetic algorithm is effective and competitive compared with popular first order RL methods. In addition, Memmel et al. (2022) proposed a method for dimensionality reduction and prioritized exploration with application to policy search algorithms in black-box optimisation, which enables more efficient learning of policies.

6.2.3 Introduction to Distributional Reinforcement Learning

As reinforcement learning is applied to diversified problem instances in the real world, it is common to observe parametric and other general uncertainties. In general, RL algorithms are designed to maximize the expected sum of rewards. However, maximizing in expectation is not risk-sensitive because it does not explicitly account for low probability, worst-case events (Tang et al., 2019b). In recent years, Bellemare et al. (2017a) introduced distributional RL, which instead studies the complete distribution

of future returns. There are two main drivers for research in distributional RL. The first focus is to find better distributional parameterizations to improve approximation of the actual distribution associated with the return of a given state and control. This is demonstrated in a number of works (Dabney et al., 2018d; Yang et al., 2019). The second is to model the different statistical characteristics of the distribution and try to identify the modeling methods that are more suitable for RL tasks, such as learning the quantile value of the distribution (Dabney et al., 2018a) or learning the expectile value of distribution as in Expectile distributional RL (Rowland et al., 2019). A typical application direction is risk-sensitive scenarios (Ma et al., 2020; Zhang et al., 2020c). For example, Min et al. (2019) applied distributional RL to the autonomous highway driving problem, and the results show that policy trained by distributional RL drives more efficiently, safely and faster. These requirements are all paralleled in the process industry.

6.3 Methodology

In this work, the true system is assumed to be described by a stochastic process with the Markov property (i.e., discrete time system evolution depends only on the current state and control action). Our strategy seeks to use stochastic search algorithms to find the optimal policy for a supply chain system under the presence of uncertainty. Additionally, we leverage the use of risk sensitive measures to allow for the identification of a more robust and risk-averse policy.

In practice, the workflow for the identification of RL controllers is as presented in Mowbray et al. (2022a). Specifically, due to the data intensity of RL, we conduct an initial policy search via offline simulation of an approximate process model. Once a policy has been identified in the offline model, it may then be transferred to the real system for the purposes of online decision-making. This workflow is depicted by Fig. 6.2.

6.3.1 Stochastic search for Reinforcement Learning (SS-RL)

Different stochastic optimisation algorithms exhibit different characteristics. For example, the ES emphasizes the evolutionary ability of groups, the ABC algorithm

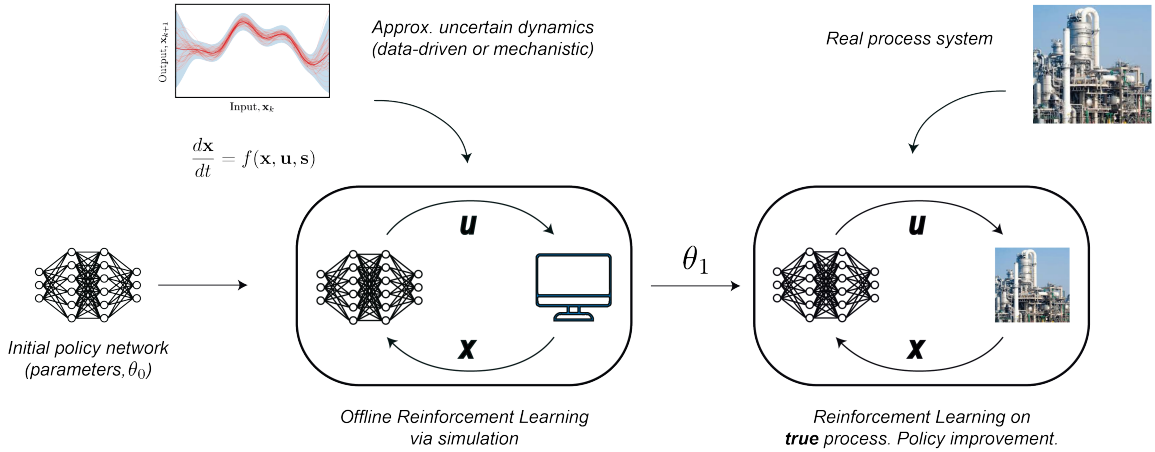


Figure 6.2: Reinforcement Learning for process systems and supply chain optimisation.

emphasizes cooperation between groups, and the PSO algorithm emphasizes group learning. Therefore, some algorithms are better at exploring the search space, whereas others are better at exploiting it. The tradeoff between exploration and exploitation has proved to be the key to affect optimisation performance (e.g., the accuracy and convergence speed of stochastic optimisation algorithms) (Chen et al., 2009; Črepinšek et al., 2013). Balancing the exploration and exploitation of the search space has been effectively demonstrated in some algorithms. For example, del Rio-Chanona et al. (2019) proposed a stochastic search algorithm that combines SA, PSO and random search, and Anye Cho et al. (2021) combined ABC and PSO for optimizing nonlinear ordinary differential equations. Both of their strategies are developed by creating a hierarchical system where the best solutions are given to the most exploitative algorithms, whereas the worst solutions are given to the most explorative algorithms (Ali and Tawhid, 2017).

Aside from providing a good exploitative-explorative balance, it is also important to ensure that the stochastic search algorithm is as efficient as possible. Park et al. (2005b) proposed a space-reduction strategy that improved the efficiency of their PSO algorithm. Therefore, we have combined hybridization and space reduction strategies in tailoring our stochastic search algorithm, which uses SA, ES, PSO and ABC. The steps of this algorithm are described as follows (following from Algorithm 6.1):

Step 1, Initialization: In this phase, $\underline{\mathrm{N}}$ particles are randomly placed in the parameter space of the neural network, $\theta \in \mathbb{R}^{n_\theta}$.

Step 2, Evaluation and Classification: The fitness function value of each

particle is evaluated (see Algorithm 6.2), and the particles are sorted into four groups of equal size (i.e. 25% are allocated to each), depending on the quality of their evaluation. The group to which they are assigned determines which stochastic search algorithm is used to subsequently perturb the particle position.

Step 3, Space Exploration and Exploitation: The above classification ensures a good balance between exploitation and exploration within the algorithm's framework. The N_{SA}, N_{ES}, N_{PSO} and N_{ABC} particles are used as inputs for SA, ES, PSO and ABC algorithms, respectively. For a detailed description of the SA, ES, PSO and ABC algorithm, see E.1 - E.4.

Step 4: During the search process, the individual best position θ_{b}^{n} and the global best position θ^{*} are recorded.

Step 5, Repeat: After N_c iterations the algorithm either returns to the **Step 2** or terminates.

Step 6, Space Reduction: We gradually restrict the search space to a high performing region of the parameter space. After every N_s iterations, the search space is dynamically adjusted according to the distance between the θ^{*} and current bounds on the search space.

Algorithm 6.1 Hybrid stochastic search algorithm

Input: The number of maximum search iterations M; dimensionality J of the optimisation problem; population size N; upper bound θ^{UB} and lower bound θ^{LB} on the search space; number of iterations N_c to reclassify the particles; number of iterations N_s to reduce search space and step size Δ;

1. Initialization: Randomly generate the position $\theta_{i,j}$ of N particles, $i \in \{1, \ldots, N\}$,$j \in \{1, \ldots, J\}$, and evaluate each particle by Algorithm 6.2

2. Sort the N particles in descending fitness function value, then classify them into four groups from best to worst performing: N_{SA}, N_{ES}, N_{PSO}, N_{ABC}.

for $m = 1, \ldots, M$ **do**

3. Pass the N_{SA}, N_{ES}, N_{PSO} and N_{ABC} particles to the SA, ES, PSO and ABC algorithms, respectively, to update their respective positions. All individual's positions are subject to upper and lower bounds, such that $\theta_{i,j} = \left[\theta_{\mathrm{j}}^{\mathrm{LB}}, \theta_{\mathrm{j}}^{\mathrm{UB}}\right]$.

4. Evaluate all particles via Algorithm 6.2 and keep record of the current individual best position θ_{b}^{n} $\forall n$ and the global best position θ^{*}.

if $m \mod N_c = 0$ **then**

5a. Conduct **step 2** and then continue.

else

5b. Continue.

end if

if $m \mod N_s = 0$ **then**

6a. Update upper bound: $\theta^{\mathrm{UB}} := \theta^{\mathrm{UB}} - (\theta^{\mathrm{UB}} - \theta^{*}) \times \Delta$ and then continue.

6b. Update lower bound: $\theta^{\mathrm{LB}} := \theta^{\mathrm{LB}} + (\theta^{\mathrm{LB}} - \theta^{*}) \times \Delta$ and then continue.

else

6c. Continue.

end if

end for

Output: Global best position of the population θ^{*} and objective function value $F_{\mathrm{fitness}}(\theta^{*})$

In this work, a recurrent neural network (RNN) is used to parameterize our policy in RL as it allows for the use of prior information for decision-making. The RNN must be trained to adjust its weights so that the output corresponds to an optimal control action. In this case, the controls are a deterministic function of state, however, the network could be modified to enable construction of a stochastic policy. A brief

introduction to RNN's and how they have been implemented in this work is given in Appendix E.5. Since the stochastic search algorithm proposed has great explorative characteristics due to its structure and choice of sub-algorithms, it is tailored to undertake the responsibility of exploring the control and state space in RL. In addition, the proposed algorithm does not need to estimate a gradient for policy improvement, which effectively reduces the number of computations per episode and mitigates the common exploding gradient problem observed in RNNs. To this end, we use the above optimisation algorithm (Algorithm 6.1) to optimise the RL policy . N particles in the stochastic search Algorithm 6.1 represents N policy networks, and the position of particles represents the weight of policy networks. The steps for evaluating each candidate policy are outlined in Algorithm 6.2. A summary of the hybridized stochastic search algorithm proposed in this work, is presented in Fig. 6.3.

Algorithm 6.2 Evaluation of each candidate policy over a finite discrete time horizon

---

Input: The number of episodes K to evaluate each candidate policy; policy function $\pi(\theta, \cdot)$ to be evaluated; fitness function, $F_{\text{fitness}}(\theta, \cdot)$; and memory store, $\mathcal{B}$;

for $k = 1, ..., K$ **do**

1a. Generate initial state $\mathbf{x}_0^k$.

for $t = 0, ..., T-1$ **do**

1b. Observe state variable $\mathbf{x}_t^k$, identify control variables $\mathbf{u}_t^k = \pi\left(\mathbf{x}_t^k; \theta\right)$, simulate uncertain discrete time process dynamics, observe $\mathbf{x}_{t+1}^k$ and evaluate reward, R_{t+1}^k. Hold information in memory store, $\mathcal{B}$.

end for

1c. Calculate the returns G_0^{k} of an episode (see Eq. 6.1) and store in $\mathcal{B}$.

end for

2. Collect policy performance information from $\mathcal{B}$, provided by the returns of the K episodes.

3. Evaluate the objective function value $F_{\text{fitness}}(\theta)$ of each candidate policy network.

Output: Objective function value $F_{\text{fitness}}(\theta)$ of candidate policy.

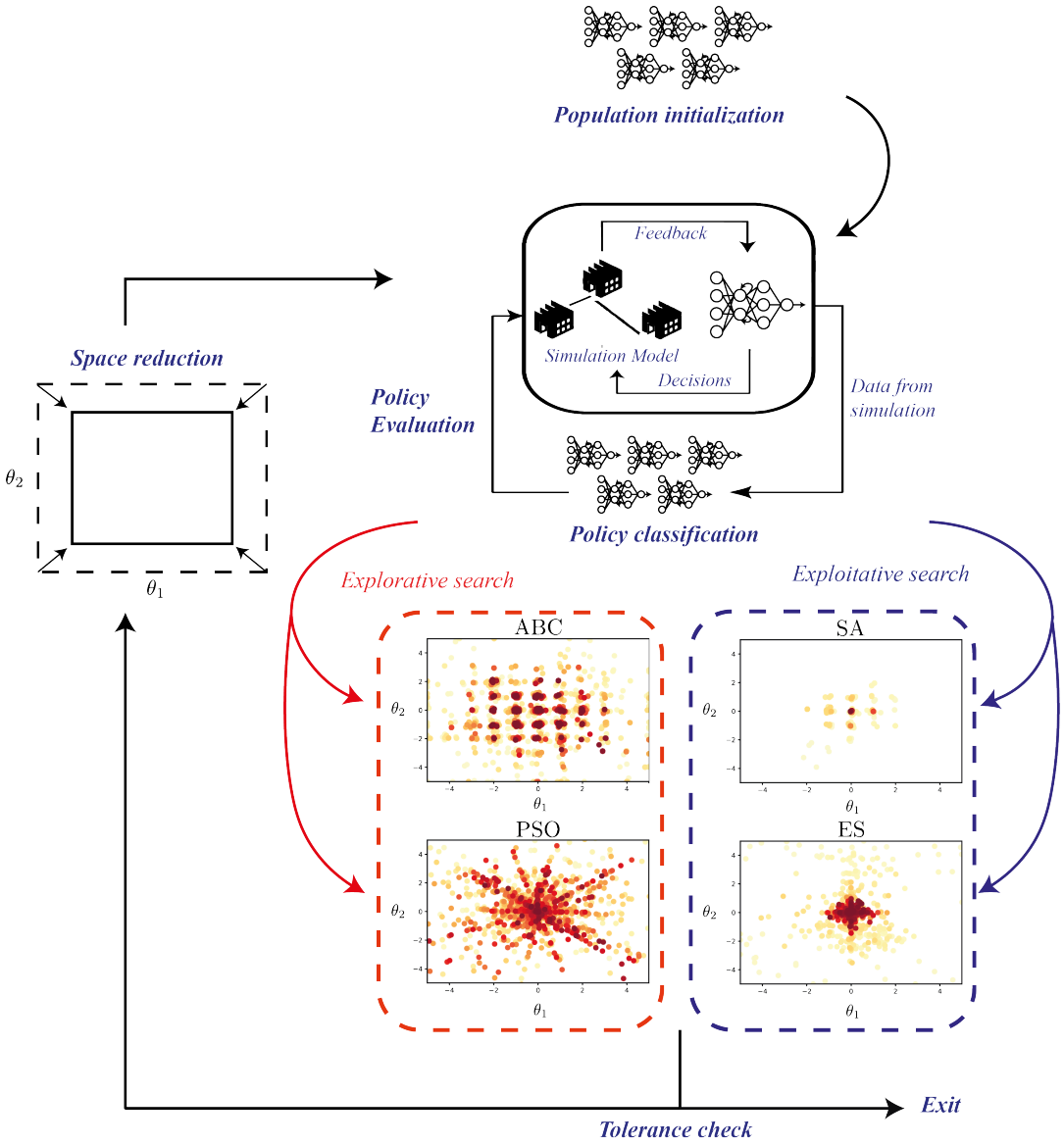


Figure 6.3: An overview of the stochastic search algorithm proposed.

Distributional Reinforcement Learning

When learning a policy for safety or operation-critical applications, it is important to consider risks and avoid catastrophic events. Only maximizing the expected value of return does not protect the decision-maker against low probability, high severity events. For this reason, we introduce the use of the value at risk (VaR) and conditional value at risk (CVaR) measures to optimise the supply chain. This approach utilises the empirical distribution of returns from a number of sampled episodes. This is otherwise known as distributional RL (Bellemare et al., 2022).

We start by first defining the VaR value V_α as the maximum possible return under a policy π. Let G be the random variable of return and $F_\pi(V_\alpha) = prob(G \leq V_\alpha)$ be the cumulative distribution function (CDF) of G. Specifically, for any $\alpha = (0, 1)$, the VaR is defined:

$$V_\alpha = \max\{g : F_\pi(g) \leq 1 - \alpha\} \Leftrightarrow V_\alpha = F_\pi^{-1}(1 - \alpha) \tag{6.5}$$

where α is the confidence level. For example, take the case of optimizing the total profit made by a supply chain: if the V_α is equal to £10 million and alpha is 0.9, it means that the maximum profit made by the supply chain under the policy with probability less than or equal to 0.1 will be £10 million. However, the definition of V_α does not quantify the quality of operational decision making within the tail of performances that are observed with probability less than or equal to $1 - \alpha$. A metric that does quantify this is the CVaR, which is defined as:

$$CVaR_\alpha = \mathbb{E}[G \mid G \leq VaR_\alpha] \tag{6.6}$$

In engineering applications, the optimisation of CVaR has advantages over VaR because it preserves convexity, as well as quantifying policy performance in the bottom (1-α)-percentile of the potential events. From the previous example, it follows by definition that the CVaR represents the expected profit associated with the outcomes observed with probability less than or equal to 0.1 (i.e. $1 - \alpha$).

As closed form expressions of the performance distribution under the policy are unavailable, we use Monte Carlo estimation (Hong and Liu, 2011) to compute the VaR and CVaR in RL. Suppose that $D = \left\{G_0^0, \ldots, G_0^K\right\}$ are $\underline{\text{K}}$ independent and identically distributed returns sorted in ascending order. Then, the VaR of G can be

estimated by:

$$V_\alpha(D) = G_0^{[K(1-\alpha)]:K} \tag{6.7}$$

where $G_0^{i:K}$ is the i^{th} worst return from the K observations expressed by D. Here, we are ordering the returns expressed by D in terms of increasing return and then taking the $K\,(1-\alpha)^{\text{th}}$ worst return. This is an approximation to the value-at-risk for probability level $1-\alpha$.

Then we can apply Eq. 6.7 to directly estimate CVaR as follows (Hong and Liu, 2011):

$$CVaR_\alpha\,(D) = V_\alpha(D) + \frac{1}{K\,(1-\alpha)} \sum_{i=0}^{K} \left[G_0^{\text{i}} - V_\alpha(D)\right]^- \tag{6.8}$$

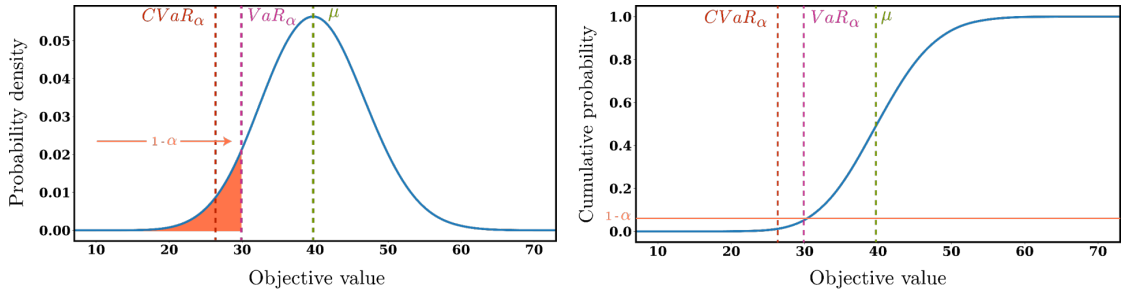
where $[a]^- = \min\{0, a\}$. Finally, we propose an objective function for RL (Eq. 6.9a), which optimises the expected value of return whilst taking into account the CVaR of the distribution. The optimisation problem is also given as Eq. 6.9b:

$$Z = \mathbb{E}[G_0] - \beta\,[b - CVaR_\alpha]^+ \tag{6.9a}$$

$$\pi^* = \arg\max_\pi \mathbb{E}[G_0] \quad s.t. \quad CVaR_\alpha \geq b \tag{6.9b}$$

where $[a]^+ = \max(0, a)$, β is the penalty factor, which is usually set to a large number to ensure that the learned policy satisfies the CVaR constraint, and $b \in \mathbb{R}$ is the constraint imposed on the CVaR. When the CVaR is less than b, the current policy will be penalized. For example, b could represent the minimum profit or loss one can make to ensure a company does not go bankrupt. A visualization of the VaR and CVaR is presented in Fig. 6.4. Fig. 6.4a) shows that by constraining or optimizing for CVaR, we seek policies that shift the tail-end of the distribution to the right, thereby reducing the probability of experiencing a very negative event. In addition, the distributional approach we propose does not need to make assumptions on the distribution of returns[1], whereas policy gradient methods generally do (Tang et al., 2019b). The steps of the stochastic search algorithm for distributional RL are outlined in Algs. 6.1 and 6.2.

[1]In other words, it is an unbiased estimate



(a) Probability density function

(b) Cumulative distribution function

Figure 6.4: Illustration of VaR_α and CVaR_α for a given probability level α under a) the probability density function and b) the cumulative distribution function.

6.4 Case studies

In this section, a multi-echelon supply chain problem from OR-gym (Hubbs et al., 2020b) is presented to illustrate the effectiveness of the proposed stochastic search algorithm for RL. Prior to presenting these results, we first benchmark our method on a virtual machine packing problem and a multi-period asset allocation problem, both of which are supply chain and operations research related problems. The computational work was carried out in Python on a HP Pavilion notebook with a Quad-core 6th Generation Intel i-5 processor with up to 2.3 GHZ and 8 GB of RAM. In all case studies the results are benchmarked by the PPO algorithm and online (mixed integer) linear programming formulations as detailed in Hubbs et al. (2020b). As the results used are leveraged from another work, we are unable to explicitly provide validation test results as evaluated for the proposed approach. However, the benchmarks are deemed appropriate because they are thoroughly justified and objective comparison can still be made with respect to the training results of PPO and the benchmarks for mathematical programming formulations provided in Hubbs et al. (2020b).

6.4.1 Virtual machine packing

Problem statement

The virtual machine packing problem (VMPP) can be regarded as a kind of bin packing problem, which is an NP-hard problem. Specifically, we benchmarked our algorithm on the multi-dimensional virtual machine problem in the OR-gym. At each control interaction (or, equivalently, each discrete time index within the finite, discrete time

horizon), an arriving virtual machine (VM) process must be allocated to one of $n_{\text{PM}} = 50$ physical machines (PMs) available. The goal is to minimise unused capacity on each PM in both the compute and memory dimensions over the horizon, which consists of 72 discrete time steps.

This problem is modelled as an MDP $(\langle \mathbb{X}, \mathbb{U}, P, R(\cdot), \gamma \rangle)$. The state, $\mathbf{x} \in \mathbb{X} \in \mathbb{R}^{2n_{PM}+3} \times \mathbb{Z}^{n_{PM}}$ is comprised both by discrete and continuous components. Specifically, it is composed of a binary representation of whether each of the PMs are available, the respective CPU and memory load of the PM at the current time, and the current CPU and memory demands of the incoming VM process. The control, $\mathbf{u} \in \mathbb{U} \subset \mathbb{Z}_+$, is to choose from one of the 50 PMs to allocate the incoming VM process to and is discrete.The PM's dynamics are deterministic and are functions of current state and control. However, the CPU and memory demands of the incoming VM are realized probabilistically, hence the dynamics are uncertain. Specifically, the CPU demand uncertainty is exogenous and described by a Gaussian distribution, with mean and variance equal to 0.16 and 1.5×10^{-4}, respectively. The memory demand uncertainty is again exogenous and described by a discrete uniform distribution with a lower bound of 0.2 and upper bound of 1. The reward function, R, is defined as the sum of differences between the current size and excess space (in terms of CPU and memory) for all PMs. Constraints are imposed to enforce that memory and CPU limits of the PMs are not violated (both upper bounds are equal to one arbitrary unit of space). In practice, this is achieved simply by allocating a large penalty to the violation (of -1000), which is not conditional to the magnitude of the constraint violation. This is something defined inherently by the OR-gym library, but could potentially induce conservative behavior from the RL controller. The rewards are undiscounted over the finite horizon, such that the discount factor, $\gamma = 1$.

The results of the approach proposed are benchmarked to mathematical programming methods and the policy gradient method, PPO. For more details of the parameters associated with the simulation model, and the respective benchmarks please see OR-gym (Hubbs et al., 2020b).

Results and discussion

Here, we simply evaluate RL when optimizing in expectation and leave investigation of the CVaR formulation (i.e. Eq. 6.9b) for Section 6.4.2 and 6.4.3. We compare the training results of the hybrid stochastic search RL algorithm with each constituent stochastic search algorithm, including PSO-RL, ABC-RL, and ES-RL (i.e. policy optimisation by PSO, ABC and ES, respectively). We benchmark all results to the PPO and MILP implementation detailed in OR-gym. All stochastic search RL algorithms are trained under the same number of total training episodes (16000) as this is in line with the training routine detailed for PPO by OR-gym. This enables objective comparison of the sample efficiencies of the two approaches. Once the training has terminated, the policies identified by the respective stochastic search algorithms are subsequently validated over 1000 episodes The results are detailed by Table 6.1.

The sample size of each candidate policy during stochastic search RL training is set to 100. The hybrid algorithm quickly learns an effective policy, with the optimal policy identified observing an expected performance of -450 over 100 episodes. This is a performance improvement of 12.0% relative to the expected score of -511 for the PPO algorithm. Further, this result is comparable to the performance of a shrinking horizon MILP benchmark, which attained an expected performance of -439 per episode, but with online decisions identified more efficiently from the function mapping provided by the policy. This highlights RL's potential for online combinatorial optimisation problems. Also, in the 1000 Monte Carlo simulations used for validation, the expected performance of an episode is -462, which indicates the sample size selected in training is appropriate.

Table 6.1: Total reward comparison of different RL algorithms in virtual machine packing. We report the mean and standard deviation (StD). The marker * indicates benchmark results acquired from Hubbs et al. (2020b). In this study, Hubbs et al. (2020b) provide training results only, which enable objective comparison between MILP, PPO and stochastic search RL.

		Solution method					
Evaluation		MILP*	PPO*	ABC-RL	ES-RL	PSO-RL	Hybrid-RL
Training	Mean	-439	-511	-460	-452	-463	-450
	StD	111	110	95	97	107	91
Validation	Mean			-480	-472	-479	-462
	StD			101	104	100	89

6.4.2 Asset allocation

Problem statement

In this section, we evaluated our algorithm on a multi-period asset allocation problem in the OR-gym. Asset allocation identifies an investment strategy that aims to balance risk and reward by apportioning a portfolio's assets, subject to some investment standards and constraints. In this RL environment, the portfolio initially consists of \$100 in cash and some initial holdings of $n_A = 3$ assets. The goal is to maximize the value of the portfolio over 10 time periods.

The problem is modelled as an MDP. The state variables, $\mathbf{x} \in \mathbb{X} \subset \mathbb{Z}_+ \times \mathbb{R}^{2n_A+1}$, include the current time index within the horizon, the current portfolio holdings of cash and assets, as well as the current prices of the assets. The control variables, $\mathbf{u} \in \mathbb{U} \subset \mathbb{R}^{n_A}$, represent decisions regarding how much to buy and sell of each asset. These decisions are bounded such that one can only buy or sell at most 2000 units of each asset at a given time index within the horizon. The evolution of the portfolio holdings is a deterministic function of the current asset prices, control decisions and cost associated with buying or selling the asset. However, the asset prices themselves are uncertain (additionally, they are not influenced by decision making and hence exogeneous). Specifically, price uncertainty is assumed Gaussian, $\mathcal{N}\left(\boldsymbol{\mu}\left(t\right), \Sigma\left(t\right)\right)$, with time varying mean, $\boldsymbol{\mu}\left(t\right) = \left[1, 10\right]^{n_A}$ and variance, $\Sigma\left(t\right) = diag\left(\left[\sigma_1^2, \ldots, \sigma_{n_A}^2\right]\right)$, with $\sigma_i\left(t\right) = 0.45$ where $\underline{i}$ indexes each asset. When all prices are known for certain in advance, the solution to this deterministic optimisation problem is \$17,956.20. However, market prices fluctuate on a daily basis, so this value has little practical reference other than providing an upper bound on the objective. The environment is also characterized by a sparse reward landscape, where the portfolio value is assigned as a reward at the end of an episode. The rest of the rewards allocated during the control horizon are equal to zero (Hubbs et al., 2020b).

Results and discussion

Similar to Section 6.4.1, here we evaluate the stochastic search RL methods relative to a robust linear programming (RLP) formulation and the PPO algorithm via the results detailed in OR-gym. Again, we utilise a total number of training episodes

for each algorithm that is consistent with the training budget of the PPO algorithm demonstrated in OR-gym. Due to the large uncertainty of price, the sample size of each candidate policy in RL training is set to 100.

Results for RL via stochastic search are given in Fig. 6.5a) and Table 6.2. Specifically, these results detail optimizing for the expected performance of the policy (i.e. Eq. 6.4). It can be seen that the hybrid algorithm performs best among the four stochastic search algorithms, and when the policy is validated over 1000 Monte Carlo simulations, the policy identified by the hybrid algorithm still provides the highest expected portfolio value. Furthermore, the policy trained by PPO in the same 200,000 training episodes can only give an expected portfolio value of about \$930, which is substantially lower than the portfolio value of the four stochastic search algorithms. Under a sparse-reward environment, there is no reward for each step in an episode that is non-terminating, which reduces the information one can derive from samples and hence express in the policy gradient. PPO also uses the generalized advantage estimate variant of the policy gradient, which is bootstrapped and constituted by an estimate of the state value function. The state value function suffers from a similar problem to the vanilla policy gradient when the reward landscape is sparse. Together, these mechanisms likely produce the 'slow learning' dynamics presented in OR-gym. RL with stochastic search shows stronger performance because it does not require estimation of first order directions for policy improvement and does not need to fit state value functions. Additionally, the RL approach outperforms the online RLP results of the benchmark provided in Hubbs et al. (2020b) by a considerable margin.

It is worth noting however, that the standard deviation of the hybrid algorithm is much larger than that of the PPO algorithm. Intuitively, a highly uncertain performance may not be operationally desirable, particularly if high losses are observed with low probability. However, the tail of policy performance may also be observed in terms of increasing profit, meaning the high standard deviation is derived from achieving very high profit with low probability[2]. Further analysis suggests that the standard deviation associated with policy performance is actually derived from the latter case for the algorithm proposed. This is highlighted by Fig. 6.5a), which visualises the

[2]Of course, if the performance distribution exhibits some symmetry or low degrees of skewness then both cases could be observed

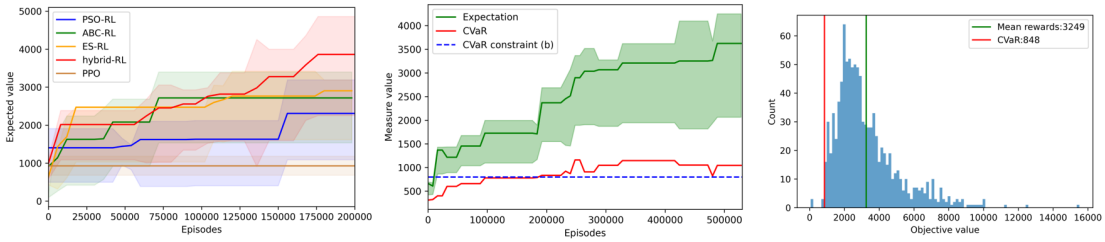


Figure 6.5: The performance of different RL algorithms and distributional RL. Plot a) shows the training curve of different RL algorithm; b) the training curve of hybrid distributional RL; and c) displays the histogram of policy performance from 1000 simulated episodes for the optimal distributional RL policy. The shaded region for PPO represents the standard deviation of the objective performance, for the stochastic search algorithms the regions are bounded by the 25^{th} and 75^{th} percentiles.

range of the 25^{th} and 75^{th} percentile of the policy performance through training via the shaded regions. Analysis of the figure highlights that the position of the mean policy performance lies closer to the 75^{th} percentile, indicating that a heavier tail lies on the side of large profit than losses and that the distribution is asymmetric.

Table 6.2: Total reward comparison of different RL algorithms in asset allocation. We report the mean and standard deviation (StD). The marker * indicates benchmark results acquired from robust linear programming (RLP) and PPO (Hubbs et al., 2020b). In this study, Hubbs et al. (2020b) provide validation results, which enable further comparison between PPO and stochastic search RL. The training results reported for mathematical programming formulations are evaluated over 100 simulations.

		Solution method					
Evaluation		RLP*	PPO*	ABC-RL	ES-RL	PSO-RL	Hybrid-RL
Training	Mean	865	~930	2716	2904	2309	3861
	StD	80	~250	1791	2294	1704	2180
Validation	Mean	~870	~910	2493	2556	2121	3578
	StD	~80	~200	1547	1626	1729	2195

The high standard deviation associated with policy performance observed from maximizing expectation in RL would obviously be negative if high losses, rather than profits, were observed with low probability. A priori one cannot know the whether the policy performance will be associated with high losses or profits in the tails and so simply additionally penalising for the standard deviation of policy performance in the fitness function is not desirable. Instead, we would prefer the policy be sensitive to risk. Therefore, we highlight the utility of the proposed distributional RL to achieve this.

The CVaR constraint is set to $b = 800$ based on the results of the RLP formulation provided in OR-gym (see Table 6.2), the sample size during training is set to 100 and the confidence level is set to $\alpha = 0.97$. Compared with the hybrid algorithm optimizing solely in expectation, which has a CVaR of \$510 in the 1000 Monte Carlo simulations, the policy identified by distributional RL has a higher CVaR of \$848 (showing a 66.3% improvement) with only a minor decrease (9.2%) on the mean reward value, indicating its superior performance in decision-making. As shown in Fig. 6.5b) and c), the CVaR constraints can be satisfied in both training and validation, and the policy exhibits better financial risk control.

6.4.3 Supply chain inventory management

Problem statement

Inventory management is closely intertwined with supply chain sustainability. The inventory management problem presented here is a $n_s = 4$ stage production-inventory system. The retailer (stage 1) is responsible for dealing with demand uncertainty from the customer given a current product inventory. This is handled by maintaining a reorder policy and ensuring deliveries from the stage upstream enable demand satisfaction from the customer. A similar problem occurs at each stage within the supply chain, apart from stage 4, which is assumed to have an infinite inventory to supply demand from stage 3. Each stage within the supply chain is required to manufacture and deliver the product ordered to the stage down the supply chain, within a given delivery (lead) time. For all stages the lead time is assumed deterministic and constant.

The problem is defined as an MDP. To ensure that the state retains the Markov property the state variables are augmented with the history of the previous $L_{\max}$ time steps, where $L_{\max}$ is the maximum lead time associated with the delivery of product from all stages. In this study $L_{max} = 10$. As a result, the state variables, $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{(n_s-1)(L_{\max}-1)}$, are defined by the current inventory and the previous $L_{\max}$ controls of stages 1-3. The controls, $\mathbf{u} \in \mathbb{U} \subset \mathbb{Z}_{+}^{n_s-1}$, are the reorder quantities made at each stage in the supply chain, which are integer values and bounded between zero and an upper bound, which is stage dependent and defined as $\mathbf{u}_{\text{ub}} = [100, 90, 80]$ for

stage 1-3, respectively. The goal of supply chain management here is to maximize the expected total profit after 30 discrete time steps. At each time step the reward function, $R(\cdot)$, allocates the profit made by the total chain in the most recent time increment, which is comprised by total sales and various costs of holding inventory, procurement and unfulfilled demand at the respective stages. There is substantial uncertainty in this problem, and this arises from variations in customer demand at stage 1. This exogeneous uncertainty is assumed to be described by a Poisson distribution with rate parameter equal to 20 arbitrary units of inventory.

In these problem classes, the bullwhip effect (Lee et al., 1997) is often observed as a result of the supply chain interactions becoming uncoordinated. The bullwhip effect refers to an extreme change in the supply position upstream caused by a small change in demand downstream. This dynamic inventory control problem poses challenge to traditional optimisation methods, provided it is subject to uncertainty and formally an MILP problem. For more details and parameters on the environment model, see InvManagement-v1 in OR-gym (Hubbs et al., 2020b).

Results and discussion

Optimizing for expectation

In this section, we further explore the training efficiency of the state-of-the-art policy gradient method, PPO, with the stochastic search methods developed in this work. As before, in addition to the hybrid stochastic search algorithm proposed above, several other stochastic search algorithms were run for benchmarking, including PSO-RL, ABC-RL, and ES-RL. In order to fairly compare the performance of these algorithms, all the hyperparameters shared by these algorithms will be set to the same value, for example, the number of particles is set to 60 and a total training budget of 66000 episodes allocated to ensure objective comparison to the results of the PPO benchmark. Fig. 6.6 shows the training curve of the stochastic search algorithms for RL applied to the inventory management problem. It can be seen from the figure that although the (best performing) randomly initialized policy of the hybrid algorithm performs poorly, it can quickly learn an effective policy over the first few thousand training episodes. After around 10000 episodes, the policy it has learned is better than the other three algorithms. This is due to the fact that the hybrid algorithm can effectively balance

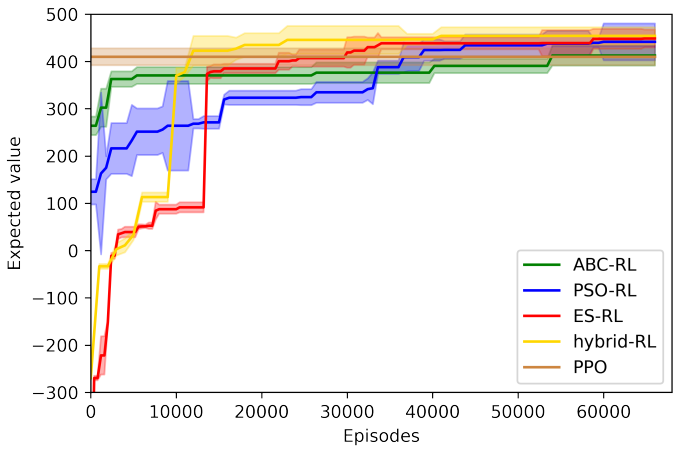


Figure 6.6: Training curve of the different RL algorithms used in the inventory management problem. The shaded areas represent the standard deviation of the rewards. The results of PPO acquired by OR-gym are also plotted to demonstrate the relative performance of the stochastic search algorithms.

the exploration and exploitation of the search space. The initial random policy of ABC performs best, but its exploitative ability is the worst, resulting in the lowest total rewards among all algorithms.

The number of sampling episodes for each candidate policy network is 10 during the training. To demonstrate the effectiveness of the final policy obtained by RL training, the final policy is then evaluated in the supply chain for 1000 episodes. The training and validation results are shown by Table 6.3, and highlight that the policy trained by the hybrid algorithm can still perform stably when evaluated in scenarios unseen in training and has good robustness. Table 6.3 also compares the stochastic search approaches described with the policy gradient, PPO method used in OR-gym after the same number of training episodes, as well as the shrinking horizon LP and MILP (the latter benchmark operates with no recourse) results demonstrated in that work. It is worth noting that the four derivative-free RL algorithms all outperformed proximal policy optimisation (PPO) in this case, demonstrating better exploration behavior. The hybrid method also outperformed the PPO implementation showing expected performance improvements of approximately 11%. The hybrid method also outperformed the MILP by 20%. It was also competitive with the LP relaxation of the problem with a performance gap of 6%.

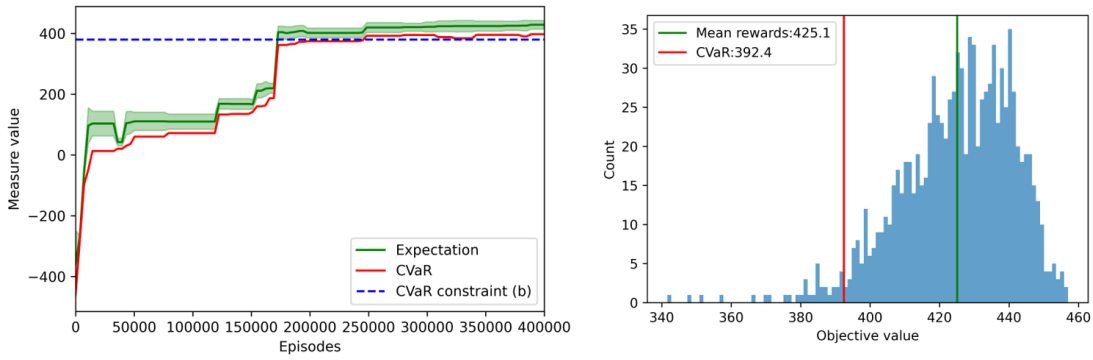
Table 6.3: Total reward comparison of different RL algorithms under 66000 training episodes. We report the mean and standard deviation (StD). The marker * indicates benchmark results acquired from shrinking horizon linear programming (SHLP), mixed integer programming (MIP) and PPO (Hubbs et al., 2020b). In this study, Hubbs et al. (2020b) provide training results only, which enable objective comparison between LP, PPO and stochastic search RL. The training results reported for mathematical programming formulations are evaluated over 100 simulations.

		Solution method						
Evaluation		SHLP*	MIP*	PPO*	ABC-RL	ES-RL	PSO-RL	Hybrid-RL
Training	Mean	485.4	378.5	409.8	412.4	448.5	442.1	454
	StD	29.1	26.1	17.9	21.2	18.8	20.2	18
Validation	Mean				411.4	428.3	426.6	431.3
	StD				16.9	27.5	28	22.2

Optimizing for risk sensitive formulations

Since our proposed hybrid stochastic search algorithm is found to outperform other algorithms, in this section we turn our attention to specifically investigate its performance for optimisation of the distributional CVaR objective. Since the CVaR is calculated by Monte Carlo sampling, a small number of sampling episodes will likely cause the algorithm to optimise for high probability events, so the number of episodes evaluated for each candidate policy network was increased to 60 during training. The confidence level was set to 0.9. The value of CVaR constraint is set to 380 based on the work provided in OR-gym.

Fig. 6.7a) shows the training curve of distributional RL and how the CVaR changes during training. The return of the policy after training is 428.8, the standard deviation is 14.2, and CVaR is 397.6 which satisfies the CVaR constraint imposed. To further demonstrate that the algorithm learns a more robust risk-averse policy, the distributional RL policy is validated over 1000 Monte Carlo samples (as in previous experiments). As shown in Fig. 6.7b), the distributional RL policy is able to obtain an average reward of 425.1 with a standard deviation of 16.4. More importantly, the CVaR of the distributional RL policy is 392.4, compared with only 365.9 when the hybrid stochastic search algorithm optimizing for expectation is evaluated in 1000 Monte Carlo simulations. This means distributional RL can offer better protection against worst case realizations of process uncertainty (7.2% improvement). It is worth mentioning that although we increased the number of sampled episodes in distributional



(a) Training curve

(b) Rewards distribution

Figure 6.7: Investigating the performance of distributional RL in inventory management problem. Plot a) shows the training curve of distributional RL. The shaded areas represent the standard deviation of the rewards. Plot b) shows a histogram of policy performance from 1,000 simulated episodes for the optimal distributional RL policy.

RL, since backpropagation is not performed and there is no value function parameterization required, the amount of computation is reduced, partially offsetting the slight decrease in data efficiency. The amount this is offset is not clear and is dependent on algorithm hyperparameters, but in the case of the PPO benchmark.

Sensitivity to different parameters

We use the control variate method to analyze the sensitivity of the three parameters, which define the RL policy training algorithm: number of particles, CVaR constraint, b, and the number of sampling episodes for each candidate policy network in distributional RL. This sensitivity study provides a basis for RL technicians to select good hyperparameters. For a more objective analysis, the learned policy is validated over 1000 Monte Carlo simulations.

Firstly, we study the impact of number of particles on the mean reward and CVaR of the supply chain because the number of particles will affect the global search ability of the algorithm. The CVaR constraint is set to $b = 380$ and the sample size of each candidate policy is 60 as before. As shown in Fig. 6.8a), the supply chain profit is not sensitive to the number of particles ranging from 40 to 100 in the hybrid algorithm, and the learned policy is stable even in 1000 Monte Carlo simulations.

However, the reason for the insensitivity to the number of particles may be that 40 particles is good enough to ensure stable performance. Therefore, we investigate the

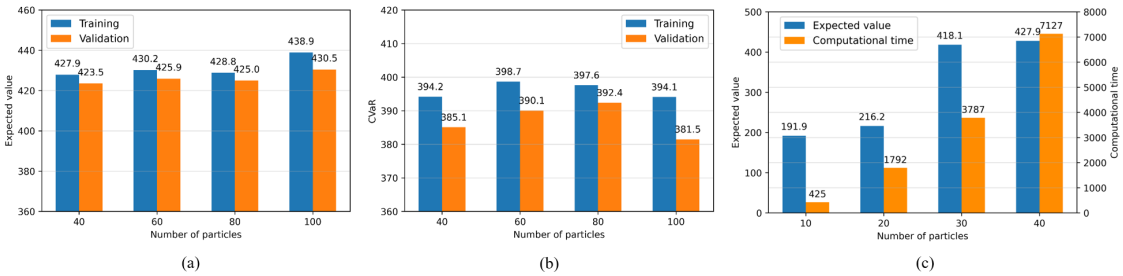


Figure 6.8: Investigating the influence of number of particles on a) mean rewards and b) CVaR. Plot c) shows the influence of number of particles at a lower range on computational time and mean rewards.

relationship between the number of particles and the computation time in a lower test range, where we introduce a termination condition: the algorithm will be terminated if the current best solution policy observes no improvement for 10 consecutive iterations. As shown in Fig. 6.8c), when the number of particles is small, the algorithm will easily fall into a local optimum and converge prematurely. On the other hand, when the number of particles increases to 30, little performance improvement is observed with the addition of further particles thereafter. It should also be highlighted that as the number of particles increased, the computational burden of the algorithm increases accordingly. In summary we deem generally that in this case study, the hybrid algorithm demonstrated no significant performance increase in using more than 40 particles. This hyperparameter selection enables the implementation to minimise the the computational cost of the algorithm.

Next, we investigated the impact of changes in CVaR constraint on the mean reward and CVaR of the supply chain system. As shown in Fig. 6.9a), by imposing a larger bound within the CVaR constraint, the mean rewards of the learned policy decrease at both training and validation. We hypothesize this is because increasing the value of the CVaR constraint forces the algorithm to give priority to satisfying CVaR constraint during the training process. As shown in Fig. 6.9b), when the CVaR bound is greater than a critical value between 380 and 400, the learned policy will not be able to satisfy CVaR constraint both during training and validation, this is because the algorithm tends to optimise low-risk events in order not to be punished for violating constraints. Noticeably, when the CVaR bound is set $b = 420$, the algorithm performs poorly. It is unrealistic to expect that CVaR will be greater than 420 by simply imposing a greater

lower bound, given that expected return obtained by optimizing solely for expectation was around that value (see Table 6.3). In this case, the algorithm was unable to find a solution policy.

Finally, we explore the influence of sample size of each candidate policy during training, where again the CVaR bound, $b = 380$ and the number of particles is $N = 60$. As can be seen in Fig. 6.9c) and d), mean policy performance decreases in training, whereas validation remains roughly constant with increasing of sample size. This is primarily due to the statistical inaccuracies associated with sample average approximations with low sample size in training. This is highlighted by lower differences between training and validation results as samples are increased. Similar observations can be made with respect to the CVaR, but additionally, the validation CVaR appears to increase with sample size. This is likely due to observing more information in training routines with higher sample numbers and better observing the influence of high risk, low probability events. However, increasing the sample size blindly will bring expense in computational cost. When the algorithm performs similarly in training and validation, it can be considered that the sample size is sufficient. The results of the sensitivity analysis on these three parameters are integrated in Table E.2 in E.6.

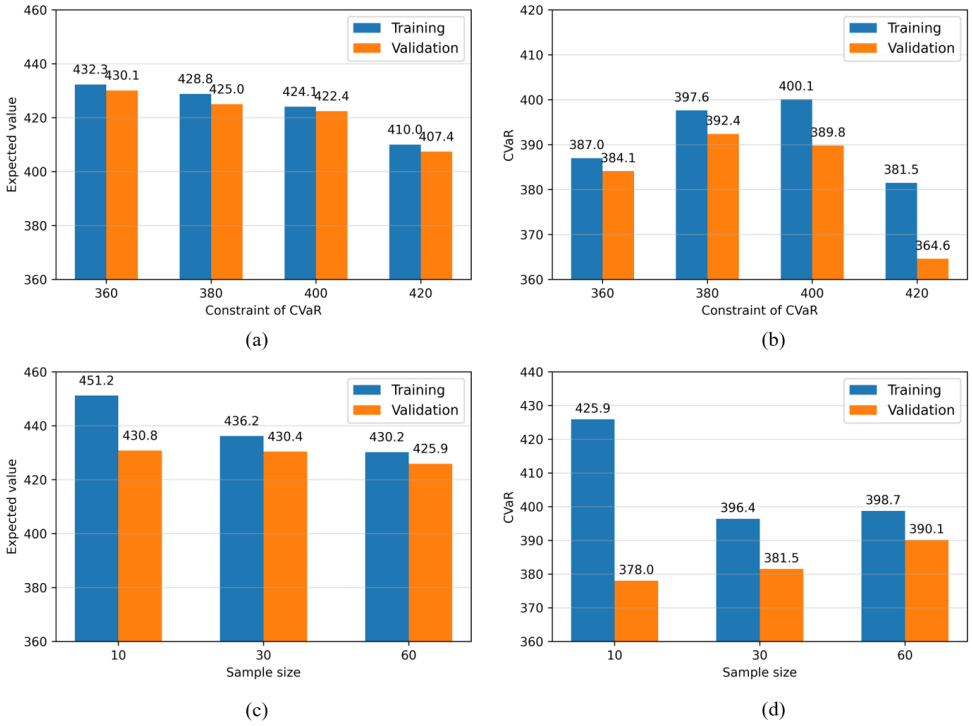


Figure 6.9: Investigating the influence of CVaR constraint on a) mean rewards and b) CVaR. Investigating the influence of sample size on c) mean rewards and d) CVaR.

6.5 Conclusions

In this article, we present a method that combines stochastic search algorithms with reinforcement learning, to address supply chain and operations research problems, this approach overcomes the shortcomings of gradient-based reinforcement learning methods. The hybrid search algorithm is based on PSO, ABC, ES and SA can balance the exploration-exploitation paradigm in the process of parameter search. We also propose a novel distributional RL framework to learn a risk-sensitive policy by introducing a CVaR penalty into the objective function.

Through a case study of a multi-echelon supply chain, the efficiency and potential of our method for the optimisation of complex stochastic systems is well demonstrated. The hybrid algorithm obtains a substantially larger return ($454) against PPO ($409.8) and other stochastic optimisation algorithms. Furthermore, the proposed distributional RL successfully learns a risk-sensitive policy, and the CVaR value given by distributional RL is greatly improved compared with expectation-based RL (7.2% improvement).

In the asset allocation case, our method is proved to be more effective than policy gradient method in a sparse-reward environment. The distributional RL can also offer more downside protection portfolio value (66.3% improvement). In addition, in the case of virtual machine packing, the results also show that stochastic search algorithm for RL is a more powerful method.

It is worth highlighting that most of the literature advocates for policy gradients (or actor-critic) RL algorithms. Here we indicate that stochastic search algorithms are more efficient and flexible in these problem instances. It is clear that the value of gradient-based search increases with the dimensionality of the parameter space, however, in many engineering applications, policy network parameters are not in the millions and hence stochastic search (or other derivate free algorithms) have value. In future work, we plan to investigate a multi-agent approach for distributional RL, consider hybrid controls (Campos et al., 2022), as well as vertically integrated decision problems.

Chapter 7

Conclusions and future work

In this section, we will summarise the major insight and contribution provided by this thesis.

In Chapter 2, we explored the fundamental theory of indirect and direct methods and the potential contributions of RL to PSE as well as the outlying challenges. We then explored particular applications of RL and the developments made by the PSE community with particular focus in highlighting the advantages over existing methods and the challenges these works approach. These advantages primarily arise in the use of model-free learning rules, which enables RL policies to optimise systems independently of assumptions on the dynamics or the associated uncertainties and instead directly through data. The challenges to RL arise in sample efficient offline policy learning, safety, robustness and closed loop stability. Subsequently, the existing methods to model and provide online decisions for uncertain batch process systems and chemical production systems were explored. We highlighted that in batch process systems, when nonlinearity is present in the dynamics, propagating the uncertainty of the system state over the time horizon becomes particularly challenging; and that ultimately all existing methods in mathematical programming, require us to make approximation to this uncertainty. We highlighted that with respect to production scheduling environments that considering uncertainty in online decision making via existing strategies is essentially intractable from the perspective of computation.

Through the work items, we have explored the development of strategies to approach the challenges highlighted in literature review. Specifically, we developed schemes to:

1. Reduce computational expense and expertise heavy tuning in offline policy learning by leveraging existing process knowledge described in process data.

2. Optimise uncertain constrained fed-batch process systems without mechanistic knowledge of the underlying system dynamics.

3. Compute online scheduling decisions for uncertain sequential production environments orders of magnitude more efficiently than optimisation approaches whilst explicitly considering uncertainty.

4. Learn policy function approximations robustly via zero-order optimisation. This approach also demonstrated relative sample efficiency over benchmark policy gradient RL approaches.

More specifically, in Chapter 3, we saw how we could use existing process data and inverse reinforcement learning techniques to parameterise and improve operational knowledge provided by existing control schemes and process operators through a two-step framework. In step 1, we synchronously abstract the control objectives and a parameterisation of the existing control scheme. Then in step 2, we demonstrate that we can improve the parameterisation further by learning from the process online under the real process objective. This framework was demonstrated on a set point tracking problem with linear system dynamics and data from an existing PID control scheme. It was demonstrated that IRL could be used successfully to abstract the existing behaviour of the PID scheme under a cost function that could be easily interpreted for safety purposes. Challenges to implementation of this approach were discussed, with the primary obstacle arising in the selection and design of characterising basis features of the behaviour of the existing scheme, such that the control objective could be abstracted as a linear combination of these features. In the second step, we highlighted the ease of model-free learning rules in subsequent policy improvement directly from the real process. The applicability of this framework will largely be challenged by the nature of the control task. If the control problem is periodically redefined (as in the case of multiple set point changes) then implementation of the scheme will be difficult. This would require abstracting a separate parameterisation for each task or investigating the use of inverse meta RL techniques (Yu et al., 2019; McClement et al., 2022). Alternatively, the current framework may be deployed for batch processes,

such as fermentation systems, where the current control system may be for example a nonlinear model predictive controller. The policy could then be improved via a batch-to-batch approach once deployed as highlighted in Petsagkourakis et al. (2020b).

In Chapter 4, a framework for safe RL is presented that utilises a fully data-driven approach for online optimisation of constrained nonlinear and uncertain fed-batch processes. The framework again consists of offline policy learning in an approximate process model, with subsequent transfer to the real process. The methodology allows for satisfaction of operational constraints with a given probability defined via the user and handling of offline process model-plant mismatch. The results demonstrated impressive performance against the best case deterministic method provided by nonlinear model predictive control (NMPC) with a 30% improvement in closed-loop performance. Of particular note, was the ability of the method proposed to handle state constraints on the real process (with the probability level desired), whereas the NMPC scheme was unable to handle constraints as desired and often failed to find a solution online due to infeasibility in the optimisation model. However, the methodology faces challenges as in current form relies on the use of Gaussian process state space models. If the model is unable to well capture the process physics then other hybrid or purely data-driven models could also be used with modification to the methodology. The use of mechanistic expressions in a hybrid model would be particularly appealing as it would allow the methodology to use a smaller dataset than in the current implementation. This would likely require exploitation of distributional approaches to RL especially if the variance of the system state is not parameterised by the resultant process model.

In Chapter 5, a framework for scheduling of uncertain sequential batch production environments is presented, which leverages a zero-order, stochastic search optimisation approach to RL and allows for flexible formulation of risk-sensitive objectives. The major benefit of this algorithm over first-order approaches to RL is that it is able to handle the non-smoothness of the mapping between the state value function and policy parameters. In the presence of non-smoothness in this map, first-order RL approaches have no guarantees in terms of convergence or optimality (Zhang et al., 2021). Additionally, estimation of risk-sensitive objectives is unbiased. This is generally not the case in other risk-averse RL formulations. The potential for application

of this approach was thoroughly investigated in two different problem instances and demonstrated to be competitive against nominal online MILP strategies in terms of objective performance. The largest optimality gap was of the order of 5%. Additionally, the algorithm was able to effectively handle common restrictions on sequential production environments through an action-masking mechanism. The RL approach was able to identify online scheduling decisions orders of magnitude faster than the MILP approach. However, this algorithm has been tailor made to smaller problem sizes. As the problem size increases the dimensionality of the state representation will also increase, likely to a size where the method, which is based in zero-order search (Balasubramanian and Ghadimi, 2022) is no longer as effective. Additionally, when assessed in validation the method was unable to handle hard constraints imposed on the production environment absolutely, but instead with very high probability (violating in 1 or 2 of 500 scenarios). The constraints that were violated had to be imposed via a penalty function in offline policy identification and could not be handled by the action-masking mechanism proposed without biasing the policy considerably. In future work, we will explore modifications to the algorithm in order to handle larger problem sizes and ensure constraint satisfaction absolutely.

Finally, in Chapter 6, we explored the development and application of a zero-order, hybrid stochastic search RL approach to inventory management in uncertain multi-echelon supply chains. Here, we directly compare our approach against the benchmark policy gradient RL implementation and mathematical programming implementations reported in Hubbs et al. (2020b). The method proposed demonstrates improved sample efficiency and performance (11% improvement) over the benchmark policy gradient and provides basis for the wider use of stochastic search approaches to RL in PSE. Currently, policy gradient methods are the go to, primarily due to their popularisation in game-based control benchmarks. In those problems the neural policy parameter space tends to be much higher dimensional, which is where the use of gradients have their advantage. In PSE, neural policy functions are typically much smaller and hence why stochastic search routines are practical. The proposed method was also deemed competitive with state-of-the-art mathematical programming approaches (20% improvement over MILP), but inherits the ability to pose risk-sensitive formulations. For example, it was demonstrated that the approach could constrain the

expected cost of the worst case policy performances through use of the conditional value-at-risk (CVaR) with an improvement of 7.2% over the case when the CVaR was left unconstrained. This is in common with the method proposed in Chapter 5 and generally something not well-explored within the PSE literature (Guerra et al., 2019; Germscheid et al., 2022). However, again, there is likely to be issues with the method as the problem size increases (i.e. stages added to the supply chain). In future work, we will look to explore the use of graph neural networks and multi-agent RL algorithms in order to handle these problems.

In the literature review associated with this work, major challenges were outlined in applied RL research as sample efficient learning, safety, robustness and closed-loop stability. This thesis has made contributions to all of these areas bar closed-loop stability. Having said this, as with all algorithmic frameworks, these methods have their weaknesses. These have been partly outlined in the previous discussion. However, further work should be directed towards all of these open areas. It is likely that for example improvements in sample efficiency will be derived by making connections from RL to other fields (Meyn, 2022) and classes of methods (Salehkaleybar et al., 2022; García-Fernández et al., 2015). Similarly, open challenges still exist in ensuring state constraint satisfaction in a wide range of process systems models. Particularly handling resource constraints in production scheduling problems (i.e. where the system is non-smooth) would be an interesting contribution. We have demonstrated that RL can be used to pose risk-sensitive formulations in this work, however major challenges still exist in algorithmic robustness to small variations in hyperparameters. Clearly, there are meta-optimisation algorithms (Claesen and De Moor, 2015; Bergstra and Bengio, 2012; Aszemi and Dominic, 2019), which can mitigate some of these challenges, but it is known that these methods are computationally expensive. The major progress will likely derive from theoretical contributions, the development of new learning rules (Bas-Serrano et al., 2021; Chen et al., 2020) and properly characterising how current methods work (Fu et al., 2019a). The final area of closed-loop stability is a particularly interesting area of research. Characterising the conditions under which a policy parameterisation induces stable behaviour is likely to be a challenging task if the function structure is nonlinear (Cooman et al., 2017), and providing guarantees is made more complex by the stochastic nature of the underlying system (Nakamura-Zimmerer

et al., 2021). However, focus directed towards this area is likely to be of greater benefit to industrial practice than necessarily demonstrating optimality on arbitrary process systems.

However, from the perspective of application, the development of RL algorithms for network production environments and large scale process systems is particularly desirable. In terms of production scheduling, RL has particular potential primarily because, in the online decision making setting, it could mitigate computationally expensive optimisation - meaning that online scheduling has the potential to become a more tractable task. This is increasingly desirable as operations become more distributed, and reactive to consumer demand and market changes. Additionally, the heuristic decisions provided by RL are identified within an optimal control framework and so are likely to provide considerable performance improvements over classic priority decision rules. As the size of the underlying problem grows this is likely to require new thinking on how best to generate decisions from policy functions, decomposing the problem via multi-agent frameworks and learning low dimensional representations of highly dimensional systems. This is unlikely to be learned end-to-end and would likely require data from existing schemes and optimisation methods as a hot-start (Zhang et al., 2020a). More generally, although focus within this thesis has been directed towards the (bio)chemical process industries, process systems thinking can be flexibly applied to other industries. For example, healthcare systems could particularly benefit from the operational decision making strategies developed within PSE (Pistikopoulos et al., 2021) and this is an area which would also greatly serve the public, especially within the UK where healthcare is a publicly funded service and efficiency improvements are perhaps even more crucial to continued operation. These are systems which are characterised by high uncertainty and limited mechanistic knowledge. As a result, they are well suited to data-driven approaches to modelling and optimisation (Nestor et al., 2019; Bellot and Schaar, 2020; Gottesman et al., 2019; Yu et al., 2021a).

Bibliography

Åarzén, K.-E., 1999: A simple event-based pid controller. *IFAC Proceedings Volumes*, **32**, no. 2, 8687–8692.

Abbeel, P. and A. Y. Ng, 2004: Apprenticeship learning via inverse reinforcement learning. Twenty-first international conference, ACM Press, Banff, Alberta, Canada, 1, [Online; accessed 2020-06-12].
URL http://portal.acm.org/citation.cfm?doid=1015330.1015430

Abdolmaleki, A., J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, 2018: *Maximum a posteriori policy optimisation*.
URL https://arxiv.org/abs/1806.06920

Achiam, J., D. Held, A. Tamar, and P. Abbeel, 2017: *Constrained policy optimization*.

Agarwal, R., D. Schuurmans, and M. Norouzi, 2020: An optimistic perspective on offline reinforcement learning. *International Conference on Machine Learning*, PMLR, 104–114.

Ahmadi, M., U. Rosolia, M. D. Ingham, R. M. Murray, and A. D. Ames, 2021: Constrained risk-averse markov decision processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11718–11725.

Ahmed, Z., N. Le Roux, M. Norouzi, and D. Schuurmans, 2019a: Understanding the impact of entropy on policy optimization. *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., PMLR, volume 97 of *Proceedings of Machine Learning Research*, 151–160.

Ahmed, Z., N. L. Roux, M. Norouzi, and D. Schuurmans, 2019b: *Understanding the impact of entropy on policy optimization*.

Akhade, S. A., N. Singh, O. Y. Gutiérrez, J. Lopez-Ruiz, H. Wang, J. D. Holladay, Y. Liu, A. Karkamkar, R. S. Weber, A. B. Padmaperuma, et al., 2020: Electrocatalytic hydrogenation of biomass-derived organics: a review. *Chemical Reviews*, **120**, no. 20, 11370–11419.

Albarghouthi, A., 2021: *Introduction to neural network verification*. URL https://arxiv.org/abs/2109.10317

Ali, A. F. and M. A. Tawhid, 2017: A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems. *Ain Shams Engineering Journal*, **8**, no. 2, 191–206, iSBN: 2090-4479 Publisher: Elsevier.

Almeder, C. and R. F. Hartl, 2013: A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer. *International Journal of Production Economics*, **145**, no. 1, 88–95.

Almquist, J., M. Cvijovic, V. Hatzimanikatis, J. Nielsen, and M. Jirstrand, 2014: Kinetic models in industrial biotechnology–improving cell factory performance. *Metabolic engineering*, **24**, 38–60.

Altiparmak, F., B. Dengiz, and A. A. Bulgak, 2002: Optimization of buffer sizes in assembly systems using intelligent techniques. *Proceedings of the Winter Simulation Conference*, IEEE, volume 2, 1157–1162.

Altman, E., 1999: *Constrained Markov decision processes: stochastic modeling*. Routledge.

Amari, S.-I. and S. C. Douglas, 1998: Why natural gradient? *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, IEEE, volume 2, 1213–1216.

Amit, R., R. Meir, and K. Ciosek, 2020: Discount factor as a regularizer in reinforcement learning. *International conference on machine learning*, PMLR, 269–278.

Amos, B., 2022: *Tutorial on amortized optimization for learning to optimize over continuous domains*.

Amos, B., I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, 2018: Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, **31**.

Anandan, P. D., C. D. Rielly, and B. Benyahia, 2022: Optimal control policies of a crystallization process using inverse reinforcement learning. *Computer Aided Chemical Engineering*, Elsevier, volume 51, 1093–1098.

Andersson, J., J. Åkesson, and M. Diehl, 2012: Casadi: A symbolic package for automatic differentiation and optimal control. *Recent advances in algorithmic differentiation*, Springer, 297–307.

Angermueller, C., D. Dohan, D. Belanger, R. Deshpande, K. Murphy, and L. Colwell, 2020: Model-based reinforcement learning for biological sequence design. *International Conference on Learning Representations*.
URL https://openreview.net/forum?id=HklxbgBKvr

Anye Cho, B., M. A. de Carvalho Servia, E. A. del Rio Chanona, R. Smith, and D. Zhang, 2021: Synergising biomass growth kinetics and transport mechanisms to simulate light/dark cycle effects on photo-production systems. *Biotechnology and Bioengineering*, **118**, no. 5, 1932–1942, iSBN: 0006-3592 Publisher: Wiley Online Library.

Arellano-Garcia, H., T. Barz, B. Dorneanu, and V. S. Vassiliadis, 2020: Real-time feasibility of nonlinear model predictive control for semi-batch reactors subject to uncertainty and disturbances. *Computers & Chemical Engineering*, **133**, 106529.

Arora, S. and P. Doshi, 2018: *A survey of inverse reinforcement learning: Challenges, methods and progress*.
URL https://arxiv.org/abs/1806.06877

Arridge, S., P. Maass, O. Öktem, and C.-B. Schönlieb, 2019: Solving inverse problems using data-driven models. *Acta Numerica*, **28**, 1–174.

Astrom, K., 1979: Maximum likelihood and prediction error methods. *IFAC Proceedings Volumes*, **12**, no. 8, 551–574.

Aström, K. J., 2008: Event based control. *Analysis and design of nonlinear control systems*, Springer, 127–147.

Aszemi, N. M. and P. Dominic, 2019: Hyperparameter optimization in convolutional neural network using genetic algorithms. *International Journal of Advanced Computer Science and Applications*, **10**, no. 6.

Attia, A. M., A. M. Ghaithan, and S. O. Duffuaa, 2019: A multi-objective optimization model for tactical planning of upstream oil & gas supply chains. *Computers & Chemical Engineering*, **128**, 216–227.

Audouze, C., F. De Vuyst, and P. Nair, 2009: Reduced-order modeling of parameterized pdes using time–space-parameter principal component analysis. *International journal for numerical methods in engineering*, **80**, no. 8, 1025–1057.

Baake, E., M. Baake, H. Bock, and K. Briggs, 1992: Fitting ordinary differential equations to chaotic data. *Physical Review A*, **45**, no. 8, 5524.

Balasubramanian, K. and S. Ghadimi, 2022: Zeroth-order nonconvex stochastic optimization: Handling constraints, high dimensionality, and saddle points. *Foundations of Computational Mathematics*, **22**, no. 1, 35–76.

Banbury, C., R. Mason, I. Styles, N. Eisenstein, M. Clancy, A. Belli, A. Logan, and P. Goldberg Oppenheimer, 2019: Development of the self optimising kohonen index network (skinet) for raman spectroscopy based detection of anatomical eye tissue. *Scientific reports*, **9**, no. 1, 1–9.

Banzhaf, W., P. Nordin, R. E. Keller, and F. D. Francone, 1998: *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc.

Bas-Serrano, J., S. Curi, A. Krause, and G. Neu, 2021: Logistic q-learning. *International Conference on Artificial Intelligence and Statistics*, PMLR, 3610–3618.

Bassett, M. H., J. F. Pekny, and G. V. Reklaitis, 1996: Decomposition techniques for the solution of large-scale scheduling problems. *AIChE Journal*, **42**, no. 12, 3373–3387.

Bazaraa, M. S., H. D. Sherali, and C. M. Shetty, 2013: *Nonlinear programming: theory and algorithms*. John Wiley & Sons.

Beaulieu, S., L. Frati, T. Miconi, J. Lehman, K. O. Stanley, J. Clune, and N. Cheney, 2020: Learning to continually learn. *arXiv:2002.09571 [cs, stat]*, arXiv: 2002.09571. URL http://arxiv.org/abs/2002.09571

Beintema, G. I., R. Tóth, and M. Schoukens, 2021: Non-linear state-space model identification from video data using deep encoders. *IFAC-PapersOnLine*, **54**, no. 7, 697–701.

Bellemare, M. G., W. Dabney, and R. Munos, 2017a: A distributional perspective on reinforcement learning. *International Conference on Machine Learning*, PMLR, 449–458.

— 2017b: A distributional perspective on reinforcement learning. *International Conference on Machine Learning*, PMLR, 449–458.

Bellemare, M. G., W. Dabney, and M. Rowland, 2022: *Distributional Reinforcement Learning*. MIT Press.

— 2023: *Distributional Reinforcement Learning*. MIT Press, http://www.distributional-rl.org.

Bellman, R., 1956: Dynamic programming. RAND CORP SANTA MONICA CA.

Bellman, R. and E. Lee, 1984: History and development of dynamic programming. *IEEE Control Systems Magazine*, **4**, no. 4, 24–28.

Bellot, A. and M. V. D. Schaar, 2020: Flexible modelling of longitudinal medical data: A bayesian nonparametric approach. *ACM Transactions on Computing for Healthcare*, **1**, no. 1, 1–15.

Ben-Tal, A., L. El Ghaoui, and A. Nemirovski, 2009: Robust optimization. *Robust optimization*, Princeton university press.

Benyahia, B., P. D. Anandan, and C. Rielly, 2021: Control of batch and continuous crystallization processes using reinforcement learning. *Computer Aided Chemical Engineering*, Elsevier, volume 50, 1371–1376.

Bergstra, J. and Y. Bengio, 2012: Random search for hyper-parameter optimization. *Journal of machine learning research*, **13**, no. 2.

Berkenkamp, F., M. Turchetta, A. P. Schoellig, and A. Krause, 2017: *Safe model-based reinforcement learning with stability guarantees*.

Bertsekas, D., 2012: *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific.

— 2022: Newton's method for reinforcement learning and model predictive control. *Results in Control and Optimization*, 100121.

Bertsekas, D. P., 1971: *Control of uncertain systems with a set-membership description of the uncertainty.*. Ph.D. thesis, Massachusetts Institute of Technology.

— 2005: Dynamic programming and suboptimal control: A survey from adp to mpc. *European Journal of Control*, **11**, no. 4-5, 310–334.

Bertsekas, D. P., D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, 1995: *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

Beykal, B., S. Avraamidou, and E. N. Pistikopoulos, 2022: Data-driven optimization of mixed-integer bi-level multi-follower integrated planning and scheduling problems under demand uncertainty. *Computers & Chemical Engineering*, **156**, 107551.

Beykal, B., S. Avraamidou, I. P. Pistikopoulos, M. Onel, and E. N. Pistikopoulos, 2020: Domino: Data-driven optimization of bi-level mixed-integer nonlinear problems. *Journal of Global Optimization*, **78**, no. 1, 1–36.

Biegler, L. T., 2007: An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, **46**, no. 11, 1043–1053.

Billingsley, P., 2008: *Probability and measure*. John Wiley & Sons.

Biradar, N. S., A. A. Hengne, S. N. Birajdar, R. Swami, and C. V. Rode, 2014: Tailoring the product distribution with batch and continuous process options in catalytic hydrogenation of furfural. *Organic Process Research & Development*, **18**, no. 11, 1434–1442.

Birge, J. R., 1997: State-of-the-art-survey—stochastic programming: Computation and applications. *INFORMS journal on computing*, **9**, no. 2, 111–133.

Błażewicz, J., E. Pesch, and M. Sterna, 2000: The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, **127**, no. 2, 317–331.

Bloss, K. F., L. Biegler, and W. Schiesser, 1999: Dynamic process optimization through adjoint formulations and constraint aggregation. *Industrial & engineering chemistry research*, **38**, no. 2, 421–432.

Boole, G., 1847: *The mathematical analysis of logic*. Philosophical Library.

Bradford, E. and L. Imsland, 2018: Economic stochastic model predictive control using the unscented kalman filter. *IFAC-PapersOnLine*, **51**, no. 18, 417–422.

Bradford, E., L. Imsland, M. Reble, and E. A. del Rio-Chanona, 2021a: Hybrid gaussian process modeling applied to economic stochastic model predictive control of batch processes. *Recent Advances in Model Predictive Control*, Springer, 191–218.

— 2021b: Hybrid gaussian process modeling applied to economic stochastic model predictive control of batch processes. *Recent Advances in Model Predictive Control*, Springer International Publishing, 191–218.
URL https://doi.org/10.1007%2F978-3-030-63281-6_8

Bradford, E., L. Imsland, D. Zhang, and E. A. del Rio Chanona, 2020: Stochastic data-driven model predictive control using gaussian processes. *Computers & Chemical Engineering*, **139**, 106844.

Bradford, E., A. M. Schweidtmann, D. Zhang, K. Jing, and E. A. del Rio-Chanona, 2018: Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate gaussian processes. *Computers & Chemical Engineering*, **118**, 143–158.

Bradley, W. and F. Boukouvala, 2021: Two-stage approach to parameter estimation of differential equations using neural odes. *Industrial & Engineering Chemistry Research*, **60**, no. 45, 16330–16344.

Bradley, W., J. Kim, Z. Kilwein, L. Blakely, M. Eydenberg, J. Jalvin, C. Laird, and F. Boukouvala, 2022: Perspectives on the integration between first-principles and data-driven modeling. *Computers & Chemical Engineering*, 107898.

Brancato, S. M., F. De Lellis, D. Salzano, G. Russo, and M. di Bernardo, 2022: *External control of a genetic toggle switch via reinforcement learning*.
URL https://arxiv.org/abs/2204.04972

Bratko, I., T. Urbančič, and C. Sammut, 1995: Behavioural cloning: phenomena, results and problems. *IFAC Proceedings Volumes*, **28**, no. 21, 143–149.

Brochu, E., V. M. Cora, and N. De Freitas, 2010: A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Bronstein, M. M., J. Bruna, T. Cohen, and P. Veličković, 2021: *Geometric deep learning: Grids, groups, graphs, geodesics, and gauges*.
URL https://arxiv.org/abs/2104.13478

Brown, L. D., T. T. Cai, and A. DasGupta, 2001: Interval estimation for a binomial proportion. *Statistical Science*, **16**, no. 2, 101–117, doi:10.2307/2676784.
URL http://www.jstor.org/stable/2676784

Brunke, L., M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, 2022: Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, **5**, 411–444.

Cafaro, D. C. and I. E. Grossmann, 2014: Strategic planning, design, and development of the shale gas supply chain network. *AIChE Journal*, **60**, no. 6, 2122–2142.

Calafiore, G. and M. C. Campi, 2005: Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, **102**, no. 1, 25–46.

Calvo, F., J. M. Gómez, L. Ricardez-Sandoval, and O. Alvarez, 2020: Integrated design of emulsified cosmetic products: A review. *Chemical Engineering Research and Design*, **161**, 279–303.

Camacho-Villalón, C. L., M. Dorigo, and T. Stützle, 2022: Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors. *International Transactions in Operational Research*.

Campi, M. C., S. Garatti, and M. Prandini, 2009: The scenario approach for systems and control design. *Annual Reviews in Control*, **33**, no. 2, 149–157.

Campos, G., N. H. El-Farra, and A. Palazoglu, 2022: Soft actor-critic deep reinforcement learning with hybrid mixed-integer actions for demand responsive scheduling of energy systems. *Industrial & Engineering Chemistry Research*.

Cannon, M., J. Buerger, B. Kouvaritakis, and S. Rakovic, 2011: Robust tubes in nonlinear model predictive control. *IEEE Transactions on Automatic Control*, **56**, no. 8, 1942–1947.

Cao, C., Y. Zhang, X. Gu, D. Li, and J. Li, 2021: An improved gravitational search algorithm to the hybrid flowshop with unrelated parallel machines scheduling problem. *International Journal of Production Research*, **59**, no. 18, 5592–5608.

Caputo, C. and M.-A. Cardin, 2022a: Analyzing Real Options and Flexibility in Engineering Systems Design Using Decision Rules and Deep Reinforcement Learning. *Journal of Mechanical Design*, **144**, no. 2, iSBN: 1050-0472 Publisher: American Society of Mechanical Engineers Digital Collection.

— 2022b: Analyzing real options and flexibility in engineering systems design using decision rules and deep reinforcement learning. *Journal of Mechanical Design*, **144**, no. 2.

Castro, P. M., A. P. Barbosa-Póvoa, H. A. Matos, and A. Q. Novais, 2004: Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Industrial & engineering chemistry research*, **43**, no. 1, 105–118.

Castro, P. M., I. E. Grossmann, and Q. Zhang, 2018: Expanding scope and computational challenges in process scheduling. *Computers & Chemical Engineering*, **114**, 14–42.

Cerda, J., G. P. Henning, and I. E. Grossmann, 1997: A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. *Industrial & Engineering Chemistry Research*, **36**, no. 5, 1695–1707.

Chachuat, B., B. Srinivasan, and D. Bonvin, 2009: Adaptation strategies for real-time optimization. *Computers & Chemical Engineering*, **33**, no. 10, 1557–1567.

Chang, H. S., J. Hu, M. C. Fu, and S. I. Marcus, 2007: *Simulation-based algorithms for Markov decision processes*. Springer.

Chang, Z., S. Song, Y. Zhang, J.-Y. Ding, R. Zhang, and R. Chiong, 2017: Distributionally robust single machine scheduling with risk aversion. *European Journal of Operational Research*, **256**, no. 1, 261–274.

Charitopoulos, V. M., L. G. Papageorgiou, and V. Dua, 2019a: Closed-loop integration of planning, scheduling and multi-parametric nonlinear control. *Computers & Chemical Engineering*, **122**, 172–192.

— 2019b: Closed-loop integration of planning, scheduling and multi-parametric nonlinear control. *Computers & Chemical Engineering*, **122**, 172–192, iSBN: 0098-1354 Publisher: Elsevier.

Chen, J., B. Xin, Z. Peng, L. Dou, and J. Zhang, 2009: Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, **39**, no. 3, 680–691, iSBN: 1083-4427 Publisher: IEEE.

Chen, L., Y. Hontoir, D. Huang, J. Zhang, and A. J. Morris, 2004: Combining first principles with black-box techniques for reaction systems. *Control Engineering Practice*, **12**, no. 7, 819–826.

Chen, M., Z. Xu, J. Zhao, Y. Zhu, and Z. Shao, 2022: Nonparametric identification of batch process using two-dimensional kernel-based gaussian process regression. *Chemical Engineering Science*, **250**, 117372.

Chen, R. T., Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, 2018a: Neural ordinary differential equations. *Advances in neural information processing systems*, **31**.

Chen, S., A. M. Devraj, F. Lu, A. Busic, and S. Meyn, 2020: Zap q-learning with nonlinear function approximation. *Advances in Neural Information Processing Systems*, **33**, 16879–16890.

Chen, S., K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, 2018b: Approximating explicit model predictive control using constrained neural networks. *2018 Annual American control conference (ACC)*, IEEE, 1520–1527.

Cheng, Y., H. Fu, J. Lyu, Z. Wang, H. Li, and X. Chen, 2019: Optimisation estimation of uncertainty integrated with production information based on bayesian fusion method. *The Journal of Engineering*, **2019**, no. 23, 9178–9182.

Chiuso, A. and G. Pillonetto, 2019: System identification: A machine learning perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, **2**, 281–304.

Choromanski, K., M. Rowland, V. Sindhwani, R. Turner, and A. Weller, 2018: Structured evolution with compact architectures for scalable policy optimization. *International Conference on Machine Learning*, PMLR, 970–978.

Chu, Y. and F. You, 2013: Integration of scheduling and dynamic optimization of batch processes under uncertainty: Two-stage stochastic programming approach and enhanced generalized benders decomposition algorithm. *Industrial & Engineering Chemistry Research*, **52**, no. 47, 16851–16869.

Claesen, M. and B. De Moor, 2015: Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

Clements, W. R., B. V. Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth, 2020: *Estimating risk and uncertainty in deep reinforcement learning*.

Clerc, M., 2010: *Particle swarm optimization*, volume 93. John Wiley & Sons.

Clopper, C. J. and E. S. Pearson, 1934: The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, **26**, no. 4, 404–413, doi:10.2307/2331986.
URL http://www.jstor.org/stable/2331986

Coates, A., P. Abbeel, and A. Ng, 2009: Apprenticeship learning for helicopter control. *Communications of the ACM*, **52**, no. 7, 97–105, publisher: ACM.

Colah, C., 2015: *Understanding lstm networks*.
URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Conrad, K., 2014: The contraction mapping theorem. *Expository paper. University of Connecticut, College of Liberal Arts and Sciences, Department of Mathematics*.

Cooman, A., F. Seyfert, M. Olivi, S. Chevillard, and L. Baratchart, 2017: Model-free closed-loop stability analysis: A linear functional approach. *IEEE Transactions on Microwave Theory and Techniques*, **66**, no. 1, 73–80.

Cover, T. M., 1999: *Elements of information theory*. John Wiley & Sons.

Cox, P. B. and R. Tóth, 2021: Linear parameter-varying subspace identification: A unified framework. *Automatica*, **123**, 109296.

Curi, S., F. Berkenkamp, and A. Krause, 2020: Efficient model-based reinforcement learning through optimistic policy search and planning. *Advances in Neural Information Processing Systems*, **33**, 14156–14170.

Dabney, W., G. Ostrovski, D. Silver, and R. Munos, 2018a: Implicit quantile networks for distributional reinforcement learning. *International conference on machine learning*, PMLR, 1096–1105.

— 2018b: Implicit quantile networks for distributional reinforcement learning. *International conference on machine learning*, PMLR, 1096–1105.

Dabney, W., M. Rowland, M. Bellemare, and R. Munos, 2018c: Distributional reinforcement learning with quantile regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

— 2018d: Distributional reinforcement learning with quantile regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, issue: 1.

D'Alessandro, D., M. Dahleh, and I. Mezic, 1999: Control of mixing in fluid flow: A maximum entropy approach. *IEEE Transactions on Automatic Control*, **44**, no. 10, 1852–1863.

Darby, M. L., M. Nikolaou, J. Jones, and D. Nicholson, 2011: Rto: An overview and assessment of current practice. *Journal of Process control*, **21**, no. 6, 874–884.

De Boer, P.-T., D. P. Kroese, S. Mannor, and R. Y. Rubinstein, 2005: A tutorial on the cross-entropy method. *Annals of operations research*, **134**, no. 1, 19–67.

De Hert, S. C. and T. Rodgers, 2017: Continuous, recycle and batch emulsification kinetics using a high-shear mixer. *Chemical Engineering Science*, **167**, 265–277.

de la Penad, D. M., A. Bemporad, and T. Alamo, 2005: Stochastic programming applied to model predictive control. *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 1361–1366.

De Vito, E., L. Rosasco, A. Caponnetto, U. De Giovannini, F. Odone, and P. Bartlett, 2005: Learning from examples as an inverse problem. *Journal of Machine Learning Research*, **6**, no. 5.

Dedopoulos, I. T. and N. Shah, 1995: Optimal short-term scheduling of maintenance and production for multipurpose plants. *Industrial & engineering chemistry research*, **34**, no. 1, 192–201.

Degrave, J., F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, et al., 2022: Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, **602**, no. 7897, 414–419.

Deisenroth, M. and C. E. Rasmussen, 2011: Pilco: A model-based and data-efficient approach to policy search. *Proceedings of the 28th International Conference on machine learning (ICML-11)*, Citeseer, 465–472.

del Rio-Chanona, E. A., N. r. Ahmed, D. Zhang, Y. Lu, and K. Jing, 2017: Kinetic modeling and process analysis for desmodesmus sp. lutein photo-production. *AIChE Journal*, **63**, no. 7, 2546–2554.

del Rio-Chanona, E. A., E. Manirafasha, D. Zhang, Q. Yue, and K. Jing, 2016: Dynamic modeling and optimization of cyanobacterial c-phycocyanin production process by artificial neural network. *Algal Research*, **13**, 7–15.

del Rio Chanona, E. A., P. Petsagkourakis, E. Bradford, J. A. Graciano, and B. Chachuat, 2021: Real-time optimization meets bayesian optimization and derivative-free optimization: A tale of modifier adaptation. *Computers & Chemical Engineering*, **147**, 107249.

del Rio-Chanona, E. A., J. L. Wagner, H. Ali, F. Fiorelli, D. Zhang, and K. Hellgardt, 2019: Deep learning-based surrogate modeling and optimization for microalgal biofuel production and photobioreactor design. *AIChE Journal*, **65**, no. 3, 915–923, iSBN: 0001-1541 Publisher: Wiley Online Library.

Demirhan, C. D., F. Boukouvala, K. Kim, H. Song, W. W. Tso, C. A. Floudas, and E. N. Pistikopoulos, 2020: An integrated data-driven modeling & global optimization approach for multi-period nonlinear production planning problems. *Computers & Chemical Engineering*, **141**, 107007.

Dias, L. S. and M. G. Ierapetritou, 2020: Integration of planning, scheduling and control problems using data-driven feasibility analysis and surrogate models. *Computers & Chemical Engineering*, **134**, 106714.

Dias, L. S., R. C. Pattison, C. Tsay, M. Baldea, and M. G. Ierapetritou, 2018: A simulation-based optimization framework for integrating scheduling and model predictive control, and its application to air separation units. *Computers & Chemical Engineering*, **113**, 139–151.

Dogru, O., N. Wieczorek, K. Velswamy, F. Ibrahim, and B. Huang, 2021: Online reinforcement learning for a continuous space system with experimental validation. *Journal of Process Control*, **104**, 86–100.

Dong, Y., X. Tang, and Y. Yuan, 2020: Principled reward shaping for reinforcement learning via lyapunov stability theory. *Neurocomputing*, **393**, 83–90.

Doshi-Velez, F. and B. Kim, 2017: *Towards a rigorous science of interpretable machine learning*.
URL https://arxiv.org/abs/1702.08608

Duarte, B., P. Saraiva, and C. Pantelides, 2004: Combined mechanistic and empirical modelling. *International Journal of Chemical Reactor Engineering*, **2**, no. 1.

Dziak, J. J., D. L. Coffman, S. T. Lanza, R. Li, and L. S. Jermiin, 2020: Sensitivity and specificity of information criteria. *Briefings in bioinformatics*, **21**, no. 2, 553–565.

Ellis, M., H. Durand, and P. D. Christofides, 2014: A tutorial review of economic model predictive control methods. *Journal of Process Control*, **24**, no. 8, 1156–1178.

EMA, E. M. A., 2000: Ich topic q 7. good manufacturing practice for active pharmaceutical ingredients. *Step 5: Note for Guidance on Good Manufacturing Practice for Active Pharmaceutical Ingredients (CPMP/ICH/4106/00)*.

Emenike, S. N. and G. Falcone, 2020: A review on energy supply chain resilience through optimization. *Renewable and Sustainable Energy Reviews*, **134**, 110088, iSBN: 1364-0321 Publisher: Elsevier.

Engstrom, L., A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, 2019: Implementation matters in deep rl: A case study on ppo and trpo. *International conference on learning representations*.

Esmaili, N., C. Sammut, and G. Shirazi, 1995: Behavioural cloning in control of a dynamic system. *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, IEEE, volume 3, 2904–2909.

Estrada-Wiese, D., E. A. del Río-Chanona, and J. A. Del Río, 2018: Stochastic optimization of broadband reflecting photonic structures. *Scientific Reports*, **8**, no. 1, 1–9, iSBN: 2045-2322 Publisher: Nature Publishing Group.

Farina, M., L. Giulioni, L. Magni, and R. Scattolini, 2014: *An mpc approach to output-feedback control of stochastic linear discrete-time systems*.

Farina, M., L. Giulioni, and R. Scattolini, 2016: Stochastic linear model predictive control with chance constraints–a review. *Journal of Process Control*, **44**, 53–67.

Filippi, C., G. Guastaroba, and M. G. Speranza, 2020: Conditional value-at-risk beyond finance: a survey. *International Transactions in Operational Research*, **27**, no. 3, 1277–1319, iSBN: 0969-6016 Publisher: Wiley Online Library.

Finn, C., P. Christiano, P. Abbeel, and S. Levine, 2016a: *A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models*. URL https://arxiv.org/abs/1611.03852

Finn, C., S. Levine, and P. Abbeel, 2016b: Guided cost learning: Deep inverse optimal control via policy optimization. *International conference on machine learning*, PMLR, 49–58.

Fisac, J. F., A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, 2018: A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, **64**, no. 7, 2737–2752.

Fisher, M., 2020: Flushing out the true cause of the global toilet paper shortage amid coronavirus pandemic. *Washington Post, April*, **7**.

Fleming, J., B. Kouvaritakis, and M. Cannon, 2014: Robust tube mpc for linear systems with multiplicative uncertainty. *IEEE Transactions on Automatic Control*, **60**, no. 4, 1087–1092.

Floudas, C. A. and X. Lin, 2004: Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, **28**, no. 11, 2109–2129.

Forrester, J. W., 1997: Industrial dynamics. *Journal of the Operational Research Society*, **48**, no. 10, 1037–1041.

Fox, M. S., 1994: Isis: a retrospective. *Intelligent scheduling*, **1**, 3–28.

François-Lavet, V., R. Fonteneau, and D. Ernst, 2015: How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011*.

Frazier, P. I., 2018: *A tutorial on bayesian optimization*.

Frigola, R., 2015: *Bayesian time series learning with Gaussian processes*. Ph.D. thesis, University of Cambridge.

Fu, J., A. Kumar, M. Soh, and S. Levine, 2019a: Diagnosing bottlenecks in deep q-learning algorithms. *International Conference on Machine Learning*, PMLR, 2021–2030.

Fu, J., K. Luo, and S. Levine, 2017: *Learning robust rewards with adversarial inverse reinforcement learning*.
URL https://arxiv.org/abs/1710.11248

Fu, Y., H. Wang, G. Tian, Z. Li, and H. Hu, 2019b: Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm. *Journal of Intelligent Manufacturing*, **30**, no. 5, 2257–2272.

Fuentes-Cortes, L. F., A. Flores-Tlacuahuac, and K. D. Nigam, 2022: Machine learning algorithms used in pse environments: A didactic approach and critical perspective. *Industrial & Engineering Chemistry Research*.

Fujisawa, Y., Y. Otomo, Y. Ogata, Y. Nakamura, R. Fujita, Y. Ishitsuka, R. Watanabe, N. Okiyama, K. Ohara, and M. Fujimoto, 2019: Deep-learning-based, computer-aided classifier developed with a small dataset of clinical images surpasses board-certified dermatologists in skin tumour diagnosis. *British Journal of Dermatology*, **180**, no. 2, 373–381.

Fushiki, T., 2011: Estimation of prediction error by using k-fold cross-validation. *Statistics and Computing*, **21**, no. 2, 137–146.

Gao, J., C. Ning, and F. You, 2019: Data-driven distributionally robust optimization of shale gas supply chains under uncertainty. *AIChE Journal*, **65**, no. 3, 947–963, iSBN: 0001-1541 Publisher: Wiley Online Library.

Gao, W.-f. and S.-y. Liu, 2012: A modified artificial bee colony algorithm. *Computers & Operations Research*, **39**, no. 3, 687–697, iSBN: 0305-0548 Publisher: Elsevier.

García-Fernández, Á. F., L. Svensson, M. R. Morelande, and S. Särkkä, 2015: Posterior linearization filter: Principles and implementation using sigma points. *IEEE transactions on signal processing*, **63**, no. 20, 5561–5573.

Gautschi, W., 1996: Orthogonal polynomials: applications and computation. *Acta numerica*, **5**, 45–119.

Ge, Z., Z. Song, S. X. Ding, and B. Huang, 2017: Data mining and analytics in the process industry: The role of machine learning. *Ieee Access*, **5**, 20590–20616.

Georgiadis, G. P., A. P. Elekidis, and M. C. Georgiadis, 2019: Optimization-based scheduling for the process industries: from theory to real-life industrial applications. *Processes*, **7**, no. 7, 438.

Germscheid, S. H., A. Mitsos, and M. Dahmen, 2022: Demand response potential of industrial processes considering uncertain short-term electricity prices. *AIChE Journal*, e17828.

Ghosh, D., E. Hermonat, P. Mhaskar, S. Snowling, and R. Goel, 2019: Hybrid modeling approach integrating first-principles models with subspace identification. *Industrial & Engineering Chemistry Research*, **58**, no. 30, 13533–13543.

Gijsbrechts, J., R. N. Boute, J. A. Van Mieghem, and D. Zhang, 2021: Can deep reinforcement learning improve inventory management? performance on dual sourcing, lost sales and multi-echelon problems. *Manufacturing & Service Operations Management*.

Glavic, M., R. Fonteneau, and D. Ernst, 2017: Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine*, **50**, no. 1, 6918–6927.

Glover, F., 1989: Tabu search—part i. *ORSA Journal on computing*, **1**, no. 3, 190–206.

Goodfellow, I., Y. Bengio, and A. Courville, 2016: *Deep learning*. MIT press.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, 2014: Generative adversarial nets. *Advances in neural information processing systems*, **27**.

Gopaluni, R. B., A. Tulsyan, B. Chachuat, B. Huang, J. M. Lee, F. Amjad, S. K. Damarla, J. W. Kim, and N. P. Lawrence, 2020: Modern machine learning tools for monitoring and control of industrial processes: A survey. *IFAC-PapersOnLine*, **53**, no. 2, 218–229.

Gottesman, O., F. Johansson, M. Komorowski, A. Faisal, D. Sontag, F. Doshi-Velez, and L. A. Celi, 2019: Guidelines for reinforcement learning in healthcare. *Nature medicine*, **25**, no. 1, 16–18.

Gottipati, S. K., B. Sattarov, S. Niu, Y. Pathak, H. Wei, S. Liu, K. M. J. Thomas, S. Blackburn, C. W. Coley, J. Tang, S. Chandar, and Y. Bengio, 2020: Learning to navigate the synthetically accessible chemical space using reinforcement learning. *arXiv:2004.12485 [cs]*, arXiv: 2004.12485.
URL http://arxiv.org/abs/2004.12485

Göttl, Q., D. G. Grimm, and J. Burger, 2022: Automated synthesis of steady-state continuous processes using reinforcement learning. *Frontiers of Chemical Science and Engineering*, **16**, no. 2, 288–302.

Grand-Clément, J., 2021: From convex optimization to mdps: A review of first-order, second-order and quasi-newton methods for mdps. *arXiv preprint arXiv:2104.10677*.

Greenberg, I., Y. Chow, M. Ghavamzadeh, and S. Mannor, 2022: Efficient risk-averse reinforcement learning. *arXiv preprint arXiv:2205.05138*.

Gros, S. and M. Zanon, 2019a: Data-driven economic nmpc using reinforcement learning. *IEEE Transactions on Automatic Control*, **65**, no. 2, 636–648.

— 2019b: Data-driven economic nmpc using reinforcement learning. *arXiv:1904.04152 [cs]*, arXiv: 1904.04152.
URL http://arxiv.org/abs/1904.04152

Grossmann, I., 2005: Enterprise-wide optimization: A new frontier in process systems engineering. *AIChE Journal*, **51**, no. 7, 1846–1857.

Grossmann, I. E., R. M. Apap, B. A. Calfa, P. García-Herreros, and Q. Zhang, 2016: Recent advances in mathematical programming techniques for the optimization of process systems under uncertainty. *Computers & Chemical Engineering*, **91**, 3–14.

Guerra, O. J., A. J. Calderón, L. G. Papageorgiou, and G. V. Reklaitis, 2019: Integrated shale gas supply chain design and water management under uncertainty. *AIChE Journal*, **65**, no. 3, 924–936.

Gupta, A., P. Matta, and B. Pant, 2021: Graph neural network: Current state of art, challenges and applications. *Materials Today: Proceedings*, **46**, 10927–10932.

Gupta, D. and C. T. Maravelias, 2017: A general state-space formulation for online scheduling. *Processes*, **5**, no. 4, 69.

Gupta, D., C. T. Maravelias, and J. M. Wassick, 2016: From rescheduling to online scheduling. *Chemical Engineering Research and Design*, **116**, 83–97.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine, 2018: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, PMLR, 1861–1870.

Han, B.-A. and J.-J. Yang, 2020: Research on adaptive job shop scheduling problems based on dueling double dqn. *IEEE Access*, **8**, 186474–186495.

Hansen, N., D. V. Arnold, and A. Auger, 2015: Evolution strategies. *Springer handbook of computational intelligence*, Springer, 871–898.

Hansen, P., N. Mladenović, and J. A. Moreno Pérez, 2010: Variable neighbourhood search: methods and applications. *Annals of Operations Research*, **175**, no. 1, 367–407.

Harjunkoski, I., C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand, and J. Wassick, 2014: Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering*, **62**, 161–193.

Haroon, K., A. Arafeh, S. Cunliffe, P. Martin, T. Rodgers, Ć. Mendoza, and M. Baker, 2020: Comparison of individual and integrated inline raman, near-infrared, and mid-infrared spectroscopic models to predict the viscosity of micellar liquids. *Applied Spectroscopy*, **74**, no. 7, 819–831.

Haupt, R., 1989: A survey of priority rule-based scheduling. *Operations-Research-Spektrum*, **11**, no. 1, 3–16.

He, W., G. Li, P. Hao, and Y. Zeng, 2019: Maximum entropy method-based reliability analysis with correlated input variables via hybrid dimension-reduction method. *Journal of Mechanical Design*, **141**, no. 10.

Heckelei, T. and H. Wolff, 2003: Estimation of constrained optimisation models for agricultural supply analysis based on generalised maximum entropy. *European review of agricultural economics*, **30**, no. 1, 27–50.

Hedrick, E., K. Hedrick, D. Bhattacharyya, S. E. Zitney, and B. Omell, 2022: Reinforcement learning for online adaptation of model predictive controllers: Application to a selective catalytic reduction unit. *Computers & Chemical Engineering*, **160**, 107727.

Heess, N., J. J. Hunt, T. P. Lillicrap, and D. Silver, 2015: Memory-based control with recurrent neural networks.

Heirung, T. A. N., J. A. Paulson, J. O'Leary, and A. Mesbah, 2018: Stochastic model predictive control—how does it work? *Computers & Chemical Engineering*, **114**, 158–170.

Hernández-Lobato, J. M. and R. Adams, 2015: Probabilistic backpropagation for scalable learning of bayesian neural networks. *International conference on machine learning*, PMLR, 1861–1869.

Hernández-García, A. and P. König, 2018: *Data augmentation instead of explicit regularization*.
URL https://arxiv.org/abs/1806.03852

Ho, J. and S. Ermon, 2016: Generative adversarial imitation learning. *Advances in neural information processing systems*, **29**.

Hochreiter, S. and J. Schmidhuber, 1997: Long short-term memory. *Neural Computation*, **9**, no. 8, 1735–17380, publisher: MIT Press.

Holtorf, F., A. Mitsos, and L. T. Biegler, 2019: Multistage nmpc with on-line generated scenario trees: Application to a semi-batch polymerization process. *Journal of Process Control*, **80**, 167–179.

Hong, L. J. and G. Liu, 2009: Simulating sensitivities of conditional value at risk. *Management Science*, **55**, no. 2, 281–293.

— 2011: Monte Carlo estimation of value-at-risk, conditional value-at-risk and their sensitivities. *Proceedings of the 2011 Winter Simulation Conference (WSC)*, IEEE, 95–107.

Hong, S., K. Jang, J. Lee, H. Yoon, I. Moon, et al., 2020: Optimal evacuation route prediction in fpso based on deep q-network. *Computer Aided Chemical Engineering*, Elsevier, volume 48, 1867–1872.

Horgan, C. C., M. Jensen, A. Nagelkerke, J.-P. St-Pierre, T. Vercauteren, M. M. Stevens, and M. S. Bergholt, 2021: High-throughput molecular imaging via deep-learning-enabled raman spectroscopy. *Analytical Chemistry*, **93**, no. 48, 15850–15860.

Hubbs, C. D., C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, 2020a: A deep reinforcement learning approach for chemical production scheduling. *Computers & Chemical Engineering*, **141**, 106982.

Hubbs, C. D., H. D. Perez, O. Sarwar, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, 2020b: *Or-gym: A reinforcement learning library for operations research problems*.

Huh, S. and I. Yang, 2020: Safe reinforcement learning for probabilistic reachability and safety specifications: A lyapunov-based approach. *arXiv preprint arXiv:2002.10126*.

Hüllen, G., J. Zhai, S. H. Kim, A. Sinha, M. J. Realff, and F. Boukouvala, 2020: Managing uncertainty in data-driven simulation-based optimization. *Computers & Chemical Engineering*, **136**, 106519.

Hüllermeier, E. and W. Waegeman, 2019: Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *arXiv preprint arXiv:1910.09457*.

Hussain, K., M. N. M. Salleh, S. Cheng, and R. Naseem, 2017: Common benchmark functions for metaheuristic evaluation: A review. *JOIV: International Journal on Informatics Visualization*, **1**, no. 4-2, 218–223.

Hussein, A. S., C. M. Elias, and E. I. Morgan, 2019: A realistic model predictive control using single and multiple shooting in the formulation of non-linear programming model. *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, IEEE, 1–6.

Ibrahim, D., Z. Kis, K. Tak, M. M. Papathanasiou, C. Kontoravdi, B. Chachuat, and N. Shah, 2021: Model-Based Planning and Delivery of Mass Vaccination Campaigns against Infectious Disease: Application to the COVID-19 Pandemic in the UK. *Vaccines*, **9**, no. 12, 1460, publisher: Multidisciplinary Digital Publishing Institute.

Ilyas, A., L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, 2018: Are deep policy gradient algorithms truly policy gradient algorithms?

ISA, I. S. o. A., 2022: *Isa95, enterprise-control system integration*. https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95, accessed: 2022-03-28.

Jaderberg, M., V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al., 2017: Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

Jakobsen, H. A., 2008: Chemical reactor modeling. *Multiphase Reactive Flows*.

Jaynes, E. T., 1957: Information theory and statistical mechanics. *Physical review*, **106**, no. 4, 620.

Jiang, Q., X. Yan, and B. Huang, 2019: Review and perspectives of data-driven distributed monitoring for industrial plant-wide processes. *Industrial & Engineering Chemistry Research*, **58**, no. 29, 12899–12912.

Jin, C., Z. Yang, Z. Wang, and M. I. Jordan, 2020: Provably efficient reinforcement learning with linear function approximation. *Conference on Learning Theory*, PMLR, 2137–2143.

Jing, K., Y. Tang, C. Yao, E. A. del Rio-Chanona, X. Ling, and D. Zhang, 2018: Overproduction of l-tryptophan via simultaneous feed of glucose and anthranilic acid from recombinant escherichia coli w3110: Kinetic modeling and process scale-up. *Biotechnology and bioengineering*, **115**, no. 2, 371–381.

Jones, D. R., M. Schonlau, and W. J. Welch, 1998: Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, **13**, no. 4, 455–492.

Joshi, T., S. Makker, H. Kodamana, and H. Kandath, 2021: *Application of twin delayed deep deterministic policy gradient learning for the control of transesterification process*.

Juan, A. A., J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira, 2015: A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, **2**, 62–72.

Kadlec, P., B. Gabrys, and S. Strandt, 2009: Data-driven soft sensors in the process industry. *Computers & chemical engineering*, **33**, no. 4, 795–814.

Kakade, S. M., 2001: A natural policy gradient. *Advances in neural information processing systems*, **14**.

Kanervisto, A., J. Karttunen, and V. Hautamäki, 2020a: Playing minecraft with behavioural cloning. *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, H. J. Escalante and R. Hadsell, Eds., PMLR, volume 123 of *Proceedings of Machine Learning Research*, 56–66.
URL https://proceedings.mlr.press/v123/kanervisto20a.html

Kanervisto, A., C. Scheller, and V. Hautamäki, 2020b: Action space shaping in deep reinforcement learning. *2020 IEEE Conference on Games (CoG)*, IEEE, 479–486.

Kang, J., A. U. Raghunathan, and S. Di Cairano, 2015: Decomposition via admm for scenario-based model predictive control. *2015 American Control Conference (ACC)*, IEEE, 1246–1251.

Kannan, G., P. Sasikumar, and K. Devika, 2010: A genetic algorithm approach for solving a closed loop supply chain model: A case of battery recycling. *Applied mathematical modelling*, **34**, no. 3, 655–670, iSBN: 0307-904X Publisher: Elsevier.

Kara, A. and I. Dogan, 2018: Reinforcement learning approaches for specifying ordering policies of perishable inventory systems. *Expert Systems with Applications*, **91**, 150–158, iSBN: 0957-4174 Publisher: Elsevier.

Karaboga, D., 2005: An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes university, engineering faculty, computer ....

Karg, B., T. Alamo, and S. Lucia, 2019: Probabilistic performance validation of deep learning-based robust nmpc controllers. *arXiv:1910.13906 [cs, eess, math]*, arXiv: 1910.13906.
URL http://arxiv.org/abs/1910.13906

Karimi, N. and H. Davoudpour, 2015: A branch and bound method for solving multi-factory supply chain scheduling with batch delivery. *Expert Systems with Applications*, **42**, no. 1, 238–245, iSBN: 0957-4174 Publisher: Elsevier.

Katayama, T. et al., 2005: *Subspace methods for system identification*, volume 1. Springer.

Kelly, M., 2017: An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, **59**, no. 4, 849–904.

Kennedy, J. and R. Eberhart, 1995a: Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*, IEEE, volume 4, 1942–1948.

— 1995b: Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*, IEEE, volume 4, 1942–1948.

Khan, A., 2018: Long-term production scheduling of open pit mines using particle swarm and bat algorithms under grade uncertainty. *Journal of the Southern African Institute of Mining and Metallurgy*, **118**, no. 4, 361–368.

Khan, A. A. and A. A. Lapkin, 2022: Designing the process designer: Hierarchical reinforcement learning for optimisation-based process design. *Chemical Engineering and Processing-Process Intensification*, 108885.

Kidambi, R., A. Rajeswaran, P. Netrapalli, and T. Joachims, 2021: *Morel : Model-based offline reinforcement learning*.

Kim, J. et al., 2021a: *DQN Learning Approach to Scheduling in Multi-job Production Systems*. Ph.D. thesis, DGIST.

Kim, J. W., N. Krausch, J. Aizpuru, T. Barz, S. Lucia, P. Neubauer, and M. N. C. Bournazou, 2022: Model predictive control and moving horizon estimation for adaptive optimal bolus feeding in high-throughput cultivation of e. coli. *arXiv preprint arXiv:2203.07211*.

Kim, J. W., B. J. Park, T. H. Oh, and J. M. Lee, 2021b: Model-based reinforcement learning and predictive control for two-stage optimal control of fed-batch bioreactor. *Computers & Chemical Engineering*, **154**, 107465.

Kim, J. W., B. J. Park, H. Yoo, T. H. Oh, J. H. Lee, and J. M. Lee, 2020a: A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system. *Journal of Process Control*, **87**, 166–178.

— 2020b: A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system. *Journal of Process Control*, **87**, 166–178.

Kim, Y. and J. M. Lee, 2020: Model-based reinforcement learning for nonlinear optimal control with practical asymptotic stability guarantees. *AIChE Journal*, **n/a**, no. n/a, e16544, doi:10.1002/aic.16544.

Kingma, D. P. and J. Ba, 2014: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirk, D. E., 1998: *Optimal control theory : an introduction*. Dover Books on Electrical Engineering Ser., Dover Publications, Mineola, N.Y.

— 2004: *Optimal control theory: an introduction*. Courier Corporation.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, 1983a: Optimization by simulated annealing. *science*, **220**, no. 4598, 671–680.

Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi, 1983b: Optimization by simulated annealing. *science*, **220**, no. 4598, 671–680, iSBN: 0036-8075 Publisher: American association for the advancement of science.

Kis, Z., C. Kontoravdi, R. Shattock, and N. Shah, 2020: Resources, production scales and time required for producing rna vaccines for the global pandemic demand. *Vaccines*, **9**, no. 1, 3.

Koch, A. L., 1998: The monod model and its alternatives. *Mathematical modeling in microbial ecology*, Springer, 62–93.

Köhler, J., R. Soloperto, M. A. Müller, and F. Allgöwer, 2020: A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, **66**, no. 2, 794–801.

Kolmogorov, A. N. and S. V. Fomin, 1957: *Elements of the theory of functions and functional analysis*, volume 1. Courier Corporation.

Kondili, E., C. C. Pantelides, and R. W. Sargent, 1993: A general algorithm for short-term scheduling of batch operations—i. milp formulation. *Computers & Chemical Engineering*, **17**, no. 2, 211–227.

Konishi, S. and G. Kitagawa, 1996: Generalised information criteria in model selection. *Biometrika*, **83**, no. 4, 875–890.

Kouvaritakis, B. and M. Cannon, 2016: Model predictive control. *Switzerland: Springer International Publishing*, 38.

Krishnamoorthy, D., B. Foss, and S. Skogestad, 2019: A primal decomposition algorithm for distributed multistage scenario model predictive control. *Journal of Process Control*, **81**, 162–171.

Krishnamoorthy, D. and S. Skogestad, 2022: Real-time optimization as a feedback control problem-a review. *Computers & Chemical Engineering*, 107723.

Krishnamoorthy, D., M. Thombre, S. Skogestad, and J. Jäschke, 2018: Data-driven scenario selection for multistage robust model predictive control. *IFAC-PapersOnLine*, **51**, no. 20, 462–468.

Krueger, D., E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. L. Priol, and A. Courville, 2021: Out-of-distribution generalization via risk extrapolation (rex). *Proceedings of the 38th International Conference on Machine Learning*,

M. Meila and T. Zhang, Eds., PMLR, volume 139 of *Proceedings of Machine Learning Research*, 5815–5826.
URL https://proceedings.mlr.press/v139/krueger21a.html

Kuleshov, V. and D. Precup, 2014: Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*.

Kumar, A., R. Agarwal, T. Ma, A. Courville, G. Tucker, and S. Levine, 2021: Dr3: Value-based deep reinforcement learning requires explicit regularization. *arXiv preprint arXiv:2112.04716*.

Kumar, A., R. Agarwal, G. Tucker, L. Li, D. Precup, and A. Kumar, 2020a: Workshop: Offline reinforcement learning, neural Information Processing Systems Online Conference 2020.

Kumar, A., J. Hong, A. Singh, and S. Levine, 2022: *When should we prefer offline reinforcement learning over behavioral cloning?*.

Kumar, A., A. Zhou, G. Tucker, and S. Levine, 2020b: *Conservative q-learning for offline reinforcement learning*.

Langson, W., I. Chryssochoos, S. Raković, and D. Q. Mayne, 2004: Robust model predictive control using tubes. *Automatica*, **40**, no. 1, 125–133.

Lara, C. L., J. D. Siirola, and I. E. Grossmann, 2020: Electric power infrastructure planning under uncertainty: stochastic dual dynamic integer programming (SDDiP) and parallelization scheme. *Optimization and Engineering*, **21**, no. 4, 1243–1281, iSBN: 1573-2924 Publisher: Springer.

Laroque, C., A. Klaas, J.-H. Fischer, and M. Kuntze, 2012: Fast converging, automated experiment runs for material flow simulations using distributed computing and combined metaheuristics. *Proceedings of the 2012 Winter Simulation Conference (WSC)*, IEEE, 1–12.

Larson, J., M. Menickelly, and S. M. Wild, 2019: Derivative-free optimization methods. *arXiv preprint arXiv:1904.11585*.

Lasi, H., P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, 2014: Industry 4.0. *Business & information systems engineering*, **6**, no. 4, 239–242.

Laskin, M., K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, 2020: Reinforcement learning with augmented data. *Advances in neural information processing systems*, **33**, 19884–19895.

Lawrence, N. P., M. G. Forbes, P. D. Loewen, D. G. McClement, J. U. Backström, and R. B. Gopaluni, 2022: Deep reinforcement learning with shallow controllers: An experimental application to pid tuning. *Control Engineering Practice*, **121**, 105046.

Lawrence, N. P., G. E. Stewart, P. D. Loewen, M. G. Forbes, J. U. Backstrom, and R. B. Gopaluni, 2020: Optimal pid and antiwindup control design as a reinforcement learning problem. *arXiv:2005.04539 [cs, eess, math]*, arXiv: 2005.04539.
URL http://arxiv.org/abs/2005.04539

LeCun, Y., Y. Bengio, and G. Hinton, 2015: Deep learning. *nature*, **521**, no. 7553, 436–444.

Lee, H. L., V. Padmanabhan, and S. Whang, 1997: Information distortion in a supply chain: The bullwhip effect. *Management science*, **43**, no. 4, 546–558, iSBN: 0025-1909 Publisher: Informs.

Lee, J. H., 2011: Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, **9**, no. 3, 415–424.

Lee, J. H., J. Shin, and M. J. Realff, 2018a: Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, **114**, 111–121.

— 2018b: Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, **114**, 111–121, iSBN: 0098-1354 Publisher: Elsevier.

Lee, J. M. and J. H. Lee, 2005: Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes. *Automatica*, **41**, no. 7, 1281–1288.

Lehman, J., J. Chen, J. Clune, and K. O. Stanley, 2017: Es is more than just a traditional finite-difference approximator.

Letsios, D., J. T. Bradley, R. Misener, N. Page, et al., 2021: Approximate and robust bounded job start scheduling for royal mail delivery offices. *Journal of Scheduling*, **24**, no. 2, 237–258.

Leurent, E., D. Efimov, and O.-A. Maillard, 2020: *Robust-adaptive control of linear systems: beyond quadratic costs*.

Levine, S., A. Kumar, G. Tucker, and J. Fu, 2020: *Offline reinforcement learning: Tutorial, review, and perspectives on open problems*.
URL https://arxiv.org/abs/2005.01643

Li, C. and I. E. Grossmann, 2021: A review of stochastic programming methods for optimization of process systems under uncertainty. *Frontiers in Chemical Engineering*, **2**, 622241.

Li, P., H. Arellano-Garcia, and G. Wozny, 2008: Chance constrained programming approach to process optimization under uncertainty. *Computers & chemical engineering*, **32**, no. 1-2, 25–45.

Li, Y., 2017: Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Li, Y., W. Huang, R. Wu, and K. Guo, 2020: An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem. *Applied Soft Computing*, **95**, 106544.

Li, Y., N. Li, H. E. Tseng, A. Girard, D. Filev, and I. Kolmanovsky, 2021: *Safe reinforcement learning using robust action governor*.

Li, Z. and C. A. Floudas, 2014: Optimal scenario reduction framework based on distance of uncertainty distribution and output performance: I. single reduction via mixed integer linear optimization. *Computers & Chemical Engineering*, **70**, 50–66.

— 2016: Optimal scenario reduction framework based on distance of uncertainty distribution and output performance: Ii. sequential reduction. *Computers & Chemical Engineering*, **84**, 599–610.

Li, Z. and M. Ierapetritou, 2008a: Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, **32**, no. 4-5, 715–727.

Li, Z. and M. G. Ierapetritou, 2008b: Robust optimization for process scheduling under uncertainty. *Industrial & Engineering Chemistry Research*, **47**, no. 12, 4148–4157.

Liashchynskyi, P. and P. Liashchynskyi, 2019: Grid search, random search, genetic algorithm: a big comparison for nas. *arXiv preprint arXiv:1912.06059*.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, 2015: Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lim, S., A. Joseph, L. Le, Y. Pan, and M. White, 2018: Actor-expert: A framework for using q-learning in continuous action spaces. *arXiv preprint arXiv:1810.09103*.

Lin, J. T. and C.-J. Huang, 2014: A simulation-based optimization approach for a semiconductor photobay with automated material handling system. *Simulation Modelling Practice and Theory*, **46**, 76–100.

Lin, Z. and Z. Bai, 2011: *Probability inequalities*. Springer Science & Business Media.

Lindgren, G., 2012: *Stationary stochastic processes: theory and applications*. CRC Press.

Liu, D., Q. Wei, D. Wang, X. Yang, and H. Li, 2017: Overview of adaptive dynamic programming. *Adaptive Dynamic Programming with Applications in Optimal Control*, Springer International Publishing, Cham, 1–33.
URL https://doi.org/10.1007/978-3-319-50815-3_1

Liu, Q., Z. Wang, X. He, and D. Zhou, 2014: A survey of event-based strategies on control and estimation. *Systems Science & Control Engineering: An Open Access Journal*, **2**, no. 1, 90–97.

Liu, Y. and I. Karimi, 2008: Scheduling multistage batch plants with parallel units and no interstage storage. *Computers & Chemical Engineering*, **32**, no. 4-5, 671–693.

Lodi, A., 2010: Mixed integer programming computation. *50 years of integer programming 1958-2008*, Springer, 619–645.

Lofberg, J., 2003: Approximations of closed-loop minimax mpc. *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, IEEE, volume 2, 1438–1442.

Löfberg, J., 2003: *Minimax approaches to robust model predictive control*, volume 812. Linköping University Electronic Press.

Lombardi, O., F. Holik, and L. Vanni, 2016: What is shannon information? *Synthese*, **193**, no. 7, 1983–2012.

Lorenzo, P. R., J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, 2017: Particle swarm optimization for hyper-parameter selection in deep neural networks. *Proceedings of the genetic and evolutionary computation conference*, 481–488.

Lu, C.-H. and C.-C. Tsai, 2008: Adaptive predictive control with recurrent neural network for industrial processes: An application to temperature control of a variable-frequency oil-cooling machine. *IEEE Transactions on Industrial Electronics*, **55**, no. 3, 1366–1375.

Lu, S., J. H. Lee, and F. You, 2020: Soft-constrained model predictive control based on data-driven distributionally robust optimization. *AIChE Journal*, **66**, no. 10, e16546.

Lucia, S. and B. Karg, 2018: A deep learning-based approach to robust nonlinear model predictive control. *IFAC-PapersOnLine*, **51**, no. 20, 511–516.

Lütjens, B., M. Everett, and J. P. How, 2019: Safe reinforcement learning with model uncertainty estimates. *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 8662–8668.

Luus, R., 1993: Application of dynamic programming to differential-algebraic process systems. *Computers & chemical engineering*, **17**, no. 4, 373–377.

Ma, X., L. Xia, Z. Zhou, J. Yang, and Q. Zhao, 2020: DSAC: Distributional soft actor critic for risk-sensitive reinforcement learning. *arXiv preprint arXiv:2004.14547*.

Ma, Y., D. Jayaraman, and O. Bastani, 2021: Conservative offline distributional reinforcement learning. *Advances in Neural Information Processing Systems*, **34**.

Ma, Y., W. Zhu, M. G. Benton, and J. Romagnoli, 2019: Continuous control of a polymerization system with deep reinforcement learning. *Journal of Process Control*, **75**, 40–47.

Magni, L., D. Pala, and R. Scattolini, 2009: Stochastic model predictive control of constrained linear systems with additive uncertainty. *2009 European Control Conference (ECC)*, IEEE, 2235–2240.

Mao, X., 2015: The truncated euler–maruyama method for stochastic differential equations. *Journal of Computational and Applied Mathematics*, **290**, no. C, 370–384, publisher: Elsevier B.V.

Maravelias, C., 2021a: *Chemical production scheduling : mixed-integer programming models and methods*. Cambridge series in chemical engineering, Cambridge University Press, Cambridge ; New York, NY.

Maravelias, C. T., 2012: General framework and modeling approach classification for chemical production scheduling. *AIChE Journal*, **58**, no. 6, 1812–1828.

— 2021b: *Chemical Production Scheduling: Mixed-integer Programming Models and Methods*. Cambridge University Press.

Maravelias, C. T. and I. E. Grossmann, 2003: New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial & engineering chemistry research*, **42**, no. 13, 3056–3074.

Maravelias, C. T. and C. Sung, 2009: Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers & Chemical Engineering*, **33**, no. 12, 1919–1930.

Marchetti, A. G., G. François, T. Faulwasser, and D. Bonvin, 2016: Modifier adaptation for real-time optimization—methods and applications. *Processes*, **4**, no. 4, 55.

Markana, A., N. Padhiyar, and K. Moudgalya, 2018: Multi-criterion control of a bioprocess in fed-batch reactor using ekf based economic model predictive control. *Chemical Engineering Research and Design*, **136**, 282–294.

Marti, R., S. Lucia, D. Sarabia, R. Paulen, S. Engell, and C. de Prada, 2015: Improving scenario decomposition algorithms for robust nonlinear model predictive control. *Computers & Chemical Engineering*, **79**, 30–45.

Martínez, B., M. Rodríguez, and I. Díaz, 2022: Cstr control with deep reinforcement learning. *Computer Aided Chemical Engineering*, Elsevier, volume 49, 1693–1698.

Mayne, D., 2016: Robust and stochastic model predictive control: Are we going in the right direction? *Annual Reviews in Control*, **41**, 184–192.

Mayne, D. Q., E. C. Kerrigan, E. Van Wyk, and P. Falugi, 2011: Tube-based robust nonlinear model predictive control. *International journal of robust and nonlinear control*, **21**, no. 11, 1341–1353.

Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. Scokaert, 2000: Constrained model predictive control: Stability and optimality. *Automatica*, **36**, no. 6, 789–814.

McAllister, R. D., J. B. Rawlings, and C. T. Maravelias, 2022: The inherent robustness of closed-loop scheduling. *Computers & Chemical Engineering*, **159**, 107678.

McBride, K. and K. Sundmacher, 2019: Overview of surrogate modeling in chemical process engineering. *Chemie Ingenieur Technik*, **91**, no. 3, 228–239.

McClement, D. G., N. P. Lawrence, M. G. Forbes, P. D. Loewen, J. U. Backström, and R. B. Gopaluni, 2022: Meta-reinforcement learning for adaptive control of second order systems.

McClement, D. G., N. P. Lawrence, P. D. Loewen, M. G. Forbes, J. U. Backström, and R. B. Gopaluni, 2021: *A meta-reinforcement learning approach to process control*.

McFarlane, R., 2018: A survey of exploration strategies in reinforcement learning. *McGill University*.

Melo, F. S., S. P. Meyn, and M. I. Ribeiro, 2008: An analysis of reinforcement learning with function approximation. *Proceedings of the 25th international conference on Machine learning*, 664–671.

Melo, F. S. and M. I. Ribeiro, 2007: Q-learning with linear function approximation. *International Conference on Computational Learning Theory*, Springer, 308–322.

Melo, M. T., S. Nickel, and F. Saldanha-da Gama, 2012: A tabu search heuristic for redesigning a multi-echelon supply chain network over a planning horizon. *International Journal of Production Economics*, **136**, no. 1, 218–230, iSBN: 0925-5273 Publisher: Elsevier.

Memarian, A., S. K. Varanasi, and B. Huang, 2021: Mixture robust semi-supervised probabilistic principal component regression with missing input data. *Chemometrics and Intelligent Laboratory Systems*, **214**, 104315.

Memmel, M., P. Liu, D. Tateo, and J. Peters, 2022: Dimensionality Reduction and Prioritized Exploration for Policy Search. *arXiv preprint arXiv:2203.04791*.

Mencarelli, L., Q. Chen, A. Pagot, and I. E. Grossmann, 2020: A review on superstructure optimization approaches in process system engineering. *Computers & Chemical Engineering*, **136**, 106808.

Méndez, C., G. Henning, and J. Cerdá, 2000: Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers & Chemical Engineering*, **24**, no. 9-10, 2223–2245.

Méndez, C. A., J. Cerdá, I. E. Grossmann, I. Harjunkoski, and M. Fahl, 2006: State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & chemical engineering*, **30**, no. 6-7, 913–946.

Mendiola-Rodriguez, T. A. and L. A. Ricardez-Sandoval, 2022: Robust control for anaerobic digestion systems of tequila vinasses under uncertainty: A deep deterministic policy gradient algorithm. *Digital Chemical Engineering*, **3**, 100023.

Meraz, M., V. Sanchez-Vazquez, and F. Martinez-Martinez, 2022: A systematic derivation of the monod equation for multi-substrate conditions. *Revista Mexicana De Ingeniería Química*, **21**, no. 2, Bio2798–Bio2798.

Merchant, A., L. Metz, S. S. Schoenholz, and E. D. Cubuk, 2021: Learn2hop: Learned optimization on rough landscapes. *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., PMLR, volume 139 of *Proceedings of Machine Learning Research*, 7643–7653.
URL https://proceedings.mlr.press/v139/merchant21a.html

Mesbah, A., 2016: Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, **36**, no. 6, 30–44.

Mesbah, A., I. V. Kolmanovsky, and S. Di Cairano, 2019: Stochastic model predictive control. *Handbook of Model Predictive Control*, Springer, 75–97.

Mesbah, A., S. Streif, R. Findeisen, and R. D. Braatz, 2014: Stochastic nonlinear model predictive control with probabilistic constraints. *2014 American control conference*, IEEE, 2413–2419.

Meyn, S., 2022: *Control Systems and Reinforcement Learning*. Cambridge University Press.

Michie, D., M. Bain, and J. Hayes-Miches, 1990: Cognitive models from subcognitive skills. *IEE control engineering series*, **44**, 71–99.

Min, K., H. Kim, and K. Huh, 2019: Deep distributional reinforcement learning based high-level driving policy determination. *IEEE Transactions on Intelligent Vehicles*, **4**, no. 3, 416–424, iSBN: 2379-8904 Publisher: IEEE.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, 2016: Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, PMLR, 1928–1937.

Mnih, V., N. Heess, and A. Graves, 2014: Recurrent models of visual attention. *Advances in neural information processing systems*, **27**.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, 2013: Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, 2015a: Human-level control through deep reinforcement learning. *nature*, **518**, no. 7540, 529–533, iSBN: 1476-4687 Publisher: Nature Publishing Group.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, 2015b: Human-level control through deep reinforcement learning. *Nature*, **518**, no. 7540, 529–52933, publisher: Nature Publishing Group.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., 2015c: Human-level control through deep reinforcement learning. *nature*, **518**, no. 7540, 529–533.

Mouret, S., I. E. Grossmann, and P. Pestiaux, 2011: Time representations and mathematical models for process scheduling problems. *Computers & Chemical Engineering*, **35**, no. 6, 1038–1063, doi:10.1016/j.compchemeng.2010.07.007. URL https://www.sciencedirect.com/science/article/pii/S0098135410002553

Mowbray, M., P. Petsagkourakis, E. A. del Rio-Chanona, and D. Zhang, 2022a: Safe chance constrained reinforcement learning for batch process control. *Computers & Chemical Engineering*, **157**, 107630, iSBN: 0098-1354 Publisher: Elsevier.

Mowbray, M., R. Smith, E. A. Del Rio-Chanona, and D. Zhang, 2021: Using process data to generate an optimal control policy via apprenticeship and reinforcement learning. *AIChE Journal*, e17306.

Mowbray, M., D. Zhang, and E. A. D. R. Chanona, 2022b: Distributional Reinforcement Learning for Scheduling of (Bio) chemical Production Processes. *arXiv preprint arXiv:2203.00636*.

Mukhoti, J., A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal, 2021: Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*.

Munikoti, S., D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, 2022: *Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications*.
URL https://arxiv.org/abs/2206.07922

Muñoz-Cobo, J.-L., R. Mendizábal, A. Miquel, C. Berna, and A. Escrivá, 2017: Use of the principles of maximum entropy and maximum relative entropy for the determination of uncertain parameter distributions in engineering applications. *Entropy*, **19**, no. 9, 486.

Murphy, K. P., 2022: *Probabilistic machine learning: an introduction*. MIT press.

— 2023: *Probabilistic Machine Learning: Advanced Topics*. MIT Press.
URL probml.ai

Nagabandi, A., I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, 2018: Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*.

Najjarbashi, A. and G. J. Lim, 2019: A variability reduction method for the operating room scheduling problem under uncertainty using cvar. *Operations Research for Health Care*, **20**, 25–32.

Nakamura-Zimmerer, T., Q. Gong, and W. Kang, 2021: *Neural network optimal feedback control with enhanced closed loop stability*.
URL https://arxiv.org/abs/2109.07466

Nemirovski, A. and A. Shapiro, 2006: Scenario approximations of chance constraints. *Probabilistic and randomized methods for design under uncertainty*, 3–47.

Nestor, B., M. B. McDermott, W. Boag, G. Berner, T. Naumann, M. C. Hughes, A. Goldenberg, and M. Ghassemi, 2019: Feature robustness in non-stationary health records: caveats to deployable model performance in common clinical machine learning tasks. *Machine Learning for Healthcare Conference*, PMLR, 381–405.

Neu, G., A. Jonsson, and V. Gómez, 2017: *A unified view of entropy-regularized markov decision processes*.

Ng, A. Y., D. Harada, and S. Russell, 1999: Policy invariance under reward transformations: Theory and application to reward shaping. *Icml*, volume 99, 278–287.

Ng, A. Y., S. Russell, et al., 2000: Algorithms for inverse reinforcement learning. *Icml*, volume 1, 2.

Nguyen, S., Y. Mei, B. Xue, and M. Zhang, 2019: A hybrid genetic programming algorithm for automated design of dispatching rules. *Evolutionary computation*, **27**, no. 3, 467–496.

Nian, R., J. Liu, and B. Huang, 2020: A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, **139**, 106886.

Nikita, S., A. Tiwari, D. Sonawat, H. Kodamana, and A. S. Rathore, 2021: Reinforcement learning based optimization of process chromatography for continuous processing of biopharmaceuticals. *Chemical Engineering Science*, **230**, 116171.

Nnaisense, 2022a: .
URL https://evotorch.ai/

— 2022b: *Empowering industry*.
URL https://nnaisense.com/case-studies/

Nocedal, J. and S. Wright, 2006: *Numerical optimization*. Springer Science & Business Media.

Nomikos, P. and J. F. MacGregor, 1994: Monitoring batch processes using multiway principal component analysis. *AIChE Journal*, **40**, no. 8, 1361–1375.

Nota, C. and P. S. Thomas, 2019a: Is the policy gradient a gradient? *arXiv preprint arXiv:1906.07073*.

— 2019b: Is the policy gradient a gradient? *arXiv preprint arXiv:1906.07073*.

— 2020: *Is the policy gradient a gradient?*.

Ochoa, G., J. A. Vázquez-Rodríguez, S. Petrovic, and E. Burke, 2009: Dispatching rules for production scheduling: A hyper-heuristic landscape analysis. *2009 IEEE congress on evolutionary computation*, IEEE, 1873–1880.

Oden, J. T., 2018: Adaptive multiscale predictive modelling. *Acta Numerica*, **27**, 353–450.

Ogasawara, H., 2019: The multiple cantelli inequalities. *Statistical Methods & Applications*, **28**, no. 3, 495–506.

Oh, T. H., H. M. Park, J. W. Kim, and J. M. Lee, 2022: Integration of reinforcement learning and model predictive control to optimize semi-batch bioreactor. *AIChE Journal*, **68**, no. 6, e17658.

Okpoti, E. S. and I.-J. Jeong, 2021: A reactive decentralized coordination algorithm for event-driven production planning and control: A cyber-physical production system prototype case study. *Journal of manufacturing systems*, **58**, 143–158.

ONS, O. f. N. S., 2021: *Uk manufacturers' sales by product: 2020 results*.

Oroojlooyjadid, A., M. Nazari, L. Snyder, and M. Takác, 2017: A deep q-network for the beer game with partial information. *arXiv preprint arXiv:1708.05924*.

Osinenko, P., D. Dobriborsci, and W. Aumer, 2022: Reinforcement learning with guarantees: a review. *IFAC-PapersOnLine*, **55**, no. 15, 123–128.

Ou, R., G. Pan, and T. Faulwasser, 2022: Data-driven multiple shooting for stochastic optimal control. *IEEE Control Systems Letters*, **7**, 313–318.

Oyebolu, F. B., R. Allmendinger, S. S. Farid, and J. Branke, 2019: Dynamic scheduling of multi-product continuous biopharmaceutical facilities: A hyper-heuristic framework. *Computers & Chemical Engineering*, **125**, 71–88.

Özkan, G., H. Hapoğlu, and M. Alpbaz, 2006: Non-linear generalised predictive control of a jacketed well mixed tank as applied to a batch process—a polymerisation reaction. *Applied Thermal Engineering*, **26**, no. 7, 720–726.

Palombarini, J. A., J. C. Barsce, and E. C. Martínez, 2018: *Generating rescheduling knowledge using reinforcement learning in a cognitive architecture*.

Pan, E., P. Petsagkourakis, M. Mowbray, D. Zhang, and A. del Rio-Chanona, 2020: *Constrained model-free reinforcement learning for process optimization*.

Pan, E., P. Petsagkourakis, M. Mowbray, D. Zhang, and E. A. del Rio-Chanona, 2021: Constrained model-free reinforcement learning for process optimization. *Computers & Chemical Engineering*, **154**, 107462.

Pan, I., L. R. Mason, and O. K. Matar, 2022: Data-centric engineering: integrating simulation, machine learning and statistics. challenges and opportunities. *Chemical Engineering Science*, **249**, 117271.

Pantelides, C. C., 1994: Unified frameworks for optimal process planning and scheduling. *Proceedings on the second conference on foundations of computer aided operations*, 253–274.

Panwalkar, S. and C. Koulamas, 2019: The evolution of schematic representations of flow shop scheduling problems. *Journal of Scheduling*, **22**, no. 4, 379–391.

Papageorgiou, L., M. C. Georgiadis, E. N. Pistikopoulos, and V. Dua, 2007: *Supply-Chain Optimization, Part II*, volume 4. John Wiley & Sons.

Park, J., J. Chun, S. H. Kim, Y. Kim, and J. Park, 2021: Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. *International Journal of Production Research*, **59**, no. 11, 3360–3377.

Park, J., Y. Seo, J. Shin, H. Lee, P. Abbeel, and K. Lee, 2022: *Surf: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning*.
URL https://arxiv.org/abs/2203.10050

Park, J.-B., K.-S. Lee, J.-R. Shin, and K. Y. Lee, 2005a: A particle swarm optimization for economic dispatch with nonsmooth cost functions. *IEEE Transactions on Power systems*, **20**, no. 1, 34–42.

— 2005b: A particle swarm optimization for economic dispatch with nonsmooth cost functions. *IEEE Transactions on Power systems*, **20**, no. 1, 34–42, iSBN: 0885-8950 Publisher: IEEE.

Parzen, E., 1962: On estimation of a probability density function and mode. *The annals of mathematical statistics*, **33**, no. 3, 1065–1076.

Paul, S. K. and P. Chowdhury, 2020: A production recovery plan in manufacturing supply chains for a high-demand item during COVID-19. *International Journal of Physical Distribution & Logistics Management*, iSBN: 0960-0035 Publisher: Emerald Publishing Limited.

Paulson, J. A., E. A. Buehler, R. D. Braatz, and A. Mesbah, 2020: Stochastic model predictive control with joint chance constraints. *International Journal of Control*, **93**, no. 1, 126–139.

Paulson, J. A. and A. Mesbah, 2018: Nonlinear model predictive control with explicit backoffs for stochastic systems under arbitrary uncertainty. *IFAC-PapersOnLine*, **51**, no. 20, 523–534.

— 2020: Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction. *IEEE Control Systems Letters*, **4**, no. 3, 719–724.

Payne, A. and P. Frow, 2006: Customer relationship management: from strategy to implementation. *Journal of marketing management*, **22**, no. 1-2, 135–168.

Peidro, D., J. Mula, M. Jiménez, and M. del Mar Botella, 2010: A fuzzy linear programming based approach for tactical supply chain planning in an uncertainty environment. *European Journal of Operational Research*, **205**, no. 1, 65–80, iSBN: 0377-2217 Publisher: Elsevier.

Peng, B., Y. Mu, J. Duan, Y. Guan, S. E. Li, and J. Chen, 2021: *Separated proportional-integral lagrangian for chance constrained reinforcement learning*.

Peng, Z., Y. Zhang, Y. Feng, T. Zhang, Z. Wu, and H. Su, 2019: Deep reinforcement learning approach for capacitated supply chain optimization under demand uncertainty. *2019 Chinese Automation Congress (CAC)*, IEEE, 3512–3517.

Perez, H. D., J. M. Wassick, and I. E. Grossmann, 2022: A digital twin framework for online optimization of supply chain business processes. *Computers & Chemical Engineering*, 107972.

Perkins, T. J. and A. G. Barto, 2002: Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, **3**, no. Dec, 803–832.

Petsagkourakis, P. and F. Galvanin, 2020: *Safe model-based design of experiments using gaussian processes*.

Petsagkourakis, P., I. O. Sandoval, E. Bradford, F. Galvanin, D. Zhang, and E. A. del Rio-Chanona, 2020a: Chance constrained policy optimization for process control and optimization. *arXiv preprint arXiv:2008.00030*.

— 2022: Chance constrained policy optimization for process control and optimization. *Journal of Process Control*, **111**, 35–45.

Petsagkourakis, P., I. O. Sandoval, E. Bradford, D. Zhang, and d. E. A. Rio-Chanona, 2020b: Reinforcement learning for batch bioprocess optimization. *Comput. Chem. Eng.*, **133**, doi:10.1016/j.compchemeng.2019.106649.
URL https://doi.org/10.1016/j.compchemeng.2019.106649

Pfrommer, S., T. Gautam, A. Zhou, and S. Sojoudi, 2022: Safe reinforcement learning with chance-constrained model predictive control. *Learning for Dynamics and Control Conference*, PMLR, 291–303.

Pistikopoulos, E. N., A. Barbosa-Povoa, J. H. Lee, R. Misener, A. Mitsos, G. V. Reklaitis, V. Venkatasubramanian, F. You, and R. Gani, 2021: Process systems engineering–the generation next? *Computers & Chemical Engineering*, **147**, 107252.

Powell, K. M., D. Machalek, and T. Quah, 2020: Real-time optimization using reinforcement learning. *Computers & Chemical Engineering*, **143**, 107077.

Powell, W. B., 2021: From reinforcement learning to optimal control: A unified framework for sequential decisions. *Handbook of Reinforcement Learning and Control*, Springer, 29–74.

Puterman, M. L., 2014a: *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

— 2014b: *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Qian, H. and Y. Yu, 2021: Derivative-free reinforcement learning: a review. *Frontiers of Computer Science*, **15**, no. 6, 1–19, iSBN: 2095-2236 Publisher: Springer.

Qin, R. and J. Zhao, 2022: High-efficiency generative adversarial network model for chemical process fault diagnosis. *IFAC-PapersOnLine*, **55**, no. 7, 732–737.

Rafiei, M. and L. A. Ricardez-Sandoval, 2018: Stochastic back-off approach for integration of design and control under uncertainty. *Industrial & Engineering Chemistry Research*, **57**, no. 12, 4351–4365.

— 2020: Integration of design and control for industrial-scale applications under uncertainty: a trust region approach. *Computers & Chemical Engineering*, **141**, 107006.

Rajeswaran, A., I. Mordatch, and V. Kumar, 2020: *A game theoretic framework for model based reinforcement learning*.
URL https://arxiv.org/abs/2004.07804

Rashidinejad, P., B. Zhu, C. Ma, J. Jiao, and S. Russell, 2021: Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, **34**, 11702–11716.

Rasmussen, C. E., 2006: Gaussian processes for machine learning. MIT Press.

Rawlings, J. B., D. Q. Mayne, and M. Diehl, 2017: *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI.

Reed, P., 2020: George e. davis (1850–1907): Transition from consultant chemist to consultant chemical engineer in a period of economic pressure. *ambix*, **67**, no. 3, 252–270.

Remko, V. H., 2020: Research opportunities for a more resilient post-COVID-19 supply chain–closing the gap between research findings and industry practice. *International Journal of Operations & Production Management*, **40**, no. 4, 341–355, iSBN: 0144-3577 Publisher: Emerald Publishing Limited.

Riedmiller, M., J. Peters, and S. Schaal, 2007a: Evaluation of policy gradient methods and variants on the cart-pole benchmark. *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, IEEE, 254–261.

— 2007b: Evaluation of policy gradient methods and variants on the cart-pole benchmark. *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, IEEE, 254–261.

Riedmiller, S. and M. Riedmiller, 1999: A neural reinforcement learning approach to learn local dispatching policies in production scheduling. *IJCAI*, Citeseer, volume 2, 764–771.

Rippin, D., 1993: Batch process systems engineering: a retrospective and prospective review. *Computers & chemical engineering*, **17**, S1–S13.

Rockafellar, R. and S. Uryasev, 2002: Conditional value-at-risk for general deviation measure. *J. Banking Financ*, **26**, 1443–1471.

Rockafellar, R. T. and S. Uryasev, 2000: Optimization of conditional value-at-risk. *Journal of risk*, **2**, 21–42, iSBN: 1465-1211 Publisher: Citeseer.

Rockafellar, R. T., S. Uryasev, et al., 2000: Optimization of conditional value-at-risk. *Journal of risk*, **2**, 21–42.

Rodriguez, J. D., A. Perez, and J. A. Lozano, 2009: Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE transactions on pattern analysis and machine intelligence*, **32**, no. 3, 569–575.

Rohani, S., 2017: *Coulson and Richardson's chemical engineering. Volume 3B, Process control*. Fourth edition.nd ed., Butterworth-Heinemann, Kidlington, Oxford, container-title: Coulson and Richardson's chemical engineering. Volume 3B, Process control.

Ronen, B. and R. Karp, 1994: An information entropy approach to the small-lot concept. *IEEE Transactions on Engineering Management*, **41**, no. 1, 89–92.

Rowland, M., R. Dadashi, S. Kumar, R. Munos, M. G. Bellemare, and W. Dabney, 2019: Statistics and samples in distributional reinforcement learning. *International Conference on Machine Learning*, PMLR, 5528–5536.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., 2015: Imagenet large scale visual recognition challenge. *International journal of computer vision*, **115**, no. 3, 211–252.

Ruszczyński, A. and A. Shapiro, 2003: Stochastic programming models. *Handbooks in operations research and management science*, **10**, 1–64.

Ryu, M., Y. Chow, R. Anderson, C. Tjandraatmadja, and C. Boutilier, 2019: Caql: Continuous action q-learning. *arXiv preprint arXiv:1909.12397*.

Sachio, S., M. Mowbray, M. M. Papathanasiou, E. A. del Rio-Chanona, and P. Petsagkourakis, 2022: Integrating process design and control using reinforcement learning. *Chemical Engineering Research and Design*, **183**, 160–169.

Sadeghian, A., N. M. Jan, O. Wu, and B. Huang, 2022: Robust probabilistic principal component regression with switching mixture gaussian noise for soft sensing. *Chemometrics and Intelligent Laboratory Systems*, **222**, 104491.

Saenz de Ugarte, B., A. Artiba, and R. Pellerin, 2009: Manufacturing execution system–a literature review. *Production planning and control*, **20**, no. 6, 525–539.

Salehkaleybar, S., S. Khorasani, N. Kiyavash, N. He, and P. Thiran, 2022: *Momentum-based policy gradient with second-order information*.
URL https://arxiv.org/abs/2205.08253

Salimans, T., J. Ho, X. Chen, S. Sidor, and I. Sutskever, 2017a: Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

— 2017b: Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Sammut, C., S. Hurst, D. Kedzier, and D. Michie, 1992: Learning to fly. *Machine Learning Proceedings 1992*, Elsevier, 385–393.

Sand, G. and S. Engell, 2004: Modeling and solving real-time scheduling problems by stochastic integer programming. *Computers & chemical engineering*, **28**, no. 6-7, 1087–1103.

Santos, F., R. Fukasawa, and L. Ricardez-Sandoval, 2021a: An integrated machine scheduling and personnel allocation problem for large-scale industrial facilities using a rolling horizon framework. *Optimization and Engineering*, **22**, no. 4, 2603–2626.

— 2021b: An integrated machine scheduling and personnel allocation problem for large-scale industrial facilities using a rolling horizon framework. *Optimization and Engineering*, **22**, no. 4, 2603–2626, iSBN: 1573-2924 Publisher: Springer.

Sarin, S. C., H. D. Sherali, and L. Liao, 2014: Minimizing conditional-value-at-risk for stochastic scheduling problems. *Journal of Scheduling*, **17**, no. 1, 5–15.

Sarkis, M., A. Bernardi, N. Shah, and M. M. Papathanasiou, 2021: Decision support tools for next-generation vaccines and advanced therapy medicinal products: present and future. *Current Opinion in Chemical Engineering*, **32**, 100689.

Schulman, J., X. Chen, and P. Abbeel, 2018a: Equivalence between policy gradients and soft q-learning. *arXiv:1704.06440 [cs]*, arXiv: 1704.06440.
URL http://arxiv.org/abs/1704.06440

Schulman, J., S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, 2017a: *Trust region policy optimization*.

Schulman, J., P. Moritz, S. Levine, M. Jordan, and P. Abbeel, 2018b: *High-dimensional continuous control using generalized advantage estimation*.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, 2017b: Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Schwefel, H.-P. and G. Rudolph, 1995: Contemporary evolution strategies. *European conference on artificial life*, Springer, 891–907.

Schweidtmann, A. M., E. Esche, A. Fischer, M. Kloft, J.-U. Repke, S. Sager, and A. Mitsos, 2021: Machine learning in chemical engineering: a perspective. *Chemie Ingenieur Technik*, **93**, no. 12, 2029–2039.

Scokaert, P. O. and J. B. Rawlings, 1999: Feasibility issues in linear model predictive control. *AIChE Journal*, **45**, no. 8, 1649–1659.

Šeda, M., 2007: Mathematical models of flow shop and job shop scheduling problems. *International Journal of Physical and Mathematical Sciences*, **1**, no. 7, 307–312.

See, J., S. Jamaian, R. Salleh, M. Nor, and F. Aman, 2018: Parameter estimation of monod model by the least-squares method for microalgae botryococcus braunii sp. *Journal of Physics: Conference Series*, IOP Publishing, volume 995, 012026.

Shady, S., T. Kaihara, N. Fujii, and D. Kokuryo, 2020: Automatic design of dispatching rules with genetic programming for dynamic job shop scheduling. *IFIP International Conference on Advances in Production Management Systems*, Springer, 399–407.

Shah, N., C. Pantelides, and R. Sargent, 1993: A general algorithm for short-term scheduling of batch operations—ii. computational issues. *Computers & chemical engineering*, **17**, no. 2, 229–244.

Shang, C. and F. You, 2019: A data-driven robust optimization approach to scenario-based stochastic model predictive control. *Journal of Process Control*, **75**, 24–39.

Shapiro, A., D. Dentcheva, and A. Ruszczynski, 2021: *Lectures on stochastic programming: modeling and theory*. SIAM.

Shapiro, A. and A. Philpott, 2007: A tutorial on stochastic programming. *Manuscript. Available at www2. isye. gatech. edu/ashapiro/publications. html*, **17**.

Sharma, N. and Y. Liu, 2022: A hybrid science-guided machine learning approach for modeling chemical processes: A review. *AIChE Journal*, **68**, no. 5, e17609.

Shehab, E., M. Sharp, L. Supramaniam, and T. A. Spedding, 2004: Enterprise resource planning: An integrative review. *Business process management journal*.

Shim, J. and J. M. Lee, 2022: Synthesis of distillation sequence with thermally coupled configurations using reinforcement learning. *Computer Aided Chemical Engineering*, Elsevier, volume 49, 169–174.

Shin, J., T. A. Badgwell, K.-H. Liu, and J. H. Lee, 2019: Reinforcement learning–overview of recent progress and implications for process control. *Computers & Chemical Engineering*, **127**, 282–294.

Silver, D., 2009: *Reinforcement Learning and Simulation-Based Search in Computer Go*. Ph.D. thesis, Dept. of Comp. Sci., Univ. of Alberta, Edmonton, AB, Canada.

Silver, D., J. A. Bagnell, and A. Stentz, 2010: Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, **29**, no. 12, 1565–1592, publisher-place: London, England publisher: SAGE Publications.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, and A. Bolton, 2017a: Mastering the game of go without human knowledge. *nature*, **550**, no. 7676, 354–359, iSBN: 1476-4687 Publisher: Nature Publishing Group.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., 2017b: Mastering the game of go without human knowledge. *nature*, **550**, no. 7676, 354–359.

Simmons-Edler, R., B. Eisner, E. Mitchell, S. Seung, and D. Lee, 2019: Q-learning for continuous actions with cross-entropy guided policies. *arXiv:1903.10605 [cs]*, arXiv: 1903.10605.
URL http://arxiv.org/abs/1903.10605

Singer, M., 2001: Decomposition methods for large job shops. *Computers & Operations Research*, **28**, no. 3, 193–207.

Singh, V. and H. Kodamana, 2020: Reinforcement learning based control of batch polymerisation processes. *IFAC-PapersOnLine*, **53**, no. 1, 667–672.

Sniedovich, M., 1978: Dynamic programming and principles of optimality. *Journal of Mathematical Analysis and Applications*, **65**, no. 3, 586–606.

Sobol', I. M., 1967: On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, **7**, no. 4, 784–802.

Sörensen, K., M. Sevaux, and F. Glover, 2018: A history of metaheuristics. *Handbook of heuristics*, Springer, 791–808.

Spielberg, S., A. Tulsyan, N. P. Lawrence, P. D. Loewen, and R. Bhushan Gopaluni, 2019: Toward self-driving processes: A deep reinforcement learning approach to control. *AIChE Journal*, **65**, no. 10, e16689.

Sternberg, W. and M. P. Deisenroth, 2017: Identification of gaussian process state-space models.

Steurtewagen, B. and D. Van den Poel, 2021: Adding interpretability to predictive maintenance by machine learning on sensor data. *Computers & Chemical Engineering*, **152**, 107381.

Stops, L., R. Leenhouts, Q. Gao, and A. M. Schweidtmann, 2022: Flowsheet synthesis through hierarchical reinforcement learning and graph neural networks. *arXiv preprint arXiv:2207.12051*.

Stork, M., R. Tousain, J. Wieringa, and O. H. Bosgra, 2003: A milp approach to the optimization of the operation procedure of a fed-batch emulsification process in a stirred vessel. *Computers & chemical engineering*, **27**, no. 11, 1681–1691.

Strassen, V., 1969: Gaussian elimination is not optimal. *Numerische mathematik*, **13**, no. 4, 354–356.

Stuart, A. M., 2010: Inverse problems: a bayesian perspective. *Acta numerica*, **19**, 451–559.

Stützle, T. et al., 1998: An ant approach to the flow shop problem. *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, Verlag Mainz, Wissenschaftsverlag Aachen, volume 3, 1560–1564.

Su, H. T., T. J. McAvoy, and P. Werbos, 1992: Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial & engineering chemistry research*, **31**, no. 5, 1338–1352, iSBN: 0888-5885 Publisher: ACS Publications.

Subramanian, K., C. T. Maravelias, and J. B. Rawlings, 2012: A state-space model for chemical production scheduling. *Computers & chemical engineering*, **47**, 97–110.

Subramanian, S., S. Lucia, R. Paulen, and S. Engell, 2021: Tube-enhanced multi-stage model predictive control for flexible robust control of constrained linear systems with additive and parametric uncertainties. *International Journal of Robust and Nonlinear Control*, doi:10.1002/rnc.5486.
URL http://dx.doi.org/10.1002/rnc.5486

Such, F. P., V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, 2017: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*.

Sun, Q. and Z. Ge, 2021: A survey on deep learning for data-driven soft sensors. *IEEE Transactions on Industrial Informatics*, **17**, no. 9, 5853–5866.

Sundaramoorthy, A. and C. T. Maravelias, 2008: Simultaneous batching and scheduling in multistage multiproduct processes. *Industrial & engineering chemistry research*, **47**, no. 5, 1546–1555.

Sundaramoorthy, A., C. T. Maravelias, and P. Prasad, 2009: Scheduling of multistage batch processes under utility constraints. *Industrial & engineering chemistry research*, **48**, no. 13, 6050–6058.

Sundström, O., D. Ambühl, and L. Guzzella, 2010: On implementation of dynamic programming for optimal control problems with final state constraints. *Oil & Gas Science and Technology–Revue de l'Institut Français du Pétrole*, **65**, no. 1, 91–102.

Suresh, V. and D. Chaudhuri, 1993: Dynamic scheduling—a survey of research. *International journal of production economics*, **32**, no. 1, 53–63.

Sutton, R., D. McAllester, S. Singh, and Y. Mansour, 2000: Policy gradient methods for reinforcement learning with function approximation. Neural information processing systems foundation, 1057–1063, iSSN: 10495258.

Sutton, R. S. and A. G. Barto, 2018a: *Reinforcement learning: An introduction*. MIT press.

— 2018b: *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., D. A. McAllester, S. P. Singh, Y. Mansour, et al., 1999: Policy gradient methods for reinforcement learning with function approximation. *NIPs*, Citeseer, volume 99, 1057–1063.

Swanson, A., 2020: Global trade sputters, leaving too much here, too little there. *The New York Times*, **10**.

Sweeney, K. D., D. C. Sweeney II, and J. F. Campbell, 2019: The performance of priority dispatching rules in a complex job shop: A study on the upper mississippi river. *International Journal of Production Economics*, **216**, 154–172.

Tang, L., S. Jiang, and J. Liu, 2010: Rolling horizon approach for dynamic parallel machine scheduling problem with release times. *Industrial & engineering chemistry research*, **49**, no. 1, 381–389.

Tang, Y. C., J. Zhang, and R. Salakhutdinov, 2019a: *Worst cases policy gradients*.

— 2019b: Worst cases policy gradients. *arXiv preprint arXiv:1911.03618*.

Tanyimboh, T. and C. Sheahan, 2002: A maximum entropy based approach to the layout optimization of water distribution systems. *Civil Engineering and Environmental Systems*, **19**, no. 3, 223–253.

Teh, Y. W., M. Welling, S. Osindero, and G. E. Hinton, 2003: Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, **4**, no. Dec, 1235–1260.

Thangavel, S., R. Paulen, and S. Engell, 2020: Robust multi-stage nonlinear model predictive control using sigma points. *Processes*, **8**, no. 7, 851.

Thebelt, A., J. Wiebe, J. Kronqvist, C. Tsay, and R. Misener, 2022: Maximizing information from chemical engineering data sets: applications to machine learning. *Chemical Engineering Science*, **252**, 117469.

Thomas, P. and D. Nicholas, 2018: The fourth industrial revolution: Shaping new era. *Journal of International Affairs*, **72**, no. 1, 17–22.

Tian, Y. and E. N. Pistikopoulos, 2018: Synthesis of operable process intensification systems—steady-state design with safety and operability considerations. *Industrial & Engineering Chemistry Research*, **58**, no. 15, 6049–6068.

Tjoa, I. B. and L. T. Biegler, 1991: Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial & Engineering Chemistry Research*, **30**, no. 2, 376–385.

Todaro, C. C. and H. C. Vogel, 2014: *Fermentation and biochemical engineering handbook*. William Andrew.

Tran, T. H., L. M. Nguyen, and K. Scheinberg, 2022: Finding optimal policy for queueing models: New parameterization. *arXiv preprint arXiv:2206.10073*.

Trentesaux, D., 2009: Distributed control of production systems. *Engineering Applications of Artificial Intelligence*, **22**, no. 7, 971–978.

Tsay, C., J. Kronqvist, A. Thebelt, and R. Misener, 2021: Partition-based formulations for mixed-integer optimization of trained relu neural networks. *Advances in Neural Information Processing Systems*, **34**, 3068–3080.

Tsay, C., A. Kumar, J. Flores-Cerrillo, and M. Baldea, 2019: Optimal demand response scheduling of an industrial air separation unit using data-driven dynamic models. *Computers & Chemical Engineering*, **126**, 22–34, iSBN: 0098-1354 Publisher: Elsevier.

Tsitsiklis, J. N., 1994: Asynchronous stochastic approximation and q-learning. *Machine learning*, **16**, no. 3, 185–202.

Turan, E. M. and J. Jäschke, 2021: Multiple shooting for training neural differential equations on time series. *IEEE Control Systems Letters*, **6**, 1897–1902.

Umlauft, J., T. Beckers, and S. Hirche, 2018: Scenario-based optimal control for gaussian process state space models. *2018 European Control Conference (ECC)*, IEEE, 1386–1392.

Uraikul, V., C. W. Chan, and P. Tontiwachwuthikul, 2007: Artificial intelligence for monitoring and supervisory control of process systems. *Engineering applications of artificial intelligence*, **20**, no. 2, 115–131.

Urpí, N. A., S. Curi, and A. Krause, 2021: Risk-averse offline reinforcement learning. *arXiv preprint arXiv:2102.05371*.

Valdez-Navarro, Y. I. and L. A. Ricardez-Sandoval, 2019: A novel back-off algorithm for integration of scheduling and control of batch processes under uncertainty. *Industrial & Engineering Chemistry Research*, **58**, no. 48, 22064–22083.

van de Berg, D., T. Savage, P. Petsagkourakis, D. Zhang, N. Shah, and E. A. del Rio-Chanona, 2022: Data-driven optimization for process systems engineering applications. *Chemical Engineering Science*, **248**, 117135.

Van Hasselt, H., A. Guez, and D. Silver, 2016: Deep reinforcement learning with double q-learning. *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Van Houdenhoven, M., J. M. Van Oostrum, E. W. Hans, G. Wullink, and G. Kazemier, 2007: Improving operating room efficiency by applying bin-packing and portfolio techniques to surgical case scheduling. *Anesthesia & Analgesia*, **105**, no. 3, 707–714.

Van Overschee, P. and B. De Moor, 1993: Subspace algorithms for the stochastic identification problem. *Automatica*, **29**, no. 3, 649–660.

Van Seijen, H., H. Van Hasselt, S. Whiteson, and M. Wiering, 2009: A theoretical and empirical analysis of expected sarsa. *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, IEEE, 177–184.

Vanderbei, R. J. et al., 2020: *Linear programming*. Springer.

Vela, C. R., S. Afsar, J. J. Palacios, I. Gonzalez-Rodriguez, and J. Puente, 2020: Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling. *Computers & Operations Research*, **119**, 104931.

Velez, S. and C. T. Maravelias, 2013: Multiple and nonuniform time grids in discrete-time mip models for chemical production scheduling. *Computers & Chemical Engineering*, **53**, 70–85.

Verderame, P. M., J. A. Elia, J. Li, and C. A. Floudas, 2010: Planning and scheduling under uncertainty: a review across multiple sectors. *Industrial & engineering chemistry research*, **49**, no. 9, 3993–4017.

Verhaegen, M., 2015: Subspace techniques in system identification. *Encyclopedia of Systems and Control*, Springer, 1386–1396.

Vijayakumar, B., P. J. Parikh, R. Scott, A. Barnes, and J. Gallimore, 2013: A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research*, **224**, no. 3, 583–591.

Von Stosch, M., R. Oliveira, J. Peres, and S. F. de Azevedo, 2014: Hybrid semi-parametric modeling in process systems engineering: Past, present and future. *Computers & Chemical Engineering*, **60**, 86–101.

Wabersich, K. P. and M. N. Zeilinger, 2021: *A predictive safety filter for learning-based control of constrained nonlinear dynamical systems*.

Wächter, A. and L. T. Biegler, 2006: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, **106**, no. 1, 25–57.

Wang, L., Z. Pan, and J. Wang, 2021: A review of reinforcement learning based intelligent optimization for manufacturing scheduling. *Complex System Modeling and Simulation*, **1**, no. 4, 257–270.

Wang, X. and S. M. Disney, 2016: The bullwhip effect: Progress, trends and directions. *European Journal of Operational Research*, **250**, no. 3, 691–701.

Wang, Y., D. Zhao, D. Rodríguez-Padrón, and C. Len, 2019a: Recent advances in catalytic hydrogenation of furfural. *Catalysts*, **9**, no. 10, 796.

Wang, Z. and C. Georgakis, 2019: Identification of hammerstein-weiner models for nonlinear mpc from infrequent measurements in batch processes. *Journal of Process Control*, **82**, 58–69.

Wang, Z., H.-X. Li, and C. Chen, 2019b: Incremental reinforcement learning in continuous spaces via policy relaxation and importance weighting. *IEEE transactions on neural networks and learning systems*, **31**, no. 6, 1870–1883.

Waschneck, B., A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek, 2018: Optimization of global production scheduling with deep reinforcement learning. *Procedia Cirp*, **72**, 1264–1269.

Watkins, C. J. C. H., 1989: Learning from delayed rewards.

Waubert de Puiseau, C., R. Meyes, and T. Meisen, 2022: On reliability of reinforcement learning based production scheduling systems: a comparative survey. *Journal of Intelligent Manufacturing*, 1–17.

Wei, L., R. McCloy, and J. Bao, 2022: Discrete-time contraction-based control of nonlinear systems with parametric uncertainties using neural networks. *Computers & Chemical Engineering*, 107962.

Wen, M. and U. Topcu, 2018: Constrained cross-entropy method for safe reinforcement learning. *Advances in Neural Information Processing Systems*, **31**.

Whitelam, S. and I. Tamblyn, 2020: Learning to grow: Control of material self-assembly using evolutionary reinforcement learning. *Physical Review E*, **101**, no. 5, 052604.

Wieland, F.-G., A. L. Hauber, M. Rosenblatt, C. Tönsing, and J. Timmer, 2021: On structural and practical identifiability. *Current Opinion in Systems Biology*, **25**, 60–69.

Wierstra, D., T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, 2014: Natural evolution strategies. *The Journal of Machine Learning Research*, **15**, no. 1, 949–980.

Wilcox, R. R., 2011: *Introduction to robust estimation and hypothesis testing*. Academic press.

Williams, C. K. and C. E. Rasmussen, 2006: *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Wills, A., T. B. Schön, L. Ljung, and B. Ninness, 2013: Identification of hammerstein–wiener models. *Automatica*, **49**, no. 1, 70–81.

Wolf, I. J. and W. Marquardt, 2016: Fast nmpc schemes for regulatory and economic nmpc–a review. *Journal of Process Control*, **44**, 162–183.

Wu, Z., L. Sun, W. Zhan, C. Yang, and M. Tomizuka, 2020: Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. *IEEE Robotics and Automation Letters*, **5**, no. 4, 5355–5362, doi:10.1109/LRA.2020.3005126.

Wulfmeier, M., P. Ondruska, and I. Posner, 2016: Maximum entropy deep inverse reinforcement learning. *arXiv:1507.04888 [cs]*, arXiv: 1507.04888.
URL http://arxiv.org/abs/1507.04888

Yan, Z. and J. Wang, 2012: Model predictive control of nonlinear systems with unmodeled dynamics based on feedforward and recurrent neural networks. *IEEE Transactions on Industrial Informatics*, **8**, no. 4, 746–756.

Yang, D., L. Zhao, Z. Lin, T. Qin, J. Bian, and T.-Y. Liu, 2019: Fully parameterized quantile function for distributional reinforcement learning. *Advances in neural information processing systems*, **32**.

Yang, J.-H. and H.-F. Lam, 2019: An innovative bayesian system identification method using autoregressive model. *Mechanical Systems and Signal Processing*, **133**, 106289.

Yang, S.-B., Z. Li, and J. Moreira, 2022: A recurrent neural network-based approach for joint chance constrained stochastic optimal control. *Journal of Process Control*, **116**, 209–220.

Yang, Z. R. and S. Chen, 1998: Robust maximum likelihood training of heteroscedastic probabilistic neural networks. *Neural Networks*, **11**, no. 4, 739–747.

Ye, Y., J. Li, Z. Li, Q. Tang, X. Xiao, and C. A. Floudas, 2014: Robust optimization and stochastic programming approaches for medium-term production scheduling of a large-scale steelmaking continuous casting process under demand uncertainty. *Computers & Chemical Engineering*, **66**, 165–185.

Yoo, H., H. E. Byun, D. Han, and J. H. Lee, 2021a: Reinforcement learning for batch process control: Review and perspectives. *Annual Reviews in Control*, **52**, 108–119.

Yoo, H., B. Kim, J. W. Kim, and J. H. Lee, 2021b: Reinforcement learning based optimal control of batch processes using monte-carlo deep deterministic policy gradient with phase segmentation. *Computers & Chemical Engineering*, **144**, 107133.

Yoo, H., V. M. Zavala, and J. H. Lee, 2021c: *A dynamic penalty function approach for constraints-handling in reinforcement learning*.

Yska, D., Y. Mei, and M. Zhang, 2018: Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. *European Conference on Genetic Programming*, Springer, 306–321.

Yu, C., J. Liu, S. Nemati, and G. Yin, 2021a: Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, **55**, no. 1, 1–36.

Yu, J. and P. Guo, 2020: Run-to-run control of chemical mechanical polishing process based on deep reinforcement learning. *IEEE Transactions on Semiconductor Manufacturing*, **33**, no. 3, 454–465.

Yu, L., T. Yu, C. Finn, and S. Ermon, 2019: Meta-inverse reinforcement learning with probabilistic context variables. *Advances in neural information processing systems*, **32**.

Yu, T., A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, 2021b: *Combo: Conservative offline model-based policy optimization*.

Yu, T., G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma, 2020: *Mopo: Model-based offline policy optimization*.

Yu, Z. J. and L. T. Biegler, 2019: Advanced-step multistage nonlinear model predictive control: Robustness and stability. *Journal of Process Control*, **84**, 192–206.

Zanon, M. and S. Gros, 2020: Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control*.

Zanon, M., V. Kungurtsev, and S. Gros, 2020: Reinforcement learning based on real-time iteration nmpc. *arXiv:2005.05225 [cs, eess]*, arXiv: 2005.05225.
URL http://arxiv.org/abs/2005.05225

Zeilinger, M. N., D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones, 2014: On real-time robust model predictive control. *Automatica*, **50**, no. 3, 683–694.

Zhang, C., W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi, 2020a: Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems*, **33**, 1621–1632.

Zhang, D., E. A. Del Rio-Chanona, P. Petsagkourakis, and J. Wagner, 2019a: Hybrid physics-based and data-driven modeling for bioprocess online simulation and optimization. *Biotechnology and bioengineering*, **116**, no. 11, 2919–2930.

Zhang, D., T. R. Savage, and B. A. Cho, 2020b: Combining model structure identification and hybrid modelling for photo-production process predictive simulation and optimisation. *Biotechnology and Bioengineering*, **117**, no. 11, 3356–3367.

Zhang, J., 2004: A reliable neural network model based optimal control strategy for a batch polymerization reactor. *Industrial & Engineering Chemistry Research*, **43**, no. 4, 1030–1038.

— 2008: Batch-to-batch optimal control of a batch polymerisation process based on stacked neural network models. *Chemical Engineering Science*, **63**, no. 5, 1273–1281.

Zhang, J., A. S. Bedi, M. Wang, and A. Koppel, 2020c: Cautious reinforcement learning via distributional risk in the dual domain. *arXiv preprint arXiv:2002.12475*.

Zhang, K., A. Koppel, H. Zhu, and T. Başar, 2020d: Global convergence of policy gradient methods to (almost) locally optimal policies. *arXiv:1906.08383 [cs, eess, math, stat]*, arXiv: 1906.08383.
URL http://arxiv.org/abs/1906.08383

Zhang, M. S., M. A. Erdogdu, and A. Garg, 2021: *Convergence and optimality of policy gradient methods in weakly smooth settings*.
URL https://arxiv.org/abs/2111.00185

Zhang, Q., J. L. Cremer, I. E. Grossmann, A. Sundaramoorthy, and J. M. Pinto, 2016: Risk-based integrated production scheduling and electricity procurement for continuous power-intensive processes. *Computers & chemical engineering*, **86**, 90–105.

Zhang, R., S. Song, and C. Wu, 2019b: Robust scheduling of hot rolling production by local search enhanced ant colony optimization algorithm. *IEEE Transactions on Industrial Informatics*, **16**, no. 4, 2809–2819.

Zhang, S. and R. S. Sutton, 2017: A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*.

Zhang, Y., X. Jin, Y. Feng, and G. Rong, 2018: Data-driven robust optimization under correlated uncertainty: a case study of production scheduling in ethylene plant. *Computers & Chemical Engineering*, **109**, 48–67.

Zhao, S. and C. C. Mi, 2021: A two-stage real-time optimized ev battery cooling control based on hierarchical and iterative dynamic programming and mpc. *IEEE Transactions on Intelligent Transportation Systems*.

Zheng, Y., X. Wang, and Z. Wu, 2022: Machine learning modeling and predictive control of the batch crystallization process. *Industrial & Engineering Chemistry Research*, **61**, no. 16, 5578–5592.

Zhong, Z., E. A. del Rio-Chanona, and P. Petsagkourakis, 2021: Data-driven distributionally robust mpc using the wasserstein metric. *arXiv preprint arXiv:2105.08414*.

— 2022: Distributionally robust mpc for nonlinear systems. *IFAC-PapersOnLine*, **55**, no. 7, 606–613.

Zhou, Z., M. Bloem, and N. Bambos, 2018: Infinite time horizon maximum causal entropy inverse reinforcement learning. *IEEE Transactions on Automatic Control*, **63**, no. 9, 2787–2802, publisher: IEEE.

Zhu, C., R. Ni, Z. Xu, K. Kong, W. R. Huang, and T. Goldstein, 2021a: Gradinit: Learning to initialize neural networks for stable and efficient training. *Advances in Neural Information Processing Systems*, **34**, 16410–16422.

Zhu, W., R. Rendall, I. Castillo, Z. Wang, L. H. Chiang, P. Hayot, and J. A. Romagnoli, 2021b: Control of a polyol process using reinforcement learning. *IFAC-PapersOnLine*, **54**, no. 3, 498–503.

Ziebart, B. D., 2010: Modeling purposeful adaptive behavior with the principle of maximum causal entropy.

Ziebart, B. D., A. L. Maas, J. A. Bagnell, A. K. Dey, et al., 2008: Maximum entropy inverse reinforcement learning. *Aaai*, Chicago, IL, USA, volume 8, 1433–1438.

Ziemann, I., A. Tsiamis, H. Sandberg, and N. Matni, 2022: How are policy gradient methods affected by the limits of control? *arXiv preprint arXiv:2206.06863*.

Črepinšek, M., S.-H. Liu, and M. Mernik, 2013: Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, **45**, no. 3, 1–33, iSBN: 0360-0300 Publisher: ACM New York, NY, USA.

Appendix A

Appendices for Background and Literature Review

A.1 Derivation of the state value function

We wish to identify a policy, which minimises the sum of stage costs in expectation. This can be quantified through the Bellman equation (i.e. the state value function). The derivation of the state value function relies heavily on three key points: 1) that the cost of a policy is defined in terms of the sum of stage costs, 2) the system obeys the Markov property and 3) the expectation operator is linear (Billingsley, 2008):

$$\begin{aligned} V_\pi(\mathbf{x}_t) &= \mathbb{E}_\pi\big[G_t \mid X_t = \mathbf{x}_t\big] \\ &= \mathbb{E}_\pi\big[\varphi(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{x}_{t+1}) + \gamma G_{t+1} \mid X_t = \mathbf{x}_t\big] \\ &= \mathbb{E}_\pi\Big[\varphi(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{x}_{t+1}) \mid X_t = \mathbf{x}_t\Big] + \gamma \mathbb{E}_\pi\Big[\mathbb{E}_\pi\big[G_{t+1} \mid X_{t+1} = \mathbf{x}_{t+1}\big] \mid X_t = \mathbf{x}_t\Big] \\ &= \mathbb{E}_\pi\Big[\varphi(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{x}_{t+1}) + \gamma V_\pi(\mathbf{x}_{t+1}) \mid X_t = \mathbf{x}_t\Big] \end{aligned} \tag{A.1}$$

where $V_\pi : \mathcal{X} \to \mathbb{R}$. The expression detailed by Eq. A.1 is also known as the state value function, which can be interpreted as the expected "cost-to-go" from the current state, $\mathbf{x}_t$, under policy, π.

A.2 Dynamic programming: policy iteration and value iteration

Both value iteration and policy iteration firstly randomly initialize and store in memory an estimate of the state value function, $V^0_{\pi^0}(\mathbf{x}) \approx V_{\pi^0}(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$ associated with the random policy π^0. Then they alternate between two distinct phases known as policy evaluation and policy improvement, which are defined by recursive application of specific operators. Both operations leverage knowledge of the fact that both the conditional probability mass functions descriptive of the dynamics and policy, and cost function[1].

In the policy evaluation phase, the objective is to quantify the true expected cost to go, $V_{\pi^k}(\mathbf{x})$, of following the current policy, π^k, in a given state. This is achieved by applying an operator known as the Bellman operator, $\mathcal{T}_\pi(\cdot)$, to the current estimate of the value function for a number of iterations. For example, at the j^{th} iterate of policy evaluation, the Bellman operator is applied to the current estimate of the value function, under policy π^k via:

$$\begin{aligned}\mathcal{T}_\pi(V^j_{\pi^k}(\mathbf{x}_t)) &= V^{j+1}_{\pi^k}(\mathbf{x}_t) \\ &= \sum_{\mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t)} \pi^k(\mathbf{u}_t \mid \mathbf{x}_t) \sum_{\mathbf{x}_{t+1} \in \mathcal{X}} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)\big[\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \gamma V^j_{\pi^k}(\mathbf{x}_{t+1})\big], \\ &\qquad\qquad \forall \mathbf{x}_t \in \mathcal{X}\end{aligned} \tag{A.2}$$

where the probabilities, $\pi(\mathbf{u}|\mathbf{x})$ are appropriately updated if $\mathbb{U} \neq \hat{\mathbb{U}}(\mathbf{x}_t)$. Application of the operator identifies an updated estimate for $V_{\pi^k}(\mathbf{x})$. The motivation for the recurrent application of the Bellman operator is that it is technically a contraction mapping. This means that according to the contraction mapping theorem (Kolmogorov and Fomin, 1957; Conrad, 2014), for any two estimates of the value function, $V^{j+1}_{\pi^k}(\mathbf{x})$ and $V^j_{\pi^k}(\mathbf{x})$, under a given policy, π^k:

$$\|\mathcal{T}_\pi(\mathbf{V}^{j+1}_{\pi^k}) - \mathcal{T}_\pi(\mathbf{V}^j_{\pi^k})\|_\infty \leq \gamma \|\mathbf{V}^{j+1}_{\pi^k} - \mathbf{V}^j_{\pi^k}\|_\infty \tag{A.3}$$

where γ is the discount factor as previously defined; $\mathbf{V}$ is a vector representation of the state values for all states in the state set; and $\|\cdot\|_\infty$ is the l_∞ norm. This is

[1]Provided that these items are known, and both the state and control spaces are countable, we then have a system of $|\mathcal{X}|$ equations with $|\mathcal{X}|$ unknowns (i.e. the state values of each state in the state set). This forms a linear programming problem, which is also formalised in Appendix A.4.

a technical condition but it ensures iterative application of the Bellman operator, to an estimate of the state value function, converges to a fixed point (or equivalently the true state value function associated with the current policy), such that:

$$V_{\pi^k}^{j+1}(\mathbf{x}) = V_{\pi^k}^{j}(\mathbf{x}) = V_{\pi^k}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}$$

The second phase, known as policy improvement, then applies a slightly different operator known as the Bellman optimality operator, $\mathcal{T}_{\pi^*}(\cdot)$, which acts on the most recent estimate of the value function, $V_{\pi^k}^{j}$, descriptive of policy π^k. However, as one may gather from the name of the operation, instead of improving the estimate of the state value function under the policy, this operation aims to improve the current policy. This is achieved by choosing the controls, which greedily minimise the expected cost-to-go under the current estimate of the state value function:

$$\begin{aligned}\mathcal{T}_{\pi^*}(V_{\pi^k}^{j}(\mathbf{x}_t)) &= V_{\pi^{k+1}}^{0}(\mathbf{x}) \\ &= \min_{\mathbf{u}_t \in \mathbb{U}(\mathbf{x}_t)} \sum_{\mathbf{x}_{t+1} \in \mathcal{X}} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)\big[\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \gamma V_{\pi^k}^{j}(\mathbf{x}_{t+1})\big], \\ &\qquad\qquad \forall \mathbf{x}_t \in \mathcal{X}\end{aligned} \tag{A.4}$$

Greedily minimising the expected cost-to-go is appealing, because it is also an operation that satisfies the contraction mapping theorem. Specifically, for two state value functions V_{π^k} and $V_{\pi^{k+1}}$:

$$\|\mathcal{T}_{\pi^*}(\mathbf{V}_{\pi^{k+1}}) - \mathcal{T}_{\pi^*}(\mathbf{V}_{\pi^k})\|_\infty \leq \gamma \|\mathbf{V}_{\pi^{k+1}} - \mathbf{V}_{\pi^k}\|_\infty \tag{A.5}$$

This means that by applying the Bellman optimality operator to the current state value function, one can iteratively identify a state value function that satisfies the necessary and sufficient conditions provided by the Bellman optimality equation (i.e. Eq. 2.7a). This is guaranteed as $k \to \infty$.

The major differences between value iteration and policy iteration algorithms simply arises in the number of iterations of policy evaluation (or the number of times the Bellman operator, $\mathcal{T}_\pi(\cdot)$) is applied to all states before a policy improvement step is taken. In policy iteration the Bellman operator is applied until a convergence tolerance criterion is satisfied, whereas just one step is taken in value iteration (Sutton and Barto, 2018a). Clearly, this implies that one step of value iteration has a lower computational burden than policy iteration. Specifically, one step of value iteration

has a computational time complexity that is polynomial in the cardinality of the state set, $|\mathcal{X}|$, and provided that no state dependent control constraints are imposed, linear in the cardinality of the control set, $|\mathbb{U}|$. Additionally, both DP algorithms have memory requirements that scale linearly with $|\mathcal{X}|$. In practice this provides obstacle to the implementation of DP methods for most practical PSE decision-making problems. The value iteration algorithm is expressed by Algorithm A.1.

Algorithm A.1 The Value Iteration Algorithm

Input: $V_{\pi^0}(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$; tolerance criterion, δ; $\Delta = \infty$; $k = 0$;

while $\Delta > \delta$ **do**

$\Delta = 0$

for $\mathbf{x}_t \in \mathcal{X}$ **do**

$V_{old} = V_{\pi^k}(\mathbf{x}_t)$

$V_{\pi^{k+1}}(\mathbf{x}_t) = \min_{\mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t)} \sum_{\mathbf{x}_{t+1} \in \mathcal{X}} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) \big[\varphi(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \gamma V_{\pi^k}(\mathbf{x}_{t+1})\big]$

$\Delta = \max(\Delta, |V_{old} - V_{\pi^{k+1}}(\mathbf{x}_t)|)$

$k = k + 1$

end for

end while

Output: Optimal state value function, $V_{\pi^*}(\mathbf{x})$ and policy π^* that satisfies $V_{\pi^*}(\mathbf{x})$.

A.3 Markov Decision Processes

The MDP is a formal description of Eq. 2.8, but defined *without* state constraints (Chang et al., 2007). Instead, the problem of identifying an optimal policy for a finite horizon MDP is stated:

$$\mathcal{P}(\pi) := \begin{cases} \min\limits_{\pi} V_\pi(X_0 = \mathbf{x}_0) \\ \text{s.t.} \\ X_0 \sim p(\mathbf{x}_0) \\ X_{t+1} \sim p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) \\ U_t \sim \pi(\mathbf{u}_t \mid \mathbf{x}_t) \\ \mathbf{u}_t \in \hat{\mathbb{U}}(\mathbf{x}_t) \\ \mathbf{x}_t \in \mathcal{X} \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{A.6}$$

where, under the assumption that state and control sets are continuous, $X_0 \sim p(\mathbf{x}_0)$ describes a probability density function (pdf) over initial states, and the policy, π, is defined as stochastic and therefore a conditional probability density function (cpdf). This is a general way to think of closed-loop decision-making, given that a deterministic function can be recovered if all probability mass (or density) is placed on a single control. Stochastic policies may be directly parameterised in direct methods, or by choosing the control which minimises the state value function with some probability in indirect methods. We will see in the following sections, the use of stochastic policies is a key concept within both direct and indirect RL algorithms.

A.4 Linear programming formulations for determining the optimal state value function

Here we present formulation of a linear programming problem for determining the optimal state value function, given knowledge of the conditional probability density functions descriptive of discrete time dynamics, and under the conditions that states are countable and control sets continuous, and the cost function is linear in the state and control. The problem then reduces to finding the $|\mathcal{X}|$ state values, $\mathbf{V} = [V(\mathbf{x}_1), \ldots, V(\mathbf{x}_{|\mathcal{X}|})]$, given the initial state distribution $p(\mathbf{x})$.

$$\mathcal{P}(\mathbf{V}) := \begin{cases} \min\limits_{\mathbf{V}} \sum\limits_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) V_\pi(\mathbf{x}) \\ \text{s.t.} \\ V_\pi(\mathbf{x}) \geq \sum\limits_{\mathbf{x}'\in\mathcal{X}} p(\mathbf{x}'|\mathbf{x},\mathbf{u})[\varphi(\mathbf{x},\mathbf{u}) + \gamma V_\pi(\mathbf{x}')], \quad \forall \mathbf{u} \in \hat{\mathbb{U}}(\mathbf{x}),\ \forall \mathbf{x} \in \mathbb{X} \end{cases} \tag{A.7}$$

A.5 Maximum entropy optimisation

The principle of maximum entropy states that: given testable information, $\mathcal{D} = \{\mathbf{z}_n, \mathbf{r}(\mathbf{z}_n)\}_{n=1}^N$, where $\mathbf{r}(\mathbf{z}) = [r_1(\mathbf{z}), \ldots, r_{n_w}(\mathbf{z})]$, is a function of a discrete random variable[2], $Z \sim p_{gt}(\mathbf{z})$; the best approximating probability mass function (pmf), $p(\mathbf{z}) \approx$

[2]For notation purposes, the description presented assumes a discrete random variable, but the ideas also extend to continuous random variables.

$p_{gt}(\mathbf{z})$, is the one that has maximum information entropy, $H(Z)$, and satisfies the testable information. The information entropy of a random variable is defined as:

$$H(Z) = -\sum_{\mathbf{z}\in\mathbb{W}} p(\mathbf{z}) \log p(\mathbf{z})$$
$$H(Z) = \mathbb{E}_{p(\mathbf{z})}\big[I_Z(\mathbf{z})\big]$$

where $\mathbb{W} \in \mathbb{R}^{n_u}$ is the support of Z and $I_Z(\mathbf{z}) = -\log p(\mathbf{z})$ is the Shannon information, which can be thought as the information content of observing a realisation of the random variable[3]. Information entropy can therefore be thought of as the expected information gain associated with observing an additional realisation of the random variable. Hence, the approximating pmf, $p(\mathbf{z})$, with greatest information entropy can be thought as the least biased choice. The primal problem to identify $p(\mathbf{z})$ is provided by the following formulation:

$$\mathcal{P}(p(\mathbf{z})) := \begin{cases} \min\limits_{p(\mathbf{z}),\forall\mathbf{z}\in\mathbb{W}} -\mathbb{E}_{p(\mathbf{z})}\big[I_Z(\mathbf{z})\big] \\ \text{s.t.} \\ \dfrac{1}{N}\displaystyle\sum_{\mathbf{z}\in\mathcal{D}} r_i(\mathbf{z}) = \sum_{\mathbf{z}\in\mathbb{W}} p(\mathbf{z}) r_i(\mathbf{z}), \quad \forall i \in \{1,\dots,n_w\} \\ \displaystyle\sum_{\mathbf{z}\in\mathbb{W}} p(\mathbf{z}) = 1 \\ p(\mathbf{z}) \geq 0, \quad \forall \mathbf{z} \in \mathbb{W} \end{cases} \tag{A.8}$$

The first constraint restricts the distribution to satisfy testable information in expectation (i.e. it is a moment matching constraint), whereas the final two constraints enforce normalisation and non-negativity of the distribution. By forming the Lagrange function, $\mathcal{L}$ and setting $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}p(\mathbf{z})} = 0$, one recovers that the solution of Eq. A.8 has exponential form:

$$p(\mathbf{z}) = A(\lambda_1,\dots,\lambda_{n_w},\cdot)\exp\left(\sum_{i=1}^{n_w}\lambda_i r_i(\mathbf{z})\right), \quad \forall \mathbf{z}\in\mathbb{W} \tag{A.9}$$

where A is the partition function, which ensures normalisation of the distribution and $\lambda_i \in \mathbb{R},\ \forall i \in \{1,\dots,n_w\}$ are the Lagrange multipliers associated with the constraint

[3]The basic intuition follows from the range of a logarithm over the domain provided by the range of a valid probability mass function. As the probability of a realisation, $\mathbf{z}$, tends to zero, the Shannon information tends to infinity. Likewise as the probability of an event tends to one, the Shannon information tends to zero. Hence the less likely an event is, the more information it provides about the random variable.

to satisfy testable information. Solving the primal problem identifies the Lagrange multipliers, which provide parameters to the exponential distribution in Eq. A.9 with support, $\mathbb{W}$. Given that the primal problem is convex, the solution identified will be a global optimum. However, if the cardinality of the support, $|\mathbb{W}|$, is large or the support is continuous the number of constraints imposed on the primal problem becomes very large, to the point that the primal problem may become computational intractable to solve. Given that the problem is convex, one can instead exploit duality to solve the problem. The dual problem for Eq. A.8 is equivalent to maximising the log-likelihood under the exponential distribution, which provides the dual function to the primal formulation of Eq. A.8 (Ziebart, 2010):

$$\log p(\mathcal{D} \mid \lambda_1, \ldots, \lambda_{n_w}) = \sum_{\mathbf{z} \in \mathcal{D}} \log \left(\frac{1}{\sum_{\mathbf{z} \in \mathbb{W}} \exp\left(\sum_{i=1}^{n_w} \lambda_i r_i(\mathbf{z}) \right)} \exp\left(\sum_{i=1}^{n_w} \lambda_i r_i(\mathbf{z}) \right) \right) \tag{A.10}$$

The dual function can be obtained by substituting Eq. A.9 into the constraint enforcing normalisation of the distribution as detailed by Eq. A.8. This identifies that the partition function, $A = \frac{1}{\sum_{\mathbf{z} \in \mathbb{W}} \exp\left(\sum_{i=1}^{n_w} \lambda_i r_i(\mathbf{z}) \right)}$, and essentially removes the presence of Lagrange multipliers relating to normalisation and non-negativity from entering into the dual function. This means that the maximum-entropy log-likelihood function is simply a function of the Lagrange multipliers related to the constraint of testable information. However, it does require one to estimate the partition function. Hence, in the case that $n_w << |\mathbb{W}|$, solving the dual problem may be more computationally attractive than solving the primal problem. As a result, finding $\lambda_i, \forall i \in \{1, \ldots, n_w\}$ that maximise Eq. A.10 recovers the exponential distribution, which best approximates the random variable, $Z \sim p_{gt}(\mathbf{z})$, subject to information provided by $\mathcal{D}$. This is discussed further in Appendix B.3 within the context of sequential decision making.

Appendix B

Appendices for research item: Using process data to generate an optimal control policy via apprenticeship and reinforcement learning

B.1 The Policy Gradient Theorem

The value $G(\boldsymbol{\tau})$ of a process trajectory is equal to the discounted sum of rewards R_{t+1} accumulated along its path

$$G\left(\boldsymbol{\tau}\right) = \sum_{t=0}^{T-1} \gamma^t R_{t+1}$$

Given the stochasticity of the underlying process, we wish to find a policy $\pi^*(\cdot, \theta,$ which maximises the expected trajectory value, or the utility $J(\boldsymbol{\tau})$:

$$J\left(\boldsymbol{\tau}\right) = \int p\left(\boldsymbol{\tau}|\theta\right) \ \mathrm{G}\left(\boldsymbol{\tau}\right) \ d\boldsymbol{\tau}$$

In order to iteratively improve the policy, we wish to find the gradient of the utility with respect to the policy parameters θ, providing a direction for policy improvement. The gradient may be expressed analytically via the following, which leverages the use of a logarithmic identity:

$$
\begin{aligned}
\nabla_{\theta_{(j)}} J(\boldsymbol{\tau}) &= \nabla_{\theta} \int p(\boldsymbol{\tau}|\theta)\, G(\boldsymbol{\tau})\ d\boldsymbol{\tau} \\
&= \int \nabla_{\theta} p(\boldsymbol{\tau}|\theta)\ \mathrm{G}(\boldsymbol{\tau})\ d\boldsymbol{\tau} \\
&= \int p(\boldsymbol{\tau}|\theta)\, \frac{\nabla_{\theta} p(\boldsymbol{\tau}|\theta)}{p(\boldsymbol{\tau}|\theta)} G(\boldsymbol{\tau})\ d\boldsymbol{\tau} \\
&= \int p(\boldsymbol{\tau}|\theta)\, \nabla_{\theta} \log p(\boldsymbol{\tau}|\theta) G(\boldsymbol{\tau})\ d\boldsymbol{\tau} \\
&= \mathbb{E}_{\tau \sim p(\boldsymbol{\tau}|\theta)} \left[G(\boldsymbol{\tau})\, \nabla_{\theta} \log p(\boldsymbol{\tau}|\theta) \right]
\end{aligned}
$$

This gradient may be estimated via the sample average approximation. The parameters of the policy may then be improved by stochastic gradient ascent:

$$
\theta_{(j+1)} \ = \ \theta_{(j)} + \omega \nabla_{\theta_{(j)}} J(\boldsymbol{\tau})
$$

where $\omega \in \mathbb{R}_{+}$ is a learning rate or step size. This iterative procedure of policy improvement acts to make trajectories of high value more probable under the policy. Clearly, approximate second-order methods such as ADAM may also be used.

B.2 Long-short term memory (LSTM) policy networks

Figure B.1.A details the LSTM policy network, which was implemented in this work. Figure B.1.B details a simplified description of the mathematical operations internal to LSTM cells.

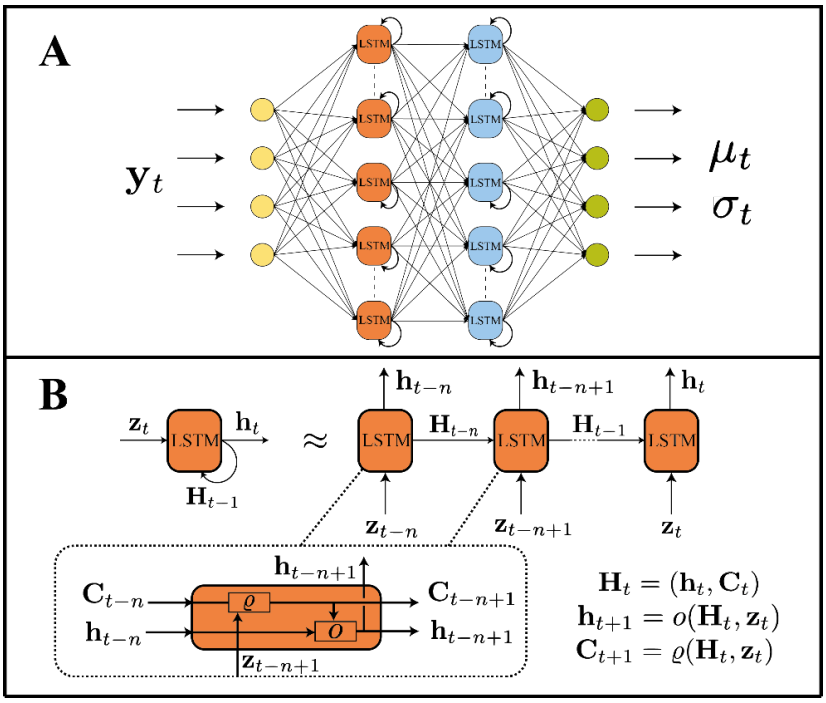


Figure B.1: **A**: Neural network parameterisation of the policy. Each of the nodes in the hidden layers is a self-contained recurrent LSTM cell. **B**: A general, simplified description of the mathematical operations internal to each LSTM cell. The internal representation $\mathbf{H}$ represents a parameterisation of previously observed states $\mathbf{y}$.

B.3 The principle of maximum entropy and maximum entropy Inverse Reinforcement Learning

In this work, we are concerned with recovering a control policy (a conditional probability density function), which induces equivalent observed trajectory features, $\boldsymbol{\upsilon}^{\pi} \in \mathbb{R}^d$, to that of an existing expert policy,$\boldsymbol{\upsilon}^{E} \in \mathbb{R}^d$. There is likely to exist a set of conditional probability density functions (or optimal control policies), $\Pi = \{\pi_1^*, \ldots, \pi_O^*\}$, which achieve this. Each control policy $\pi^* \in \Pi$ could correspond to the optimal control policy for a given MDP. All of the corresponding MDPs are defined by the same probabilistic process dynamics, $p\left(\mathbf{x}_{t+1}|\mathbf{x}_t,\ \mathbf{u}_t\right) p\left(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}, \mathbf{x}_t,\ \mathbf{u}_t\right)$, state and control spaces, but differing reward functions, $R\left(\cdot\right)$.

The question that arises therefore, is "which control policy or in which MDP is it optimal to learn"? The information theoretic answer to this question lies in the work provided by Jaynes (1957), where the Principle of Maximum Entropy (PoME) was first presented. The PoME says that the probability density function, which explains existing (testable) information about a target probability density function, and has

the greatest entropy, is optimal. This is a general framework, which has been applied to many different problem settings.

Following the PoME, the Max.Ent. IRL formulation, finds a reward function, that constructs a control policy $\pi(\mathbf{u}|\mathbf{y})$, which:

1. Induces a distribution over trajectories, $p(\boldsymbol{\tau})$.

2. Matches the trajectory features $\boldsymbol{\upsilon}$ corresponding to $p\left(\boldsymbol{\tau}\right)$ to those $(\boldsymbol{\upsilon}^{E})$ observed from empirical demonstrations. $\boldsymbol{T} = [\boldsymbol{\tau}_{1}^{E},\ \ldots,\ \boldsymbol{\tau}_{K}^{E}]$ of the target expert policy, π^{E}, in expectation.

3. Subject to maximising the entropy, $H = -\sum_{\boldsymbol{\tau}} p\left(\boldsymbol{\tau}\right) \log p\left(\boldsymbol{\tau}\right)$, of the distribution, $p(\boldsymbol{\tau})$, induced by the policy learned $\pi(\mathbf{u}|\mathbf{y})$.

In the following analysis, we highlight the relation of PoME to Section 3.3.4, by optimising over the inducing distribution $p\left(\boldsymbol{\tau}\right)$ of state trajectories. For more information we redirect the interested reader to the original works. In consideration of a linear reward function $R = \widehat{\alpha}^{T}\boldsymbol{\varphi}\left(\mathbf{y}\right)$, the undiscounted return from a given trajectory $\boldsymbol{\tau}$ is formalised $G\left(\boldsymbol{\tau}\right) = \sum_{t=1}^{T} R\left(\mathbf{y}_{t}\right)$. The primal problem expressed by the Max.Ent. IRL framework can then be written:

$$\begin{aligned} \min_{p(\boldsymbol{\tau})} \ & \sum_{\boldsymbol{\tau}} p\left(\boldsymbol{\tau}\right) \log p\left(\boldsymbol{\tau}\right) \\ \text{s.t} \quad & \sum_{\tau} p\left(\boldsymbol{\tau}\right) G(\boldsymbol{\tau}) = \mathbb{E}\left[G(\boldsymbol{\tau}^{E})\right] \\ & \sum_{\boldsymbol{\tau}} p\left(\boldsymbol{\tau}\right) = 1 \end{aligned}$$

Forming the Lagrangian function $\mathcal{L}\left(\boldsymbol{\alpha},\ p\left(\boldsymbol{\tau}\right)\right)$, one obtains:

$$\begin{aligned} \mathcal{L}\left(\alpha,\, p\left(\boldsymbol{\tau}\right)\right) = \ & \sum_{\boldsymbol{\tau}} p\left(\boldsymbol{\tau}\right) \log p\left(\boldsymbol{\tau}\right) \\ & + \lambda_{1}\left(\mathbb{E}\left[G(\boldsymbol{\tau}^{E})\right] - \sum_{\boldsymbol{\tau}} p\left(\boldsymbol{\tau}\right) G(\boldsymbol{\tau})\right) + \lambda_{2}\left(1 - \sum_{\boldsymbol{\tau}} p\left(\boldsymbol{\tau}\right)\right) \end{aligned}$$

where $\lambda_{1} \in \mathbb{R}$ and $\lambda_{2} \in \mathbb{R}$. Differentiating with respect to the induced distribution $p(\boldsymbol{\tau})$, and evaluating first order conditions, it follows that:

$$\frac{\mathrm{d}\mathcal{L}\left(\boldsymbol{\alpha},\ p\left(\boldsymbol{\tau}\right)\right)}{\mathrm{d}p(\boldsymbol{\tau})} = \ \log p\left(\boldsymbol{\tau}\right) + \ 1 - \lambda_{1} G(\boldsymbol{\tau}) - \lambda_{2}$$

when $\frac{\mathrm{d}\mathcal{L}(\alpha,\ p(\boldsymbol{\tau}))}{\mathrm{d}p(\boldsymbol{\tau})} = 0$,

$$\log p\left(\boldsymbol{\tau}\right) + 1 - \lambda_1 G(\boldsymbol{\tau}) - \lambda_2 = 0 \implies p\left(\boldsymbol{\tau}\right) = \exp\{\lambda_1 G(\boldsymbol{\tau}) + \lambda_2 - 1\}$$

$$\therefore p\left(\boldsymbol{\tau}\right) = \widehat{Z} \exp\{\lambda_1 G(\boldsymbol{\tau})\}$$

This result implies that the maximum entropy model of the induced distribution $p(\boldsymbol{\tau})$ belongs to the exponential family and that this distribution assigns exponentially greater probability mass on trajectories, which observe high cumulative rewards. To find an expression for the constant $\widehat{Z}$, we make use of the normalisation constraint enforced in the primal problem, such that if assumed constant for all $\boldsymbol{\tau}$:

$$\sum_{\tau} \widehat{Z} \exp\left(\lambda_1 G(\boldsymbol{\tau})\right) = 1$$
$$\widehat{Z} = \frac{1}{Z} = \frac{1}{\sum_{\boldsymbol{\tau}} \exp\left(\lambda_1 G(\boldsymbol{\tau})\right)}$$

where Z is the partition function. In practice, we do not know $G(\boldsymbol{\tau}^E)$ prior to conducting AL. However, we do possess knowledge of the trajectory features $\boldsymbol{\upsilon}^E$. Hence, the problem described in Section 3.3.4, can then be recovered by identifying that $\boldsymbol{\alpha} = \lambda_1 \widehat{\boldsymbol{\alpha}}$**,** and equating $\lambda_1 G\left(\boldsymbol{\tau}\right) = \boldsymbol{\alpha}^T \boldsymbol{\upsilon}$. Drawing attention back to the primal problem, it is clear that the introduction of discounted trajectory features introduces non-convexity into problem, as the constraint of testable information is no longer affine. As discussed in Section 3.3.4, the MaxEnt IRL problem then follows:

$$\max_{\boldsymbol{\alpha}} \prod_{k=1}^{K} p(\boldsymbol{\tau}^{\boldsymbol{E}}_{\boldsymbol{(k)}} \mid \boldsymbol{\alpha})$$
$$p(\boldsymbol{\tau} \mid \boldsymbol{\alpha}) = \frac{\exp\left(\boldsymbol{\alpha}^T \boldsymbol{\upsilon}(\boldsymbol{\tau})\right)}{Z}$$

where the weights of the linear reward function are analogous to the Lagrange multipliers of the original problem. In order to estimate the partition function Z, this work proposes to perform policy optimisation in the underlying MDP as constructed with linear reward function $R = \alpha^T \varphi(\mathbf{y})$**.** Theoretically, the policy should possess the property of maximum entropy. This can be achieved by performing policy optimisation under a reward function, which is augmented with an entropy incentive: $R^H = \alpha^T \varphi\left(\mathbf{y}\right) + \beta H_\pi$, where $H_\pi = -\sum_{\mathbf{u}} \pi\left(\mathbf{u}|\mathbf{x}\right) \log \pi\left(\mathbf{u}|\mathbf{x}\right)$ is the entropy of the

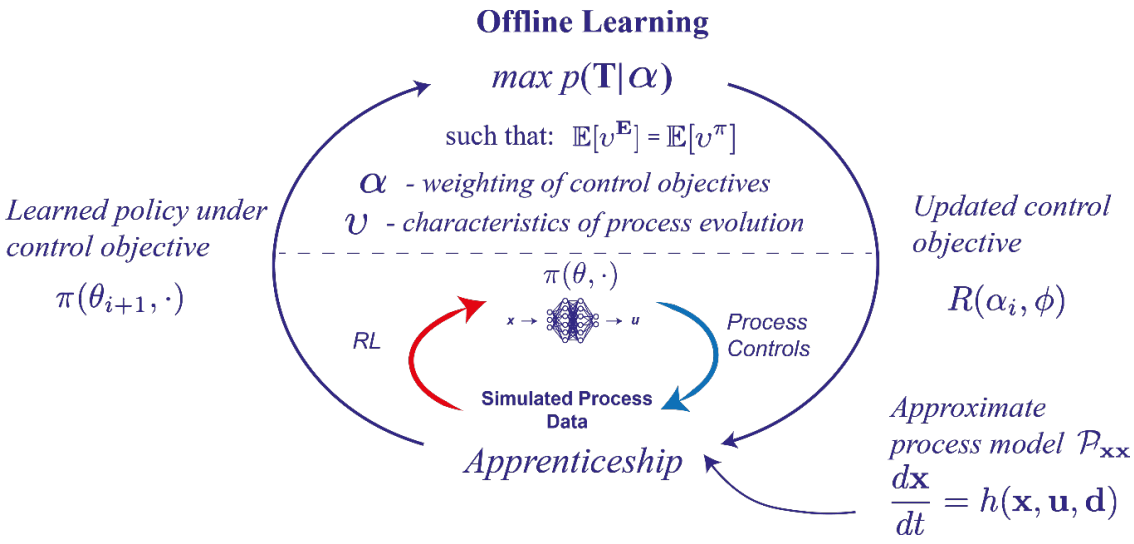


Figure B.2: Process flow diagram for offline learning as proposed in this work.

policy π at a given control interval, and $\beta \in \mathbb{R}_{\geq 0}$ is a non-negative real number. In practice, the parameter β is hand tuned and specific to the implementation, hence in the RL sense it is difficult to guarantee the property of maximum entropy absolutely. In this work, we found $\beta = 0$, recovered the desired behaviour, although performance may be improved by hand tuning β or drawing upon works such as soft actor critic (SAC). Having identified the optimal policy under the current iterate of $\boldsymbol{\alpha}^T$, gradient steps can be taken to maximise the log-likelihood function as follows:

$$
\begin{aligned}
\nabla_{\boldsymbol{\alpha}} \log p(\boldsymbol{\tau}^E \mid \boldsymbol{\alpha}) &= \boldsymbol{\upsilon}^E - \nabla_{\boldsymbol{\alpha}} \log Z(\boldsymbol{\alpha}) \\
\nabla_{\boldsymbol{\alpha}} \log Z(\boldsymbol{\alpha}) &= \frac{1}{Z(\boldsymbol{\alpha})} \nabla_{\boldsymbol{\alpha}} Z(\boldsymbol{\alpha}) \\
&= \frac{1}{Z(\boldsymbol{\alpha})} \sum_{\tau} \boldsymbol{\upsilon}(\boldsymbol{\tau}) \exp(\boldsymbol{\alpha}^T \boldsymbol{\upsilon}(\boldsymbol{\tau})) \\
&= \sum_{\tau} p(\boldsymbol{\tau} \mid \boldsymbol{\alpha}) \boldsymbol{\upsilon}(\boldsymbol{\tau}) \\
\therefore \nabla_{\boldsymbol{\alpha}} \log p(\boldsymbol{\tau} \mid \boldsymbol{\alpha}) &= \boldsymbol{\upsilon}^E - \mathbb{E}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau} \mid \alpha)}[\boldsymbol{\upsilon}]
\end{aligned}
$$

Figure B.2 summarises the general idea for AL (offline learning) provided by this work.

B.4 Policy characterisation

Algorithm B.1 Monte Carlo Policy Evaluation

Input: A policy $\pi(\mathbf{u}|\mathbf{y};\theta)$; state features $\boldsymbol{\varphi}(\mathbf{y}) = [\varphi_1(\mathbf{y}), \ldots, \varphi_d(\mathbf{y})]$; a number of episodes N; episode length T; a model of the process dynamics $\mathcal{M}(\cdot)$; an initial state distribution $\eta \sim p(\mathbf{x}_0)$; discount factor γ;

Output: Trajectory feature expectations $\mathbb{E}[\boldsymbol{\upsilon}] \in \mathbb{R}^d$

for $n = 1, \ldots, N$ **do**

1a. Initialise process state $\mathbf{x}_0$, observe $\mathbf{y}_0$ and store projection to state feature space $\boldsymbol{\varphi}^n \in \mathbb{R}^d$

for $t = 1, \ldots, T-1$ **do**

1b. Select action $\mathbf{u}_t \sim \pi(\mathbf{u}_t \mid \mathbf{y}_t; \theta)$

1c. Observe transition of process state $\mathbf{y}_{t+1}$ given dynamics $\mathcal{M}(\cdot)$

1d. Project process state $\mathbf{y}_{t+1}$ into state feature space $\boldsymbol{\varphi}^{(n)} \in \mathbb{R}^d$

end for

2. Calculate and store discounted sum of state features such that $\boldsymbol{\upsilon}^{(n),\gamma} = \sum_{t=0}^{T} \gamma^t \boldsymbol{\varphi}^{(n)}(\mathbf{y}_t)$

end for

3. Calculate trajectory feature expectations $\mathbb{E}[\boldsymbol{\upsilon}^{\gamma}] = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\upsilon}^{(n),\gamma}$

B.5 Approximate Process Model

Mass Balance

$$\text{Input} = M_R.V_A.C_{\text{AO}}.\delta\text{t} \qquad \text{Gen.} = -\left(-r_A\right).V_R.M_R.\delta\text{t};$$

$$\text{Output} = M_R.V_A.C_A\delta\text{t}; \qquad \text{Acc} = M_R.V_R.\delta C_A$$

$$(-r_A) = kC_A = A.C_A.\exp^{-\frac{E}{\text{RT}}}$$

Hence as $\delta t \to 0$,

$$\frac{d\text{C}_A}{d\text{t}} = \frac{V_A}{V_R}\left(C_{\text{AO}} - C_A\right) - A.C_A.e^{-\frac{E}{\text{RT}}}$$

Table B.1: Parameter definition and values within process model

Parameter	Description	Values
$\mathbf{V_A}$	Inflow rate, m^3s^{-1}	0.50
$\mathbf{T_i}$	Inlet temperature, K	Not defined in deviation variable model
$\mathbf{C_{A0}}$	Inlet concentration, kmol m^{-3}	Manipulated variable
$\mathbf{C_A}$	Outlet Concentration, kmol m^{-3}	Control variable, SS = 12
$\mathbf{V_R}$	Volume of reactor m^3	1
$\mathbf{A}$	Pre-exponential factor s^{-1}	148.41
$\mathbf{E}$	Activation energy, kJ $kmol^{-1}$	11×10^3
$\mathbf{R}$	Universal gas constant, kJ K^{-1} $kmol^{-1}$	8.314
$\mathbf{h_e}$	Convective HTC, kW m^{-2} K^{-1}	100
$\mathbf{A_e}$	Effective area of heating element m^2	3.5
$\mathbf{T_e}$	Temperature of heating element K	Manipulated Variable
ρ	Density of system kg m^{-3}	820 (Benzene)
$\mathbf{C_p}$	Specific heat capacity of system kJ kg^{-1} K^{-1}	1.4 (Benzene)
$\mathbf{T}$	Temperature of Reactor, K	Control Variable, SS = 353
$\mathbf{dH_r}$	Enthalpy of reaction kJ $kmol^{-1}$	2000

Table B.2: Assumptions made in derivation of the underlying process dynamics

Mass Balance	Energy Balance
A and B are of the same molecular mass	No phase change and specific heat capacity of the streams and reactor are constant and approximately equal.
Reactor is perfectly filled and mixed	Agitation imparts no energy into the system
Density of the system remains constant and the system is dilute	Reaction is irreversible and first order.
Mass flowrate in and out are equal and constant	Heating element or jacket is thin walled

Taking deviation variables and linearizing non-linear exponential via:

$$f(x,y) = f(x_o,y_o) + \frac{\mathrm{d}f}{\mathrm{d}x}|_{(x_o,y_o)}(x-x_o) + \frac{\mathrm{d}f}{\mathrm{d}y}|_{(x_o,y_o)}(y-y_o)$$

$$\left(\frac{\mathrm{d}C_A}{d\mathrm{t}} - \frac{\mathrm{d}C_{A_{\mathrm{ss}}}}{d\mathrm{t}}\right) = \frac{V_A}{V_R}\left((C_{A0} - C_{A0_{\mathrm{ss}}}) - (C_A - C_{A\mathrm{ss}})\right) - (r_A - r_{A_{\mathrm{ss}}})$$

$$\frac{\mathrm{d}C_A^*}{d\mathrm{t}} = \frac{V_A}{V_R}(C_{A0}^* - C_A^*) - (k_1 C_A^* + k_2 T^*)$$

where $k_1 = Ae^{-\frac{E}{RT_{\mathrm{ss}}}}$ and $k_2 = \frac{\mathrm{AE}C_{A\mathrm{ss}}}{RT_{\mathrm{ss}}^2}e^{-\frac{E}{RT_{\mathrm{ss}}}}$, where constant terms associated with linearization of the exponential cancel, and the first derivative terms of the steady

state rate constant are zero.

Energy Balance

$$\text{Input} = \left(\rho V_A.C_p.T_i. + h_e A_e \left(T_e - T\right)\right)\delta\text{t} \qquad \text{Gen.} = \left(-r_A\right).\text{dH}_R.V_R.\delta\text{t}$$
$$\text{Output} = \rho.V_A.C_p.T.\delta\text{t} \qquad \text{Acc.} = \rho.C_p.V_R\delta T$$

Hence as $\delta t \to 0$,

$$\frac{\text{d}T}{d\text{t}} = \frac{V_A}{V_R}.(T_i - T) + \frac{h_e A_e}{\rho C_p.V_R}\left(T_e - T\right) - \frac{\text{dH}_R.(-r_A)}{\rho C_p}$$

Taking deviation variables and linearizing non-linear exponential:

$$\frac{dT^*}{d\text{t}} = \frac{V_A}{V_R}.({T_i}^* - T^*) + \frac{h_e A_e}{\rho C_p.V_R}\left({T_e}^* - T^*\right) - \left({k_1}'{C_A}^* + {k_2}'T^*\right)$$

where $k_1' = \frac{\text{dH}_R}{\rho C_p}k_1$ and $k_2' = \frac{\text{dH}_R}{\rho C_p}k_2$. Since the inlet temperature T_i is assumed constant i.e. $T_i^* = 0$, we can reformulate:

$$\frac{dT^*}{d\text{t}} = -\frac{V_A}{V_R}T^* + \frac{h_e A_e}{\rho C_p.V_R}\left({T_e}^* - T^*\right) - \left({k_1}'{C_A}^* + {k_2}'T^*\right)$$

Hence allowing for condensation into linear state space form:

$$\frac{\text{d}C_A^*}{d\text{t}} = -\left(\frac{V_A}{V_R} + k_1\right)C_A^* - k_2T^* + \frac{V_A}{V_R}C_{A0}^*$$
$$\frac{\text{d}T^*}{d\text{t}} = \left(-{k_2}' - \frac{V_A}{V_R} - \frac{h_e A_e}{\rho C_p.V_R}\right)T^* - {k_1}'{C_A}^* + \frac{h_e A_e}{\rho C_p.V_R}{T_e}^*$$

Or

$$\frac{d}{d\text{t}}\begin{bmatrix} C_A^* \\ T^* \end{bmatrix} = \begin{bmatrix} -\left(\frac{V_A}{V_R} + k_1\right) & -k_2 \\ -{k_1}' & -\left({k_2}' + \frac{V_A}{V_R} + \frac{h_e A_e}{\rho C_p.V_R}\right) \end{bmatrix}\begin{bmatrix} C_A^* \\ T^* \end{bmatrix} + \begin{bmatrix} \frac{V_A}{V_R} & 0 \\ 0 & \frac{h_e A_e}{\rho C_p.V_R} \end{bmatrix}\begin{bmatrix} C_{A0}^* \\ {T_e}^* \end{bmatrix}$$

This is related to the definition of $h(\cdot)$ discussed in section 3.4 of the article associated

$$h\left(\mathbf{x}_t^*,\, \mathbf{u}_t^*\right) = \begin{bmatrix} -\left(\frac{V_A}{V_R} + k_1\right) & -k_2 \\ -{k_1}' & -\left({k_2}' + \frac{V_A}{V_R} + \frac{h_e A_e}{\rho C_p.V_R}\right) \end{bmatrix}\mathbf{x}_t^* + \begin{bmatrix} \frac{V_A}{V_R} & 0 \\ 0 & \frac{h_e A_e}{\rho.C_p.V_R} \end{bmatrix}\mathbf{u}_t^*$$

B.6 Generation of demonstrated trajectories and control bounds

Table B.3: Tuned PID controllers facilitated by the MATLAB/Simulink package.

Controller	PID Parameters	Concentration Control Loop	Temperature Control Loop
PID1	K_C	43.07	89.30
	K_I	428.54	683.41
	K_D	0.29	0.33
PID2	K_C	11.36	8.77
	K_I	51.66	19.89
	K_D	0.19	0.39

Table B.4: Bounds of the action space to ensure limits of actuation

Action Bound	Concentration Control Loop ($\mathbf{C^*_{A0}}$), kmol m^{-3}	Temperature Control Loop ($\mathbf{T^*_e}$), K
Upper Bound $\mathbf{u}_{\text{UB}}$	30	60
Lower Bound $\mathbf{u}_{\text{LB}}$	-30	-60

B.7 Case study I and II hyperparameters

Table B.5: Hyperparameters of the Algorithms for offline learning with notation as referenced in Algorithms 1 and 2. In Algorithm 1, different numbers of training epochs were utilised depending on the iteration of policy optimisation as defined in Algorithm 2. The index 1 refers to the number of epochs used in iteration 1 and 2: refers to the number of epochs used from iteration two onwards.

Algorithm	Hyperparameter	Value
Algorithm 2	Maximum Iterations $N_{\max}$	15
	Learning rate κ	0.05
Algorithm 1	Episodes of Monte Carlo Policy Evaluation K	500
	Training Epochs $[N_1,\ N_{2:}]$	[150, 50]
	Learning rate ω	0.008

B.8 Case study III hyperparameters

Table B.6: Hyperparameters of the Algorithms for offline learning with notation as referenced in Algorithms 1 and 2. In Algorithm 1, different numbers of training epochs were utilised depending on the iteration of policy optimisation as defined in Algorithm 2. The index 1 refers to the number of epochs used in iteration 1 and 2: refers to the number of epochs used from iteration two onwards.

Algorithm	Hyperparameter	Value
Algorithm 2	Maximum Iterations $N_{\max}$	10
	Learning rate κ	0.05
Algorithm 1	Episodes of Monte Carlo Policy Evaluation K	500
	Training Epochs $[N_1,\ N_{2:}]$	[10, 10]
	Learning rate ω	0.008

B.9 Case study data requirements and computational time

It should be noted that the following is specific to a serial implementation on an Intel® Xeon® W-2123 CPU with 32.0 GB RAM. All processing times could be improved primarily through a parallel computing configuration of the code or use of more advanced hardware. All values detailed are specific to the hyperparameter settings as in B.7 and B.8.

Table B.7: Data and computational time requirements for Case study II. Parallel implementation and potential code-level improvements would reduce the time requirement substantially.

Algorithm	(Hyper)parameter	Value
Offline Training (CSII)	Trajectories Simulated	125×10^3
	Computational Time Requirements (min)	300
Online Training (CSII)	Trajectories Simulated	10×10^3
	Computational Time Requirements (min)	24

Table B.8: Data and computational time requirements for Case study III. Note the effect of knowledge transfer within the framework with respect to data efficiency and time requirement. Parallel implementation and potential code-level improvements would reduce the time requirement substantially.

Algorithm	(Hyper)parameter	Value
Offline Training (CSIII)	Trajectories Simulated	40×10^3
	Computational Time Requirements (min)	96
Online Training (CSIII)	Trajectories Simulated	10×10^3
	Computational Time Requirements (min)	24

Appendix C

Appendices for research item: safe chance constrained reinforcement learning

C.1 Gaussian process state space modelling

C.1.1 Training of Gaussian process models

Selection of the appropriate hyperparameters, $\hat{\boldsymbol{\lambda}} = [\boldsymbol{\lambda}, \sigma_n^2] \in \mathbb{R}^{n_\lambda+1}$ for a covariance function provides considerable improvement in the predictive abilities of GPs and can be viewed as a parallel to parameter estimation for mechanistic process models. The tuning procedure acts to maximise the marginal log-likelihood $p(\mathbf{Y}_j^T|\boldsymbol{\Upsilon}, \hat{\lambda})$ of the state specific, noisy output data points $\mathbf{Y}_j$, provided with the respective input measurements $\boldsymbol{\Upsilon}$ and hyperparameters $\hat{\lambda}$:

$$\log p(\mathbf{Y}_j^T|\boldsymbol{\Upsilon}, \hat{\lambda}) = -\frac{1}{2}(\mathbf{Y}_j(K + \sigma_n^2 I_N)^{-1}\mathbf{Y}_j^T + \log|K + \sigma_n^2 I_N| + N\log 2\pi) \tag{C.1}$$

Gradient-based optimisation may then be deployed to find $\hat{\lambda}$, which maximise the likelihood of observing our output data, given the covariance function chosen and the input data. This problem is non-convex and so typically multi-start schemes are deployed to find the best solution.

C.1.2 Obtaining function realisations from GP state space models

In this work, we are concerned with obtaining function realisations from a Gaussian process state space model. The state space model is composed of n_x individual Gaussian process models of each state. Here, we use the method proposed by Strassen (1969); Bradford et al. (2020); Umlauft et al. (2018). The method aims to update the posterior distribution of the Gaussian process model according to the initial dataset $\mathcal{D}$ used for model construction, as well as the states and control inputs observed during each trajectory evolution. This combined dataset is denoted $\mathcal{D}_+$.

Algorithm C.1 Function Realisations via GP State Space Model for Decision-making Under Uncertainty

---

Input: Experimental dataset $\mathcal{D}$; GP state space model $f_{GPSS} = [f_{GP}^1(\boldsymbol{\upsilon}), \ldots, f_{GP}^{n_x}(\boldsymbol{\upsilon})]$ with hyperparameters $\hat{\Lambda} = [\hat{\boldsymbol{\lambda}}_1, \ldots, \hat{\boldsymbol{\lambda}}_{n_x}]$ trained on $\mathcal{D}$; Control Policy $\pi(\mathbf{u}|\mathbf{x})$; Finite horizon length T; initial state distribution $p(\mathbf{x}_0)$; Memory for state $\mathcal{B}_x$ and control $\mathcal{B}_u$ trajectories, as well as for information related to decision making $\mathcal{B}_\pi$ for use in subsequent policy optimisation.;

1. Set $\mathcal{D}_+ = \mathcal{D}$

2. Draw $\mathbf{x}_0 \sim p(\mathbf{x}_0)$. Append $\mathbf{x}_0$ to $\mathcal{B}_x$

for $t = 1, \ldots, T-1$ **do**

3a. Observe $\mathbf{x}_{t-1}$, sample $\mathbf{u}_{t-1} \sim \pi(\mathbf{u}|\mathbf{x})$ and concatenate, such that $\boldsymbol{\upsilon}_{t-1} = \left[\mathbf{x}_{t-1}^T \mathbf{u}_{t-1}^T\right]^T$

3b. Condition the GP state space model on $(\mathcal{D}_+, \boldsymbol{\upsilon}_{t-1})$ to obtain the predictive posterior: $p(\mathbf{x}_t|\boldsymbol{\upsilon}_{t-1}, \mathcal{D}_+) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\upsilon}_{t-1}; \mathcal{D}_+), \boldsymbol{\Sigma}(\boldsymbol{\upsilon}_{t-1}; \mathcal{D}_+))$

3c. Draw next state from posterior of the GP state space model, $\mathbf{x}_t \sim p(\mathbf{x}_t|\boldsymbol{\upsilon}_{t-1}, \mathcal{D}_+)$

3d. Update $\mathcal{D}_+ = [\mathcal{D}_+^T \; d_{N+t}^T]^T$, where $d_{N+t} = [\boldsymbol{\upsilon}_{t-1}^T \; \mathbf{x}_t^T]$ and append $\mathbf{x}_t$ and $\mathbf{u}_{t-1}$ to $\mathcal{B}_x$ and $\mathcal{B}_u$, respectively

end for

Output: Function realisation stored in $\mathcal{B}_x$ and $\mathcal{B}_u$ and information related to decision making $\mathcal{B}_\pi$ (to be explained in **Algorithm 4.2**)

The use of Algorithm C.1 allows for proper propagation of model uncertainty and sampling of functions from the GP. In essence, it is desired to obtain state sequences $\mathbf{x}_{0:T} = [\mathbf{x}_0, \ldots, \mathbf{x}_T]$, which are expressive of Eq. 4.1 and represent a realisation of process uncertainty. As samples $\mathbf{x}_t$ are drawn from the posterior they are added, along with the respective input $\boldsymbol{\upsilon}_{t-1}$, to the dataset $\mathcal{D}_+$ upon which the GP is conditioned. This leads to a subsequent update of the GP posterior distribution (via C.2 - C.3) considering previous samples $\mathbf{x}_{t-1}$ as noiseless observations, with retention of the original covariance function hyperparameters $\hat{\boldsymbol{\lambda}}$. This means that if the updated GP posterior were to be queried at the previous input $\boldsymbol{\upsilon}_{t-1}$, the exact realisation of $\mathbf{x}_{t-1}$ would be drawn again i.e. the GP would express $\mathbf{x}_{t-1}$ deterministically. Such an outcome highlights the algorithm's utility in effective function space sampling and implies that future process evolution is explicitly dependent upon the past realisations of uncertainty.

We can express the updated posterior distribution of the j^{th} GP after transition from one discrete time index at $t = t_0$ to $t = t_1$ as follows:

$$\begin{aligned} \mu_j(\boldsymbol{\upsilon}^*; \mathcal{D}_+) &= K_*^{+^T} \Sigma^{+^{-1}} \mathbf{Y}_j^{+^T} \\ \sigma_j^2(\boldsymbol{\upsilon}^*; \mathcal{D}_+) &= k(\boldsymbol{\upsilon}^*, \boldsymbol{\upsilon}^*) - K_*^{+^T} \Sigma^{+^{-1}} K_*^+ \end{aligned} \tag{C.2}$$

where $\mathbf{Y}_j^+ = [\mathbf{Y}_j, y_j^+] \in \mathbb{R}^{1\times(N+1)}$, where $y_j^+ \in \mathbb{R}$ is state $x_j \in \mathbb{R}$ observed at time index $t = t_1$; The updated covariance matrices are expressed as follows:

$$K_*^{+^T} = \left[K_*^T, k(\upsilon_*, \upsilon^+)\right] \qquad \Sigma^{+-1} = \begin{bmatrix} K + \sigma_n I_N & K_+ \\ K_+^T & k(\boldsymbol{\upsilon}^+, \boldsymbol{\upsilon}^+) \end{bmatrix}^{-1} \tag{C.3}$$

where $K_+^T = [k(\boldsymbol{\upsilon}_+, \boldsymbol{\upsilon}_i), \ldots, k(\boldsymbol{\upsilon}_+, \boldsymbol{\upsilon}_N)] \in \mathbb{R}^{1\times N}$, and $\boldsymbol{\upsilon}_+ \in \mathbb{R}^{n_\upsilon}$ is the state and control input pair at time index $t = t_0$. This process is repeated iteratively for state transitions thereafter, which means that the memory and computation requirements will grow quadratically and cubically, respectively, with the time horizon T. Updating K_*^+, Y_j^+ is relatively easy, however, updating $\Sigma^{+^{-1}}$, is slightly more involved due to inversion. In order to do this we use the method from Strassen (1969) as proposed in Bradford et al. (2020); Umlauft et al. (2018). We refer the reader to these works for more information.

C.2 Validation of Gaussian process models used in case study

Table C.1 details the results of leave-one-out cross validation of the Gaussian process state space model used in this case study. Specifically, the results reported assess multi-step ahead predictions, which correspond to forecasting the entire batch given an initial state and control profile. Results are reported as the average across all possible different folds (of which there are 32). Predictions from the GP were drawn using the mean of the posterior. The dataset used to construct the Gaussian process models is available at https://github.com/mawbray/Lutein-Dataset

Table C.1: Multistep prediction mean absolute percentage error (MAPE) of leave-one-out cross validation of Gaussian process state space model used in Case study.

Component of State	MAPE (%)
Biomass	2.5
Nitrate	4.3
Lutein	2.2

C.3 Proximal policy optimisation, The advantage function and entropy regularisation

PPO is at its core a policy gradient (PG) method. PG methods have previously been discussed, and so this work directs the interested reader to the original paper Sutton et al. (1999) and other recent work Petsagkourakis et al. (2020b). PPO utilises a specific instance of the PG, known as the advantage policy gradient (APG). The APG Mnih et al. (2016) is a powerful, low variance form of the policy gradient, which utilises the generalised advantage function estimate A_{φ} (GAE), rather than the action-value estimate, as in vanilla policy gradient methods Sutton et al. (1999). Further detail on the GAE and PPO is provided by C.3.1 and C.3.3, respectively. In practice, the investigation found the addition of an entropy regularisation term useful in RL training. Entropy regularisation is widely studied in the RL literature, and at a high level provides mechanism to ensure the policy does not converge deterministically to a

poor local optimum. This is particularly important in view of RL as a set of sampling-based algorithms Neu et al. (2017); Ahmed et al. (2019a) and is discussed further in C.3.2.

C.3.1 The advantage function

The advantage function Schulman et al. (2018b) is formalised:

$$V^{\pi}(\mathbf{x}_t) = \mathbb{E}_{\pi}\Bigg[\sum_{t'=t}^{T-1} R_{t'+1}|\mathbf{x} = \mathbf{x}_t\Bigg] \qquad Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = \mathbb{E}_{\pi}\Big[R_{t+1} + \gamma V^{\pi}(\mathbf{x}')|\mathbf{x} = \mathbf{x}_t, \mathbf{u} = \mathbf{u}_t\Big]$$
$$A^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t) - V^{\pi}(\mathbf{x}_t) \tag{C.4}$$

and, represents the difference between the expected returns under a policy in the current state, V^{π}, and the returns accumulated from selecting a given control in the current state and the current policy thereafter, Q^{π}. In RL practice, parameterisation of the value function V_{ψ} is required in order to approximate the true value function V^{π}, such that $V_{\psi} \approx V^{\pi}$. Decision as to the model structure and initialisation of the parameters asserts bias into estimation of the advantage function. This is reduced through use of the generalised advantage function estimate $\hat{A}^{\pi}$ (GAE). The GAE provides a mechanism to explicitly *trade off* variance and bias, by maximising the information provided by the reward signal. Explicitly, the GAE is formalised as:

$$\begin{aligned} \hat{A}^{\pi}_t &= \delta_{t+1} + (\rho\gamma)\delta_{t+2} + \ldots + (\rho\gamma)^{T-t+1}\delta_T \\ \delta_{t+1} &= R_{t+1} + \gamma V_{\psi}(\mathbf{x}_{t+1}) - V_{\psi}(\mathbf{x}_t) \end{aligned} \tag{C.5}$$

The parameter $\rho = [0, 1]$ provides the mechanism to balance the bias and variance. Values closer to 1 reduce bias by utilising more information from the reward signal, but at the compromise of increasing the variance of the estimate. The opposite applies as values tend to 0.

C.3.2 Entropy regularisation

There is a rich literature on maximum entropy (Max.Ent.) RL Ahmed et al. (2019b); Haarnoja et al. (2018); Ziebart (2010). Instead of simply optimizing for the process objective and accumulated reward, $G(\boldsymbol{\tau})$, Max.Ent. RL also optimises for the expected

entropy of the stochastic policy learned. As a result, we can formulate the Max.Ent. RL objective, J_H as follows:

$$J_H = \mathbb{E}\big[G(\boldsymbol{\tau}) + H_\pi\big] \tag{C.6}$$

where $H_\pi = -\mathbb{E}_\pi\big[\log \pi(\mathbf{u}|\mathbf{x})\big]$ is the entropy of the policy. Typically, in practice, this objective is maximised via a regularisation term i.e. not as an extrinsic addition of entropy to the reward signal, and therefore not optimised via the PG. It is thought that entropy regularisation provides two main benefits: 1) it modifies the optimisation landscape 'for the better', and in some cases provides a smoother landscape than the vanilla objective, and 2) the use of entropy plays some role in tackling the exploration-exploitation paradigm, i.e. by regularising entropy, exploration is encouraged, preventing convergence to a suboptimal deterministic policy. The use of entropy was found to be particularly helpful in this study, aiding the learning dynamics. It could be perceived that the constraint boundary provides a discontinuity in the reward landscape, and the promotion of exploration via entropy provides mechanism to 'escape' local optima.

C.3.3 Entropy regularised proximal policy optimisation

PPO aims to provide conservative policy updates, by utilising the concept of trust region optimisation. The idea of trust region optimisation in the RL sense, is to constrain the update of an initial policy, such that the ultimate policy remains within a given distance of the initial in policy space. This distance could e.g. be quantified by the Kullback-Liebler divergence. One algorithm known as trust-region policy optimisation (TRPO) necessitates estimate of the Hessian of the approximate KL divergence with respect to the policy parameters Schulman et al. (2017a) (this also shares similarities with the natural policy gradient Kakade (2001)). PPO sidesteps this complexity through approximation of the 2nd order TRPO update with a first order update - instead of explicitly enforcing this as a hard constraint, PPO enforces this via a penalty method Schulman et al. (2017b). This means that PPO is more computationally efficient than TRPO and provides flexible use of different function approximators (policy parameterisations).

The objective function L^{CLIP} formalised within the PPO framework follows:

$$
\begin{aligned}
r_t(\theta) &= \frac{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\pi_{\theta_{old}}(\mathbf{u}_t|\mathbf{x}_t)} \\
L^{CLIP}(\theta) &= \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t^\pi, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^\pi)\Big]
\end{aligned}
\tag{C.7}
$$

where, $\epsilon = [0, 1]$ and $\hat{A}_t^\pi$ is the advantage function, as discussed previously. By clipping the ratio r, updates corresponding to negative advantages are clipped with a ratio of $r = 1 + \epsilon$, whereas updates with positive advantages are clipped at $r = 1 - \epsilon$. The minimum is taken in order to provide a pessimistic update and enforce what could be interpreted as a trust-region. A full entropy regularised PPO algorithm is presented by Algorithm C.3.3.

Algorithm C.2 Entropy Regularised Proximal Policy Optimisation

Input: Approximate state space model or process dynamics $f_{SS}(\cdot)$; Initial control policy $\pi(\mathbf{u}|\mathbf{x};\theta_0)$; Initial critic $V(\mathbf{x}_t,\psi_0)$; Reward function $R_{xx'}$; Finite horizon length T; initial state distribution $p(\mathbf{x}_0)$; entropy penalty term $\beta \in \mathbb{R}^+$; Learning rate $w^\pi \in \mathbb{R}^+$; Learning rate $w^V \in \mathbb{R}^+$; Strategies for updating the learning rates (schedules) $f_w^\pi(\cdot)$ and $f_w^V(\cdot)$; Memory $\mathcal{B}_{info}$ for information required for policy optimisation; K episodes; Learning updates per batch J; batchsize of M trajectories; tolerance criterion

1. $i = 0$

while not converged **do**

2a. Obtain a batch of K rollouts over horizon of T discrete intervals, via $\pi(\mathbf{u}|\mathbf{x};\theta_i)$, f_{SS}, and $p(\mathbf{x}_0)$

2b. Return trajectory information i.e. rewards $R^{(k)}_{0:T-1} = [R^{(k)}_1,\ldots,R^{(k)}_T]$ under $R_{xx'}$ for the sequence of controls $\mathbf{u}^{(k)}_{0:T-1} = [\mathbf{u}^{(k)}_0,\ldots,\mathbf{u}^{(k)}_{T-1}]$ and states $\mathbf{x}^{(k)}_{0:T} = [\mathbf{x}^{(k)}_0,\ldots,\mathbf{x}^{(k)}_T]$, corresponding to each rollout and store in $\mathcal{B}_{info}$

2c. $j = 0$

while $j < J$ **do**

2ci. Perform policy optimisation by sampling the information of M trajectories from $\mathcal{B}_{info}$, calculating the respective importance ratios r_t via Eq. C.7 and GAEs via Eq. C.5:

$\theta_{i+1} = \theta_i + w_i^\pi \nabla_\theta \big[\frac{1}{MT}\sum_{m=1}^{M}\sum_{t=0}^{T-1} L^{CLIP}(\mathbf{x}_t^{(m)},\mathbf{u}_t^{(m)},\mathbf{x}_{t+1}^{(m)},\theta_i) + \beta H_\pi(\pi(\mathbf{u}_t^{(m)}|\mathbf{x}_t^{(m)}))\big]$

2cii. Update the critic $V(\mathbf{x},\psi_i)$ on the same data sampled in **2c.i.** and the respective returns, G_t:

$\psi_{i+1} = \psi_i - \nabla_{\psi_i} V(\mathbf{x},\psi_i)(V(\mathbf{x}_t^{(m)},\psi_i) - G_t^{(m)})$

2ciii. Update the learning rate : $w^\pi_{i+1} = f^\pi_w(w_i)$

2civ. Update the learning rate : $w^V_{i+1} = f^V_w(w_i)$

2cv. $i+=1$, $j+=1$

end while

2d. Reset memory $\mathcal{B}_{info}$

2e. Assess tolerance criterion

end while

Output: Optimal Constrained Policy $\pi^*_C(\theta)$ trained under $\boldsymbol{\xi}^*$

C.4 Evaluating joint constraint satisfaction empirically

In this work, we are concerned with the satisfaction of the joint chance constraints expressed by:

$$F_X(0) = \mathbb{P}(X \leq 0) = \mathbb{P}(\bigcap_{i=0}^{T} \{\mathbf{x}_i \in \hat{\mathbb{X}}_i\}) \tag{C.8}$$

where $\hat{\mathbb{X}}_i$ is the tightened joint constraint set and

$$X = \max_{(t,j)\in\{0,\ldots,T\}\times\{1,\ldots,n_g\}} A_j \mathbf{x}_t - b_j,$$

defines the maximum constraint violation during process evolution. As analytical expression of Eq. C.8 is not available, it is proposed to instead estimate it via Monte Carlo sampling. Hence we can define the empirical cumulative distribution function (ecdf) via S Monte Carlo samples:

$$F_X(0) \approx F_{SA}(0) = \frac{1}{S} \sum_{s=1}^{S} \mathbf{1}(X^{(s)} \leq 0) \tag{C.9}$$

where $\mathbf{1}$ is the indicator function. However, due to the limits imposed by finite samples, the approximation is likely to include error. Therefore, in order to account for this we deploy a concept from the *binomial proportion confidence interval* literature. Specifically, the Clopper–Pearson interval Clopper and Pearson (1934), which enables us to ensure the probability of joint satisfaction with a given confidence level, $1 - \upsilon$, on the basis of empirical observation. This is expressed by Lemma 3, which is recycled from Petsagkourakis et al. (2020a).

Lemma 3 ***Joint chance constraint satisfaction via the Clopper-Pearson confidence interval*** *Clopper and Pearson (1934); Brown et al. (2001): Consider the realisation of $F_{SA}(0)$ based on S independently and identically distributed samples. The lower bound of the true value $F_{LB}(0)$ may be defined with a given confidence $1-\upsilon$, such that:*

$$\begin{aligned} &\mathbb{P}(F_X(0) \geq F_{LB}(0)) \geq 1 - \upsilon \\ &F_{LB}(0) = 1 - \mathit{betainv}(\upsilon, S + 1 - SF_{SA}(0), SF_{SA}(0)) \end{aligned} \tag{C.10}$$

where betainv$(\cdot)$ is the inverse of the Beta cumulative distribution function with parameters $\{S+1-SF_{SA}(0)\}$ and $\{SF_{SA}(0)\}$.

C.5 Further Information on Benchmark

In construction of the benchmark provided, a direct collocation scheme was implemented in python. The code is available at https://github.com/mawbray/Lutein_DO. In the case of NMPC and online optimisation, an approximate problem was solved if a solution could not be found online an approximate problem was solved instead. This was conducted via the following formulation:

$$\begin{aligned}
&\max_{\mathbf{u}_{t':T-1},\boldsymbol{\epsilon}} \sum_{t=t'}^{T-1} R_{t+1} - \mathbf{z}\boldsymbol{\epsilon} && \text{(see Eqs. 4.33 and 4.32)}\\
&\text{s.t.}\\
&\quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) && \text{(see Eqs. 4.30 and 4.31)}\\
&\quad \mathbf{u}_t \in \hat{\mathbb{U}}\\
&\quad A\mathbf{x}_t - b \leq \boldsymbol{\epsilon} && \text{(see Eqs. 4.7 and 4.32)}\\
&\quad \mathbf{0} \leq \boldsymbol{\epsilon}\\
&\quad \forall t \in \{t', ..., T-1\}
\end{aligned} \tag{C.11}$$

where $\mathbf{z} = [1, 10, 10]$ and $\mathbf{x}_{t'}$ is observed from the uncertain process. This approximate problem modifies the objective function to incentivize minimisation of constraint violation. It should be stressed that this problem is only solved if a solution cannot be found to the original problem. This was typically the case when the optimisation was initialised such that $\mathbf{x}_{t'}$ had already violated the constraints and arose from the inability to handle constraints.[1]

C.6 Hyperparameters for Learning in Case Study

[1]Note that there is a notational mistake in the supporting information provided in Mowbray et al. (2022a). The information provided here is correct and was used to generate the results presented in this thesis.

Table C.2: Miscellaneous hyperparameters specific to Proximal policy optimisation algorithm used in this work.

Parameter	Value
Episodes, K	200
Nodes per LSTM layer of Policy Net.	30
LSTM Layers in Policy Net.	4
Activation function in output layer of Policy Net.	ReLU6
Nodes per LSTM layer in Value Net.	30
LSTM layers in Value Net.	2
Activation function in output layer of Value Net.	Leaky ReLU
Policy learning rate, w^{π}	5×10^{-3}
Value learning rate, w^{V}	5×10^{-3}
GAE weight, ρ	0.99
Batch size, M	100
Weight updates, J	2
Clipping factor, ϵ	0.2
Discount factor, γ	0.99
Entropy regularisation weights, β	5×10^{-2}

Appendix D

Appendices for research item: distributional reinforcement learning for scheduling of chemical production processes

D.1 Particle swarm and simulated annealing (PSO-SA) hybrid algorithm

The general intuition behind stochastic search is provided by Algorithm 5.3.4. Most stochastic search algorithms adhere to the description provided, which is summarised as follows. In **step 1** a population of parameters for a policy (network) is instantiated by a space filling procedure of choice. Then, in **step 2**, the following steps are performed *iteratively*: a) a population of neural policies are constructed from the population of parameters; b*i-iv*) the population is evaluated via Monte Carlo according to the estimator defined; b*v*) the current best policy is tracked and stored in memory (if a criterion is satisfied); and, c) the population parameters are then perturbed according to a stochastic search optimisation algorithm. After a number of optimisation iterations, the best policy identified is output. The number of optimisation iterations is defined according to the available computational budget or according to a threshold on the rate of improvement.

In this work, we use a hybrid particle swarm optimisation - simulated annealing stochastic search optimisation algorithm with a search space reduction strategy Park et al. (2005a) in order to identify the optimal scheduling policy parameters, $\theta^* \in \mathbb{R}^{n_\theta}$. Hybridisation of particle swarm optimisation and simulated annealing enables us to balance exploitation and exploration of the parameter space. The search space reduction strategy was found to improve performance, especially in the *harder* experiments conducted i.e. for experiment E3-8.

D.1.1 Particle swarm optimisation

Particle swarm optimisation is a stochastic search algorithm initially proposed in Kennedy and Eberhart (1995a), which takes inspiration from the behaviour of animals such as fish and birds in a shoal or murmuration, respectively. The population is defined by a number of individuals, P. Each individual (candidate will be as a synonymous term) within the population is first initialised with a velocity, $\mathbf{v}_i^0 \in \mathbb{R}^{n_\theta}$, and a position, θ_i^0, $\forall i \in \{1, \ldots, P\}$. All individual's positions are subject to upper and lower bounds, such that $\theta_i = [\theta_{LB}, \theta_{UB}]^{n_\theta}$. Similarly, all velocities are subject to upper bounds, such that $\mathbf{v}_i = [-\infty, v_{max}]^{n_\theta}$, where $v_max \in \mathbb{R}$ is typically defined as:

$$v_{max} = c_3(\theta_{UB} - \theta_{LB})$$

where $c_3 = [0, 1]$. Each candidate is then rated with respect to an objective function, $F_{SA} : \mathbb{R}^{n_\theta} \to \mathbb{R}$ that one would like to minimise. A note of: the candidate's current best position, $\mathbf{b}_i^* \in \mathbb{R}^{n_\theta}$; the locally best position, $\mathbf{g}_i^* \in \mathbb{R}^{n_\theta}$, within a neighbourhood of n_h individuals; and, of the global best, θ^*, is then made. The optimisation procedure may then proceed by updating the position of each individual in the population iteratively via the following equation:

$$\begin{aligned} \mathbf{v}_i^{k+1} &= \omega \mathbf{v}_i^k + c_1 r_1 \big(\mathbf{b}_i^* - \theta_i^k\big) + c_2 r_2 (\mathbf{g}_i^* - \theta_i^k) \\ \theta_i^{k+1} &= \theta_i^k + \mathbf{v}_i^{k+1} \end{aligned} \tag{D.1}$$

where $\omega \in \mathbb{R}_+$, $c_1 \in \mathbb{R}_+$ and $c_2 \in \mathbb{R}_+$ may be interpreted to represent a weighting for inertia, individual acceleration and global acceleration, respectively. The individual and local best components of the total acceleration are adjusted stochastically by the definition of $r_1 \sim U(0, 1)$ and $r_2 \sim U(0, 1)$. From Eq. D.1, the update of the individual's position makes use of information both from the trajectory of the individual but

the trajectories collected across the swarm.

D.1.2 Simulated annealing

Simulated annealing is also a stochastic search optimisation algorithm, which is superficially related to structural transitions of physical systems under temperature change. The algorithm updates the position of an individual, θ_i^k, probabilistically based on the improvement provided in the proposed update with respect to the objective, F_{SA}, and the current *temperature*, $T \in \mathbb{R}_+$, of the algorithm. Specifically, an updated candidate, $\bar{\theta}_i^k$, is proposed and accepted or rejected as follows:

$$\bar{\theta}_i^k = \theta_i^k + \mathbf{w}_i^k \tag{D.2a}$$

where $\mathbf{w}_i^k = [-1, 1]^{n_\theta}$ is described according to a distribution of choice on the space $[-1, 1]^{n_\theta}$. Eq. D.2a details the perturbation of a candidate's position. The evaluation and acceptance or rejection of the proposed position follows:

$$\Delta E = F_{SA}(\theta_i^k) - F_{SA}(\bar{\theta}_i^k) \tag{D.2b}$$

$$\theta_i^{k+1} = \begin{cases} \bar{\theta}_i^k, & \text{if } \Delta E > 0 \text{ or } \bar{z} \geq \exp(\frac{\Delta E}{T}) \\ \theta_i^k, & \text{otherwise} \end{cases} \tag{D.2c}$$

where $\bar{z} \sim U(0, 1)$. From Eq. D.2c, it is seen that if the proposed candidate does not improve with respect to the objective function, then it is accepted probabilistically to balance exploration and exploitation.

Generally, schemes are implemented to update the temperature, T, given that it controls the acceptance rate. The lower the value of T, the higher the probability of acceptance. The larger the value of T the lower the probability of acceptance. Due to the nature of the hybridisation used in this work, large constant values of T were used.

D.1.3 Search space reduction

Search space reduction strategies are known to improve the performance of general stochastic search algorithms. In this work, preliminary experiments demonstrated that the use of a space reduction strategy improved performance of θ^* as identified by

the algorithm. The strategy follows the reasoning of the work provided in Park et al. (2005a):

$$\begin{aligned} \theta_{LB}^{k+1} &= \theta_{LB}^{k} + \alpha(\theta_{LB}^{k} - \theta^{*}) \\ \theta_{UB}^{k+1} &= \theta_{UB}^{k} - \alpha(\theta_{UB}^{k} - \theta^{*}) \end{aligned} \tag{D.3}$$

where $\alpha = [0, 1]$ represents a learning rate. Algorithm D.1.3 details the hybridization of the constituent components discussed in this section.

In this work, two logic conditions are implemented to trigger exploitative search and space reduction and are defined as follows:

$$\text{logic}_{SA} = \begin{cases} \text{True}, & \textbf{if } k \bmod 5 = 0 \\ \text{False}, & \textbf{otherwise} \end{cases} \tag{D.4}$$

$$\text{logic}_{SSR} = \begin{cases} \text{True}, & \textbf{if } (k \bmod 5 = 0) \ \& \ (k > 20) \\ \text{False}, & \textbf{otherwise} \end{cases} \tag{D.5}$$

D.1.4 Policy network structure selection

Generally, it was observed that the performance of π learned via the method proposed was dependent upon proper selection of the neural network structure. It was found that a neural network 3 layer network with: an input layer of $n_x = 2N + 2u + 1$ nodes, where N is the number of tasks and n_u is the number of units; hidden layer 1 composed of 10 feedforward nodes; hidden layer 2 composed of 4 Elman recurrent (tanh) nodes; hidden layer 3 composed of feedforward 2 nodes; and, an output layer composed of n_u nodes. Hyperbolic tangent activation functions were applied across hidden layer 2, a sigmoid over hidden layer 3, and a ReLU6 function over the output layer. The investigation of deeper networks was generally led by research supporting their use for approximation of non-smooth functions. The use of recurrency within the network was used to handle the potential partial observability of the problem when uncertainty was present in case study.

Algorithm D.1 A hybrid particle swarm optimisation - simulated annealing algorithm with a search space reduction strategy

Input: Initial upper, θ^0_{UB}, and lower bounds, θ^0_{LB}, on the parameter search space; Population size, P; Maximum velocity, $\mathbf{v}_{max} = [v_{max,1}, \ldots, v_{max,n_\theta}]$; Search space reduction step size, α; Particle swarm optimisation algorithm, $g_{PSO}(\cdot)$; Simulated annealing algorithm, $g_{SA}(\cdot)$; Temperature, T; Cooling schedule, $g_T(\cdot)$; Search space reduction rule, g_{SSR}; Objective function, $F_{SA}(\cdot)$; Memory buffer, $\mathcal{B}_{info}$; Maximum number of search iterations, K; Logic condition to trigger simulated annealing search optimisation algorithm, logic_{SA}; Logic condition to trigger search space reduction, logic_{SSR}

1. Generate initial population of individual parameters, $\Theta^1 = \{\theta^1_1, \ldots, \theta^1_P\}$. Each parameter setting $\theta^1_i = [\theta^1_{i,1}, \ldots, \theta^1_{i,n_\theta}] \in \Theta^1$ is generated such that, $\theta_{i,j} \sim U(\theta_{LB,j}, \theta_{UB,j})$, $\forall j \in \{1, \ldots, n_\theta\}$.

2. Initialize a velocity for each individual in the population via the following strategy: $\mathbf{v}^0_i = (2\mathbf{a} - 1)(\theta_{UB} - \theta_{LB})$, where $\mathbf{a} = [a_1, \ldots, a_{n_\theta}]$, and $a_i \sim U(0, 1)$. Define $\mathbb{V}^1 = \{\mathbf{v}_i, \forall i \in \{1, \ldots, P\}\}$.

for k = 1, ..., K **do**

3a. Evaluate $J^k_i = F_{SA}(\theta_i),\ \forall \theta_i \in \Theta^k$. Append J_i to $\mathcal{B}_{info},\ \forall i$

3b. Determine the current local best, $\mathbf{g}^*_i$, for a neighbourhood composed of n_h individuals, for $\theta_i \in \Theta^k$. Define $\mathbb{G}^k = \{\mathbf{g}^*_i, \forall i \in \{1, \ldots, P\}\}$.

3c. According to $J^j_i \in \mathcal{B}_{info},\ \forall j \in \{1, \ldots, k\}$, determine whether θ^k_i is the best parameter setting observed by the individual. Store this position as the current individual best, $\mathbf{b}^*_i$ and store in $\mathbb{B}^k = \{\mathbf{b}^*_i, \forall i \in \{1, \ldots, P\}\}$

if logic_{SSR} **then**

3d. Reduce search space: $\theta^k_{UB}, \theta^k_{LB} = g_{SSR}(\mathbb{G}^k, \theta^k)$

end if

if logic_{SA} **then**

3e.i. Conduct simulated annealing and update individual and neighbourhood best: $\Theta^k, \mathbb{G}^k, \mathbb{B}^k, \mathcal{B}_{info} = g_{SA}(\Theta^k, \mathbb{G}^k, \mathbb{B}^k, \mathcal{B}_{info}, F_{SA}, T^k)$, via e.g. Eq. D.2

3e.ii. Update temperature: $T^{k+1} = g_T(T^k)$

end if

3f. Update population via Particle Swarm: $\Theta^{k+1}, \mathbb{V}^{k+1} = g_{PSO}(\Theta^k, \mathbb{G}^k, \mathbb{B}^k)$ via e.g. Eq. D.1. Ensure $\theta^{k+1}_i \in \Theta^{k+1}$ and $\mathbf{v}^{k+1}_i \in \mathbb{V}^{k+1}$ satisfy constraints provided by θ_{LB}, θ_{UB} and $\mathbf{v}_{max}$

end for

4. Determine global best parameters, θ^*, from $\mathbb{B}^K$ and $\mathcal{B}_{info}$

Output: θ^*

D.2 Definition of the production scheduling problem

The following text defines the problem case studies presented in this work. Full code implementation of the system defined here is available at https://github.com/mawbray/ps-gym and is compatible with stable RL baseline algorithms (as it is coded via the recipe provided by OpenAI gym custom classes).

D.2.1 Problem definition

We consider a multi-product plant where the conversion of raw material to product only requires one processing stage. *We assume* there is an unlimited amount of raw material, resources, storage and wait time (of units) available to the scheduling element. Further, we assume that the plant is completely reactive to the scheduling decisions of the policy, π, although this assumption can be relaxed if decision-making is formulated within an appropriate framework as shown in Hubbs et al. (2020a). The scheduling element *must decide* the sequencing of tasks (which correspond uniquely to client orders) on the equipment (units) available to the plant. The following operational rules are imposed on the scheduling element:

1. A given unit l has a maximum batch size for a given task i. Each task must be organized in campaigns (i.e. processed via multiple batches sequentially) and completed once during the scheduling horizon. All batch sizes are predetermined, but there is uncertainty as to the processing time (this is specific to task and unit).

2. Further, the task should be processed before the delivery due date of the client, which is assumed to be an uncertain variable (the due date is approximately known at the start of the scheduling horizon, but is confirmed with the plant a number of periods before the order is required by the client).

3. There are constraints on the viable sequencing of tasks within units (i.e. some tasks may not be processed before or after others in certain units).

4. There is a sequence and unit dependent cleaning period required between operations, during which no operations should be scheduled in the given unit.

5. Each task may be scheduled in a subset of the available units.

6. Some units are not available from the beginning of the horizon and some tasks may not be processed for a fixed period from the start of the horizon (i.e. they have a fixed release time).

7. Processing of a task in a unit must terminate before another task can be scheduled in that unit.

The objective of operation is to minimise the makespan and the tardiness of task (order) completion. This means that once all the tasks have been successfully processed according to the operational rules defined, then the decision making problem can be terminated. As in the original work, we formalize the notation of a number of problem sets and parameters in Table D.1. We try to keep notation consistent with that work. It should be noted that in this work, we transcribe the problem as a discrete-time formulation. The original work Cerda et al. (1997) utilised a continuous-time formulation. We formalize the notation of a number of problem sets and parameters in Table D.1.

Table D.1: Table of problem parameters and sets. *D.T.I. is shorthand for discrete time indices.

Sets	Notation
Tasks (orders) to be processed	$\mathbb{I} = \{1, \ldots, N\} \subset \mathbb{Z}_+$
Available units	$\mathbb{L} = \{1, \ldots, n_u\} \subset \mathbb{Z}_+$
Available units for task i	$\mathbb{L}_i \subseteq \mathbb{L}$
The task most recently processed in unit l	$\mathbb{M}_l \subset \mathbb{Z}$; $\lvert\mathbb{M}_l\rvert \leq 1$
Tasks which have been completely processed	$\mathbb{T}_f \subset \mathbb{Z}$
Feasible successors of task i in unit l	$\mathbb{SU}_{il} \subset \mathbb{Z}_+$
Feasible successors of task i	$\mathbb{SU}_i = \cup_{l \in \mathbb{L}} \mathbb{SU}_{il} \subset \mathbb{Z}_+$
The task currently being processed in unit l	$\mathbb{O}_l$
Parameters	**Notation**
Number of tasks	$N \in \mathbb{R}_+$
Due date of client order (task)	$\tau_i \in \mathbb{Z}_+$
Number of units	$n_u \in \mathbb{Z}_+$
Batch size of unit l for task i	$B_{il} \in \mathbb{R}_+$
Number of batches required to process task i in unit l	$NB_{il} \in \mathbb{Z}_+$
Sequence dependent set up time	$\hat{v}_{ilt} \in \mathbb{Z}_+$
Release time of tasks in D.T.I.	$(\text{RTT})_i \in \mathbb{Z}_+$
Release time of units in D.T.I.	$(\text{RTU})_l \in \mathbb{Z}_+$
Sequence dependent cleaning time (task i succeeds m)	$TCL_{mil} \in \mathbb{Z}_+$
Miscellaneous Variables	**Notation**
Variable indicating campaign production of task i starts at time t in unit l	$W_{ilt} \in \mathbb{Z}_2$
Variable indicating unit l is processing task i at time t	$\bar{W}_{ilt} \in \mathbb{Z}_2$
Integer variable denoting task i is being processed in unit l at time t	$w_{lt} \in \mathbb{U}$
Estimated D.T.I. for unit l to process a batch of task i	$\bar{PT}_{il} \in \mathbb{Z}$
Actual D.T.I. for unit l to process batch n of task i	$PL_{inl} \in \mathbb{Z}$
Actual D.T.I. to finish processing current campaign in unit l at time t	$\delta_{lt} \in \mathbb{Z}$
Current inventory of task (client order) i at time t	$I_{it} \in \mathbb{R}_+$
D.T.I. until due date of task i at time t	$\rho_{it} \in \mathbb{Z}$
The length of a discrete time index	$dt \in \mathbb{R}_+$

D.2.2 Formulating discrete-time scheduling problems as Markov decision processes

The methodology presented in Section 6.3 utilises formulation of the scheduling problem as an MDP, which is an approximation to solving the finite-horizon stochastic optimal control problem (as detailed in Eq. 5.3). Construction of the problem follows.

Firstly, the time horizon is discretised into $T = 200$ finite periods of length $dt = 0.5$ days, where $t \in \{0, \ldots, T\}$ describes the time at a given index in the discrete time horizon.

We hypothesise that the system is made completely observable by the following state representation:

$$\mathbf{x}_t = \Big[I_{1t}, \ldots, I_{Nt}, w_{1t}, \ldots, w_{n_u t}, \delta_{1t}, \ldots, \delta_{n_u t}, \rho_{1t}, \ldots, \rho_{Nt}, t\Big]^T \in \mathbb{R}^{2N+2n_u+1} \tag{D.6}$$

where I_{it} $\forall i \in \mathbb{I}$ quantifies the current inventory of client orders in the plant; the tasks processed within units in the plant over the previous time interval, w_{lt} $\forall l \in \mathbb{L}$; the discrete time indices remaining until completion of the task campaigns being processed in all units, δ_{lt} $\forall l \in \mathbb{L}$; the discrete time indices remaining until orders (tasks) are due, ρ_{it} $\forall i \in \mathbb{I}$; and, the current discrete time index, t.

Similarly, we define the control space, $\mathbf{u} = [u_1, \ldots, u_{n_u}]^T \in \mathbb{U}$ as the set of integer decisions (tasks) that could be scheduled in the available units. Hence, one can define $\mathbb{U} = \bigcup_{l=1}^{n_u} \mathbb{U}_l \subset \mathbb{Z}_+^{n_u}$, where $\mathbb{U}_l = \mathbb{I} \cup \{N+1\}$, and $N+1$ is an integer value, which represents the decision to idle the unit for one time period.

A sparse reward function is defined as follows:

$$R = \begin{cases} \mathbf{d}^T \mathbf{x}_{t+1}, & \text{if } t = T-1 \text{ or } \mathbb{T}_f = \mathbb{I} \\ 0, & \text{otherwise} \end{cases} \tag{D.7}$$

where $\mathbf{d} \in \mathbb{Z}^{n_x}$ denotes a vector specified to penalise order lateness and makespan; $\mathbb{I} \subseteq \mathbb{Z}$ denotes the set of tasks (or client orders); and, $\mathbb{T}_f \subseteq \mathbb{Z}$ denotes the set of tasks, which have been completed. If $\mathbb{T}_f = \mathbb{I}$ is satisfied, the decision-making problem is terminated. Definition of $\mathbf{d}$ follows:

$$\mathbf{d} = \big[\mathbf{0}_{1,N}, \mathbf{0}_{1,n_u}, \mathbf{0}_{1,n_u}, \mathbf{1}_{1,N}, -1\big]^T \in \mathbb{Z}^{2N+2n_u+1} \tag{D.8}$$

where $\mathbf{0}_{1,j}$ represents a row vector of zeros of dimension j; and, $\mathbf{1}_{1,j}$ represents a row vector of ones of dimension j. How the definition provided by Eq. D.7 enforces

operational objectives is clarified by full definition of the dynamics, as presented subsequently.

The inventory balance is defined to highlight that the underlying state space model is both non-smooth and subject to uncertain variables:

$$I_{it+1} = I_{it} + \sum_{n=1}^{NB_{il}} \sum_{l \in \mathbb{L}_i} B_{il} W_{ilt-\hat{\upsilon}_{ilt}-\sum_{\hat{n}=1}^{n} PL_{i\hat{n}l}} \quad \forall i \in \mathbb{I} \tag{D.9}$$

where B_{il} is the maximum batch size (kg) of unit l for task i; NB_{il} is the integer number of batches required to satisfy the client's order size in unit l; PL_{inl} is a random variable that indicates the number of discrete time indices required to produce batch n of task i in unit l and represents a realization of an uncertain variable; $W_{ilt} \in \mathbb{Z}_2$ is a binary variable, indicating a production campaign for task i was first scheduled in unit l at time t; $\hat{\upsilon}_{ilt}$ defines the current sequence dependent unit set up time required to process order i in unit l at time t. The sequence dependent unit set up time is defined as follows:

$$\hat{\upsilon}_{ilt} = \begin{cases} \sum_{m \in \mathbb{M}_l} \max(\max([\mathrm{TCL}_{mil} + t_{ml} - t]^+, [(\mathrm{RTT})_i - t]^+), [(\mathrm{RTU})_l - t]^+), & \text{if } |\mathbb{M}_l| = 1 \\ \max([(\mathrm{RTT})_i - t]^+, [(\mathrm{RTU})_l - t]^+), & \text{otherwise} \end{cases} \tag{D.10}$$

where $\mathbb{M}_l \subset \mathbb{Z}_+$, defines a set denoting the task most recently processed in unit l, which finished at time, $t_{ml} \leq t$, such that the cardinality of the set $|\mathbb{M}_l| \in \mathbb{Z}_2$; $\mathrm{TCL}_{mil} \in \mathbb{R}_+$ defines the cleaning times between successive tasks m and i; $(\mathrm{RTT})_i \in \mathbb{Z}_+$ defines the release time (in number of discrete time indices) a task i cannot be processed for at the start of the horizon; $(\mathrm{RTU})_l \in \mathbb{Z}_+$ defines the release time of unit l.

It is apparent that in order to handle the requirements for cleaning time between processing of successive tasks m and i in unit l, we directly incorporate the cleaning time required and all other mandated setup time (collectively denoted $\hat{\upsilon}_{ilt}$) into the total processing time of task i in unit l when first scheduled at time t. The total processing time for a task i in unit l at time $\hat{t}_l$ is then equivalent to $\sum_{n=1}^{NB_{il}} PL_{inl} + \hat{\upsilon}_{il\hat{t}_l}$.

Further, if a different task $\hat{i}$ is scheduled by the policy at time, $\bar{t}$, in unit l, before the end of the production campaign for task i (i.e. where $W_{il\hat{t}_l}(\sum_{n=1}^{NB_{il}} PL_{inl} + \hat{\upsilon}_{il\hat{t}_l}) + \hat{t}_l > \bar{t} > \hat{t}_l$), then the task indicator $W_{il\hat{t}_l} = 0$ is redefined.

Similarly, if a production campaign of task i successfully terminates at time t in unit l, then: $W_{il\hat{t}_l} = 0$; $\mathbb{M}_l = \{i\}$; $t_{ml} = t$; and, the task is added to a list of completed

orders, $\mathbb{T}_f = \mathbb{T}_f \cup \{i\}$. In both cases redefinition of the binary variable is retroactive to the initial scheduling decision.

Next, we update a representation of 'lifting variables' Gupta and Maravelias (2017) that indicate current unit operations:

$$w_{lt+1} = u_{lt} \quad \forall l \in \mathbb{L} \tag{D.11}$$

where $u_{lt} \in \mathbb{Z}_+$ is the scheduled decision at the previous time index. This is deemed a necessary part of the state representation, given that it provides information regarding the operational status of the available equipment at the current time index. For example: if $u_{lt} = N + 1$, then the unit is idle; if $0 \leq u_{lt} < N + 1$, then it is not. A control must be predicted at each discrete time index. Typically, the length of a production campaign will be greater than the duration of discrete time interval, dt. To handle this if $u_{lt} = u_{lt-1}$ it is deemed that the scheduling element is indicating to continue with the current operation in unit l.

To determine the amount of processing time left for a given campaign in a unit l, we define:

$$\begin{aligned} A_{lt} &= \sum_{i \in \mathbb{I}_L} W_{ilt} \left(\sum_{n=1}^{NB_{il}} PL_{inl} + \hat{v}_{ilt} - \delta_{lt} \right) + \delta_{lt} \\ \delta_{lt+1} &= A_{lt} - 1 \quad \forall l \in \mathbb{L} \end{aligned} \tag{D.12}$$

The formulation assumes that we know the processing time required for batch n of task i in unit l, $PL_{i,n,l}$, ahead of its realization. In reality it is treated as a random variable, which is only observed when a given batch in a production campaign terminates. This is discussed further in Section D.2.3. The final equation comprising the system updates the number of discrete time indices until the order i is to be completed for delivery to the client, ρ_{it}:

$$\rho_{it+1} = \begin{cases} [\rho_{it}]^-, & \text{if } i \in \mathbb{T}_f \\ \rho_{it} - 1, & \text{otherwise} \end{cases} \quad \forall i \in \mathbb{I} \tag{D.13}$$

where $[y]^- = \min(0, y)$. The use of the logic condition enforced in Eq. D.13 is essentially redundant from the point of view of providing information about completed tasks, given that this is equivalently expressed by the inventory representation in the state vector. However, it is particularly useful in the allocation of rewards as it provides means to easily allocate penalty for tardy completion of tasks as expressed in Eq. D.7.

D.2.3 A forecasting framework for handling future plant uncertainty

In this study, the processing time of a batch, PL_{inl}, and the due date of the client orders, τ_i, are described by uniform and poisson distributions, respectively. This is detailed bu Eq. D.14a and D.14b:

$$PL_{inl} \sim U(\max(1, \bar{PT}_{il} - c), \bar{PT}_{il} + c) \quad \forall n \in \{1, \ldots, NB_{il}\}, \quad \forall l \in \mathbb{L}_i, \quad \forall i \in \mathbb{I} \tag{D.14a}$$

where $c \in \mathbb{Z}_+$ defines the variance of the distribution ($c = 1$ in this work); and, $\bar{PT}_{il}$ is the expected processing time for a batch of task i in unit l. The due date uncertainty is described as follows:

$$\tau_i \sim P(\bar{\tau}_i) \quad \forall i \in \mathbb{I} \tag{D.14b}$$

where $\bar{\tau}_i$ is the expected due date. In practice, we do not observe the realizations of these variables in advance, and hence maintain estimates according to their expected values within the state representation (i.e. Eq. D.6). In the case of the due date, the client is required to confirm the delivery date two days before the order - at which point τ_i is observed and updated within the state. Whereas the true processing time is only observed at the end of the operation.

D.2.4 Defining the initial system state

Eq. D.9 - D.13 represent the discrete-time plant dynamics (i.e. Eq. 5.1) that we are concerned with. To initiate the system, it is necessary to define an initial state, $\mathbf{x}_0$, that represents the plant at the start of the scheduling horizon, $t = 0$. This is described by the following set of expressions:

$$I_{i0} = 0 \quad \forall i \in \mathbb{I} \tag{D.15a}$$

which assumes that at the start of the horizon, the plant holds no inventory of the products one desires to produce;

$$w_{l0} = N + 1 \quad \forall l \in \mathbb{L} \tag{D.15b}$$

denotes that at the start of the horizon all units are idled. Even if this is not the case (i.e. the unit is unavailable for other reasons, such as production of another task which

it began before the start of the scheduling horizon), the allocation appears satisfactory according to experiments that follow in Section 5.5;

$$\delta_{l0} = (\text{RTU})_l \quad \forall l \in \mathbb{L} \tag{D.15c}$$

Eq. D.15c assumes that the unit is required to be idled for at least a period equivalent to its release time. At the start of the horizon, the number of discrete time indices until a task is due to be delivered to a client is equivalent to the task due date:

$$\rho_{i0} = \bar{\tau}_i \quad \forall i \in \mathbb{I} \tag{D.15d}$$

It should also be noted that at $t = 0$, $\mathbb{M}_l = \{\}$ $\forall l$ and $\mathbb{T}_f = \{\}$.

Having provided detail of the state, controls space and plant dynamics, we now turn our attention to identification of the feasible controls.

D.2.5 Defining the set of feasible controls

We seek to identify the set of controls $\hat{\mathbb{U}}_t \subset \mathbb{U} \subset \mathbb{Z}^{n_u}$, which innately satisfy the constraints imposed on the scheduling element.

Here, we define a number of sets that enable us to identify $\bar{\mathbb{U}}_t$:

1. A given unit l can process a subset, $\mathbb{I}_l \subseteq \mathbb{I}$ of the tasks that require processing.

2. If unit l has just processed task i then there exists a set of tasks which may be processed subsequently, $\mathbb{SU}_{il}$.

3. There exists a set, $\mathbb{O}_l = \{i\}$, which describes the task currently being processed in unit l. At $t = 0$, $\mathbb{O}_l = \emptyset$. If the unit is idled, then $\mathbb{O}_l = \emptyset$.

4. As previously mentioned there exists a set, $\mathbb{T}_f$, descriptive of those tasks, which have already been processed successfully. At $t = 0$, $\mathbb{T}_f = \emptyset$.

5. Lastly, again, there exists a set, $\mathbb{M}_l$ descriptive of the task most recently processed in unit l. Likewise, at $t = 0$, $\mathbb{M}_l = \emptyset$.

It follows then that any given time in processing, one can define the set of controls

that partially satisfy the imposed operational rules as:

$$\bar{\mathbb{U}}_t = \bigcup_{l=1}^{n_u} \begin{cases} (\mathbb{I}_l \cup \{N+1\}) \backslash \mathbb{T}_f & \text{if } \mathbb{O}_l = \emptyset \ \& \ \mathbb{M}_l = \emptyset \\ ((\mathbb{I}_l \cup \{N+1\}) \cap \mathbb{SU}_{ml}) \backslash \mathbb{T}_f \text{ where, } m \in \mathbb{M}_l & \text{if } \mathbb{O}_l = \emptyset \ \& \ |\mathbb{M}_l| = 1 \\ \mathbb{O}_l & \text{if } \mathbb{O}_l \neq \emptyset \end{cases} \tag{D.16}$$

Note, however, in this instance we cannot innately satisfy the control set $\hat{\mathbb{U}}_t$, as $\hat{\mathbb{U}}_t \subset \bar{\mathbb{U}}_t$. Specifically, $\bar{\mathbb{U}}_t$ permits the scheduling of the same task i in two different units l and l' at the same time index t. We propose to handle this through use of a penalty function, which we define as:

$$\begin{aligned} \phi &= R - \kappa_g \left\| [g(\mathbf{x}, \mathbf{u})]^+ \right\|_2 \\ g(\mathbf{x}, \mathbf{u}) &= [g_1, \ldots, g_N] \\ g_i &= \sum_{l \in \mathbb{L}_i} \bar{W}_{ilt} - 1 \end{aligned} \tag{D.17}$$

where $\kappa_g = 250$ and $\bar{W}_{ilt} \in \mathbb{Z}_2$ is a lifting variable that indicates unit l is processing task i at time t, such that if $u_{lt} = i$, then $\bar{W}_{ilt} = 1$ and, otherwise $\bar{W}_{ilt} = 0$. Note that technically, under the discrete-time state space model formulation used in this work (i.e. Eq. D.11), this could also be considered as a state inequality constraint. However, due to the uncertainty associated with the evolution of the state, this formalism will be explored in future work. By deploying the methodology proposed in Section 6.3, but modifying the rounding policy such that it is defined $f_r : \mathbb{W} \to \bar{\mathbb{U}}$, we can ensure that the decisions of the policy function, π satisfy the constraints imposed on the scheduling problem (see D.2.1) as originally proposed in Cerda et al. (1997) by maximising the *penalised return* defined under ϕ.

D.3 Definition of experimental data used in computational experiments

This section details the data used to define the case studies investigated in Section 5.5. Please see Tables D.2, D.3, D.4, D.5, D.6 for information regarding: the feasible processing of tasks in units and respective maximum batch sizes; nominal processing times; the viable successors of a given task; the cleaning times required between

successive tasks; and information regarding order sizes, due dates and release times, respectively.

Table D.2: Maximum task batch size (kg/batch) for every unit. RTU* denotes the finite release time of the unit in days. The length of a discrete time index corresponds to 0.5 days.

	Unit, l			
Task (order)	1	2	3	4
T1	100			
T2			210	
T3	140		170	
T4		120		
T5		90		130
T6	280	210		
T7			390	290
T8				120
T9	200			
T10	250	270		
T11			190	
T12			140	150
T13	120		155	
T14		115		
T15		130		145
RTU*	0.0	3.0	2.0	3.0

Table D.3: Order processing times (days/batch), PT_{il}. The length of a discrete time index corresponds to 0.5 days.

	Unit, l			
Task (order), T$_i$	1	2	3	4
T1	2.0			
T2			1.0	
T3	1.0		1.0	
T4		1.5		
T5		1.5		1.0
T6	2.5	2.0		
T7			1.0	1.5
T8				2.0
T9	1.5			
T10	2.5	2.0		
T11			1.5	
T12			2.0	1.5
T13	3.0		1.0	
T14		2.5		
T15		1.0		2.0

Table D.4: Set of feasible successors.

Task (order)	Feasible Successors
T1	T6, T9, T10, T13
T2	T3, T11, T13
T3	T1, T2, T7, T9, T10, T11, T12
T4	T5, T10, T14, T15
T5	T4, T6, T7, T8, T12, T14
T6	T1, T3, T4, T9, T13, T14, T15
T7	T2, T5, T8, T12
T8	T7, T11, T12, T15
T9	T1, T3, T6, T10, T13
T10	T1, T3, T4, T9, T13, T14, T15
T11	T2, T12, T13
T12	T3, T5, T7, T8, T11, T15
T13	T1, T2, T3, T6, T7, T9, T10, T11, T12
T14	T4, T5, T6, T10, T15
T15	T4, T6, T7, T8, T10, T12, T14

Table D.5: Cleaning times required between pairs of orders (days) in all units. The length of a discrete time index corresponds to 0.5 days.

	Succeeding Task														
Task (order)	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
T1						0.5			1.0	0.5			1.5		
T2			1.0								1.0		1.5		
T3	1.0	0.5					0.5		1.5	0.5	1.0	2.0			
T4					0.5					0.5				2.0	1.0
T5				0.5		0.5	1.0	0.5				0.5		0.5	
T6	1.5		0.5	0.5					1.0				0.5	1.0	1.5
T7		2.0			1.0			0.5				1.0			
T8							1.5				0.5	0.5			1.5
T9	2.0		1.0			0.5				1.5			3.0		
T10	1.0		0.5	0.5	1.0				2.5				0.5	2.0	1.0
T11		1.0										0.5	2.5		
T12			1.0		1.5		2.0	1.0			0.5				1.0
T13	1.5	0.5	2.0			2.0	2.5		0.5	0.5	1.0	1.5			
T14				0.5	0.5	0.5				0.5					0.5
T15				0.5		0.5	0.5	0.5		1.5		0.5		0.5	

Table D.6: Order sizes, due dates (days) and release times. The length of a discrete time index corresponds to 0.5 days.

Task (order) T_i	P1 Order Size, M_i (kg)	P2 Order size, M_i (kg)	Due date, $\bar{\tau}_i$ (days)	Release time of order, $(RTO)_i$ (days)
T1	700	700	10	0
T2	1050	850	22	5
T3	900	900	25	0
T4	1000	900	20	6
T5	650	500	28	0
T6	1350	1350	30	2
T7	950	950	17	3
T8	850	850	23	0
T9		450	30	2
T10		650	21	6
T11		300	30	0
T12		450	28	1.5
T13		200	15	0
T14		700	29	0
T15		300	40	5.5

D.4 Results for distributional formulation: problem instance 2

In this section, we present the results from investigation of experiments D3-8. The difference between these experiments and E3-8, is that now we are interested in optimizing for a different measure of the distribution, $p_\pi(z)$. In E3-8, the objective was optimisation in expectation, μ_Z. In D3-8, the objective is optimisation for the expected cost of the worst 20% of the penalised returns, i.e. Eq. 5.9a, with $\beta = 0.2$. Again, the optimisation procedure associated with the proposed method utilises a sample size of $n_I = 50$. All policies were then evaluated for 500 MCs.

Table D.7: Results for distributional RL from experimental conditions detailed by Table 5.1. Results that are emboldened detail those policies that show improved CVaR over the expected RL formulation (as detailed in Table 5.4).

Method	Reference	μ_Z	σ_Z	$\bar{\mu}_\beta$	F_{LB}
Proposed	D3	-116.0	10.4	-130.6	1.0
	D4	-151.8	10.2	**-166.6**	0.98
	D5	-123.4	18.2	-150.2	1.0
	D6	-166.5	22.2	**-198.7**	1.0
	D7	-127.1	20.3	-152.9	0.99
	D8	-167.0	23.7	**-199.9**	0.98

D.5 Misspecification of plant uncertainty

The processing time uncertainty has the same form as Eq. D.14a, but with misspecification of the parameters of the distribution by an additive constant, $k_{pt} \in \mathbb{Z}_+$. In this case, the actual plant processing time uncertainty, Eq. D.14a, is redefined:

$$PL_{inl} \sim U\big(\max(1, \bar{PT}_{il} - \hat{c}), \bar{PT}_{il} + \hat{c}\big), \quad \forall n \in \{1, \ldots, NB_{i,l}\}, \quad \forall l \in \mathbb{L}_i, \quad \forall i \in \mathbb{I}$$

where $\hat{c} = c + k_{pt}$ (in this work $c = 1$). Similarly, the rate of the Poisson distribution descriptive of the due date uncertainty (Eq. D.14b) is misspecified. Here, however, the misspecification is treated probabilistically, such that the rate, $\bar{\tau}_i, \ \forall i \in \mathbb{I}$, is perturbed by a relatively small amount. Specifically, we redefine the due date uncertainty of the real plant as follows:

$$\tau_i \sim P(\hat{\tau}_i), \quad \forall i \in \mathbb{I}$$

where $\hat{\tau}_i = \bar{\tau}_i + k_{dd} z_i,\ z_i \sim U(-1, 1)\ \forall i \in \mathbb{I}$ and $k_{dd} \in \mathbb{Z}_+$. Details of k_{pt} and k_{dd} investigated in this work are provided by Table 5.6 in Section 5.5.5.

D.6 The probability of constraint satisfaction

In this work, we are interested in satisfying hard constraints on the scheduling decisions (control inputs) to a plant. In the case that the underlying plant is subject to uncertainties, we evaluate this constraint probabilistically. This means we are interested in quantifying the probability, F_U, that the constraints on the control inputs, $\mathbf{u}_t \in \hat{\mathbb{U}}_t,\ \forall t$ are satisfied:

$$F_U = \mathbb{P}\Big(\bigcap_{i=0}^{T-1} \{\mathbf{u}_i \in \hat{\mathbb{U}}_i\} \Big) \tag{D.18}$$

In order to estimate F_U, we gain empirical approximation, F_{SA}, known as the empirical cumulative distribution function, by sampling. In this work, $n_I = 500$ Monte Carlo samples were used to estimate F_{LB} in the results reported as follows:

$$F_{SA} = \frac{1}{n_I} \sum_{i=1}^{n_I} Y^i \tag{D.19}$$

where Y^i is a random variable, which takes a value of $Y^i = 1$, if $\mathbf{u}_t \in \hat{\mathbb{U}}_t, \forall t$, and $Y^i = 0$ if $\exists\ \mathbf{u}_t \notin \hat{\mathbb{U}}_t, \forall t$. Due to the nature of estimation from finite samples, the F_{SA} is prone to estimation error. Hence, we employ a concept from the *binomial proportion confidence interval* literature, known as the Clopper–Pearson interval (CPI). The use of the CPI helps to ensure the probability of joint satisfaction with a given confidence level, $1 - \upsilon$. This is expressed by Lemma 4, which is recycled from Mowbray et al. (2022a).

Lemma 4 ***Joint chance constraint satisfaction via the Clopper-Pearson confidence interval****: Consider the realisation of F_{SA} based on n_I independently and identically distributed samples. The lower bound of the true value F_{LB} may be defined with a given confidence $1 - \upsilon$, such that:*

$$\begin{aligned} &\mathbb{P}(F_U \geq F_{LB}) \geq 1 - \upsilon \\ &F_{LB} = 1 - \mathit{betainv}(\upsilon, n_I + 1 - n_I F_{SA}, n_I F_{SA}) \end{aligned} \tag{D.20}$$

where $\mathit{betainv}(\cdot)$ is the inverse of the Beta cumulative distribution function with parameters $\{n_I + 1 - n_I F_{SA}\}$ and $\{n_I F_{SA}\}$.

Appendix E

Appendices for research objective: Distributional reinforcement learning for optimisation of multi-echelon supply chains

E.1 Simulated annealing

Simulated annealing is a stochastic search algorithm proposed in Kirkpatrick et al. (1983b), inspired by the solid annealing process. The SA algorithm starts from a high initial temperature, and with the continuous decrease of temperature parameters, it can jump out of the local optimal solution and finally approach the global optimal solution. The basic steps of the SA we present in Algorithm E.1 are as follows:

Step 0: Initialization: initial temperature T (sufficiently large), initial solution state θ (the starting point of the algorithm), the number of iterations N_{iter} for each T.

Step 1: Generate a new solution $\theta^{'}$.

Step 2: Calculate the increment $\Delta T = F_{\text{fitness}}(\theta) - F_{\text{fitness}}\ (\theta')$, where $F_{\text{fitness}}(\theta)$ is the evaluation function. If $\Delta T < 0$, θ' is accepted as the new current solution, otherwise, θ' is accepted as the new current solution with the probability of $\exp(-\Delta T/\text{T})$.

Step 3: If the termination condition is satisfied, output the current solution as the optimal solution, and the program is terminated. Decrease T gradually, go to step (1).

Algorithm E.1 Simulated Annealing

Input: Temperature T; initialization θ; upper bound θ_{UB} and lower bound θ_{LB} on the search space; cooling rate C; and, the number of iterations N_{iter}

for $k = 1, \ldots, N_{\text{iter}}$ **do**

1. Generate a new solution θ'

2. If $\Delta T > 0$ & $\exp\left(-\frac{\Delta \text{T}}{\text{T}}\right) < U(0,\ 1)$: $\theta^{k+1} = \theta^k$; **else**: $\theta^{k+1} = \theta'$

3. $\text{T} = \text{T} \times C$

4. $k := k + 1$

end for

Output: Final solution θ^* and objective function value $F_{\text{fitness}}(\theta^*)$

E.2 Evolution strategy

The two common selection schemes for evolution strategy (ES) algorithms are $(\mu+\lambda)$-ES and (μ, λ)-ES (Schwefel and Rudolph, 1995; Hansen et al., 2015). In this work, we implement an adaptive $(\mu + \lambda)$-ES, which is shown in Algorithm E.2.

Algorithm E.2 Adaptive $(\mu + \lambda)$-ES

Input: Number of parents μ; number of kids λ; $k := 0$; and the number of iterations N_{iter}; upper bound θ_{UB} and lower bound θ_{LB} on the search space;

1. Initialize parameters θ and mutation strength of μ individuals

for $k = 1, \ldots, N_{\text{iter}}$ **do**

2a. Crossover: randomly recombine individuals from parents μ to produce offspring λ.

2b. Mutation: apply mutation operator to the λ kids.

2c. Evaluate the $\mu + \lambda$ individuals.

2d. Select μ individuals for survival.

2e. $k := k + 1$

end for

Output: Optimal solution θ^* and objective function value $F_{\text{fitness}}(\theta^*)$

E.3 Particle swarm optimisation

PSO algorithm is a new parallel meta-heuristic algorithm first proposed by Kennedy and Eberhart (1995b). The particles in the algorithm determine the search pattern according to their own experience and the experience of other particles, and constantly approach the optimal solution until the termination condition is reached. Here we implement a PSO algorithm with shrinkage factor (Clerc, 2010). Each particle i is first randomly initialized with a position vector $\theta_i = [\theta_{i,1}, \ldots, \theta_{\text{i,D}}]$ and velocity vector

$v_i = [v_{i,1}, \ldots, v_{\mathrm{i,D}}]$ in D-dimensional space. The population is defined by a number of particles N. All particles update their own speed according to two extreme values: one is the individual extreme value (θ_i^{b}); the other is the global extreme value (θ^*). The update formula of the position vector and velocity vector of all particles is as follows:

$$\omega = \frac{2}{\left|2 - (r_1 + r_2) - \sqrt{(r_1 + r_2)^2 - 4(r_1 + r_2)}\right|} \tag{E.1a}$$

$$v_i^{k+1} = \omega[v_i^k + c_1 r_1 \left(\theta_i^{\mathrm{b}} - \theta_i^k\right) + c_2 r_2(\theta^* - \theta_i^k)] \tag{E.1b}$$

$$\theta_i^{k+1} = \theta_i^k + v_i^{k+1} \tag{E.1c}$$

where ω is a shrinkage factor, which determines how much the speed of the last iteration is reserved. It is one of the important parameters of PSO, the global search ability and local search ability of the algorithm can be balanced by adjusting it. c_1 and c_2 are the learning factor of the algorithm. It is generally believed that a larger c_1 will make all particles linger too much in the local area, which is not conducive to the global search of the algorithm. However, a larger $c_2$will make the particles fall into the local optimal value prematurely and reduce the accuracy of the solution. r_1 and r_2 are random numbers between 0 and 1. The algorithm is also outlined in Alg. E.3.

Algorithm E.3 Particle Swarm Optimisation

Input: Initialize the number of particle N, upper bound θ_{UB} and lower bound θ_{LB} on the search space, maximum velocity $v_{\max} = [v_{max,1}, v_{max,2}, \ldots v_{max,D}]$, individual acceleration c_1 and global acceleration c_2.

for each particle $i = 1, ..., N$ **do**

1a. Initialize the position θ_i and velocity v_i of particles i.

1b. Evaluate particle i and set $\theta_i^{\mathrm{b}} = \theta_i$

end for

2. $\theta^* = \arg\max_i F_{\mathrm{fitness}}(\theta_i^{\mathrm{b}})$

for $i = 1, ..., N$ **do**

3a. Update the velocity v_i and position θ_i of particle i.

3b. Calculate the value of individual fitness function.

3c. If $F_{\mathrm{fitness}}(\theta_i) > F_{\mathrm{fitness}}(\theta_i^{\mathrm{b}})$ **then** $\theta_i^{\mathrm{b}} = \theta_i$.

3d. If $F_{\mathrm{fitness}}(\theta_i) > F_{\mathrm{fitness}}(\theta^*)$ **then** $\theta^* = \theta_i$

3e. If termination criterion satisfied **then** break.

end for

Output: Optimal position θ^* and objective function value $F_{\mathrm{fitness}}(\theta^*)$

E.4 Artificial bee colony

The ABC algorithm was proposed in Karaboga (2005), inspired in the intelligent behavior of honey bee colonies. Since bees generate new candidate solutions by updating the random dimension vector of their parent solutions, this search pattern is good at exploration but poor at exploitation (Gao and Liu, 2012).

It is assumed that the dimension of solving the problem is D and the number of particles is N. The position of honey source i is expressed as $\theta_i^k = [\theta_{i1}^k, \theta_{i2}^k, \dots \theta_{\mathrm{iD}}^k]$ in k iterations. The initial position of honey source i is randomly generated in the search space. Employed bees search around the honey source i based on Eq. E.2 to produce a new honey source:

$$\theta_i^{k+1} = \theta_i^k + \varphi(\theta_i^k - \theta_j^k) \tag{E.2}$$

where $j \neq i$, φ is uniformly distributed random number. Then onlooker calculate the selection probability matrix based on Eq. E.3, then use the roulette selection method to choose one of their sources and retain the honey source by greedy selection.

$$M = \frac{1}{N}\sum_{i}^{N} \mathrm{fit}_i \tag{E.3a}$$

$$F_i = e^{\frac{\mathrm{fit}_i}{M}} \tag{E.3b}$$

$$P_i = \frac{F_i}{\sum_{i=1}^{N} F_i} \tag{E.3c}$$

where fit_i is the fitness value of honey source i.

During the search, if the honey source θ_i does not get a better honey source after k_{limit} iterations, the honey source θ_i^k will be abandoned and the corresponding employed bees will be transformed into scout bees. Scout bees will randomly generate a new honey source instead of θ_i in the search space.

Algorithm E.4 Artificial bee colony search routine

Input: The number of iteration $\underline{\mathrm{N}_{\mathrm{iter}}}$, , upper bound θ_{UB} and lower bound θ_{LB} on the search space, the number of particles $\underline{\mathrm{N}}$;

1. Initialize the food source memory θ_i

for $k = 1, \ldots, N_{\mathrm{iter}}$ **do**

2a. Employed bees set out to find the next honey source and make greedy choices.

2b. Onlooker calculate the selection probability matrix, then execute the roulette selection method to choose one of their sources, and repeat the greedy selection.

2c. Scout bees conduct a domain search to find possible new food sources.

2d. Remember the location of the best food source available.

e. $k := k + 1$

end for

Output: The best food source θ and objective function value $F_{\mathrm{fitness}}(\theta)$

E.5 Recurrent Neural Network

Recurrent neural networks (RNNs) have previously been used as a solution to modeling (Su et al., 1992) and control law parameterization (Mnih et al., 2014). The RNNs generates an output at each time step and has a recursive connection between the hidden units. This enables them to exploit information from the previous states, a feature that allows RNNs to be successfully applied to partially observable problems and belief MDPs. In this work, we use RNNs to parameterize the policy $\pi\left(\mathbf{x}_t; \mathbf{h}_{t-1}, \theta\right)$. A detailed representation of a simple RNN is depicted in Fig. E.1. The RNNs maintains a hidden state $\mathbf{h}_t$, that is an abstraction of the information provided by the previous sequence of states. This hidden state is updated at each time index based on new information provided by the current state, which is then fed forward through the activation function $\sigma\left(\cdot\right)$ in the network to ultimately influence the control prediction of the network. Here $\sigma\left(\cdot\right)$ is the Tanh function. Similar to general artificial neural networks, the parameters of RNNs can be optimised via standard tools and algorithms.

In this work, we build a neural network with two hidden layers. The first hidden layer uses RNNs, and the second hidden layer uses a fully connected feedforward layer. Each case study has a slightly different neural network structure with different number of parameters. The detailed structure of each neural network and the number of their parameters are presented in Table E.1. In order to facilitate neural network learning and mitigate the effects of mathematical bias in the state representation, we

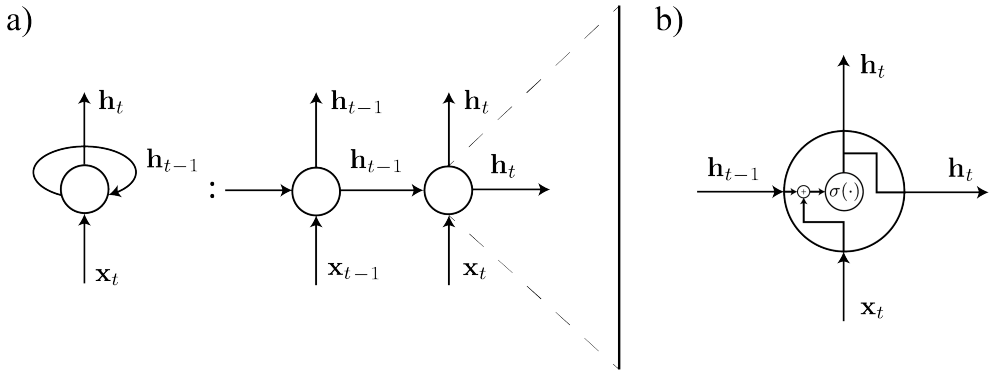


Figure E.1: Recurrent neural networks as a) unfolded computational graph and b) hidden node.

standardize all data so that the input data are of the same order of magnitude. The metrics used for standardization are fixed during training, and estimated from the range of state variables.

Table E.1: Recurrent neural network detailed structure and number of parameters.

Hyperparameter	Virtual Machine Packing	Asset Allocation	Supply Chain
Input layer nodes	153	7	33
Hidden recurrent layer 1 nodes	10	10	10
Hidden recurrent layer 2 nodes	5	5	5
Output linear layer nodes	1	3	3
Number of parameters	1711	263	523

E.6 Results of sensitivity analysis

Table E.2: Results of sensitivity analysis for different parameters.

		Experiment Number								
		1	2	3	4	5	6	7	8	9
Training Parameters	Particles	40	60	80	100	80	80	80	60	60
	CVaR bound	380	380	380	380	360	400	420	380	380
	Sample Size	60	60	60	60	60	60	60	10	30
Policy Training	Expectation	427.9	430.2	428.8	438.9	432.3	424.1	410	451.2	436.2
	Std. Dev.	16.3	14.9	14.2	21.4	21.8	13	17.5	13.2	20.1
	CVaR	394.2	398.7	397.6	394.1	387.0	400.1	381.5	425.9	396.4
Policy Validation	Expectation	423.5	425.9	425.0	430.5	430.1	422.4	407.4	430.8	430.4
	Std. Dev.	18.4	18.4	16.4	24.3	22.5	16.3	21.3	27.8	24.8
	CVaR	385.1	390.1	392.4	381.5	384.1	389.8	364.6	378	381.5