Automated Planning with Hybrid Domain Models: A Method to Improve Continuous Process Descriptions

Submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

by

Rubiya Yasmin Reba

University of HUDDERSFIELD

September, 2021

Acknowledgement

"Praise is to Allah by Whose grace good deeds are completed"

(Sunan Ibn Majah Hadith No. 3803)

All praises are due to Almighty Allah for making it possible for me to complete my Ph.D. programme. I would like to acknowledge the contribution of my parents for their continuous trust and support in my life. I pray Allah grant them longer life in good health and faith. My special appreciation goes to my sisters, brother-in-laws, brother and sister-in-law for loving and caring me so much.

I would like to acknowledge the help of my supervisor Professor T. L. McCluskey for his guidelines, suggestions and encouragement during the challenging moments of my PhD life. My final acknowledgement goes to my dear friend Rabia Jilani and her sister Aisha Jilani for their hospitality, generosity and advice.

Abstract

Recent advances in Automated Planning not only involve the improvements in planning efficiency but also the enhancement in granularity in which planning domains are modelled. A significant progression is in the move from discrete domain models to mixed discrete-continuous models i.e. hybrid domain models. While planning with hybrid domains has been studied for decades, the knowledge engineering of those domain models is still a challenge, particularly for real-time complex domains. It is imperative to understand how to effectively and efficiently formulate the planning models to achieve maximum productivity with minimum wasted effort or cost. One of the main engineering challenges of hybrid domain models involves encoding the frequent fluctuation of underlying processes with continuous updates in the world state. The occurring numerical changes with the variation of parameters can be too complex to be formulated accurately by human manual efforts.

This thesis proposes a method utilising machine learning techniques which results in the formulation of a run-time representative estimation of continuous changes in varied process parameters. The method incorporates statistical analysis to acquire process models from real-world data for hybrid planning domains. We assume that domain knowledge has been already encoded in an initial hybrid domain model (in this thesis, that is a PDDL+ model). We then use the method to create an improved process model (within the same encoding language of PDDL+) which is embedded into the process specification of the original, pre-engineered domain model.

By exploiting the quantitative data from hybrid planning domains, firstly the proposed approach, with the help of statistical methods, identifies the associations (i.e. dependencies or inter-dependencies) between a single outcome and single/multiple predictor numeric variables in the underlying process. Based on the deduced statistical relationship among variables, the appropriate linear regression technique with corresponding statistical tests are nominated and implemented to formulate the process model. Then the constructed process model is embedded/adjusted into the pre-engineered hybrid do-

main models. The learned process model, in the form of a mathematical function, automatically approximates/adjusts the quantity of an outcome variable with the continuous variations in different predictor features in order to efficiently and accurately control the corresponding process in hybrid planning domains.

To empirically evaluate our approach, we utilise pre-engineered models (in PDDL+) of an Urban Traffic Control (UTC) domain and a Coffee domain. For the UTC domain, we experiment with the real-time traffic data that is collected from AIMSUN simulator utilised in the SimplifAI project (McCluskey, Vallati and Franco 2017). Besides, for coffee domain, we collect the real-time data from an observational study that is conducted by Easthope (2015). The evaluation results demonstrate that the automatically learned values of numeric process variables by our method are more rational than the formulation values of process variables declared statically in the original domain models. Besides, it reveals that the learned process models can provide more accurate simulation output, which can consequently lead to higher-quality plans. Along with that, by automatically identifying the effective process variables and removing the irrelevant ones from the learned process models, it can assist the knowledge engineering tasks of modelling/adjusting the dynamically changing process variables with their values in the hybrid planning domains, without declaring them statically.

Keywords: *Automated Planning, Knowledge Engineering, Machine Learning, Mixed Discrete-Continuous Representations, Process Models, Correlation Analysis, Regression Analysis*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

In this era of automation, Human life is surrounded by different types of intelligent systems which play a variety of roles in the society such as medical care, education, military operations, factory automation, business transaction, intelligent transportation, and household assistance etc. Intelligent systems employ Artificial Intelligence (AI) to perform tasks that generally require human intelligence like planning, learning, reasoning, problem solving, perception, and knowledge representation. Intelligent Autonomous Systems (IAS) is the study of developing such intelligent agents that can perceive the surrounding environment and response accordingly without human assistance. Planning and Learning are the major capabilities that such automated systems should have. Basically, Planning is a cognitive process which involves reasoning with knowledge about actions and change to generate strategies or plans to achieve a goal (Preece et al. 2015). Whereas, Learning is a process of getting new knowledge, making connections with the prior knowledge and transfer knowledge in order to improve performance (Ambrose et al. 2010).

Automated Planning (AP), also known as AI planning, is a pivotal branch of AI that synthesises plans or a series of actions. Through the plan execution, it guides the intelligent agents transforming the environment from initial state to a desired goal state. The abstract architecture of AP has logically separated into two parts: Planning Engine and Domain Model (McCluskey 2012) as shown in Fig 1.1. A planning engine (Planner) is a generic software system to automatically generate plans based on the given domain knowledge. A domain model as a formal specification of the planning problem, provides information of the application area facilitating the planner to apply techniques accordingly.

In general real-world planning applications require models representing both discrete and continuous changes to variables such as numeric resources for completing a task. Unlike the classical planner with discrete domain knowledge, the hybrid planner has the ability to handle both discrete changes and continuous transition with hybrid (mixed discrete-continuous) domain knowledge. The hybrid planning domain has additional

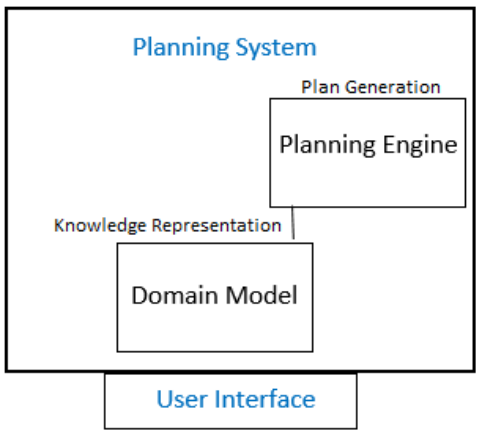


Fig. 1.1 Abstract architecture of planners (McCluskey 2012)

expressive features to model the time-dependent continuous processes and also the exogenous events that are occurred by the uncertain environment. It enables the hybrid planner to reason with external events and their interaction with continuously changing numeric values in order to generate real-time plans.

In AI, Machine Learning (ML) is considered as compulsory behaviour of an intelligent system for enabling the learning ability from experiences, observations or by direct instructions automatically. Based on the learning outcomes, the system can make better decisions or predictions, develop skills and improve actions accordingly without human or programmed instructions (Carbonell et al. 1983). There has been a long history of utilising ML techniques in Automated Planning (AP), with the intent of automatically extract, refine, organise and exploit the domain knowledge. Consequently, The acquired knowledge assists the planning agents to improve the planning performance, plan quality and domain theory (Zimmerman and Kambhampati 2003, Jiménez et al. 2012). Also, to some extent, it mitigates the knowledge engineering (KE) issues by automatically acquiring adequate domain models instead of manual encoding (Tate et al. 2012, Tate and Wickler 2015).

A domain model represents the conceptualised and formalised knowledge within a planning application as analysed, reasoned and deployed by the planners for plan generation. To describe the discrete and continuous behaviours with time constraints, a hybrid (planning) domain model supplies advanced modelling components over classical domains. Domain modelling languages for instance PDDL+ (Fox and Long 2002), ANML (Smith et al. 2008), $\int$-PDDL+ (Ramirez et al. 2017) are applied to encode hybrid domains according to the problem statement. The enhanced expressivity of such languages brings the opportunity of handling external events to support controllable uncertain situations. Besides it allows the time-dependent processes for ceaseless state update and

the action executions for instantaneous state transition. Recently, the PDDL+ modelling language has become more acceptable in the AI planning world to solve real-time problems. It has overcome the limitations of PDDL family by supporting the predefined exogenous events to avoid plan failures in the dynamic world (Fox and Long 2002). In addition, it can involve linear (Shin and Davis 2005, Coles et al. 2012, Coles and Coles 2014) and non-linear continuous changes (Penna et al. 2009, Bryce et al. 2015) along with the simultaneous changes on different numeric fluent or quantities by initiating and terminating the processes.

In Automated Planning (AP), Knowledge Engineering (KE) is the process of capturing and formulating the application knowledge, which are encoded into the domain model for use in the planning engines (McCluskey, Vaquero and Vallati 2017). The overall KE process involves the acquisition, formulation, validation and maintenance of domain knowledge. There are three basic KE methods that are followed in the planning community to build a domain model such as (1) coding manually, or simply hand-coded by domain/planning experts, known as Knowledge Engineers, (2) employing the UML based tool/method, e.g. itSIMPLE (Vaquero et al. 2007), and (3) deploying the hierarchical, object based tool, e.g. GIPO (Simpson et al. 2007). A Knowledge Engineer, having expert knowledge and experience in domain modelling languages, formulates the (planning) application knowledge and encodes them into the domain models using a text editor or any built-in editor e.g. PDDL Studio (Plch et al. 2012). The most commonly used KE method among them is (1) hand-coding by the Knowledge Engineers. Besides, the existing KE tools (mentioned in the methods 2 and 3) do not support the hybrid (planning) domains (e.g. the PDDL+ domain model) that can represent the real-time planning applications (Shah et al. 2013). Therefore the AP community depends on the skills of Knowledge Engineers in order to manually formulate/encode the hybrid domain model with PDDL+ representation.

Automated Planning with Hybrid Domains (APHD) has shown its eligibility to solve real-time planning problems, dealing with numeric and temporal constraints (Piotrowski et al. 2016, Scala et al. 2016). However, the set up and use of hybrid planning has great challenges in terms of (a) computational difficulty of solving planning problems and (b) engineering difficulty in creating the knowledge model of hybrid domains (McCluskey and Vallati 2017). To handle the dynamic behaviours of real-time planning applications, a hybrid planner depends on the domain knowledge to apply planning techniques accordingly. Therefore, the performance of a planner along with flexible configuration and technical reformulation as well as the plan quality are quite affected by the effective engineering of domain knowledge (Vallati et al. 2015). This thesis focuses on the area (b) to get more effective domain models in order to improve planner performance with

efficient planning outcomes. The thrust is to utilise ML techniques to induce process knowledge (i.e. continuous changes) in PDDL+ hybrid domains. In particular, this thesis exploits ML techniques and statistical methods to automatically learn the continuous time-dependent changes occurred in the real-time (planning) domains.

In this thesis, we propose a machine learning based approach with statistical analysis to automatically acquire process models (PM) from real-world data (i.e. quantitative time-series data) for hybrid planning applications. We assume that the domain knowledge has already been created and encoded in an initial hybrid domain model within a PDDL+ representation, and the learned process model will be integrated/adjusted into the process specification of pre-engineered hybrid domain model. The intent is to better represent the underlying process of already engineered hybrid models in terms of automatically approximating and adjusting the dynamically varying process parameters with their values, by utilising training data.
A process model (PM), in the form of mathematical function, represents the causal relationship between an outcome and predictor (one or more) numeric variables in the process of hybrid domains. It estimates the effects of continuously fluctuating predictors on outcome variable by means of continuous assignment expression. At first, our proposed approach, with the help of statistical methods, infers the associations among the outcome and predictor variables in the respective process from quantitative data. Based on the inferred relationship among process variables, the appropriate (linear) regression technique with corresponding statistical test is nominated and implemented in order to formulate the process model. Finally the induced PM is integrated/adjusted into the process description of pre-engineered hybrid planning domains. While investigating the automatic learning of process models, this thesis is distinguished in doing that in the context of the need to use learned domain model as the input to a plan generation engine, and therefore the learned process model must be created within the original language (in this thesis, that is PDDL+).

1.1 Motivation

The hybrid planning techniques have the potential to be used in real-world applications in which the successful AP operations are quite dependent on the effective knowledge engineering (KE) of domains. Although languages such as PDDL+ has been adapted to model real-time autonomous systems, it is still a challenge for the knowledge engineers to handle its expressivity with respect of process description. Some issues usually arise during the creation of domain models along with process knowledge engineering such as model quality (dynamic or static), accuracy, adequacy, consistency, correctness and completeness (McCluskey and Vallati 2017, McCluskey et al. 2016). The rationale to

learn the process knowledge for hybrid AI Planners are summarised below:

i **Difficulties in pre-engineering -** Modelling a domain requires Knowledge Engineering (KE) experts for this particular application, where all possible features should conform to the system requirements. While reasonable specifications of the model are formulated they may not be so accurate and need some dynamic testing. As an example, the process specification that simulates flow through one of the road junctions in the traffic domain is given below:

```
(:process flowrun_green
   :parameters (?p - stage ?r1 ?r2 - link)
   :precondition (and
      (active ?p)
      (> (occupancy ?r1) 0.0)
      (> (turnrate ?p ?r1 ?r2) 0.0)
      (< (occupancy ?r2) (capacity ?r2))
   )
   :effect (and
      (increase (occupancy ?r2) (* # t (turnrate ?p ?r1 ?r2)))
      (decrease (occupancy ?r1) (* #t (turnrate ?p ?r1 ?r2)))
   )
)
```

Fig. 1.2 *flowrun_green* process of Traffic Domain (McCluskey and Vallati 2017)

In Figure 1.2, the process flowrun green allows car to flow from road1 to road2 if the corresponding green is on. One of the preconditions to start this specific process is the road2 occupancy must be less than the road2 capacity. KE experts make some approximations for these two values based on the historical/real sensor data, then calculate the vehicle flow rate or turn rate accordingly. But there might be some better approximations for different cases (Figure 1.3) based on the road situations. Besides, the vehicle flow rate may also depend on other factors such as the density of intersections, the number of non-motor vehicles, the gradient of roads, the speed limit, the density of bus stops and the number of lanes (He and Zhao 2013). Those influencing factors usually vary in different road junction scenarios. Therefore, it is crucial to specify a general process model (e.g. *flowrun_green*) for all road links with different influencing factors (features). Consequently, it becomes a challenge for KE experts to identify effective features affecting the outcome process variable dynamically.

```
(:process flowrun_green
   :parameters (?p - stage ?r1 ?r2 - link)
   :precondition (and
      (active ?p)
      (> (occupancy ?r1) 0.0)
      (> (turnrate ?p ?r1 ?r2) 0.0)
      (< (occupancy ?r2) 8/10 (capacity ?r2))
   )
   :effect (and
      (increase (occupancy ?r2) (* # t (turnrate ?p ?r1 ?r2)))
      (decrease (occupancy ?r1) (* #t (turnrate ?p ?r1 ?r2)))
   )
)
```

(a) Approximation 1

```
(:process flowrun_green
   :parameters (?p - stage ?r1 ?r2 - link)
   :precondition (and
      (active ?p)
      (> (occupancy ?r1) 0.0)
      (> (turnrate ?p ?r1 ?r2) 0.0)
      (> (occupancy ?r2) 8/10 * (capacity ?r2) and
           (< (occupancy ?r2) 9/10 (capacity ?r2) )
      )
   )
   :effect (and
      (increase (occupancy ?r2) (* #t * 1/2 * (turnrate ?p ?r1 ?r2)) )
      (decrease (occupancy ?r1) (* #t * 1/2 * (turnrate ?p ?r1 ?r2)))
   )
)
```

(b) Approximation 2

Fig. 1.3 *flowrun_green* process with different approximations

ii **Hard to encode –** One of the main KE issues of hybrid domains is the modelling of process knowledge with dynamic variants. The components of process specification change continuously, so that it is more difficult in expressing complex relations/ hybrid representations than in classical planning. Succinctly, the underlying processes involve frequent fluctuations of numeric parameters with the continuous updates in the world state. Hence, it becomes difficult to encode the complex numeric changes by hand every single time it happens.

iii **Changes over time –** In this dynamic world, the process description may need to change over time to reflect the changing reality. For remodelling the processes, manual/static assumptions may not be accurate while only depending on the skills

of KE experts (Bryce, Benton and Boldt 2016, McCluskey, Vaquero and Vallati 2017). Therefore, it will be better if they are automatically learned and adjusted in the physical system.

iv **Existing Domain Learning Systems –** Machine Learning (ML) is considered to be an important area from which tools to assist the knowledge engineering (KE) process can be formed. These techniques can automatically extract, exploit, refine and adjust the domain knowledge such as model description, state variables, planning outcomes, or achievable goals in the planning system (Zimmerman and Kambhampati 2003, Jiménez et al. 2012). Empowering the KE approaches with ML techniques, can mitigate the occurring KE issues and human-made errors associated with the development, debug and maintenance of the complex domains (Tate and Wickler 2015, Tate et al. 2012). Despite the long history of applying ML in KE tools, some learning aspects still need to be explored, particularly learning with the time dimension for real-world domains (Arora et al. 2018). Besides, the existing automated knowledge engineering tools such as AMAN (Zhuo and Kambhampati 2013), LOCM (Cresswell et al. 2013), RIM (Zhuo et al. 2013), and NLOCM (Gregory and Lindsay 2016) do not support hybrid domains and need to broaden up their scopes to bring them into practical use (Jilani et al. 2014, Arora et al. 2018).

1.2 Aims and Objectives

This thesis aims to take an initiative to tackle the challenges of engineering hybrid (planning) domains in terms of modelling process descriptions (i.e. the continuous changes). One of the main challenges is to manually capture and encode the continuously changing quantities of varying (numeric) parameters in the underlying processes of real-world planning domains. This thesis focuses on utilising ML techniques to acquire the process knowledge defines in the hybrid (planning) domain, with the intent to automatically model/adjust the dynamically changing process parameters with their values, without declaring them statically/manually. The aim is to automatically refine and improve the hybrid domain model, which can enhance the simulation accuracy. By upgrading the simulation accuracy, the purpose of this thesis is to provide higher-quality plans that are applicable to the real world applications. Besides, by mitigating the knowledge engineering challenges, it can support the knowledge engineers to build more accurate and rational hybrid domain models in order to generate realistic plans. As a consequence, the hybrid planning engines will become more accessible, and ubiquitous in everyday life.

The proposed approach is to collect real/simulated data from the running of processes in the domain, and, utilising a pre-engineered hybrid domain model, use a method in-

corporating machine learning tactics from the collected data to induce an improved process description. It provides a process model with appropriate feature selection and efficient linear approximation as output. To effectively learn the process model, this thesis's emphasise is on applying statistical modelling using regression analysis in hybrid hypothesis space. A regression model is used to infer the relationship between two or more numeric variables that estimate the outcome variable based on the predictor features.

The main focus of this thesis is to improve the hybrid domain model in respect of **Dynamicity** and **Versatility**, by automatically inducing the process models. A PDDL+ process simulates continuous changes in the numeric variables that are initiated by changes in the world. Therefore, learning the process model from real-time data with PDDL+ formulation, will lead to a more rational (hybrid) planning model without having to declare the dynamic knowledge as static facts. Also, it will enhance the **Dynamicity** of the planning model by updating the learning data set in order to capture and adjust/integrate the changing requirements. Besides, by adding new predictor variables in the learning data set, it will enable the planning model to adapt other numeric features automatically in the process that will boost its **Versatility**. For example in figure 1.4, the outcome variable (y) may have relationship with other predictor variables (i.e. $X3$) in a process. In addition, the values of y and X may need to modify or update over time with the changes in dynamic states. On this account, it is required to update the existing parameter values, add new rows of values, or add new feature columns in the learning data set, so as to keep the process model up-to-date.

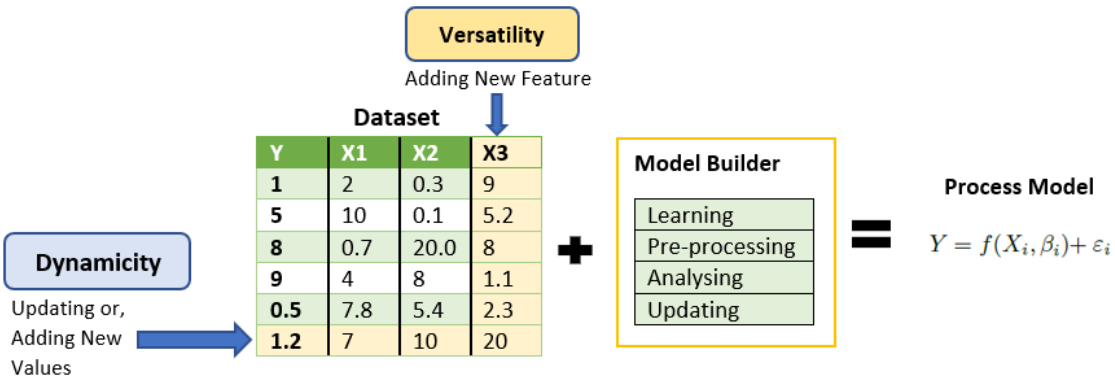


Fig. 1.4 Improve the Hybrid Domain model in terms of Dynamicity and Versatility

The primary objectives of this thesis are summarised below:

i **Automated acquisition of process model -** the Process Model (PM) will be learned and acquired automatically from the partial domain knowledge and input data. For instance, the UTC domain model (Appendix: C.1) in SimplifAI project includes three processes: *keepgreen* (keeps the green/intergreen on and updates

the greentime value), *flowrun_green* (allows car to flow if the corresponding green is on) and *keepinter* (calculate the timing between colour interchange). In the UTC domain, the most important / tricky process is the *flowrun_green*, where the other ones are a device to make the model work. Therefore, *flowrun_green* process will be experimented in order to embed the learned process model with the intent of estimating the traffic flow rate automatically.

The traffic flow, in terms of vehicles per hour, is the rate at which vehicles pass a given point on the roadway. In the urban road network, the vehicle flow rate is influenced by different factors e.g. the density of bus stops, the speed limit, cross flow, incoming and outgoing road saturation, which can vary according to the road infrastructure (He and Zhao 2013). The proposed approach induces Process Models, by exploiting real-time traffic data, that can estimate the vehicle turn rate for each road link with the variation in corresponding influencing factors. In that way, the automated acquisition of process models will enhance the **versatility** of hybrid (planning) domain model by automatically adopting/adjusting many different features in the PM.

ii **Refine the process knowledge -** the automatically acquired process models will modify the current domain knowledge (i.e. the process description) that will be applied in the hybrid planning engine. In that way, it will keep the application domain up-to-date and allow planners to make plans according to the environment, so as to achieve the **dynamicity** of the hybrid domain models. For example: in urban traffic network, the custom complex mathematical models with predefined policies and manual dealing process are not enough to cope with unpredictable situations such as road accidents, natural calamities, and road repairs. The learned process model embedded with UTC planning domain can automatically adjust the timings of signal phases according to the estimated traffic flow rate in a particular road situation.

iii **Automated configuration of process model for simulation and testing -** AI hybrid planning techniques with PDDL+ model can generate more efficient, optimised and goal-directed strategies automatically from real time sensor/historical data (Vallati et al. 2016). It involves manual changing in the PDDL+ representation for multiple times to simulate and test the data in different scenarios. This manual configuration procedure may contain errors which can be avoided with the automated acquisition of process model.

iv **Assist knowledge engineers in model construction -** By automatically learning the continuous fluctuation in underlying processes of hybrid planning domains, it will help to construct more accurate and adequate hybrid domain models than

hand crafted.

v **Improve plan quality -** it will assist the planning engine to make a large number of correct plans suitable for ongoing dynamic events. The induced process model will help the planner to adjust itself with global constraints and produce more improved plans.

vi **Bring into community use -** the main difference between classical and hybrid domain is the ability to modelling continuous change. The current automated knowledge engineering tools are not suitable to handle continuous dynamics of the domain. These tools has made the classical planners usable for common users. But modelling the hybrid domain is still time consuming and needs experts knowledge. Therefore, to make the hybrid planning engine more accessible and open up to general use, automated hybrid domain acquisition is required.

1.3 Thesis Contribution

This thesis contributes to the area of research into knowledge engineering for AI planning with real-world hybrid domains. The main contributions are highlighted as follows:

* To the best of our knowledge, this is the first work on learning continuous effects of processes within planner-acceptable formulations of hybrid planning domains. Besides, without declaring statically, it learns the dynamic values of numeric variables from real-world historical/sensor data and adjust them in the process effects automatically. In that way, it facilitates the knowledge engineering task of modelling processes with dynamically changing parameters in the real-world hybrid (planning) domains.

* It controls a process according to the dynamic fluctuations in corresponding process parameters that reflect the changing reality of real-world domains. By automatically handling the dynamicity/versatility of underlying processes in the hybrid (planning) domains, the proposed approach will support the knowledge engineers to construct realistic planning domains more efficiently.

* It approximates the more rational and pragmatic value of outcome variable in the running process that upgrades the simulation accuracy. Consequently, the improved simulation output aids the planning engine to produce high-quality plans.

* It automatically identifies the significant/ineffective process variables (i.e. numeric features) along with their interdependencies that is crucial for engineering/re-engineering the process knowledge with dynamically varying parameters. In

other words, the proposed approach can assist the knowledge engineers by automatically choosing effective process parameters and identifying/removing the irrelevant ones from the process models.

* We explore different linear regression techniques along with statistical methods that reveals their potentiality of facilitating knowledge engineering tasks. Besides, this thesis has shed some light on the knowledge engineering gaps in modelling hybrid domains with processes, which can be filled up to some extent by utilising the statistical analysis with regression techniques. The research community, specifically the planning community, will benefit from this thesis that proposes a new approach of modelling process effects/continuous changes in the numeric features (i.e. process parameters) for hybrid planning domains.

Following are the published works that have been achieved during the Ph.D. research:

□ **Workshop Papers**

Title: Acquiring Process Knowledge in Hybrid Planning Domains using Machine Learning
Authors: Rubiya Reba, Rabia Jilani, Alan Lindsay and Lee McCluskey
Workshop: The Knowledge Engineering for Planning and Scheduling Workshop (KEPS)
Location: Nancy, France
Date: June 14-19, 2020

Title: Acquiring Process Knowledge in Hybrid Planning Domains using Machine Learning
Authors: Rubiya Reba, Rabia Jilani, Alan Lindsay and Lee McCluskey
Workshop: The 35th Workshop of the UK PLANNING AND SCHEDULING Special Interest Group (UK PlanSIG 2020)
Location: Online
Date: 16th December 2020

□ **Conference Paper**

Title: Refining Process Descriptions from Execution Data in Hybrid Planning Domain Models
Authors: Alan Lindsay, Santiago Franco, Rubiya Reba, and Thomas L. McCluskey.
Book title: Proceedings of the International Conference on Automated Planning and Scheduling
Conference: The 2020 ICAPS conference

Location: Nancy, France
Date: October, 2020

1.4 Thesis Structure

This thesis is organised into eight chapters. The contents of each chapter are summarised below.

- **Chapter 1** provides a brief introduction of this thesis. It explains the aims and objectives of this research work. Besides, it discusses our motivation for this work. The contribution of this thesis in the knowledge engineering process are also highlighted in this chapter.

- **Chapter 2** explains the concept of Automated Planning with Hybrid Domains (APHD) that reflect the realistic hybrid systems in order to solve practical problems in real-world applications. It gives an overview of various domain modelling languages that are used to encode classical/temporal planning problems in Automated Planning. Besides, it discusses the modelling features with semantic and syntactic structure of PDDL+ language that are utilised to encode the hybrid planning domains. This chapter also demonstrate the potentiality of hybrid (planning) domains over classical domains in terms of producing realistic plans. Some state-of-the-art hybrid planners (i.e. planning engines) are also mentioned that can generate plans with PDDL+ domains. This chapter ends with an overview of some applications of APHD solving real-world problems.

- **Chapter 3** reviews some earlier and recent published works on learning heuristics and domain (action) models in classical/temporal planning domains by exploiting Machine Learning (ML) techniques. Also, it briefly explains the existing research works on implementing ML in hybrid planning domains. Some current approaches of inducing process models/learning continuous effects and their differences from our proposed approach are discussed. Finally it highlights the current research gap in the field of learning/refining the process knowledge for hybrid planning domains.

- **Chapter 4** introduces the process model (i.e. process modelling) that is implemented throughout this thesis. This chapter explores the statistical methods of data collection, data pre-processing/preparation and data analysis that have been utilised in this thesis. Besides, it investigates several ML techniques (i.e. linear regression) that have been exploited to formulate the process models from data. Finally, some statistical tests are suggested that prove/verify the significance of our constructed process models.

- **Chapter 5** details our proposed PMI method in hybrid planning domains. It describes the steps of acquiring a process model (PM) from input data and its integration into the pre-engineered hybrid domains. Besides, it mentions some evaluation metric in order to assess the learned domain models.

- **Chapter 6** demonstrates the real-world case studies that have been chosen to experiment/implement our PMI method such as Coffee domain (i.e. making espresso with personalised taste) and UTC domain from SimplifAI (i.e. controlling traffic signal phase according to flow rate). The rationales for employing PM in order to improve those hybrid planning domains are also given.

- **Chapter 7** It describes the empirical analysis that is conducted with the cases mentioned in chapter 6. Besides, it demonstrates the evaluation results with learned domain models in the corresponding cases.

- **Chapter 8** concludes this thesis by summarising the overall work that has been accomplished so far and providing the future direction to improve/enhance our work further. It also discusses the limitations of this research work.

CHAPTER 2

Automated Planning with Hybrid Domains (APHD)

With the advancement of AI Planning technology, the Automated Planning with Hybrid Domains (APHD) is becoming more acceptable in order to solve real-world problems. A real-time system, also known as hybrid system, involves both analogue and digital computations of system components for completing any task. Therefore, a planner or planning engine must have the ability to handle two kind of primary behaviours such as instantaneous discrete change and continuous transition over time in respect of producing realistic plans.

Basically, classical planning techniques are applicable in static environment, where a single agent (or planner) works with a set of non-temporal and deterministic actions to achieve a predefined goal. On the other hand, the hybrid planning techniques are capable to generate temporal plans with instantaneous as well as durative action sequence in the dynamic world. In addition, it allows the interaction between discrete events and continuous processes during the state transition of physical components.

To implement the hybrid planning strategies in realistic applications, a planning engine requires detailed knowledge specification of the hybrid system. In AI planning, a domain model is used to describe the application requirements including the definition of objects, classes of different objects and their relations, all probable actions and functional properties. Since the hybrid system involves time-dependent discrete-continuous changes in the numeric resources, the hybrid domain model is used instead of classical domain model to express such entities.

The formal model of hybrid domain is based on the hybrid automaton, which can represent the mixed discrete-continuous system and contain numeric variables, control modes, invariant, events, jump and flow conditions. Domain modelling languages, for example, PDDL+ (Fox and Long 2002), $\int$-PDDL+ (Ramirez et al. 2017), ANML (Smith et al. 2008) are utilised to encode hybrid domains according to the problem statement. The expressive power of such hybrid modelling languages for use with automated hybrid planner brings the opportunity of handling temporal processes, exogenous events, instantaneous and durative action executions in the current state.

Recently, the PDDL+ modelling language has become popular in the planning community and a centre point of research in the AI planning field in comparison with other

planning languages. It has overcome the limitations of classical/temporal planning languages by supporting the predefined exogenous events to avoid plan failures in the dynamic world (Fox and Long 2002). Moreover, PDDL+ planning can involve linear (Coles and Coles 2014, Coles et al. 2012, Shin and Davis 2005) and non-linear continuous changes (Bryce et al. 2015, Della Penna et al. 2009) with the help of processes. Besides, this plan specification language allows simultaneous changes on different numeric fluent or quantities by initiating and terminating the concurrent processes and actions. These qualities have widen up the scope of applying hybrid planning techniques in real-world applications. However, the set up and use of hybrid planning has great challenges in terms of (a) computational difficulty of solving the planning problems and (b) engineering difficulty in creating the knowledge model of hybrid domains (McCluskey and Vallati 2017).

A hybrid planning engine, also known as hybrid planner, applies planning techniques based on the application knowledge described in the (hybrid) domain models. Therefore, a planner performance along with the plan quality, flexible configuration and reformulation of planning strategies are quite affected by the effective engineering of the domain knowledge (Vallati et al. 2015). On this account, Knowledge Engineering (KE) plays a vital role in the field of Automated Planning (AP). It involves the acquisition, formulation, validation and maintenance of (planning) application knowledge that are finally codified into the domain models (Jilani et al. 2014). To encode a (planning) domain model, the current KE approaches require domain experts/knowledge engineers who must have expertise, knowledge and hands-on experience in the corresponding planning application (Arora et al. 2018). However, the manual encoding of complex real-world domains induces human-made errors, during the creation of domain models, due to the limitations of human knowledge, capturing dynamically changing states, or handling unpredictable events. Besides, it becomes a challenge to develop, debug and maintain a real-time domain models by hand, as the granularity of planning applications becomes higher (McCluskey et al. 2002).

To mitigate the KE issues that are occurred by manual encoding of domain models, AI planning community is utilising Machine Learning (ML) techniques in order to acquire the domain models automatically (Arora et al. 2018). The long history of exploiting ML techniques in Automated Planning (AP) has shown its eligibility to automatically extract, refine, organise and exploit the domain knowledge (see Chapter 3, for detailed discussion). Machine Learning (ML), as a compulsory behaviour of an Intelligent Autonomous System (IAS), enables an agent to learn from experiences/observations, perceive the world, develop skills, make better decisions and perform/improve the actions accordingly without human or programmed instructions (Carbonell et al. 1983). In that

way, it assists the planning agents/knowledge engineers to improve the planning performance, plan quality and domain theory by automatically compensating the incomplete domain knowledge (Zimmerman and Kambhampati 2003, Jiménez et al. 2012).

In this thesis, our main focus is on the aforementioned area (b) to get more effective (hybrid) domain models by tackling the challenges of engineering hybrid domains in terms of modelling process descriptions (i.e. the continuous changes). The thrust is to exploit ML techniques and statistical methods with an intent to automatically induce the process models (see Chapter 4, for detailed discussion). In other words, the proposed ML based approach (explained in chapter 5) automatically learns the continuous time-dependent changes that are occurred in real-time (planning) domains. The primary objective of this thesis is to automatically refine and improve the hybrid domain models in respect of **Dynamicity** and **Versatility** (mentioned in chapter 1), by utilising the ML techniques with statistical analysis.

This chapter begins with an introduction of Automated Planning, followed by a description of primary elements of an automated planning system (section 2.1). Then it details the modelling languages for classical and hybrid planning domains, along with the comparison between classical and hybrid planning domains that emphasises the need for implementing hybrid domains in AI (Automated) planning applications (section 2.2). The basic definition of hybrid systems/hybrid domains is also given (section 2.3). Besides, it describes the modelling components of PDDL+ language in hybrid domains with example (section 2.4), proceeds to mentioning the existing hybrid planners that support PDDL+ models (section 2.5) . At the end, some real world applications of hybrid planning domains with PDDL+ formulation are highlighted (section 2.6).

2.1 Automated (AI) Planning

Generally, Planning is a cognitive process of human brain. It involves reasoning, learning from experience and knowledge to generate a sequence of actions in order to achieve a desired goal (Rogers et al. 2011). In Artificial Intelligence, Planning is one of the major capabilities of an Autonomous Intelligent Systems (AIS) or robots that enables the system to solve a given problem automatically without the human intervention (Ghallab et al. 2016). Automated planning has a varied range of applications such as control spacecraft mission or planetary exploration, manage non-player characters in embedded games, handle large-scale machineries or refinery process of chemical plants, regulate autonomous vehicles, virtual human or humanoid robots and so on (Ghallab et al. 2004).

2.1.1 Motivation

There are theoretical and practical reasons that Automated Planning has become a potential research field in AI (Ghallab et al. 2004, 2016). Theoretically, it provides rational and deliberative behaviours to the autonomous agents enabling them to reason about actions. In that way, planning is one of the main requisite to design an Autonomous Intelligent System (AIS). Practically autonomous agents can handle repetitive monotonous as well as complex risky tasks more efficiently than human. Based on this discussion, planning plays an integral part of AI.

2.1.2 Conceptual Model

The conceptual model for automated planning is illustrated in figure 2.1 that describes the basic prototype of a planning system (Ghallab et al. 2004). It has three main components: a state-transition system, a controller and a planner.

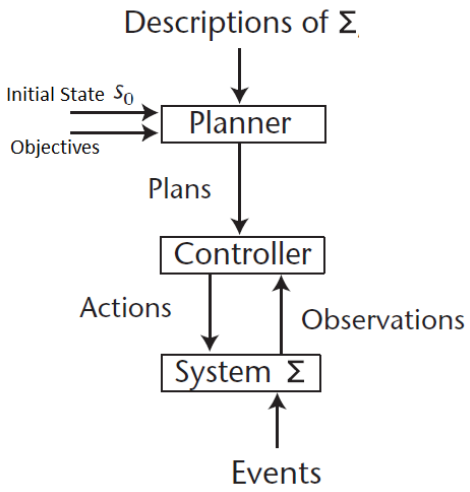


Fig. 2.1 A Conceptual Model for Automated Planning (Ghallab et al. 2004)

State-transition system, Σ

State-transition system, Σ (or discrete-event system) represents all possible situations of a real-world system. It is a 4-tuple $\Sigma = (S, A, E, \gamma)$, where,

$S = S_0, S_1, S_2, \ldots$ is a finite set of all probable situations that a world can be. S_0 is the initial state which describes the current position of the system;

$A = a_1, a_2, \ldots$ is a finite set of all possible actions that can be applied to change the state of the system;

$E = e_1, e_2, \ldots$ is a finite set of events that can be triggered or occurred by the internal dynamics of the system;

$\gamma : S \times (A \cup E) \rightarrow 2^S$ is a state-transition function that takes current states of the world S and action A or event E as input. It changes the current state of the system by applying the actions or events and concludes a new set of states.

Planner
A planner takes a description of the system Σ, initial state S_0 and objectives (or goal states) explaining a planning problem. Then it constructs a plan (a policy or a sequence of action) in order to achieve the given objective or reach the goal state.

Controller
A controller receives observations including the complete information about the current state of the system. Then it provides applicable actions based on current state and given plans to the state-transition function γ.

2.2 Planning Domain Representation Languages

A Planning system is logically separated into two mandatory parts: Planning Engine and Domain Model (McCluskey 2012). The planning engine creates plan based on the knowledge encoded/represented in the domain model. In the usual planning system, an user inputs the domain model to the planning engine in order to get a plan (illustrated in figure 1.1). Generally, a domain model contains objects of different types/classes, actions, states and goals. In the following sections, different types of modelling languages such as STRIPS, ADL, PDDL are briefly described.

2.2.1 STRIPS

STRIPS (Stanford Research Institute Problem Solver) is the first planning system implemented in the Shakey, the first automated mobile robot (Fikes and Nilsson 1971). Fikes and Nilsson (1971) developed STRIPS as a planning component in the software of Shakey. Afterwards the language used in the STRIPS became the formal input language of the planners. The STRIPS representation of States, Actions and Goals is deliberated with an example shown in figure 2.2 below.

2.2.1.1 STRIPS representation of Domain Model

- A STRIPS state is denoted by a conjunction of propositional positive-literals. The literals are function-free and grounded. It assumes situations to be false if they are not mentioned in the states which is called 'Closed World Assumption'.
- A STRIPS action contains four elements are as follows:

- preconditions – a conjunction of function-free positive literals which must be true before applying the action
- Positive literals – it will be added to the current state
- Negative literals – it will be removed from the current state
- Effect – a conjunction of function-free literals which represents the resulting state or new state. Literals which are not stated in the effect remain unchanged.

- A goal is a partially specified state which will be satisfied if all literals are true.

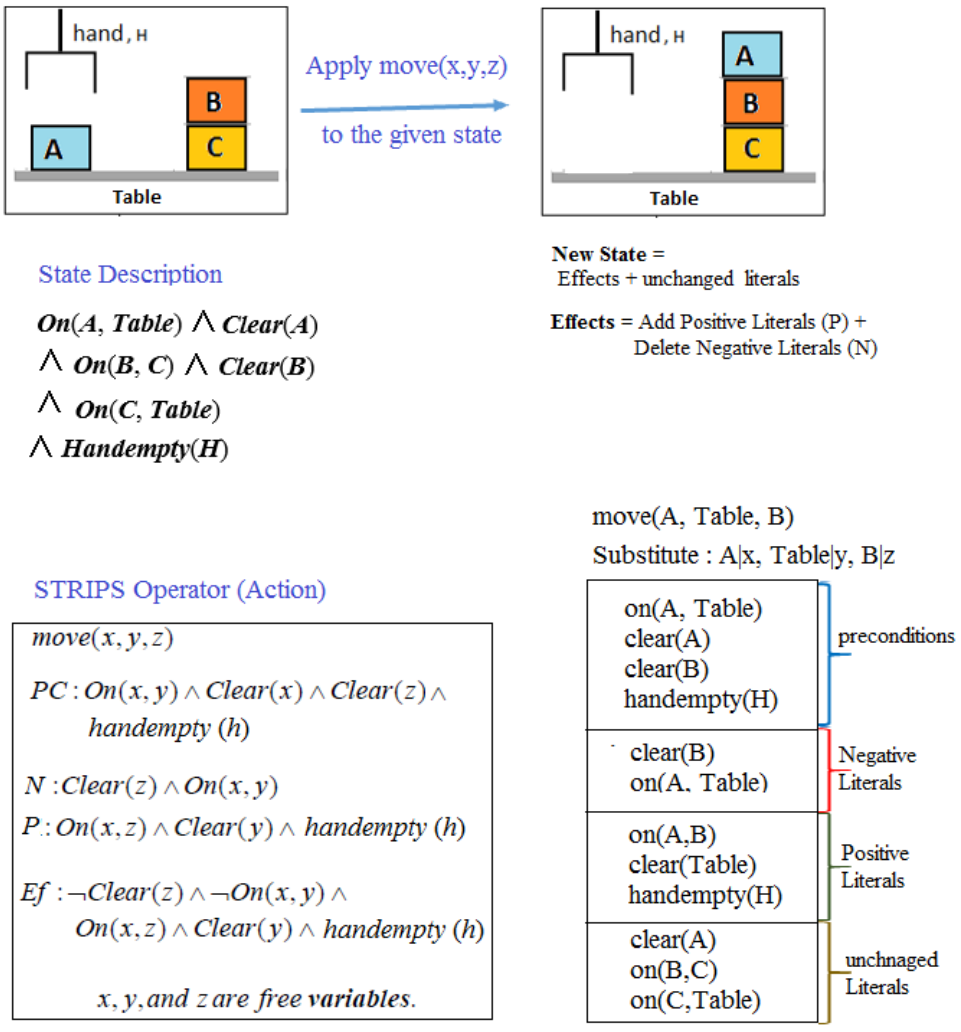


Fig. 2.2 STRIPS Representation of Block-World Domain

2.2.2 ADL

ADL (Action Description Language) is an extension of the STRIPS language which overcomes the syntactic and semantic constraints (Pednault 1989). Pednault (1989) has developed this action language to handle the real-world problems where STRIPS is

insufficient. The modifications of STRIPS in the ADL are explained with an example shown in figure 2.3 below.

2.2.2.1 ADL representation of Domain Model

- The ADL States can be expressed by positive and negative literals. Unspecified literals are considered as unknown rather than false, known as the Open World Assumption.
- The Goal states can be a conjunction or dis-junction of the literals and may contain quantified variables.
- In addition, Pednault (1989) has added some advanced features for instance: Conditional effects, types and built-in equality predicate.

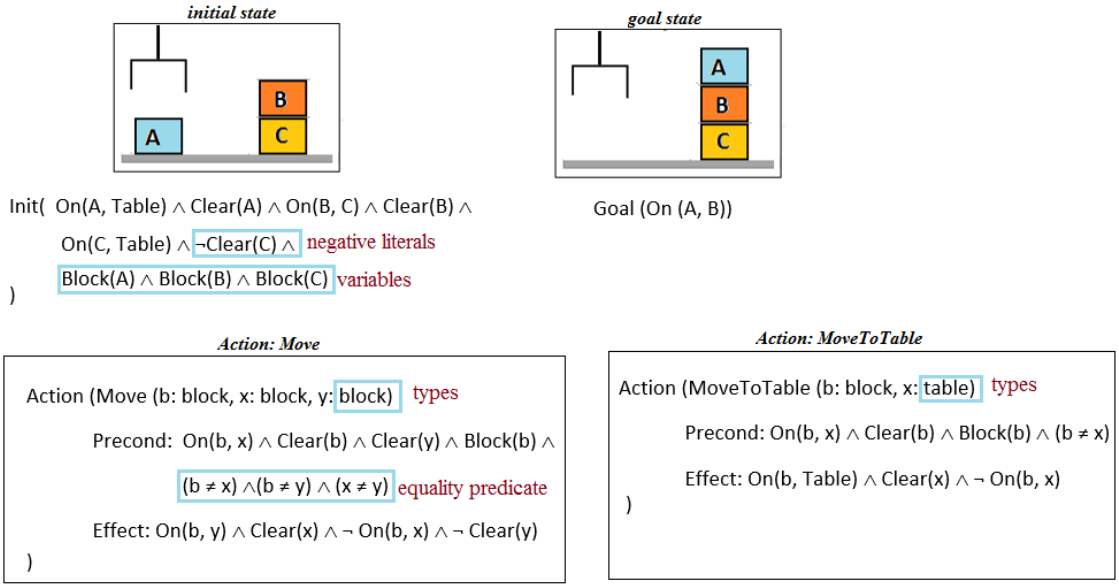


Fig. 2.3 ADL representation of Block-world Domain

2.2.3 PDDL

McDermott (2000) describes the Planning Domain Definition Language (PDDL) as a formal specification of planning problems. Initially, it is considered as the standard domain input language for the AIPS-98 planning competition to regulate the competing planners. Syntactically it is inspired by some basic modelling languages such as ADL (Action Description Language) (Pednault 1989), UMCP (Erol et al. 1994) and UCPOP (Weld and Penberthy 1992). Generally, a PDDL model contains objects of different types/classes, actions, initial states and goals. Besides that, it includes some additional features in the syntax, for example, requirements, constants, and predicates (Boolean value of an Object).

2.2.3.1 PDDL representation of Planning Domain

In PDDL model, the description of a planning domain is divided into two parts: Domain definition and Problem definition.

- **Domain Definition -** Basically it defines the domain name, predicates of objects and description of all possible actions in the planning domain. The basic structure of domain definition is shown in figure 2.4a below.
- **Problem Definition -** The problem definition mainly includes the objects, initial state and goal description of the planning problem. Usually, the PDDL planning problem is written in the format as presented in figure 2.4b below.

```
(define (domain DOMAIN_NAME)
        (: requirements [: strips] [: equality] [: typing] [: adl]) ;; Declares Packages
        (: predicates   ...)
        (: action ACTION1_NAME
                (: parameters (...)) ;; the list of variables
                (: precondition (...))
                (: effect (....))
        )
)
```

(a) Planning domain description in PDDL representation

```
(define (problem PROBLEM_NAME)
        (: domain DOMAIN_NAME)
        (: objects OBJ1 OBJ2 ... OBJ_N)
        (: init (....))
        (: goal (and ....))
)
```

(b) Planning problem description in PDDL representation

Fig. 2.4 Basic Structure of PDDL planning model

2.2.3.2 Different versions of PDDL

The earliest version of PDDL is the PDDL 1.2 (McDermott et al. 1998) that was used in the first International Planning Competition (IPC) (McDermott 2000). With the progressive versions of PDDL, it has become the official language of IPC. Different versions of PDDL with feature extensions are listed in the table 2.1 below.

Table 2.1 PDDL versions with feature extensions

Versions	Feature Extensions
PDDL 2.1 (Fox and Long 2003)	• Temporal planning with durative actions • Durative actions with fixed time period • Continuous/Discrete effects • Plan optimisation with metric minimisation/maximisation • Numeric fluent for continuous (non-binary) variables
PDDL 2.2 (Edelkamp and Hoffmann 2004)	• Derived Predicates to handle facts dependencies • Timed initial literals for exogenous events
PDDL 3.0 (Gerevini and Long 2005)	• Preferences for the quality measurement of a plan • State-trajectory constraints with safety conditions or operating conditions
PDDL 3.1 (Kovacs 2011)	• Object-Fluent with numerical and object-type functions

Table 2.1 summaries the improved features of PDDL versions for modelling classical/temporal planning domains. However, such classical/temporal domains have limitations for modelling realistic problems in terms of handling both discrete and continuous changes with non-linear time limits and external events. As mentioned earlier, a real-time hybrid system, also defined as hybrid domain, may require interaction between discrete and continuous components with time-dependent dynamics. Therefore, to generate realistic plans, a planning agent must be able to interact with continuously changing quantities and obtain up-to-date information about their numeric features.

2.2.4 Classical vs Hybrid Planning Domains

For overcoming the limitations of classical domain, the hybrid domain (planning) models have been supplemented by two main components which are processes and events. In comparison with classical domain, a hybrid domain involves the time-dependent

discrete-continuous changes in numeric resources. Besides, it also allows interaction between discrete events and continuous processes. Figure 2.5 illustrates the comparison between classical and hybrid planning domains.

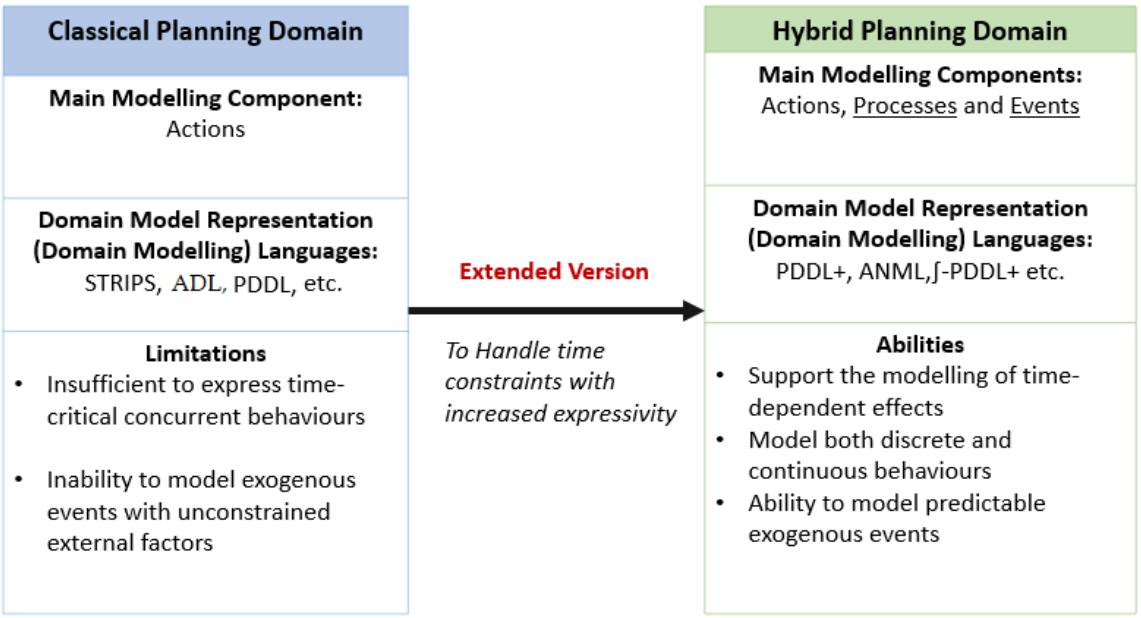


Fig. 2.5 Comparison between Classical and Hybrid Planning Domains

The hybrid domain modelling language, named PDDL+ (Fox and Long 2006) is the remarkable extension of PDDL2.1 (Fox and Long 2003) which can handle both continuous and discrete changes in the planning domain. Fox and Long (2006) has developed PDDL+ for modelling real world problems where other PDDL languages are insufficient. A real-time system or Hybrid system (mentioned in next section 2.3) can be represented by a hybrid automaton (Henzinger 1996) with mixed discrete-continuous states. Therefore, the theory of Hybrid Automata (HA) is applied in the PDDL+ semantics. In the section 2.4, the objectives of PDDL+ along with the modelling features, semantic/syntactic structure and the comparisons with PDDL2.1 are discussed in details.

2.3 Hybrid Systems/ Hybrid Domains

In a dynamic world, realistic problems may require both discrete and continuous changes in the feature components to solve tasks. These problems are described by hybrid systems where the analogue features are controlled by digital components (Thomas and Henzinger 1996). Specifically a hybrid system (also known as hybrid domain) is a real-time dynamic system that exhibits both discrete and continuous behaviours by embedding the digital element in the analogue environment. For example, controlling traffic flow by digital traffic signal, electric kettle with temperature control, automatic thermo-

stat with temperature adjustment, chemical refinery processes controlled by valves and embedded computer system with digital and analogue devices etc.

2.4 Modelling Hybrid Domains with PDDL+

The main objective of PDDL+ is to model the realistic problem such as Petroleum refinery production, Planetary lander and the Satellite observation domain which involve continuous and discrete fluctuations in numeric resources (Fox and Long 2002, 2006). Besides, Fox and Long (2006) explained that PDDL+ has the potentiality to connect AI planning research with practical applications.

2.4.1 Modelling Features

PDDL+ has three main components to model a hybrid domain: durative/instantaneous Action, Event and Process. Process execution occurs continuous fluctuations on numeric quantities that can be initiated and terminated by an action or event. An action is taken by the agent and an event can be triggered by the external environment that brings discrete modification in the system. This three-part structure of modelling processes with continuous effects is referred as the *start-process-stop* model. In addition, PDDL+ allows the activation or deactivation of multiple continuous processes with discrete changes at the same time which is called *Happening*. Figure 2.6 illustrates the recharging process of rover domain in PDDL+ planning model (Fox and Long 2002). In this example, when the sun arrives an action *"active-charger"* induces the *"charging"* process which increases charge-level consuming the sunlight. Besides, when sun disappears or the charge is full, an event *"stop-charging"* halts the *"charging"* process.

```
(:action activate-charger
 :parameters (?r - rover)
 :precondition (and (in-sun ?r)
                    (< (charge ?r) (capacity ?r)))
 :effect ( charging ?r))


(:process charging
 :parameters (?r - rover)
 :precondition (and (<= (charge ?r) (capacity ?r))
                    (in-sun ?r)
                    (charging ?r))
 :effect (increase (charge ?r) (* #t (charge-rate ?r))))


(:event stop-charging
 :parameters (?r - rover)
 :precondition (or (= (charge ?r) (capacity ?r))
                   (not (in-sun ?r)))
 :effect (not (charging ?r)))
```

Fig. 2.6 The recharging process of rover domain in PDDL+ model (Fox and Long 2002)

2.4.2 Semantic and Syntactic structure

Henzinger (1996) defined the semantics of Hybrid Automata (HA) based on the labelled transition system, therefore PDDL+ semantics include HA along with labelled transition system. Figure 2.7 depicts a model of hybrid automaton for the recharging process in rover domain (Fox and Long 2002). In this model, labelled edges denote the events or discrete changes that update the system from one control mode to another. Here, the control switch *"active-charger"* changes the charge-level through the real-valued variables such as *charge*, *recharge-rate*, and $\frac{d}{dt}$*charge*.

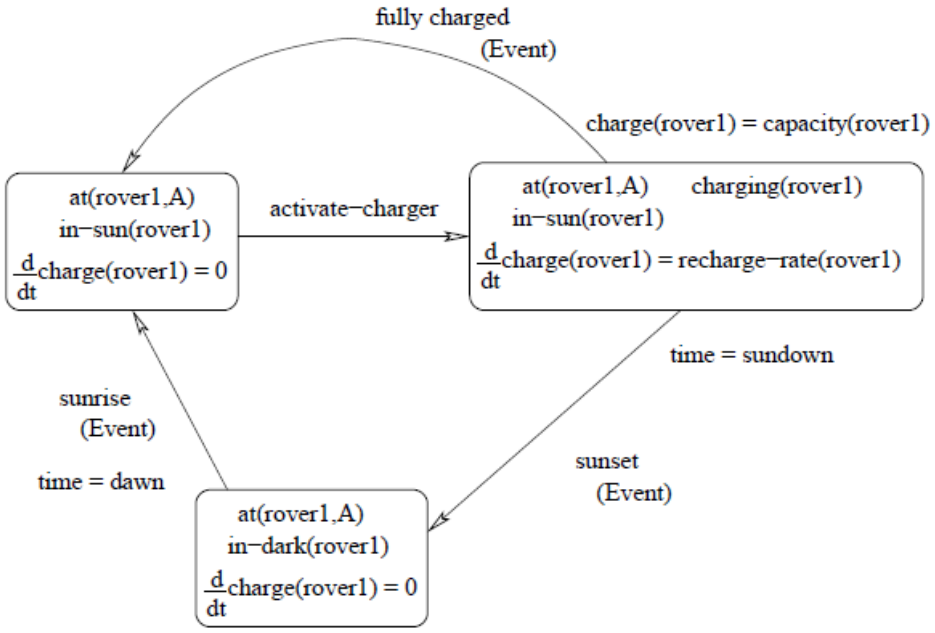


Fig. 2.7 The recharging process of rover domain in Hybrid Automaton model (Fox and Long 2002)

The syntactic structure of PDDL+ is formed on the basis of PDDL languages in conjunction with durative actions of fixed-length from PDDL2.1 and timed initial literals from PDDL2.2. PDDL+ events are similar to PDDL actions with a prerequisite of numeric preconditions. PDDL+ processes are similar to the actions defined in PDDL 2.1 with an exception of always containing #t literal for time-dependent continuous effects. The execution time period of a process instance is denoted by #t over which the preconditions of running process are satisfied. The numeric post-condition (effect) of a process is time-dependent, that brings the continuous change by updating numeric expressions. As an example, we can consider the *"charging"* process in rover domain (figure 2.6). Where the charge-level of rover $?r$ increases continuously as a function of the *charge-rate* of $?r$. Such continuous change in the charge-level is expressed by the numeric expression below:

(increase (charge ?r) (#t (charge-rate ?r)))*

2.4.3 PDDL2.1 Vs PDDL+

PDDL2.1 is restricted in expressing the continuous changes (McDermott 2003). Comparatively, PDDL+ is more expressive to model the continuous as well as the discrete behaviours with the *start-process-stop* model. PDDL2.1 is limited in modelling the discretised time-dependent effects. On the contrary, PDDL+ can discretise the time period in order to execute the exogenous events of world and reason about their consequences.

2.5 Hybrid (PDDL+) planners

Piotrowski et al. (2016) has developed the first heuristic planner named DiNo that can handle non-linear process and events with full PDDL+ features. To manage the complex non-linear system dynamics, DiNo depends on the novel SRPG+ (Staged Relaxed Planning Graph+) heuristic with the discretise-and-validate approach. As a predecessor of DiNo, UPMurphi (Della Penna et al. 2012) also supports the full set of PDDL+ features by implementing the discretise and validate approach. It can solve realistic planning problems of hybrid systems by the explicit model checking algorithms on top of the CMurphi model checker (Della Penna et al. 2009). Nevertheless, UPMurphi has limited scalability with the lack of heuristics, particularly the domain-specific heuristics. Scala et al. (2016) develop an automated planning system which is called Expressive Numeric Heuristic Search Planner (ENHSP). It applies the interval-based (generalised) relaxation for hybrid planning that improves the numeric planning heuristics. It allows wider range of mathematical functions with non-linear conditions in the planning problems. Moreover, it supports simple/non-linear numeric planning with classical domains (i.e. PDDL2.1) as well as PDDL+ hybrid domains including autonomous processes and discrete events.

Bryce et al. (2015) introduces the SMT (Satisfiability Modulo Theories) based planning in hybrid systems that can adopt nonlinear continuous changes by SMT encoding. Later on, the dReal SMT is integrated with the proposed HNSolver algorithm where the PDDL+ planning is modelled as a HA (Hybrid Automata) Network (Bryce, Bogomolov, Heinz and Schilling 2016). This approach relies on the language of dReach where the PDDL+ problems are translated into complicated hybrid system problems. Besides, it can not handle the full set of language features contained in PDDL+, specifically the exogenous events (Cashmore et al. 2016). To overcome those limitations, Cashmore et al. (2020) presents a new SMT encoding for PDDL+ models named as SMTPlan+. It is a SMT-based PDDL+ planner for hybrid systems that supports full range of PDDL+ features including external events.

2.6 Applications

Automated Planning with Hybrid Domains (APHD) have been applied in a wide range of practical applications for solving real-time problems. For instance, Della Penna et al. (2010) represents a complete PDDL+ model for the batch chemical plant. By utilising the UPMurphi (Della Penna et al. 2009) hybrid planning tool, it generates a set of *production polices* for the plant. The aim is to produce saline solution with a given concentration. Besides, Fox et al. (2012) cast the multiple battery load management problem as a hybrid (PDDL+) planning problem. Where, they build the effective policies for battery switching under uncertainty. Their experimental results with two battery system demonstrate that the proposed approach can improve 5% to 15% battery lifetime. Piacentini et al. (2016) presents PDDL+ modelling for solving the Unit Commitment (UC) problem in the electrical power industry. They compare the planned solutions with a traditional MIP (Mixed Integer Programming) approach that shows $\sim 3\%$ similarity with MIP. Vallati et al. (2016) introduces a hybrid planning based traffic control system with PDDL+ models to control vehicle flow in the urban road network. The goal is to handle unexpected traffic congestion during uncertain events such as road accidents/repair, natural calamities etc. Their experimental analysis shows that the proposed PDDL+ approach can generate effective signal plans with unexpected traffic situations by considering the current traffic condition and the network structure. Other real-world applications of APHD are Policy learning for autonomous feature tracking (Magazzeni et al. 2014), Power substation management (Bell et al. 2009), Planning for persistent underwater autonomy (Palomeras et al. 2016, Cashmore et al. 2017), Autonomous maintenance of submerged oil and gas infrastructures (Maurelli et al. 2016) etc.

CHAPTER 3

Machine Learning (ML) in Automated Planning (AP)

In AI, Machine Learning (ML) is one of the intelligent behaviours of an autonomous system. It enables the system to learn from experience, develop skills and improve actions automatically without any human assistance or programmed instructions (Mitchell et al. 2013). Typically, the learning process of ML is classified in terms of how much reasoning or deliberation the learner has to undergo (Russell and Norvig 1995). The basic ML approaches are: learning by example (supervised), learning by observation (unsupervised), learning by analogy (semi-supervised) and learning by discovery (Reinforcement).

According to the surveys conducted by Jiménez et al. (2012) and Zimmerman and Kambhampati (2003), there has been a long history of using ML techniques to aid Automated Planning (AP). Jiménez et al. (2012) reviews the ML techniques in AP for defining the action models and control knowledge that can improve the planning performance. Besides, Zimmerman and Kambhampati (2003) observe the Automated Planning (AP) systems and Machine Learning (ML) techniques to analysis the role of learning in the planning. They suggest that the integration of ML components with AP systems can accelerate the planning process, enhance plan quality and refine domain theory. Applying ML techniques to assist AP can fall into approximately two areas e.g. (1) Learning Heuristic (i.e. Automated Skill Acquisition) to refine the knowledge of a system already have and (2) Learning Domain Model (i.e. Automated Knowledge Acquisition) to gain the new knowledge from the environment.

This chapter concisely summarises the current ML approaches for Learning Heuristics and Domain Models in AP. Besides, some recent works on implementing ML in hybrid planning domains are discussed that implicate the necessity of ML in APHD. We also review the existing approaches for inducing process models in the hybrid systems. At the end, the current state of ML applications in learning the hybrid (planning) domain knowledge are highlighted with issues.

3.1 Learning Heuristics

The inductive learning technique can be used to generate the heuristics and search control rules from the classical domain knowledge in order to accelerate planning tasks (Leckie and Zukerman 1998). Veloso et al. (1995) demonstrates an architecture called PRODIGY which includes different ML mechanisms to learn about control knowledge with respect to improve domain and plan quality. It is also shown that AI reasoning in planning performance can be improved with regard to enhance the planner's solvability. This PRODIGY architecture together with the inductive generalisation process can repair the plans by observing the action executions and resulting states (Wang 1996). Percassi et al. (2020) combines machine learning and heuristic search in order to improve (domain-independent) planning performance with action costs. Particularly they utilise the inductive learning approach to predict the plan cost which is deployed in the proposed bound-sensitive heuristic function of a state-space planner. Another supervised learning based approach is presented by Gomoluch (2020) for domain-independent planning, where the heuristic functions are learned from the data representing multiple domains and then deployed on unseen domains. The experimental evaluation demonstrates the potentiality of proposed approach in terms of improving original heuristics. Kocsis and Szepesvári (2006) implements the Reinforcement Learning (RL) technique named the Markovian Decision Problems (MDPs) in the Monte-carlo planning algorithm to guide the action selection process and produce the optimal plans. Besides, Leonetti et al. (2016) exploits the Reinforcement Learning (RL) technique to solve complex decision-making problems with action constraints. The proposed method is called **D**omain **A**pproximation for **R**einforcement **L**earn**ING** (DARLING). De La Rosa et al. (2008) defines the decision tree (supervised learning technique) based heuristic methods along with the Enforced Hill Climbing algorithm to acquire the action policy and node evaluation order for domain-specific models. Later De la Rosa et al. (2011) presents the relational decision tree based solution for reducing node evaluations in (forward-chaining) heuristic planning, where Domain-specific Control Knowledge (DCK) is learned from the good quality solutions of a heuristic planner. Based on the learning, they build (domain-dependent) decision trees which capture the best possible actions taken by the planner. The experimental results reveal that heuristic planner integrated with learned classifiers can solve larger problems than state-of-the-art planners.

3.2 Learning Domain (Action) Models

Reinforcement Learning methods can be used to learn action models for self-improving reactive agents to solve complicated tasks (Lin 1992). Watkins and Dayan (1992) presents an algorithm named Q-learning on the basis of Markovian domains to improve the quality of actions by learning about the optimal acts. Later the Q-learning approach is joined with the Reinforcement learning to generate optimal exploration techniques by learning the joint actions values (Claus and Boutilier 1998). Sutton (1991) combines the machine learning methods with the reactive execution of the planning agent. The stated architecture is called Dyna. It observes the planning outcomes and updates the models by making changes in action effects. Lanchas et al. (2007) defines an inductive relational ML approach to learn the action-duration from plan execution and induce a new action model. The modelling procedure includes three-phases:(1) knowledge acquisition, (2) model learning and (3) redefinition of the action mode. The plan with the new action model takes less execution time to achieve a goal. McCluskey et al. (2009) develops the inductive learning based knowledge acquisition system named OPmaker2. It inputs the partial domain model along with the solution sequences of planning task and postulates the full domain model including plan heuristics as output. This method can induce domain models for efficient plan generation and reduce the problem of knowledge engineering. The LOCM (Learning Object-Centred Models) system induces the PDDL domain models from the example plans by exploiting the assumptions with the inductive learning (Cresswell et al. 2013). This novel LOCM algorithm does not require in-depth domain knowledge to analyse the action traces. Further the LOCM domain representation is generalised in the LOCM2 algorithm that is able to learn wider range of domain models (Cresswell and Gregory 2011). Later the output of LOCM2 is used in the LOP (LOCM with Optimised Plans) domain model acquisition system which takes the optimal plans as input. It observes the static relations to induce the domain models for shorter plan generation (Gregory and Cresswell 2015). Jilani et al. (2015) introduces a graph-based machine learning tool named ASCoL (Automated Static Constraint Learner) that identifies the static relations between predicates in a planning domain. It can input any type of plan traces i.e. optimal/sub-optimal goal-oriented or random walks in order to output improved domain models. Gregory and Lindsay (2016) demonstrate a domain model acquisition system (NLOCM) to learn action costs from the action traces in the numerical planning domain. In NLOCM, the constraint programming approach with the fragment of PDDL has applied to observe actions with minimal input. Segura-Muros et al. (2018) formulates an algorithm called PlanMiner-O2 based on the inductive rule learning techniques that learns action models with costs. PlanMiner-O2 can generate valid PDDL domain models dealing with high levels of incompleteness and some levels of noise.

3.3 ML in the Hybrid Planning Domains

Say et al. (2017) presents deep network learning based approach to acquire Mixed Integer Linear Program (MILP) formulation of hybrid models from observations, where the MILP-based Hybrid Planner is used instead of the PDDL+ hybrid planner that learns and optimises nonlinear hybrid planning problems with deep net models. A similar approach named *Tensorflow* is demonstrated by Wu et al. (2017) that learns plans through backpropagation to optimise the given expressions. The approach can work well on solving highly non-linear planning problems, however it does not support the PDDL family of languages as input. Bhatti et al. (2019) builds an architecture that integrates the Model Predictive Control (MPC) techniques into the automated hybrid planner (i.e. PDDL+ planner). The objective is to guide the automated planning search with effective approximation of planning heuristics. Their experimental analysis shows that the proposed approach can enhance the planning performance of state-of-the-art planners. Lindsay et al. (2020) proposes a machine learning (i.e. decision tree) based approach for refining the process description in hybrid planning domains. The presented approach identifies the significant temporal features and states of original planning domains by exploiting the observational data. The intent is to effectively capture and refine the continuous process effects along with the relationships between functions and propositions. The empirical evaluation exhibits more accurate simulation with the refined domain models that can consequently lead to higher quality plans. Besides, it can reduce the knowledge engineering efforts by automatically refine the domain models.

3.4 Inducing Process Models

Langley et al. (2002) presents an approach that induces the process models from continuous (time-series) data for scientific and engineering domains, where a process model specifies a set of processes that describe the causal relations between input and output variables in the form of differential/static equations. Their learning system called IPM (Inductive Process Modeler) inputs a set of generic processes and time-series data about quantitative variables in order to construct a process model. Unlike the tradition inductive approach in ML, IPM conducts constrained search through the space of process models that incorporates the Lagramge (Todorovski and Dzeroski 1997) program to find the best model with optimal parameter estimations. Later, Langley et al. (2003) extends the IPM method in response to deal with some specific challenges posed by overfitted models, unobservable variables, and numeric conditions on processes. The extended IPM algorithm includes a gradient-descent method along with the explicit specification of unobservable variables. This work is further improved by Todorovski et al. (2005) in a system named HIPM that induces process knowledge and organises them in a hi-

erarchical manner. The HIPM algorithm implements the heuristic beam search with knowledge-guided refinement operators for searching through the model space more effectively. The system inputs a hierarchy of generic processes, entities/hierarchy of entities with associated variables and a specified beam width. Langley and Arvay (2017) describes a more flexible inductive process modelling approach which is called FPM (Flexible Process Modeling). The FPM system automatically produces new generic processes to construct an acceptable model in the absence of sufficient processes. It implements two approaches i.e. Naive approach and Heuristic approach by utilising the SPM (Arvay and Langley 2016) program in order to induce consistent process models with corresponding differential equations.

3.5 Current State

From the discussion in section 3.1 and 3.2, we can perceive that there has been a long history of applying ML techniques in the classical/temporal planning domains with respect to improve planning heuristics and domain theory. Comparatively, there is very less work (reviewed in section 3.3) on applying ML techniques in APHD with regard to upgrade the hybrid planning heuristics, refine the hybrid domain knowledge or improve the plan simulation output. Specifically, learning the hybrid domain models with time dimension has got less attention in the AI planning community (Arora et al. 2018).
Furthermore, The existing methods/approaches (discussed in section 3.4) for inducing process models do not support the hybrid planning domains that can apply in the AI planning applications. In other words, they have experimented with processes in the population dynamics domains instead of considering the hybrid domains in AI planning applications. Besides, none of the aforementioned methods (reviewed in section 3.4) utilises the regression technique to induce process models from real-world data. For instance, Langley et al. (2003) used synthetic data instead of real-time observational data. However, induced process model in the form of mathematical equation is a complementary to this thesis, where the induced process model represents the causal relationship between dependent and independent variables.
Although, the area of ML utilisation in hybrid planning systems has received research attention in recent years but still can not make as much stride as the classical/temporal planning systems. The very recent published work done by Lindsay et al. (2020) (reviewed in section 3.3) appears as a first to learn and refine the process knowledge for hybrid planning domains. This thesis is trying to utilise ML techniques in order to improve the hybrid (planning) domain models. In particular, the proposed approach automatically learns and estimates the continuous changes of numeric fluent in the processes that are represented by process models. The learned process models are embedded/adjusted in the existing hybrid domain models with PDDL+ formulation. The aim

is to refine/upgrade the process knowledge and enhance the simulation accuracy that can consequently improve the plan quality.

CHAPTER 4

Statistical Methods and Analysis with ML Techniques

In this thesis, we propose a machine learning based approach with statistical analysis, named PMI, that automatically learns the continuous time-dependent changes occurred in hybrid (planning) domains (details in next chapter 5). Particularly, the PMI formulates Process Models (PM) by exploiting real-world data (i.e. quantitative time-series data) for hybrid planning applications (described in previous chapter 2). There has been a long history of applying Machine Learning (ML) techniques in the classical/temporal planning domains with respect to refine and improve (planning) domain knowledge (reviewed in previous chapter 3). Comparatively, there is very less work on applying ML techniques in hybrid planning domains (highlighted in section 3.3). This thesis utilises the ML techniques incorporated with statistical analysis in order to refine the hybrid domain knowledge. It involves a range of statistical methods to conduct statistical analysis of real-time data used in planning applications. In particular, it includes mathematical formulas, models and ML techniques that extract information about continuous changes in the quantity of numeric variables. Based on the knowledge extracted from continuous processes, a Process Model (PM) is induced which approximates the continuous effects in hybrid planning domains.

This chapter provides the formal definition of the kind of PM we are interested in. It investigates the classification of data types, the sources for collecting raw data and the steps of data preparation. Besides, it describes the statistical methods for analysing data and the ML techniques to formulate the PM accordingly. At the end, the statistical tests are discussed that are conducted to assess the significance of the resulting PM on the observed data.

4.1 Process Modelling

Basically, a PM is defined as a mathematical function which estimates the total variation in one quantity, y by partitioning the variability into its deterministic components $x_1, x_2, ..x_n$, plus an additional component, ϵ (Heckert et al. 2002). Where, the ϵ explains random variations (errors) in the data. As an example, according to Charles's Law (Dal-

ton 1802), the volume of gas increases proportionally with temperature increment and vice versa. The PM represents the total variation of measured pressure in a gas tank by the variability in temperature of gas. Equation 4.1 represents the general form of the PM.

$$\hat{y} = f(\vec{x}; \vec{\beta}) + \epsilon \tag{4.1}$$

Usually the term "model" denotes more than just explaining the deterministic variation in terms of the mathematical function. However, this thesis employs the "Process model", or in short "PM", as a mathematical function of numeric variables in the hybrid planning domain. Specifically it describes the continuous changes in the numeric quantities in terms of mathematical equation.
Three main components of PM: the outcome variable ($\hat{y}$), the mathematical function ($f(\vec{x}; \vec{\beta})$) and the random errors (ϵ) are briefly described below.

1. **Outcome Variable -** The outcome variable, $\hat{y}$, known as "response variable", is a quantity that is measured by the changing quantities of other variables. It is also called dependent variable because the quantity of y depends on the input values of other variables. In short, y is an outcome of ongoing process, often as a result of variations in input variables.

2. **Mathematical Function -** The mathematical function, often termed as "regression function" or "regression equation", describes the deterministic variation in the outcome variable, $\hat{y}$. It contains two parts: predictor variables, $\vec{x}$ and corresponding parameters (also known as regression coefficients), $\vec{\beta}$. The predictor variable, also called "independent variable", causes a change in the quantity of outcome variable. Whereas the parameter of corresponding predictor indicates how the variation in predictor is related with the changes in outcome variable (detailed discussion in section 4.5.2).

3. **Random Error -** Random error, ϵ defines the deviation between the estimated value of outcome variable ($\hat{y}$) and the actual value (y) in the data. It signifies the statistical function-based relationship between the outcome and predictor variables which can not be absolutely deterministic in the real world.

4.2 Data Collection

Data collection is the prerequisite step of constructing the process model (statistical model) where the quality of data collected determines the quality of resulting model. General principles of choosing the data sources and types are applied based on the problem areas such as comparative, screening/characterising, modelling and optimising

(Heckert et al. 2002). There are mainly two types of sources/methods of collecting statistical data: Primary and Secondary (Groebner et al. 2018). The primary source data are collected from an observational study or resulting from an experiment, for instance survey, simulation, questionnaire etc. On the other side, the secondary source data are collected from other researches that are made available by others for instance books, reviews, catalogue etc. Moreover, the statistical data can be classified into two main categories: qualitative and quantitative (illustrated in the diagram 4.1).

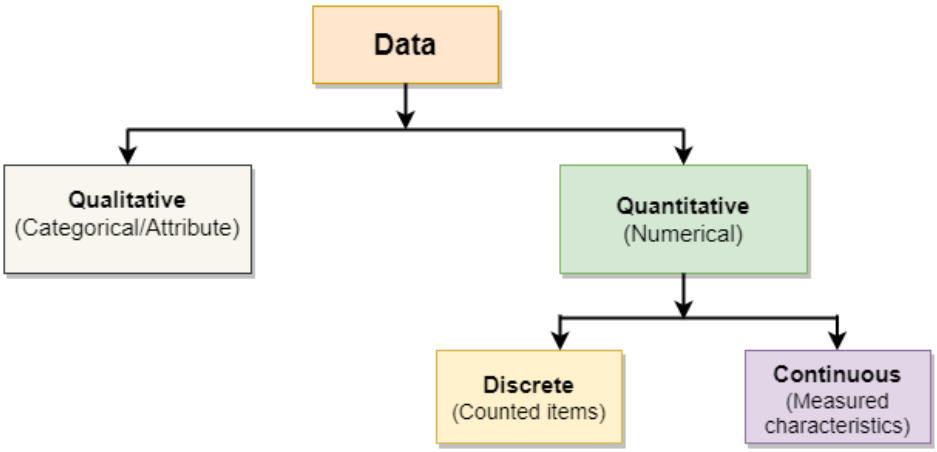


Fig. 4.1 Classification of Statistical Data (Groebner et al. 2018)

1. **Qualitative data:** Qualitative data is categorical which describes the attributes that can be observed but not measured. Some examples are gender (male/female), citizenship, name etc.

2. **Quantitative data:** Quantitative data is numerical value that is used to express amount, range or quantity. Examples of quantitative data includes height, weight, number of children etc. Quantitative variables are further classified into two parts: discrete and continuous. A discrete variable is countable and can only be a whole number, e.g. number of students in a classroom. On the contrary, a continuous variable can take any value in some interval, e.g. temperature.

4.3 Data Preparation

After collecting the data, the next step is to manipulate or transform the source data into a form that can be analysed accurately. Correct data preparation leads to an understanding of the data that facilitates to build right model (Pyle 1999). Data preparation involves a range of activities described briefly below:

4.3.1 Data Pre-processing

The real-world data often contains many errors such as having missing values, duplicate entities, incorrect attribute values or inconsistent variables which can cause a low quality of models (Danubianu 2015). Data pre-processing is a method of resolving such issues by cleaning the data and ensuring the data quality. In other words, it organises the data into a proper format, so that features can be easily interpreted by the ML algorithms.

4.3.2 Feature Encoding

Feature encoding performs data transformations such that ML algorithms can easily accept the features as input. For example, mathematical transformation using equations, binary conversion of qualitative (categorical) variables, altering the measurement scales, truncating the decimal places etc.

4.3.3 Train/Test split

After feature encoding is completed, the dataset is divided into two parts: Training and Testing sets. The training dataset are exploited to train ML algorithms with an intent to build a model that fits on this data. The fitted model tries to learn the features and intricacies of the training data, so that it can make predictions on new data (general data). To test the model hypothesis and measure the prediction accuracy, the model is applied on the testing dataset. It is worth noting that the split of train/test data may arise the model overfitting or underfitting issues that affects the predictability of the model (Jabbar and Khan 2015). Overfitting indicates that the model is fitted on the training data very closely (Figure: 4.2) but may fail to fit on the new data or predict future observations accurately. On the contrary, the underfitted model does not sufficiently fit the training samples (Figure: 4.2) and also is not generalised to other data. As a consequence, the model can not make inferences on new data with poor predictive ability. The issues of model overfitting/underfitting can be avoided by the cross validation (Prechelt 1998) discussed below.

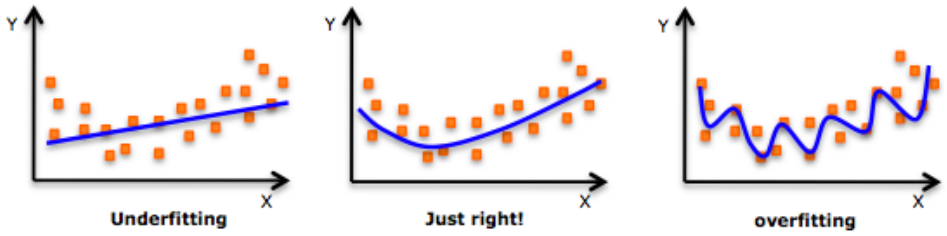


Fig. 4.2 An example of underfitting and overfitting with a model that's "just right!" (Bronshtein 2017)

4.3.4 Cross Validation

The cross validation (CV) method splits the data into k number of subsets where k-1 subsets are used as training data holding the last subset as testing data. The most commonly used CV method is K-Folds Cross Validation (Anguita et al. 2012). In K-folds cross validation, the data is randomly divided into k folds (or subsets) where k-1 folds are used for training and 1 fold for testing. The whole procedure repeats k times rotating the test set. To finalise a model, the resulting models from across the iterations are analysed based on the performance metrics such as mean square error, standard deviation etc. Figure 4.3 demonstrates an example of 5-Fold cross validation (K = 5) where the dataset is split into 5 folds. The entire procedure iterates 5 times taking one fold (orange box) as testing set and the other folds (green boxes) as training set in each iteration.

Fig. 4.3 5-Fold Cross Validation (K = 5) (Krishni 2018)

4.4 Correlation Analysis

Correlation analysis is a statistical method, which is used to measure the association between two quantitative features e.g., the relationship between a dependent, y and an independent variable, x or between two independent variables (x, x). In other words, it statistically infers how strong the relationship of each variable pair (x, y) in a bi-variate data where each x data point has a corresponding y data point. Based on the relationship significance (correlation coefficient) among features, different regression techniques (discussed in the next section 4.5) are applied with an intent to construct the process model (Equation 4.1). To calculate the correlation coefficient, we have applied the Pearson Correlation Coefficient (PCC), or simply Pearson's r method that identifies the feature associations (between x and y) along with their inter-dependencies (between x and x).

4.4.1 Pearson Correlation Coefficient (PCC)

Pearson correlation coefficient (PCC), also known as the bivariate Pearson Correlation, estimates the correlation coefficient (r) which indicates the strength and direction of

linear relationship between pairs of quantitative variables (Cohen 2013). Pearson's correlation coefficient between two sample variables x and y is denoted by r or r_{xy}, and is calculated by the formula (Zaiontz 2015*a*) below:

$$r_{xy} = \frac{cov(x, y)}{s_x s_y} \tag{4.2}$$

where,

$cov(x, y)$ is the covariance between two sample random variables x and y, and is defined by the formula (Zaiontz 2015*a*):

$$cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)} \tag{4.3}$$

s_x and s_y are the sample standard deviations of x and y respectively that are calculated as follows (Zaiontz 2015*b*):

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.4}$$

$$s_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.5}$$

In equation 4.2 , substitute the estimates of covariance and standard deviations by the formula 4.3, 4.4 and 4.5, we can obtain a formula for r_{xy} as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.6}$$

where,

n is the sample size,

x_i, y_i are the individual sample points indexed with i,

$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ are the sample mean of x and y respectively.

The values of correlation coefficient range from -1 to +1 where the sign (- or +) indicates the direction of the relationship and the magnitude implies the strength of the relationship. A correlation coefficient of +1 implies that two variable have positive linear relationship, a correlation coefficient of -1 specifies the negative linear relationship, while a correlation coefficient of zero signifies that two variables have no linear relationship (illustrated in Figure 4.4).

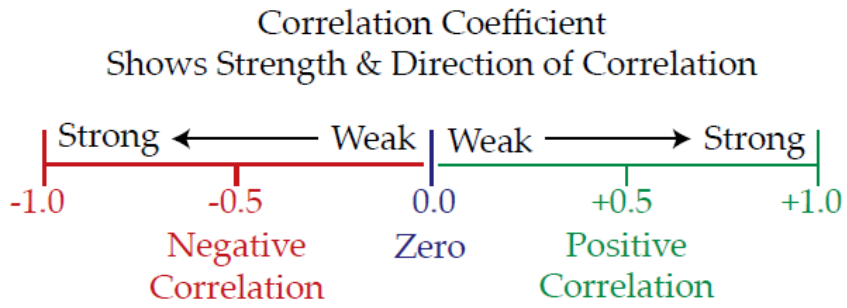


Fig. 4.4 The spectrum of the correlation coefficient (-1 to +1) (Gogtay and Thatte 2017)

The strength of linear relationship can be assessed by the general guidelines (Cohen 2013) given below:

$0.1 < |r| < 0.3$: small or weak correlation

$0.3 < |r| < 0.5$: medium or moderate correlation

$0.5 < |r|$: large or strong correlation

Pearson's correlation coefficient can only infer the linear relationships among quantitative variables, denoting that it can not assess the non-linear relationships or the relationships among qualitative (categorical) variables. Besides, it only reveals the existence of significant relationship among variables without inferring the cause and effect (Gogtay and Thatte 2017). Therefore, correlation analysis is accompanied by the regression analysis that expresses the causal relationship among variables in the form of a mathematical equation.

4.5 Regression Analysis

Regression analysis is a statistical method that is used in supervised machine learning for forecasting, prediction, time series modelling and determining cause-effect relationship between variables (Freund et al. 2006). More specifically, the goal of regression analysis is to formulate a mathematical equation, known as regression model, that defines the outcome variable (y) as a function of the predictor variables (x). The regression model predicts the value of outcome (dependent) variable based upon the value of one or more predictor (independent) variables. There are various types of regression techniques which are primarily chosen depending on three metrics: number and type of independent variable/s, type of dependent variable and shape of the regression line (Gogtay et al. 2017). Basically, it follows three steps to conduct the regression analysis: (1) Analysing the correlation, (2) Estimation of the parameters of regression model, (3) Interpretation of these parameters. Steps are discussed below:

4.5.1 Analysing the correlation (strength of relationship between variables)

Correlation analysis has been described in the previous section 4.4 that identifies the feature (variable) associations and their interdependencies based on the results of statistical test (PCC). From the values of correlation coefficient (r) and out of innumerable forms of regression, the appropriate regression technique has been nominated that best suit our learning problem depending on the type and number of independent variables. Aforementioned in the section 4.4.1, PCC can only determine the existence of linear relationship among variables, hence we have implemented different linear regression techniques driven by the following three cases to measure the causal effect of the relationship:

i y is related to a single x-variable (Simple Linear Regression)

ii y is related to multiple x-variables (Multiple Linear Regression)

iii y is related to multiple x-variables where x variables are correlated (Multicollinearity)

4.5.2 Estimation of the parameters of regression model

After determining the suitable regression technique from the first step (correlation analysis), the second step of regression analysis is to formulate the mathematical equation (shown in 4.1) by estimating the parameters or regression coefficients. The aim is to fitting the regression (linear) line through a cloud of data points, in such a manner that the distances between the regression line and the data points are minimised. This is to understand the underlying relationships and appropriately measure the outcome variable (y) on the basis of values of the predictor variables (X). The regression techniques employed in this thesis with corresponding statistical tests are described below.

4.5.2.1 **Simple Linear Regression**

Simple Linear Regression is a statistical method that estimates the relationship between a continuous dependent variable, y and a continuous or discrete independent variable, x. The regression model (process model) is represented by the equation of a straight line (Equation 4.7), known as regression line, that best fits the observed (training) data (Craven and Islam 2011).

$$\hat{y} = \beta_0 + \beta_1 * x \tag{4.7}$$

Where,
β_0 = y-intercept, i.e. the value of y where $x = 0$ and the line intersects with the y-axis
β_1 = slope of the line (also known as regression coefficient), describes the change in y due to a unit change in x

$\hat{y}$ = the predicted or estimated value of outcome variable, y based on the given value of predictor variable, x.
To find the best fit line, or simply approximate the appropriate value of β_0 and β_1, the most commonly used technique is "Least Squares Method" (James et al. 2013). For fitting the regression line through a set of data points, it estimates the parameters by minimising the Residual Sum of Squares (RSS), also known as the Sum of Squared Residuals or Errors (SSR or SSE), which is defined as follows

$$RSS = e_1^2 + e_2^2 + \cdots + e_n^2,$$

or equivalently as

$$RSS = (y_1 - \beta_0 - \beta_1 x_1)^2 + (y_2 - \beta_0 - \beta_1 x_2)^2 + \cdots + (y_n - \beta_0 - \beta_1 x_n)^2. \quad (4.8)$$

where, e_i = $y_i - \hat{y}_i$ represents the ith residual which is the difference between the ith observed response (or, actual) value and the ith response value that is predicted by the linear regression model. The residual, also known as the deviation or error (e), can be defined as the vertical distance between the regression line and the data point. Equation 4.9 represents the mathematical approach of getting the best fit line with the property that the Residual Sum of Squares, RSS, or the e_i^2 of N data points is minimum.

$$Min(RSS) = Min(\sum_{i=1}^{N} e_i^2) = Min(\sum_{i=1}^{N} (y_i - \hat{y}_i)^2) \quad (4.9)$$

To minimise the RSS, the Least Square method estimates the values of β_1 and β_0 using the basic formula below (Equation 4.10 and 4.11 respectively) (James et al. 2013):

$$\beta_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \quad (4.10)$$

$$\beta_0 = \bar{y} - (\beta_1 * \bar{x}) \quad (4.11)$$

where, N = Total data points, $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ are the mean of x and y respectively. Figure 4.5 provides the graphical illustration of a simple regression model obtained from the least squares coefficient estimates (4.10 and 4.11).

4.5.2.2 **Multiple Linear Regression**

Multiple Linear Regression is an extension of Simple Linear Regression where the outcome variable, y depends on two or more predictor variables, X. The general form of

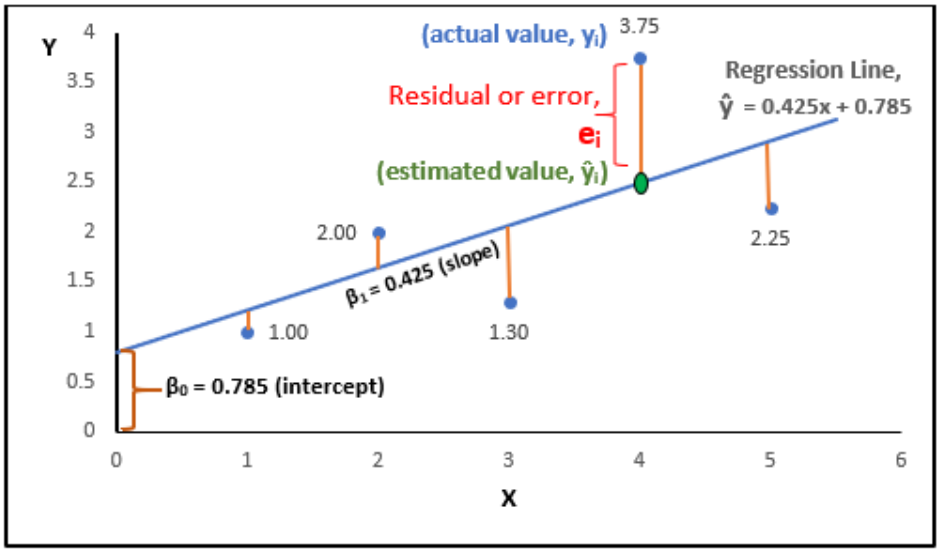


Fig. 4.5 Simple Linear Regression Model with Residual plot

multiple Linear regression model is given below (Equation 4.12) (Berry et al. 1985):

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k \tag{4.12}$$

Where, $\hat{y}$ = the estimated (predicted) value of the dependent variable, y,
X = $\{ x_1, x_2, \ldots, x_k \}$ = k distinct predictor (independent) variables
β_0 = y-intercept that is the value of y when all independent variables, X are equal to zero,
$\beta_1, \beta_2, \ldots, \beta_k$ = The regression coefficient of corresponding x variable that determines the partial contribution of respective x variable. In other words, each regression coefficient, β_i measures the change in y given a unit change in the respective x_i variable, holding all the other independent variables constant. It can be expressed as follows:

$$\beta_1 = \frac{\partial y}{\partial x_1}, \beta_2 = \frac{\partial y}{\partial x_2}, \ldots, \beta_k = \frac{\partial y}{\partial x_k}$$

In multiple linear regression, the parameters are estimated employing the same Least Square method that is used in the context of simple linear regression. The approach is to choose $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ that minimise the residual sum of squares (RSS):

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_k x_{ik})^2 \tag{4.13}$$

Unlike the simple linear regression estimates stated in 4.10 and 4.11 for finding the values of β coefficients, the method of Least Square is extended with multiple independent variables that expresses the regression line in the form below (Equation 4.14) (Zaiontz

2014*a*):

$$\hat{y} - \beta_0 = \beta_1(x_1 - \bar{x_1}) + \beta_2(x_2 - \bar{x_2}) + \cdots + \beta_k(x_k - \bar{x_k}) = \sum_{j=1}^{k} \beta_j(x_j - \bar{x_j}) \quad (4.14)$$

Aforementioned in the equation 4.13, the objective is to find a regression line (Equation 4.14) that best fits the given set of n points $(x_{11}, \ldots, x_{1k}, y_1)$, $\ldots$, $(x_{n1}, \ldots, x_{nk}, y_n)$. Hence, the best fit regression line has the given form (equation 4.15) (Zaiontz 2014*b*):

$$\hat{y} - \bar{y} = \sum_{j=1}^{k} \beta_j(x_j - \bar{x_j}) \quad (4.15)$$

where, the coefficients β_m are the solutions to the following (equation 4.16) k equations in k unknowns. The basic formula of estimating covariance between two sample variables, $cov(x, y)$ is stated in the equation 4.3.

$$cov(y, x_j) = \sum_{m=1}^{k} \beta_m * cov(x_m, x_j) \quad (4.16)$$

In multiple linear regression, the model selection process is required to select the best combination of predictor variables that build an optimal predictive (regression) model. The approach is to compare multiple models containing different subsets of predictors with the intent to select the best model with minimum prediction error. The most commonly used linear model selection technique is Stepwise Regression. The stepwise regression, also known as stepwise selection, is an iterative process of adding/dropping predictors in the regression model based on a specified criterion. Some criteria for analysing variables in order to discern the significant subset of variables are: the residual sum of squares (RSS), the residual mean square (RMS), the squared multiple correlation coefficient (R^2), the adjusted R^2, Akaike information criterion (AIC), Bayesian information criterion (BIC), F-tests or t-tests (Hocking 1976, Burnham and Anderson 2004). There are three basic methods of stepwise regression listed below (Bruce et al. 2020):

a) **Forward Selection**, this method starts with no predictor variables in the model and adds the most significant predictor variable in each iteration. The procedure terminates when none of the variable left that is statistically significant. The steps of forward stepwise selection procedure is given below (figure 4.6):

Algorithm *Forward Stepwise Selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Fig. 4.6 Forward Stepwise selection procedure (James et al. 2013)

b) **Backward Elimination**, this method initially adds all predictor variables in the model and eliminates the least significant variable in each iteration until all predictors are statistically significant. The steps of backward stepwise selection procedure is given below (figure 4.7):

Algorithm *Backward Stepwise Selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Fig. 4.7 Backward Stepwise selection procedure (James et al. 2013)

c) **Stepwise Selection**, also known as standard stepwise regression combines the forward and backward selection techniques that repeatedly adds and removes predictors as needed.

After implying the specified selection strategy, the stepwise regression estimates the coefficients of the regression model by applying Least Square method to the retained variables.

4.5.2.3 Multiple Linear Regression with Shrinkage method

The multivariate data set often contains two or more predictor variables that are correlated with each other. In such situations, the existence of near-linear relationships

among the predictor variables is referred as Multicollinearity, or collinearity. The presence of collinearity causes different problems in the regression context which ultimately degrade the predictability of the resulting model (Faraway 2002). Since the collinear variables tend to increase/decrease together, it can be difficult to determine the distinct effect of collinear variables on the outcome variable. As a consequence, it inflates the standard error of regression coefficients (β) by reducing the accuracy of estimated β values.

To deal with multicollinearity in the multiple linear regression, the first step is to detect the collinearity and determine the severity of collinearity before fitting the model. For detecting the multicollinearity, some of the common methods are listed below (Paul 2006):

- A simple way is to inspect the signs ($+/-$) of the regression coefficients. If the regression coefficients of two or more predictor variables have opposite sign from what we would expect then it may indicate the existence of multicollinearity.

- A better way is to evaluate the correlation matrix of the predictors ($r_{x,x}$) that describes the strength of relationship among variable pairs (discussed in the section 4.4). If the absolute value of an element of this matrix is larger (≥ 0.5) then it specifies a pair of high collinearity. However, the correlation matrix is not applicable where three or more variables are highly correlated even if there is no pair of particularly high correlation. In that situation, referred as Multicollinearity, the variance inflation factor (VIF) is considered instead of the correlation matrix.

- The Variance Inflation Factor (VIF) is the ratio of the variance of β_i when fitting the full model divided by the variance of β_i if fit on its own. In other words, it measures how much the variance of a regression coefficient (β_i) is inflated due to multicollinearity in the model. The basic formula of computing the VIF for each variable is given below in equation 4.17 (James et al. 2013):

$$VIF(\beta_i) = \frac{1}{1 - R^2_{X_i|X_{-i}}} \tag{4.17}$$

  Where, $R^2_{X_i|X_{-i}}$ denotes the coefficient of determination which is the squared multiple correlation coefficient (R^2) from a regression of X_i onto all of the other predictors. For instance,
  In the regression model, $y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$
  R^2_1 is obtained from regressing X_1 on X_2 and X_3 as follows:
  $X_1 = \alpha_0 + \alpha_1 X_2 + \alpha_2 X_3$
  Similarly,
  R^2_2 is obtained from, $X_2 = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_3$, and

R_3^2 is obtained from, $X_3 = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2$

A rule of thumb for interpreting the variance inflation factor is stated below (James et al. 2013):

VIF = 1 (the complete absence of collinearity)

$1 <$ VIF ≤ 5 (Moderately correlated)

VIF > 5 (Highly correlated)

After identifying the existence and severity of the multicollinearity among predictors, the next step is to apply different remedies to handle multicollinearity in the model. Some of the solutions for correcting the multicollinearity are briefly described below (Paul 2006):

- **Model Respecification**

  In some cases, Model respecification of the regression equation can lessen the impact of multicollinearity which involves the redefinition or reformation of correlated regressors (predictors). One approach is to combine the collinear variables together into a single predictor. For example, x_1, x_2, and x_3 are linearly correlated, now find some function such as $x = (x_1 + x_2)/x3$ or $x = x_1 * x_2 * x_3$ etc. that represents the correlated regressors without altering the actual information content.

- **Variable Elimination**

  The simplest method of solving multicollinearity problem is the elimination of collinear variables from the regression model. A stepwise regression as a model selection technique can be used to determine which of the variable to drop. However, it may degrade the predictive power of the model in specific cases i.e. the dropped predictors have significant influence on the outcome variable, y. Therefore, special care is required in the variable selection process in order to ensure the model quality.

- **Ridge Regression**

  Ridge regression (Hoerl and Kennard 2000) is an alternative linear regression (least square) technique that is used when the multivariate data set suffers from multicollinearity or contains a large number of variables superior to the sample size. It is also known as shrinkage or regularization method to fit a linear model containing all predictors by shrinking the coefficient estimates towards zero. The consequence of shrinking the coefficient values, is to reduce their variance and also allows the less significant predictors to have a coefficient close to zero.

  The aforementioned least squares approach of fitting regression line, estimates the value of β_0, β_1, ..., β_k that minimise the residual sum of squares (RSS) shown

below:

$$RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k}\beta_j x_{ij})^2, \text{From Equation (4.13)}$$

Ridge regression calculates the coefficient estimates using the same least squares technique with a slight difference in minimisation quantity. Specifically, the values of ridge regression coefficient, β^R are approximated to minimise RSS along with a shrinkage penalty stated in the equation 4.18 :

$$Min(\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{k}\beta_j^2) = Min(RSS + \lambda\sum_{j=1}^{k}\beta_j^2) \quad (4.18)$$

Equation 4.18 trades off two different criteria using RSS and the shrinkage penalty, $\lambda\sum_{j=1}^{k}\beta_j^2$. By minimising the RSS, the ridge regression chooses appropriate coefficient estimates that best fit the regression line. However, the second term, $\lambda\sum_{j=1}^{k}\beta_j^2$ (shrinkage penalty) shrinks the estimates of β_j, so that the predictor variables with minor contribution to the outcome variable, y, have their coefficients close to zero. The shrinkage penalty is not applied to the intercept, β_0 which is actually the mean value of the outcome, y when all predictors, $x_{i1} = x_{i2} = \cdots = x_{ik} = 0$, therefore the estimated intercept will take the form, $\beta_0 = \bar{y} = \sum_{i=1}^{n} y_i/n$.

The amount of shrinkage penalty is adjusted by the tuning parameter, λ. The range of possible λ values is 0 to ∞. When, λ = 0 specifies the penalty term that has no effect on the ridge regression and consequently generates the simple least square coefficients. However, the effect of shrinkage penalty raises with the increment of λ value to ∞ (as $\lambda \to \infty$) which brings the ridge regression coefficients close to zero. For each value of λ, ridge regression generates a different set of coefficient estimates with an intent to select the best model among them. Therefore, determining a good value for λ is critical. Cross validation (discussed in 4.3) is performed on the ridge regression for determining the appropriate tuning parameter, λ (Golub et al. 1979). The approach is to compute the cross-validation error for each value of λ and select the model with smallest cross-validation error.

4.5.3 Interpretation of parameters (Regression Output)

The last step of regression analysis is the interpretation of regression parameters, or simply the test for significance of regression model. It involves two types of statistical tests i.e. the t-test and the analysis of variance (ANOVA). The t-test is a statistical hypothesis test on the regression coefficients which indicates the test for significance of individual coefficient (Allen 2004). Conversely, ANOVA, as a generalisation of the

t-test, is a statistical method to test the significance of entire regression model (Turner and Thayer 2001).

Test on Individual Regression Coefficients (t-test)

The t-test conducts the hypothesis tests on the regression coefficients which are obtained by applying regression technique. The aim is to diagnosis the significance of individual regression coefficients in the regression model i.e. the regression model become more effective by adding a significant variable, conversely, an insignificant variable can make the regression model worse by degrading the prediction ability. To test the significance of a particular regression coefficient, β_j, the statements for hypothesis test are expressed as follows:

$$\text{Null Hypothesis, } \boldsymbol{H_0} : \beta_j = 0 \text{ [Change in } x_j \text{ does not affect } y]$$

$$\text{Alternative Hypothesis, } \boldsymbol{H_1} : \beta_j \neq 0 \text{ [Change in } x_j \text{ affects } y]$$

The test statistic used for this test is given below:

$$\boldsymbol{t} = \frac{\beta_j}{se(\beta_j)}$$

Where, $se(\beta_j)$ is the standard error of β_j. The test statistic, $\boldsymbol{t}$, follows a $\boldsymbol{t}$ distribution with (n - 2) degrees of freedom, where n is the total number of observations. The null hypothesis, $\boldsymbol{H_0}$, is true if the estimated value of the test statistic lies in the acceptance region stated below:

$$-\boldsymbol{t_{\alpha/2,n-2}} < \boldsymbol{t} < \boldsymbol{t_{\alpha/2,n-2}}$$

where, $-\boldsymbol{t_{\alpha/2,n-2}}$ and $\boldsymbol{t_{\alpha/2,n-2}}$ are the critical values for the two-sided hypothesis. $\boldsymbol{t_{\alpha/2,n-2}}$ is the percentile of the $\boldsymbol{t}$ distribution corresponding to a cumulative probability of $(1-\alpha/2)$ and α is the significance level. Alternatively, the $\boldsymbol{p}$ value corresponding to the test statistic, $\boldsymbol{t}$, can also be used. If the $\boldsymbol{p}$ value of corresponding test statistic, $\boldsymbol{t_{\beta_j}}$ is less than the significance level, α, then it is concluded that β_j is significant and the null hypothesis, $\boldsymbol{H_0}$: β_j = 0, is rejected. Consequently, the alternative hypothesis, $\boldsymbol{H_1}$ is accepted which implies that relationship exists between x_j and y.

Test for Significance of entire Regression model (Analysis of Variance Approach)

The analysis of variance (ANOVA) is a statistical procedure to test the significance of regression that is the statistical inference of the existence of linear relationship between the outcome variable and at least one of the predictor variables. The approach is to analyse the variance of the observed data which is partitioned into components that are then further employed in the test for significance of regression. The aim is to determine

whether the regression model can be applied to the observed data. The statements for the hypotheses are:

$$\text{Null Hypothesis, } \boldsymbol{H_0} : \beta_1 = \beta_2 = \cdots = \beta_k = 0 \text{ [No regression relationship]}$$

$$\text{Alternative Hypothesis, } \boldsymbol{H_1} : \beta_j \neq 0 \text{ for at least one } j$$

To test the hypothesis $\boldsymbol{H_0} : \beta_{1,2,\ldots,k} = 0$, ANOVA uses $\boldsymbol{F}$-tests which compares two types of variations, i.e. the variation between the sample means, and the variation within each of the samples. The basic formula of ANOVA test statistics is stated below, which is known as $\boldsymbol{F}$ statistic or $\boldsymbol{F}$-ratio.

$$\boldsymbol{F} = \frac{\text{Explained Variance}}{\text{Unexplained Variance}} \textbf{ Or, } \boldsymbol{F} = \frac{\text{Variation between sample means}}{\text{Variation within the samples}}$$

$$\text{Equivalently, } \boldsymbol{F} = \frac{MS_R}{MS_E}$$

where,
the regression mean square, $MS_R = \frac{\text{the regression sum of squares, } SS_R}{\text{the respective degrees of freedom, } dof(SS_R)}$, and
the error mean square, $MS_E = \frac{\text{the error sum of squares, } SS_E}{\text{the respective degrees of freedom, } dof(SS_E)}$.

If the null hypothesis, $\boldsymbol{H_0}$, is accepted then the statistic $\boldsymbol{F}$ follows the $\boldsymbol{F}$ distribution with k degrees of freedom in the numerator and $n - (k + 1)$ degrees of freedom in the denominator. Otherwise, The null hypothesis, $\boldsymbol{H_0}$, is rejected if the calculated statistic, $\boldsymbol{F}$, is such that:

$$\boldsymbol{F} > \boldsymbol{f_{\alpha,1,n-2}}$$

where, $\boldsymbol{f_{\alpha,1,n-2}}$ is the percentile of the $\boldsymbol{F}$ distribution corresponding to a cumulative probability of $(1 - \alpha)$ and α is the significance level. Alternatively, the $\boldsymbol{p}$ value corresponding to the test statistic, $\boldsymbol{F}$, can also be used. If the $\boldsymbol{p}$ value $<$ the desired level of significance, α, then $\boldsymbol{H_0} : \beta_1 = \beta_2 = \cdots = \beta_k = 0$ is false (or rejected), indicating that at least one coefficient out of $\beta_1, \beta_2, \ldots, \beta_k$ is significant. Specifically, a relation does exist between the outcome variable and either all or at least one of the predictor variables.

CHAPTER 5

The PMI Method in Hybrid Planning Domains

In this thesis, we assume that a knowledge engineer, for a particular application, has assembled, created and encoded an initial hybrid domain model within the PDDL+ representation. Our proposed method, named PMI (Process Model Improvement), results in a learned mathematical function (i.e. process model) that is used to adjust and improve a process description within that previously engineered, original domain model. In particular, the PMI method utilises machine learning techniques along with statistical analysis (discussed in previous Chapter 4) to formulate the Process Model (PM) by exploiting real-world data collected from actual execution of the process. The learned process model, in the form of a mathematical function, automatically approximates/adjusts the quantity of an outcome variable with the continuous variations in different predictor features in order to efficiently and accurately control the corresponding process in hybrid planning domains.
This chapter describes the step by step procedure of PMI method that involves data collection, data preparation, PM formulation, embedding the learned PM into the pre-engineered/original hybrid domain model, and finally the evaluation of learned domain model embedded with PM (section 5.3). Besides, it discusses the assumptions and the basic requirements of PMI method in order to formulate PM for hybrid planning domains (section 5.1). The automation abilities along with the manual steps are also highlighted here (section 5.2).

5.1 Assumption and Requirements

As mentioned earlier, the proposed PMI method assumes that an initial hybrid domain model, also called original domain model, has already been created by a domain expert/knowledge engineer, wherein the formulated Process Model (PM) is adjusted/embedded into the process description.

To formulate Process Models (PM), the basic requirements of PMI method are summarised below:

- Our proposed PMI method constructs the process models by analysing observa-

tional/simulation data with the help of ML techniques and statistical methods. Therefore, the quantitative (time series) data of numeric variables are required to be exploited by PMI method

- To initiate the procedure of PMI, a domain expert/knowledge engineer has to manually define an outcome and one/more predictor features of a process which effects need to be learned. In other words, dependent and independent variables of the process have to be identified a priori in order to formulate the process model
- The formulated process model specifies the causal (linear) relationship between an outcome (i.e. dependent variable) and one or more predictors (i.e. independent variables). Hence, the input data should contain the quantitative values of dependent and independent variables that are learned to find their interdependencies and apply (linear) regression techniques accordingly
- The learned process models estimate the quantities of numeric variables that fluctuate over time, thence it does not accept the numeric variable with constant value

5.2 Automation of PMI

The PMI method is partially automated; in particular the important, complex step of PM formulation is automated. The end to end method does require a knowledge engineer to perform the overall process. In detail:

- **Data Collection:** This step is manual.
- **Data Preparation:** This step is partially automated, where initially a knowledge engineer requires to identify the attributes or features that need to be transformed/encoded. Besides, the PMI method automatically splits the input data into training and test set implementing the cross validation method (discussed in chapter 4).
- **PM Formulation:** This step is automated. The real-world data collected from actual execution of the process is input, and it outputs a process model (i.e. a mathematical function) that approximates the quantity of an outcome variable with the continuous variations in different predictor features in the process.
- **PM Integration:** This step is partially automated in that a knowledge engineer needs to manually adjust the objects, predicates, parameter types, preconditions, and other numeric expressions etc in the process specification accordingly, while automatically embedding the learned PM into that pre-engineered process specification.

- **Evaluation of Learned Model embedded with PM:** This step is manual.
- **Deployment and Planning:** This step is manual.

5.3 Steps of PMI method

Figure 5.1 presents an abstract view of the overall PMI method, which takes quantitative (time-series) data along with the pre-engineered, original hybrid domain model (i.e. PDDL+ domain and problem definition) as input. In particular, input data contains the quantities of numeric variables that appear in the learned PM. It finally outputs the learned domain model with embedded PM that facilitates and automates the estimation of an outcome variable given a set of predictor variables (i.e. numeric features) in the underlying process.

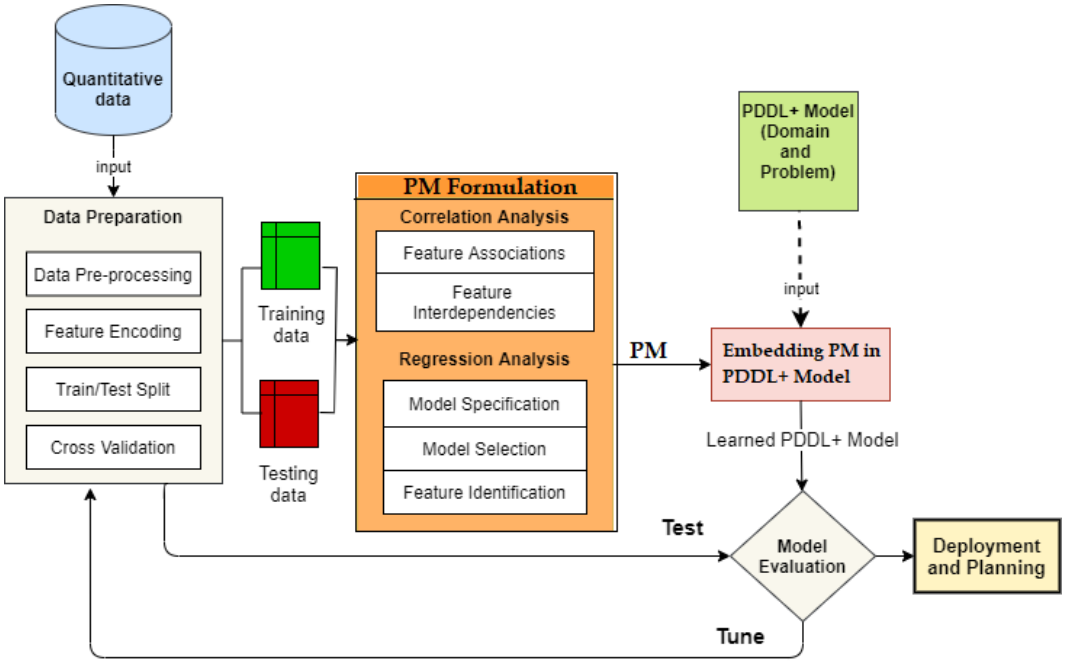


Fig. 5.1 PMI: abstract architecture

5.3.1 Data Collection

Initially, the PMI method takes the quantitative time-series data that is collected from the primary sources such as observational study/simulation output. The definition of data collection sources and the quantitative data types are provided in section 4.2.

5.3.2 Data Preparation

After collecting the raw data, the data are manipulated/transformed into a form that can be analysed accurately by the Machine Learning (ML) algorithms and statistical methods, in order to build the right model. Data preparation steps are: data pre-processing,

feature encoding, splitting data into two parts i.e. training and test set (detailed discussion in section 4.3). The training set is exploited to train ML algorithms with an intent to build a PM that that can make better predictions on new data (or, general data). To conduct the statistical test (mentioned in section 4.5.3), the testing set is utilised that evaluates the model hypothesis of learned PM and measures the prediction accuracy. PMI method divides the input data into training and test set by employing the cross validation methods (illustrated in section 4.3.4).

5.3.3 Formulation of PM

After preparing the raw data, the PMI method formulates the Process Model (PM) from the prepared data by conducting the Correlation Analysis and Regression Analysis consecutively. The earlier chapter 4 provides the detailed description of the statistical methods and ML techniques that are implemented in Correlation analysis (section 4.4) and Regression Analysis (section 4.5). The current section explains the steps of Correlation Analysis and Regression Analysis that are carried out by PMI method in order to formulate a PM.

5.3.3.1 Correlation Analysis

To formulate a PM from the input data, at first the PMI infers the relationship among variables i.e. Feature associations and Feature interdependencies by executing the correlation analysis. Feature associations define the correlation between a dependent/outcome variable (y) and one or more independent/predictor variables (X), where Feature interdependencies indicate the correlation among independent variables.
The PMI method performs the Pearson's r correlation test (mentioned in section 4.4.1) that statistically infers how strong a relationship is between two variables, based on the estimated values of the correlation coefficient (r). In simple word, it indicates that how likely the changes in one variable affect the other variable. Varying in the range between -1 and $+1$, r measures the magnitude of association, or correlation, as well as the direction of the identified features' relationship (shown in figure 4.4). The estimated Pearson Correlation Coefficient (PCC) between y and x is denoted by $r_{x,y}$, and between the two different x variables is denoted by r_{x_i,x_j}. After conducting the Pearson's r correlation test, the results can be categorised into three different cases in order to nominate the appropriate linear regression technique.

- **Case 1 : y is related to a single x-variable**
  In case 1, there is only one variable pair (x, y) with the correlation coefficient value (i.e. PCC value) $r_{x,y} \geq \pm 0.1$. Besides, the PCC value of all x-variable pairs (i.e. r_{x_i,x_j}) are less than ± 0.1. By following the general guidelines for assessing

the estimated PCC values (given in section 4.4.1), we can infer that a cause-effect relationship exists between y and a single x-variable with $r_{x,y} \geq \pm 0.1$, where x-variables are not inter-correlated. In this case, Simple Linear Regression (discussed in section 4.5.2.1) is applicable to formulate the model of causal relationship between y and a single x variable.

- **Case 2 : y is related to multiple x-variables**
  In case 2, some variable pairs (x, y) have PCC values ($r_{x,y}$) greater than or equal ± 0.1, which indicates the existence of relationship among y and more than one x-variable. Similar to case 1, all independent variables are non-correlated with $r_{x_i,x_j} < \pm 0.1$. Therefrom Multiple Linear Regression (MLR) technique such as Stepwise Regression (described in section 4.5.2.2) is suitable to model the linear relationship among y and x variables that can involve more than one x variable.

- **Case 3 : y is related to multiple x-variables where x-variables have intercorrelation**
  Similar to Case 2, some variable pairs (x, y) have PCC values $r_{x,y} \geq \pm 0.1$. Different from case 1 and case 2, at least one x-variable pair has PCC value $r_{x_i,x_j} \geq \pm 0.1$. After interpreting PCC values, we can conclude that y is dependent on multiple x-variables where some/all x-variables have interrelation. The phenomenon in which two or more predictor variables are linearly related in multiple regression is known as multicollinearity. To deal with multicollinearity, different techniques e.g. stepwise regression and ridge regression are applied concerning the severity of intercorrelation (detailed explanation in section 4.5.2.3). Therefore the Variance Inflation Factor (VIF) for predictors (i.e. x variables) is measured that quantifies the strength of intercorrelation (discussed in the next section).

Based on the inferred relationship among variables, the PMI method nominates the appropriate regression analysis, that is carried out to formulate the PM for three different cases (mentioned above). The next section describes the formulation procedure of PM by exploiting the nominated regression technique with corresponding statistical test.

5.3.3.2 Regression Analysis

The PMI method performs the regression analysis that outputs in a learned mathematical function (i.e. regression model) to express the causal relationship among variables. In other words, the formulated regression model, represented by PM, automatically approximates/adjusts the quantity of an outcome variable with the continuous variations in different predictor features.

The pseudocode below explains the formulation procedure of PM that automatically nominates the appropriate regression technique based on the PCC values, and then accordingly applies the nominated regression technique on the training data. The procedure takes a time series data (TSD) as input. TSD contains the quantitative values of a regressand (i.e. outcome variable, y) and multiple regressors (i.e. predictor variables, X) that tend to change over time. Finally, it outputs a regression model, $\hat{y}$ which is represented by our process learning hypothesis, or the learned PM.

In line 1, the procedure starts by taking the input data TSD, then it estimates the PCC values by conducting the correlation test on TSD (given in line 2). In line 3 and 4, it observes the estimated PCC values (i.e. $r_{x,y}$ and r_{x_i,x_j}) for case 1 (mentioned in previous section 5.3.3.1), where y is related with a single x-variable. If the conditions in line 3 and 4 are true then the linear regression technique is applied on the training data (shown in line 5). The regression coefficients (i.e. β_0 and β_1) are estimated that formulates the regression model.

In line 8 and 9, it assesses the estimated PCC values (i.e. $r_{x,y}$ and r_{x_i,x_j}) for case 2 (mentioned in earlier section 5.3.3.1), where y is dependent on multiple x-variables and x-variables are not correlated. If the conditions in line 8 and 9 are true then the multiple linear (i.e. stepwise) regression technique is applied on the training data (shown in line 10). The regression coefficients (i.e. β_0, β_1, β_2, ...) are estimated that formulates the regression model.

In line 8 and 12, it examines the estimated PCC values (i.e. $r_{x,y}$ and r_{x_i,x_j}) for case 3 (mentioned in earlier section 5.3.3.1), where y is related to multiple x-variables and x-variables are interrelated, this phenomenon in regression is called multicollinearity. To deal with multicollinearity, stepwise regression and ridge regression are applied concerning the severity of intercorrelation. Therefore the Variance Inflation Factor (VIF) for x-variables are measured that quantifies the strength of intercorrelation (shown in line 13).

The VIF is a transformation of the R^2 (the squared multiple correlation coefficient) resulting from predicting x by other predictor variables in the model (expressed in equation 4.17). The value of VIF > 5 confirms the existence of high collinearity and the value of VIF between 1 and 5 (i.e. $1 <$ VIF < 5) indicates the moderate correlation among independent variables (as discussed in section 4.5.2.3). If there is no collinearity existed between x-variables, then the value of VIF should be less than or equal 1 i.e. VIF ≤ 1.

Procedure 1 Automated Process Modelling with Correlation and Regression Analysis

Input: Time Series Data, $TSD = \{y, X\}, i.e.$

Regressand, $y = \{y \mid y \in \mathbb{Q}\}$

Regressors, $X = \{x_1, x_2, \ldots, x_k \mid x \in \mathbb{Q}, k \in \mathbb{N}_{>0}\}$

where, $\{\mathbb{Q}\}$= Set of Rational Numbers,

$\{\mathbb{N}_{>0}\}$= Set of Non-zero Natural Numbers

Output: Regression Model, $\hat{y} = f(X, \beta) = \beta_0 + f(x_i, \beta_{x_i})$

where, β_0 = Intercept,

β_{x_i} = Slope (Regression coefficient of y on x_i),

$x \in X$ and $i = \{1, 2, \ldots, n(X)\}$ ▷ $n(X)$, the number of elements of set X

procedure GetPM(TSD)
  $PCC \leftarrow CorrelationTest(TSD)$ ▷ PCC is the set of Correlation Coefficients
  if $\exists! r \in r_{x,y} | r \geq \pm 0.1$ **then** ▷ $r_{x,y} \subset PCC$
    if $\forall q \in r_{x_i,x_j} | q < \pm 0.1$ **then** ▷ $r_{x_i,x_j} \subset PCC$
      $(\beta_0, \beta_1) \leftarrow$ LinearRegression(y, x)
    end if
  end if
  if $\exists r \in r_{x,y} | r \geq \pm 0.1$ **then**
    if $\forall q \in r_{x_i,x_j} | q < \pm 0.1$ **then**
      (β_0,β_{x_i}) $\leftarrow$ StepwiseRegression(y, X)
    end if
    if $\exists q \in r_{x_i,x_j} | q \geq \pm 0.1$ **then** ▷ VIF is the set of Variance Inflation Factor
      $VIF \leftarrow$ MulticollinearityTest(X)
      if $\forall v \in VIF | 1 < v \leq 5$ **then**
        (β_0,β_{x_i}) $\leftarrow$ StepwiseRegression(y, X)
      end if
      if $\exists v \in VIF | v > 5$ **then**
        (β_0,β_{x_i}) $\leftarrow$ RidgeRegression(y, X)
      end if
    end if
  end if
  $p_f \leftarrow$ ANOVAtest(y, X) ▷ p-value of f-statistic, p_f
  $p_t \leftarrow$ T-test(y, X) ▷ p-value of t-test, p_t
  if $p_f < 0.05$ AND $p_t < 0.05$ **then**
    $\hat{y} \leftarrow f(X, \beta)$ ▷ $\hat{y}$ represents PM
  end if
end procedure

In this thesis, we conduct the correlation analysis as a prerequisite of regression

analysis, where the Pearson Correlation Coefficient confirms only the presence of relationship among predictor variables without quantifying the severity (medium/high) of intercorrelation. Therefore, VIF is calculated which is basically the squared Pearson Correlation Coefficient. In a nutshell, the VIF can not be less than 1, if the correlation coefficient confirms the existence of linear relationship among predictor variables.
By assessing the the severity of multicollinearity from VIF values, the procedure applies the ridge regression and stepwise regression in case of high and moderate collinearity respectively (shown in line 15 and 18 respectively).

After applying the corresponding regression technique in three different cases (discussed above), the mathematical function is formulated that represents the Regression model and from it create the learned PM. To assess the statistical significance of regression model, statistical tests are conducted such as the t-test and ANOVA test (F statistic) shown in line 22 and 23 (described in section 4.5.3). In line 24, if the p-values for both the F-test and the t-test are less than the significance level of 0.05, then we can conclude that regression model (shown in line 25) fits the data. In other words, the variation of y has explained well by the estimated regression equation that formulates the PM.
An exceptional situation may occur with all p-values > 0.05, when the outcome variable (y) does not dependent on any predictor variable (X). In short, y is not related with any x-variable. In Planning context, such type of exceptions is quite impossible/rare because Domain/Planning experts model a process (i.e. continuous change) based on some relationship assumptions between continuous numeric variables.

5.3.4 Integration of the PM

After applying the appropriate regression technique, we will get a regression model, represented in PM, to better explain the causal effect of numerically changing variables in a process. The resulting PM is integrated/adjusted into the process description of previously engineered, original hybrid domain model with PDDL+ formulation. As mentioned earlier, the PDDL+ has the ability to model continuous processes in the hybrid planning domains, whereas the dynamically changing quantities of numeric components are defined statically. In this thesis, the learned PM automatically estimates and adjusts the outcome variable based on the predictor features accordingly in the PDDL+ domains.

The learned PM exhibits the proposed process learning hypothesis with effective feature selection and appropriate approximations in terms of $\boldsymbol{\beta}$ values (stated in equation 5.1). It can identify the effective features (i.e. independent variables, x) with non-zero $\boldsymbol{\beta}$ values as well as eliminate unnecessary features with the corresponding $\boldsymbol{\beta}$ value to

0. For instance, if any x does not have impact on the resulting model, then it will set the corresponding $\boldsymbol{\beta}$ value to 0. Finally, the PM is embedded/adjusted into the process description of pre-engineered PDDL+ domain for modelling the continuous effects.

In PDDL+ hybrid domain model, the time-dependent continuous change on a numeric variable is expressed by means of a special modelling component named *process*. The generic structure of a *process* with PDDL+ representation is shown in figure 5.2 (Fox and Long 2006). where $\vec{\boldsymbol{p}}$ denotes the parameters of discrete predicates and $\overrightarrow{\boldsymbol{p_{x,y}}}$ symbolises the parameters of numeric fluent (i.e. the continuous functions). The process instance runs over the time period expressed by the special variable **#t**. The process effect occurs the time-dependent continuous changes ($increase$ or $decrease$) on the numeric fluent by updating the numeric expressions in terms of **#t** literal.

```
(:process name-process
   :parameters (p⃗ , p_{x,y}⃗)
   :precondition ( (starting-conditions p⃗ , p_{x,y}⃗) )
   :effect ( (increase (name-functions p_{x,y}⃗) (* #t 1) )
      )
)
```

Fig. 5.2 Generic structure for a PDDL+ process

In this thesis, the learned PM, in the form of numeric expression, is embedded/adjusted into the previously engineered process specification as a process effect. The following generic structure shown in figure 5.3, exhibits the PDDL+ process embedded with the learned PM (stated in equation 5.1).

$$
\begin{aligned}
\hat{y} =& f(X, \beta) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k \\
&\text{Here,} \\
\hat{y} =& \text{ Estimated value of the outcome variable, y} \\
X =& \text{ Set of Predictor variables, x} \\
\beta_0 =& y\text{-intercept, i.e. the value of } y \text{ where } x = 0 \\
\beta_i =& \text{Regression coefficient of respective } x_i \text{ variable,} \\
& \quad \text{i.e. defines the change in } y \text{ due to a unit change in } x_i \\
\beta_i =& 0 \text{ ,for } x_i \text{ that does not affect } y
\end{aligned}
\tag{5.1}
$$

```
(:process name-process
   :parameters (p⃗ , p⃗x,y)
   :precondition ( (starting-conditions p⃗ , p⃗x,y) )
   :effect ( and
      (assign (name-outcome p⃗y) (f(p⃗X, β)) )
      (increase (name-predictors p⃗X) (* #t 1) )
   )
)
```

Fig. 5.3 Generic structure for a PDDL+ process augmented with PM

In figure 5.3, $\overrightarrow{p_y}$ designates the parameter of outcome variable (y) and $\overrightarrow{p_X}$ symbolises the parameters of predictor variables (X), where, the change in $\overrightarrow{p_y}$ value is estimated by the function $f(\overrightarrow{p_X}, \beta)$ with the fluctuation in $\overrightarrow{p_X}$ values. The time-dependent continuous changes (such as $increase$ or $decrease$) take place on the numeric fluent, $\overrightarrow{p_X}$ by updating the expressions in terms of **#t** literal.

5.3.5 Evaluation of Learned Domain Model

By integrating/adjusting the Process Model (PM) into the process description of pre-engineered PDDL+ domain, we acquire a new domain model, that is called the learned domain model. Before deploying the new domain model in the planning applications, it is mandatory to evaluate the model effectiveness, accuracy and performance. There are several statistical tools or metrics for model evaluation, like Graphical residual analysis, Root Mean Squared Error (RMSE), Root Mean Squared Logarithmic Error (RMSLE), R-Squared (R^2), and Adjusted R-Squared (Adjusted R^2) etc that are applied in accordance with the type of problems. An evaluation metric provides feedback by explaining the model performance and discriminating the model results. Based on the feedback from evaluation metric, the learned model is improved and continue tuning until the desirable accuracy or result is acquired. Some evaluation metrics are briefly described below:

i **Residual analysis**

The primary tool for model evaluation is the graphical residual analysis (Heckert et al. 2002). As demonstrated in the figure 4.5, the residuals are the differences between the actual outcome values and the corresponding prediction values estimated using the regression function. The mathematical definition of the residual for the i^{th} observation in the data set is expressed below:

$$e_i = y_i - f(\vec{x_i}; \vec{\beta}) \tag{5.2}$$

Where, y_i denotes the i^{th} observed (actual) value of outcome variable in the data set, and
$f(\vec{x_i}; \vec{\beta})$ represents the regression function (defined in the equation 4.1) that outputs the corresponding predicted value of i^{th} observation in the data set
Graphical method of residual analysis illustrates the deviation of predicted value from the actual observation, particularly, the relationship between the regression model and the observational (training) data. For instance, the random distribution of errors in the residual plot indicates that the regression model fits the data well. On the contrary, the non-random structure signifies that the model fits the training data poorly.

ii **Root Mean Squared Error (RMSE)**
One of the most widely used evaluation metric is the Root Mean Squared Error (RMSE), also called the Root Mean Square Deviation (RMSD) given by the equation 5.3 (Chai and Draxler 2014). It squares the errors to prevent the cancelling of positive and negative values, therefore, provides the probable magnitude of the error terms. Besides, the square root enables to explain deviations of large numbers. As RMSE is sensitive towards outlier values, the outlier values are removed from the training data before applying this metric.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Actual_i - Predicted_i)^2}{N}} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{5.3}$$

Where, N is the total number of observations.

iii **Root Mean Squared Logarithmic Error (RMSLE)**
Unlike the RMSE, the RMSLE metric of regression calculates the log of the actual and prediction values which empowers it to handle outliers (Saxena 2019). The basic formulation of RMSLE is stated below (equation 5.4). It suffers from the biased penalty issue i.e. it incurs larger penalty when the *predicted value* $<$ *actual value*, conversely, less penalty is acquired when the *predicted value* $>$ the *actual value*. Therefore, RMSLE metric is used in certain regression problems where the underestimation is unaccepted but the overestimation is tolerated.

$$RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2} \tag{5.4}$$

5.3.6 Deployment and Planning

The last step is to deploy the final model (i.e. the learned domain model with PDDL+ representation) in the hybrid planning engines (i.e. PDDL+ planners) in order to gen-

erate real-time plans with more accurate and rational simulation output. The learned model consists of two parts: Domain definition and Problem definition. The domain definition integrates the Process Model (PM) into the process description of existing PDDL+ model. The problem definition initialises the values of regression parameters i.e. intercept, β_0 and regression coefficients, β_j for corresponding process variables.

5.4 Discussion

In the field of Artificial Intelligence (AI), Automated Planning (or, AI Planning) solves planning problems based on the (planning) application knowledge represented in domain models which is usually captured and encoded by the knowledge engineers/domain experts. Therefore AI planning depends on the experts knowledge in order to build domain models, and consequently produce effective plans. Besides, AI planning generally has limited data requirements for modelling hybrid domains. This thesis proposes a method, named PMI, that formulates Process Models (PM) by exploiting real-world data collected from actual execution of the underlying process in (hybrid) planning domains. The resulting PM, in the form of mathematical function, is integrated/adjusted into the process description of previously engineered, original (hybrid) domain model. In short, the proposed PMI method requires quantitative (time-series) data along with the pre-engineered, original hybrid domain model in order to build the learned (planning) domain model with embedded PM. Hence the proposed PMI method may introduce additional burden on Knowledge Engineering (KE) task by introducing additional requirements (i.e. relies on reliable data) and supplementary steps, for instance, data collection and data preparation etc. However, introducing additional steps on KE task for hybrid (planning) domains is a trade-off between reducing the KE effort of modelling processes, for instance, automatically identifying the significant/ineffective process variables (i.e. numeric features) along with their interdependencies, approximating the more rational and pragmatic value of process variable (i.e. outcome variable), capturing the dynamic fluctuations in process parameters and adjust them in the process effects automatically etc.

Moreover, the Machine Learning (ML) techniques along with statistical methods used in PMI provide results/outputs that are all interpretable. It means the constructed ML models (in this thesis, that is process models, or regression models) are easily verifiable and understandable for knowledge engineers/domain experts. In other words, the interpretable Process Models are easy to explain the values and accuracy of their findings. For instance, we can simply extract the relationships/interdependencies between process variables from the Pearson Correlation Coefficient (PCC) values. Besides, we can easily identify the significance of entire regression model from the p-values of t-test and ANOVA test (F-test).

CHAPTER 6

Case Studies

In this chapter we explore the domains that can be used to illustrate and evaluate the PMI Method. We investigated hybrid domains which had existing domain models, as well as novel domains.
In summary, the range of domains considered were:

- a large case study being attempted by McCluskey, Vallati and Franco (2017) aimed at generating traffic management plans

- the set of 'commonly' available PDDL+ domain models, including the Bouncing ball, Block World, Vending Machine and Coffee domains.

- the novel domain of ultra-precision surface polishing, where plans are for polishing tool movements (Walker et al. 2019)

We require a set of criteria to be true before a domain could be used:

i firstly, and most importantly, we require a reliable set of 'real' process data values to be available for the learning phase. PMI focuses on learning the continuous fluctuation of numeric values in the processes of hybrid planning domain. To learn the values of continuous features/variables, we need to exploit the quantitative (time series) data that we can harvest within the domain.

ii secondly, domain modelling needs to pose a challenge for knowledge engineers, in the sense that the initial domain model's processes descriptions are likely to be imperfect when specified a priori, and may well change often depending on environmental factors. In other words, learning process improvements during operation are important. In our study, this is often the case with physical planning domain applications, as the models of the processes used are invariably approximations, and need to be maintained.

Applying these criteria to the range of domains left us with the traffic management domain, and the coffee domain. The surface polishing domain would be, like traffic management, an excellent domain to use in this study. After exploring this application

for some time (and creating an initial process specification for polishing domain, given in Appendix F), however, we were unable to capture appropriate training data due to the pandemic situation of COVID-19 around the world (discussed in section 8.2.1). This domain we intend to use in future work (see chapter 8, for details), once the data is available. Hence, to demonstrate the feasibility of the proposed approach and to evaluate it on the real planning application, we use the PDDL+ encoding of the Coffee domain and the Urban Traffic Control (UTC) domain as case studies (detailed in section 6.1 and 6.2 respectively).

6.1 Coffee Domain

Coffee is one of the most popular beverages consumed by people around the world. Due to its variety of tastes and aromas, it has become a regular drink nowadays. Basically, coffee is prepared by brewing the roasted coffee ground with hot water. Different brewing methods are used to prepare coffee with desired flavour. Worldwide, specifically in Europe, espresso is the most common method of brewing coffee.
An espresso shot is prepared by brewing the finely ground compacted coffee with hot water under pressure. The brewing process extracts the solids and dissolved components from the ground coffee that promotes the flavours. Therefore, the proper extraction of coffee grinds defines the standard taste of an espresso shot.
The Planning Research group at KCL (King's College London) has introduced a hybrid planning based coffee making process with PDDL+ representation (KCL-Planning 2019). The KCL planning research focuses on applying the domain-independent planning in real-time applications. The PDDL+ formulation of coffee domain, proposed by KCL-Planning (2019), can brew coffee within a specified range of water temperature. The detailed discussion on PDDL+ model of coffee domain is provided below.

6.1.1 Original PDDL+ constructs to brew coffee

KCL-Planning (2019) formulates a very basic model of coffee domain that simply brews coffee with heated water. If the water is cold, it increases the water temperature. When the water temperature reaches the boiling point, the coffee brewing process is activated automatically. Table 6.1 below summaries the PDDL+ modelling components used to perform different functionalities e.g. brewing coffee, heating and cooling water etc. The PDDL+ encoding of original coffee domain (KCL-Planning 2019) is provided in Appendix A.1.

Modelling components (Action/Process/Event/Durative action)	Functionalities
:action *heatwater(w)*	Initiates the *heating(w)* process when the water *w* is cold
:process *heating(w)*	Increases the temperature of water *w* until reaches at it boiling point (i.e. 100)
:event *stop-heating(w)*	Stops the *heating(w)* process when temperature of water *w* exceeds 100
:event *boil(w)*	Activates the durative action *makecoffee(c, w)* when the water *w* is in boiling state
:durative-action *makecoffee(c, w)*	Takes actions for making the coffee *c* with hot water *w* where the temperature of water *w* should be between 60 to 80
:process *cooling(w)*	Decreases the temperature of water *w* when its temperature surpasses 18
:event *stop-cooling(w)*	Triggers the cooling state of water *w* when its temperature drops below 18

Table 6.1 PDDL+ modelling components of Original coffee domain (KCL-Planning 2019)

In the table 6.1, an action "*heatwater*" initiates a process "*heating*" that increases the water temperature. An event "*stop-heating*" halts the heating process with the water temperature $> 100°C$. Another event "*boil*" activates the brewing process that is represented by the durative action "*makecoffee*". During the brewing process, the overall temperature of water should be between $60°C$ to $80°C$. The "*cooling*" process is used to maintain the temperature of hot water within the specified range (i.e. from $60°C$ to $80°C$).

6.1.2 Rationale for using PM to improve Coffee domain

In the original coffee domain formulated by KCL-Planning (2019), the coffee brewing process only considers the water temperature to prepare an espresso shot. Other factors that could put in the PDDL+ model of coffee domain are missing e.g. brewing time, coffee dosage, coffee-to-water ratio, coffee particle size (finer or coarser grind) etc. Those factors along with the water temperature affect the extraction of coffee that decides the taste of espresso (Andueza et al. 2003, Ludwig et al. 2012, Melrose et al. 2018). KCL-Planning (2019) defines the water temperature statically without considering the extraction yield or the taste of coffee. Hence, the original coffee domain

(KCL-Planning 2019) is not a good approximation in the context of preparing espresso shots with standard taste.
In this thesis, we reformulate the PDDL+ coffee domain that demonstrates the coffee brewing process for making espresso according to taste. In other words, the coffee is brewed according to the expected coffee yield% that defines espresso taste. It maintains the quantity of coffee yield% by regulating the brewing temperature and brewing time during brewing process. The water temperature is the brewing temperature when it reaches the ground coffee and extracts flavour from it, whereas, brewing time is the amount of time that the water is in contact with coffee grounds.

6.1.3 Reformulated PDDL+ model to brew coffee

The PDDL+ model of coffee domain is reformulated that is partially inspired by its original PDDL+ construct (KCL-Planning 2019). Unlike the brewing process defined in original coffee domain, the reformulated coffee domain includes the start-process-stop model instead of using durative action. The PDDL+ modelling components of the proposed coffee domain are listed in table 6.2, where two consecutive processes are used such as *"heating"* to heat the water and *"brewing"* to brew coffee ground with heated water. Appendix B provides the entire PDDL+ model with a sample problem definition.

Modelling components (Action/Process/Event/Durative action)	Functionalities
:action *heatwater(c, w)*	Activates the process *heating(w)* when have coffee *c* and cold water *w*
:process *heating(w)*	Raises the temperature of water *w*
:event *stop-heating(w)*	Halts the process *heating(w)* once the temperature of water *w* reaches the expected brewing temperature
:action *brew-coffee(c, w)*	Starts the *brewing(c, w)* process with coffee *c* and hot water *w*
:process *brewing(c, w)*	Brews coffee *c* by calculating the extraction yield with the increment of brewing time. Assuming that brewing temperature of water *w* is constant
:event *stop-brewing(c, w)*	Stops the *brewing(c, w)* process when the expected yield is acquired
:action *serve-coffee(c, w)*	Terminates the whole coffee making procedure once the coffee *c* is expectedly brewed with water *w*

Table 6.2 PDDL+ modelling components of Learned coffee domain

This thesis analyses the coffee data that has collected from the observation study conducted by Easthope (2015). The aim is to induce the PM for extraction yield, which is then embedded in the *brewing(c,w)* process of reformulated PDDL+ model. The learned coffee model with PM (given in Appendix B.1) estimates the value of extraction yield based on the basic learning factors such as brew time and brew temperature. As a consequence, it can automatically adjust the amount of extracted coffee yield during the brewing process. Subsequently, it has extended the scope of deploying other influencing factors in the coffee brewing process. For instance, pump pressure, tamp pressure, roasting degree etc. Next section 6.1.4 explains the observational study of coffee brewing process and exposes the results derived by Easthope (2015).

6.1.4 Observational Study

In the study conducted by Easthope (2015), espresso shots (22 grams each) were sampled for each temperature: 92, 94, 96, and 98 degrees Celsius given in the table 6.3, 6.4, 6.5 and 6.6 correspondingly. Table 6.3 to 6.6 summarise the data for extraction temperature along with the brewing time and corresponding extraction yield, beverage, TDS and dose. Here, TDS (Total Dissolved Solid) percentage decides the strength of

the coffee, Dose indicates the coffee ground in grams and Yield percentage is the total amount of extracted element from the ground coffee. Yield percentage is calculated from beverage, TDS and dose by the given equation 6.1:

$$yield[\%] = beverage[g] * (\frac{TDS[\%]}{dose[g]}) \tag{6.1}$$

Cup	Temperature(°C)	Beverage(g)	Brew Time(sec)	TDS	Yield%	Dose(g)
1	92	48.3	29	8.41	18.4	22
2	92	46.6	28	8.7	18.4	22
3	92	48.2	29	8.53	18.8	22
4	92	46.2	30	9.26	19.3	22
5	92	47.1	30	9	19.3	22
6	92	47.1	32	9.11	19.5	22

Table 6.3 Data for 92°C extraction temperature

Cup	Temperature(°C)	Beverage(g)	Brew Time(sec)	TDS	Yield%	Dose(g)
1	94	46.7	29	9.09	19.3	22
2	94	46.2	30	9.22	19.3	22
3	94	47.1	32	9.01	19.3	22
4	94	46	31	9.28	19.4	22
5	94	45.4	30	9.39	19.4	22
6	94	47.8	30	9.03	19.5	22

Table 6.4 Data for 94°C extraction temperature

Cup	Temperature(°C)	Beverage(g)	Brew Time(sec)	TDS	Yield%	Dose(g)
1	96	44.8	29	9.43	19.1	22
2	96	45	29	9.43	19.2	22
3	96	46.1	30	9.24	19.3	22
4	96	45	29	9.49	19.4	22
5	96	46.2	30	9.41	19.7	22
6	96	47.2	30	9.23	19.7	22

Table 6.5 Data for 96°C extraction temperature

Cup	Temperature(°C)	Beverage(g)	Brew Time(sec)	TDS	Yield%	Dose(g)
1	98	46	28	9.25	19.2	22
2	98	47.2	29	9	19.3	22
3	98	47	30	9.08	19.4	22
4	98	48	31	8.89	19.4	22
5	98	48.9	31	9.03	20	22
6	98	48.6	30	9.06	20.1	22

Table 6.6 Data for 98°C extraction temperature

According to the study carried out by Easthope (2015) and SCA (2018), the ideal $yield$ percentage is between 18% - 22%. The study reveals that $yield\%$ raises consistently with the increment of brewing temperature. Figure 6.1 illustrates the upward trend of extraction $yield\%$ by estimating the average extraction $yield\%$ relative to brewing temperature shown in the table 6.7

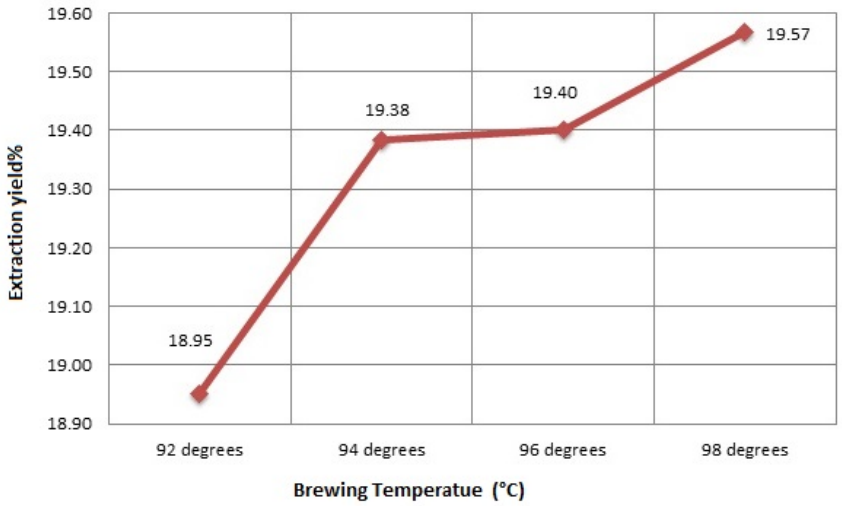


Fig. 6.1 Effects of brewing temperature on $yield\%$ (Easthope 2015)

Water Temperature (°C)	Yield Percentage (%)
92	18.95
94	19.38
96	19.40
98	19.57

Table 6.7 Average extraction $yield(\%)$ relative to brewing temperature (°C) in espresso

Aside from the extraction, brewing temperature also affects the taste of espresso shot (Korhonen 2019). There are three components extracted from the coffee grounds that determine the taste of the coffee such as caffeine (dissolved in water), volatile oils (evaporate into air for flavour and aroma) and organic acids (bring bitter taste). Based

on the research conducted by Easthope (2015), the water temperature to brew coffee grounds affects the taste of coffee as follows (table 6.8):

Water Temperature (°C)	Espresso taste
92	The coffee has high acidity but with lower body, sweetness and bitterness
98	It has medium acidity with lots of sweetness, bitterness, full body and with strange powdery mouthfeel
94 - 96	With balanced acid, high levels of sweetness, good body, and low bitterness

Table 6.8 The brewing temperature (°C) affects the taste of coffee

Besides, the brewing time affects the extraction yield for instance longer extraction time results a higher extraction yields, whereas shorter brewing time outcomes lower extraction yields (Roman Corrochano 2017). As a consequence, the taste of espresso is also influenced by the brewing time as shown in the figure 6.2 (Brushett 2014).

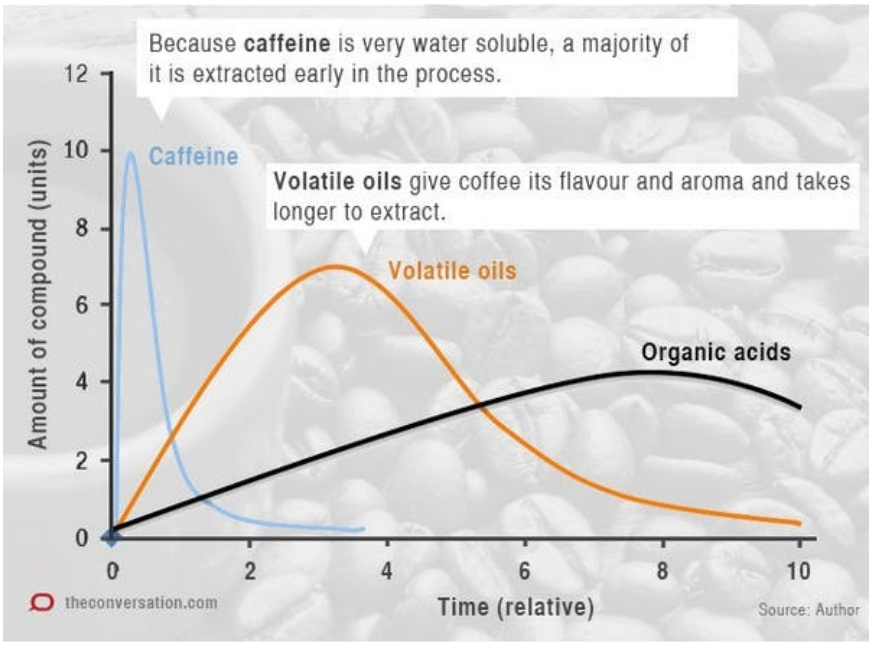


Fig. 6.2 Brewing time (extraction time) vs compounds extracted (Brushett 2014)

6.1.5 Application of PMI method for the Coffee Domain

This section explains the procedure of PMI method (described in chapter 5), that have been carried out, for learning and implementing the PM in coffee domain.

6.1.5.1 Data Collection

As a first step of PMI method (mentioned in section 5.3.1) for formulating the Process Model (PM), the statistical data is collected manually from the observational study, that has conducted by Easthope (2015). The raw data includes the forty sample reading of $22g$ espresso shots, where each ten shots has been prepared at brew temperature $92°$C, $94°$C, $96°$C and $98°$C respectively. For each espresso shot, the beverage weight (between 46-48 g) and the brew time or shot time (between 28 – 32 $seconds$) have been recorded in the data. Besides, the TDS (Total Dissolved Solid) readings are given in the data that are used to calculate the corresponding extraction yield (shown in the equation 6.1). Besides, Easthope (2015) has tasted and recorded the samples with estimated extraction yield in the observational data (summarised in the table 6.7 and 6.8). The raw data are categorised into four temperature group and is sorted by the extraction yield%. To avoid inconsistent extraction behaviour, Easthope (2015) removes the outliers (two lowest and highest data points) from the observational data. The dataset with reduced sample size is provided in the tables 6.3 - 6.6.

6.1.5.2 Data Preparation

After collecting the raw data, the data needs to be prepared that can be efficiently exploited by ML techniques in order to build PM with appropriate feature approximations. At first, the data for four temperature groups are merged, that helps to analyse all data in one go. Next, the irrelevant or inconsistent feature columns are removed from the dataset that can degrade the quality of PM. For example, the constant value of Dose(g), the weight of Beverage(g) and the reading of TDS which are only used to calculate yield% (stated in the equation 6.1). After cleaning the data, the PMI automatically split the entire dataset into two parts: training and testing sets using the Cross-Validation (CV) method (mentioned in section 5.3.2). It implements the K-fold Cross-Validation approach (discussed in section 4.3.4) by utilising the scikit-learn (or sklearn) library in a Python program. Due to the small sample size, the entire dataset is divided into 5 folds. In 5-Fold CV (K=5), the formulation of regression model repeats 5 times with 1 distinct fold as testing data and the rest 4 folds as training data (as illustrated in figure 4.3). The best fitted model is automatically selected based on the lowest MSE (Mean Squared Error) value among all resulting models, which is finally represented by the PM (detailed discussion in the next section).

6.1.5.3 Formulation of PM

From the data preparation step, we get the prepared data that can be accurately analysed by exploiting the ML techniques in order to formulate the Process Model (PM). The input data contains three quantitative features such as Brew Temperature(°C), Brew

Time(sec) and corresponding Yield%. For identifying the relationship among those continuous variables, the PMI method performs the correlation analysis (mentioned in section 5.3.3.1). On the basis of resulting correlation coefficient, the appropriate regression analysis is carried out to construct the PM. The output of correlation and regression analysis are demonstrated below, along with the statistical methods and ML techniques that have applied.

Correlation Analysis

For estimating the correlation coefficient (r), the PMI method executes the Pearson Correlation Coefficient (PCC) test on the input data (discussed in section 5.3.3.1), where the dependent variable (y) is yield% and the independent variables (X) are Brew Temperature ($x1$) and Brew Time ($x2$). Table 6.9 exhibits the outcome of correlation test which contains the values of Pearson's r such as $r_{x_1,y}$, $r_{x_2,y}$ and r_{x_1,x_2}. The magnitude of relationship between y and x variables is denoted by $r_{x_1,y}$ and $r_{x_2,y}$ correspondingly. Whereas, r_{x_1,x_2} indicates the strength of interrelation between two x variables.

	Yield% y	Brew Temperature(°C) x_1	Brew Time(sec) x_2
Yield%	1		
Brew Temperature(°C)	0.551416365	1	
Brew Time(sec)	0.529255767	−0.036273813	1

Table 6.9 Result of Correlation test (Pearson's r) for Coffee domain

The strength of linear relationship based on r value
$0.1 < \|r\| < 0.3$: small or weak correlation
$0.3 < \|r\| < 0.5$: medium or moderate correlation
$0.5 < \|r\|$: large or strong correlation

Table 6.10 The general guidelines of assessing the relationship based on Pearson's r value (Cohen 2013)

From table 6.9, we get the PCC values such as $r_{x_1,y} = 0.551416365$, $r_{x_2,y} = 0.529255767$ and $r_{x_1,x_2} = -0.036273813$. By assessing those values according to the general guidelines (stated in table 6.10), PMI method infers that yield% is highly dependent on the brew temperature and brew time with the respective values of $r_{x_1,y}$ and $r_{x_2,y}$ greater than 0.5 (green highlights in table 6.9). Besides, the absolute value of $r_{x_1,x_2} < 0.1$ (red highlight in table 6.9) signifies that brew temperature and brew time are not correlated. In conclusion, the fluctuation in brew temperature and brew time affect the value of yield%, where brew temperature and brew time are independent of each other.

Regression Analysis

After interpreting the results of correlation test, the PMI method nominates the Multiple Linear Regression (MLR) technique to formulate PM, where y is related to multiple x-variables (i.e. Case 2 in section 5.3.3.1). In other words, the outcome variable, yield% depends on two non-correlated predictor variables i.e. brew temperature and brew time. To formulate a PM, PMI method performs the stepwise regression with forward selection, that identifies and adds the most effective predictor variables in the model based on specified criterion (detailed in section 4.5.2). Here, the adjusted R-squared is used as variable selection criterion for achieving the optimal fitted linear model. We have specifically developed a python program, where the statsmodels module (Seabold and Perktold 2010) are used, that facilitates the PMI method to automatically explore the statistical data and estimate the statistical models.

As mentioned in earlier section 6.1.5.2, the input dataset is divided into two parts: training set and testing set using the 5-Fold cross validation method. The MLR technique exploits the training set to learn the regression model by approximating the regression coefficients or parameters of the model. The aim is to predict the value of outcome variable (yield%) for a given value of predictor variables such as brew temperature and brew time. The testing set is used to measure the prediction accuracy of the model in terms of Mean Squared Error (MSE). MSE is an average value of the squared errors where error is the difference between the actual and estimated value of the outcome variable. The PMI method computes the MSE value using the built-in function "*mean_squared_error*" that is imported from *sklearn.metrics* python module.

The entire procedure of constructing regression model repeats until each fold is used as a testing set at some point. In each iteration, the MLR technique with forward selection is executed to specify the model by determining and including the most effective predictor variables in the regression equation. Besides the parameters of regression model are estimated, for instance intercept, β_0 and the regression coefficient of corresponding predictor variables, β_1, β_2 etc. Finally, the best fitted model with lowest MSE is selected among all resulting models. Table - 6.11 provides the estimated parameter values of regression model in each fold along with corresponding MSE and RMSE (defined in equation 5.3) values.

KFold	Intercept β_0	Coefficient of x_1 β_1	Coefficient of x_2 β_2	MSE	RMSE
1	9.500395	0.063336	0.129335	0.171047	0.41357
2	2.17130	0.08921	0.29181	0.10417	0.32276
3	3.597629	0.103934	0.196035	**0.0280**	0.1673
4	4.296596	0.094365	0.202254	0.039457	0.19863
5	2.684641	0.105094	0.223383	0.13012	0.3607

Table 6.11 Estimated Regression Coefficients for Coffee domain in each fold of K-folds cross validation (where K = 5)

From Table - 6.11, we find the final regression equation with lowest MSE value of 0.0280 where the intercept (β_0) value is 3.597629 and the regression coefficient of brew temperature (β_1) and brew time (β_2) is 0.103934 and 0.196035 respectively. Therefore, the mathematical equation of the formulated regression model can be written as follows (equation 6.2).

$$y = \beta_0 + (\beta_1 * x_1) + (\beta_2 * x_2) \tag{6.2}$$

where, y = extracted yield, x_1 = brew temperature and x_2 = brew time
To assess the significance of predictors in the regression model (equation 6.2), PMI method conducts the t-test, where the predictor with a p-value less than 0.05 indicates at least a 95% chance of true relationship between outcome and predictor variable in the population. Besides, the ANOVA test (F statistic) is performed to evaluate the statistical significance of entire regression model. From ANOVA test, p-value less than the significance level of 0.05 suggests that the regression model fits the data. In a nutshell, the resulting regression model with p-values < 0.05, formulates the Process Model (PM). Table – 6.12 summaries the output of statistical tests. It exhibits that the p-values from both F-test and t-test are less than 0.05 (green highlights in the table). Therefore we can conclude that the estimated regression model (highlighted with green colour in table 6.11) signifies the PM for extraction yield.

Table 6.12 Statistical Significance Tests for Regression (Coffee domain)

Regression Statistics	
Multiple R	0.795877296
R Square	0.63342067
Adjusted R Square	0.587598254
Standard Error	0.27897268
Observations	19

ANOVA

	df	*SS*	*MS*	*F*	*P-value*
Regression	2	2.151630003	1.075815001	13.82337996	**0.000326095**
Residual	16	1.245212103	0.077825756		
Total	18	3.396842105			

T-test

	Coefficients	*Standard Error*	*t Stat*	*P-value*
Intercept	3.597628763	3.008066652	1.195993699	0.249123121
Brew Temp. (°C)	0.103934098	0.026020344	3.99433985	**0.00104438**
Brew Time (sec)	0.196035107	0.05719659	3.427391498	**0.003454577**

6.1.5.4 Integration of PM

The resulting PM explains the cause and effect relationship between outcome (i.e. extraction yield) and predictors (e.g. brew temperature and brew time) in the coffee brewing process. It automatically estimates the amount of extraction yield with the variations in brew temperature and brew time. The mathematical formula of PM is given in the equation 6.2, where the approximated parameter values (i.e. β_0, β_1 and β_2) are provided in the table 6.11 (KFold 3).

The learned PM is integrated and adjusted into the "*brewing*" process description of previously engineered, coffee domain (given in Appendix B.1). Figure 6.3 shows the "*brewing*" process with embedded PM. For a given brew temperature, it calculates the value of extraction yield with the variation in brew time, where brew time is incremented using the **#t** literal. Also, assuming that brew temperature is constant during the brewing process.

```
(:process brewing
:parameters (?c - coffee ?w - water)
:precondition (and (brewing ?c ?w))
:effect (and
            ;;** brewing process with PM of extraction yield ***
            ;;Assume, the temperature is constant during brewing
             (assign (yield ?c ?w)
                     (+ (beta0 ?c ?w)
                        (+ (* (beta1 ?c ?w) (temperature ?w))
                           (* (beta2 ?c ?w) (brew-time ?c ?w))
                        )
                     )
             )

             (increase (brew-time ?c ?w) (* #t 1))
        )
)
```

Fig. 6.3 Brewing process with PM in the coffee domain

6.2 Urban Traffic Control(UTC) Domain

With the increasing number of vehicles in urban road network, it has become a challenge to manage and control traffic system in a reasonable and effective way. There has been continuous progress in developing traffic signal control system with the evolution in science and technology. Basically, three main strategies are followed by UTC for controlling traffic congestion in normal situation i.e. Fixed-time multiple periods (Little et al. 1981), Model-based Predictive (Papageorgiou et al. 2007) and Adaptive traffic signal control system like SCATS (Lowrie and PR 1982) and SCOOT (Bretherton 1990). Those techniques generally depend on the complex mathematical models and predefined policies which are not compatible with unexpected UTC situations (Wei et al. 2019).
Recently, Automated Planning (AP) has shown its eligibility to apply in UTC applications, with the advancement of AI planning technology (Cenamor et al. 2014, Vallati et al. 2016). The AP based traffic control system can model the UTC problem using a declarative language in conjunction with powerful reasoning engines. Therefore, new actions along with sensor information can be easily added/modified in the model, which can keep the UTC model up-to-date in accordance with traffic conditions.

The UK research council funded SimplifAI (McCluskey, Vallati and Franco 2017) is a project spanning several years aimed at deploy AI Planning to help in Urban Traffic Management Control (UTMC). In SimplifAI, Vallati et al. (2016) introduces a hybrid planning based traffic control system with PDDL+ representation to control vehicle flow in both normal and exceptional road situations. The main focus is to deal with unforeseen traffic congestion during uncertain events such as road accidents/repair, nat-

ural calamities etc. In particular, the proposed PDDL+ encoding of UTC problem can control the traffic green phase on the basis of road network congestion. Vallati et al. (2016) emphasis on the macroscopic model of UTC rather than using microscopic model based planning approach. In microscopic model, each vehicle is represented individually along with detailed description which has limited scalability to exploit plans (Chrpa et al. 2016). On the other hand, macroscopic model overcomes the mentioned limitation by considering the flow of vehicles on a road network to cover large areas. The effectiveness of such hybrid planning-based approach with PDDL+ formulation of UTC domain has been discussed by McCluskey and Vallati (2017).

Pozanco et al. (2018) proposes an automated planning based traffic control system, named APTC (Automated Planning for Traffic Control). It overcomes two main difficulties of implementing planning systems in UTC: (1) engineering of traffic control model to exactly resemble the current road situations; and (2) the scalability of planning algorithms to produce plans in accordance with dynamic events and road diversity. To solve the aforementioned problems, the proposed system continuously learns a domain knowledge to get an accurate planning model as well as applies distributed planning approach to divide the large city network into a set of areas. Therefore, the planning problem is distributed with respect to individual areas, which is solved asynchronously. In APTC model, a traffic domain is encoded as a discrete model with simple PDDL representation based on the UTC model in IAS (Intelligent Autonomic System) developed by Gulić et al. (2016). Besides, it follows domain-dependent approach, where actions are executed to handle traffic lights according to the street density level, rather than the traffic flow. If a high density is detected, it activates the actions to turn green light on for a fixed time, leading towards the reduction of density level.

In this thesis, the UK research council funded SimplifAI project is being used as a main case study. This section (6.2) describes the basic model for UTC problem, along with the assumptions that are made in the SimplifAI. Besides, it details the PDDL+ construct that regulates the road junctions. It then highlights the rationale for using PM in the UTC domain, concluding with the application of PMI method for the UTC domain.

6.2.1 Basic Model for UTC problem

In UTC problem, a road network can be represented by a directed graph, where the edges represent the road sections and the vertices symbolise the road junctions (or intersections) that are the entry/exit points of connected road sections. Vehicles enter a region of the road network through the entry point of junction and leave from the exit point of that junction. Each road section has a maximum capacity which specifies the

maximum number of vehicles that a road section can serve. The occupancy (or queue) specifies the number of existing vehicles on a road section. Figure 6.4 depicts a road network of two junctions J_1 and J_2 with eleven connected road sections i.e. $road_1$ to $road_{11}$. The direction of arrows indicates the incoming and outgoing traffic flow for the road section $road_{10}$. Vehicles can enter the $road_{10}$ from $road_1$ and $road_{11}$ as well as can egress the $road_{10}$ by heading three different directions e.g. $road_5$, $road_7$ and $road_9$. $q(road_{10}, J_2)$ defines the occupancy (or queue) of the $road_{10}$ section which is the traffic congestion in the $road_{10}$ section towards the junction J_2.

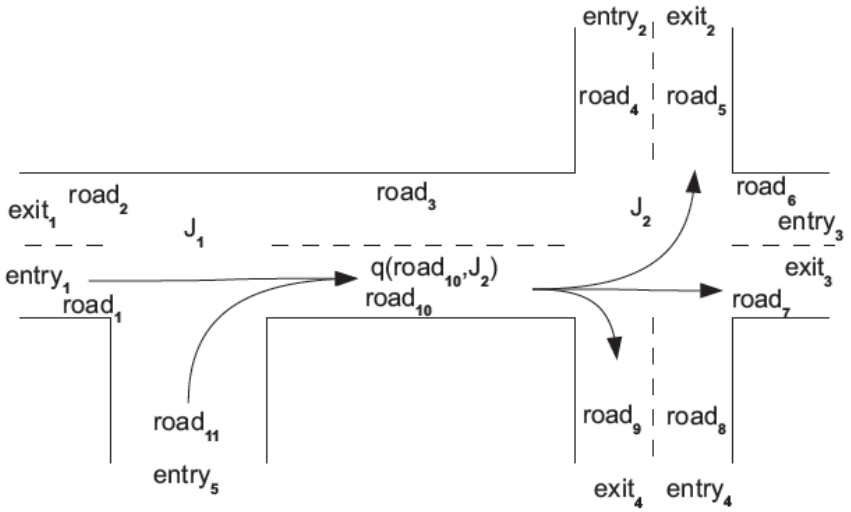


Fig. 6.4 An example of a road network in UTC model (Vallati et al. 2016)

6.2.2 Model Assumptions

In SimplifAI, the PDDL+ formulation of UTC domain only considers the junctions with connected road sections to construct the basic model of road network. Where, vehicle flow in each junction are regulated by the corresponding traffic lights. Besides, it is assumed that the flow of vehicles follows the correct lane without blocking other vehicles heading towards the different directions. Traffic congestion in each junction is measured by the flow rate between incoming and outgoing road section or road link. Flow rate, also called the turn rate, is the number of vehicles per unit time that leave a road link, r_1, pass through the associated junction, J and enter another road link, r_2 (expressed in equation 6.3). r_1 and r_2 denote the incoming and outgoing road link of the junction J respectively.

$$turnrate = \frac{n}{t} \tag{6.3}$$

Where, turnrate = traffic flow (veh/sec)

n = number of vehicles passing through r_1 to r_2 during time t

t = duration of time interval (sec)

The traffic flow between r_1 and r_2 is activated when the signal phase (i.e. green light

phase) of respective junction J is on. Each junction contains a specific number of sequential signal phases (or stages) which allows the traffic flow between connected road links. For each signal phase, the minimum and maximum phase time (green time of traffic light) along with the inter limit (amber time of traffic light) are defined. Within those ranges, the planner can handle the phase activation/deactivation based on the traffic congestion. For instance, $n3969$ is a road junction which is connected with 4 other junctions e.g. $n3972$, $n7646$, $n6612$ and $n3968$ through the road links illustrated in figure 6.5 below. It has three signal phases: $s3969\_s0$, $s3969\_s1$ and $s3969\_s2$ that triggers consecutively with a specified interval (intergreen or amber light time). The green arrow directs the traffic flow of connected road links in each signal phase.

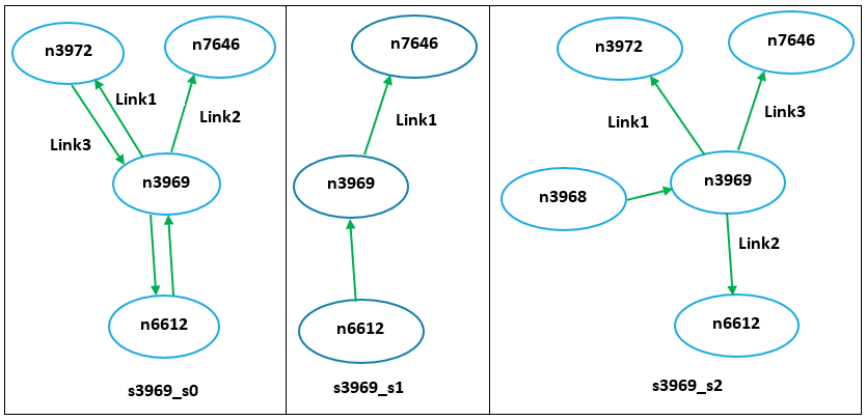


Fig. 6.5 Signal phases of junction $n3969$: Phase $s0$, $s1$ and $s2$.

6.2.3 Modelled Area and Experimental Data

For modelling the road network with PDDL+ encoding, Vallati et al. (2016) contemplates the Manchester (UK) urban area (figure 6.6). Figure 6.6 highlights some road junctions (red dots) in Greater Manchester which are used for modelling UTC domain. Due to the complex structure of junctions and the memory scalability issues of adjusting enormous junctions, the UTC model includes a specified set of junctions, named as controllable junctions, during plan generation (McCluskey and Vallati 2017). For instance, figure 6.7 illustrates an abstract view of controllable road junctions that is a part of Manchester (UK) urban region (depicted in figure 6.6) and is employed in the UTC model.

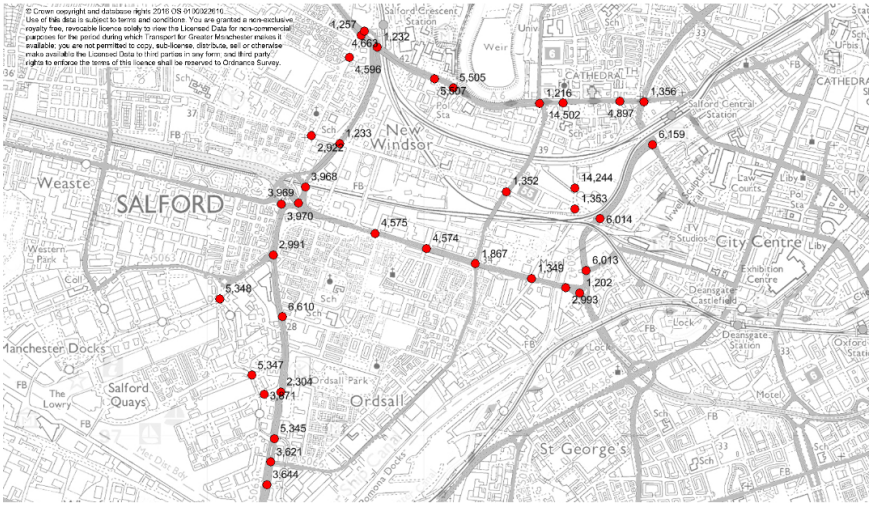


Fig. 6.6 A part of modelled area with active road junctions (red dots) in Greater Manchester (Image sourced from Transport for Greater Manchester, 2018).

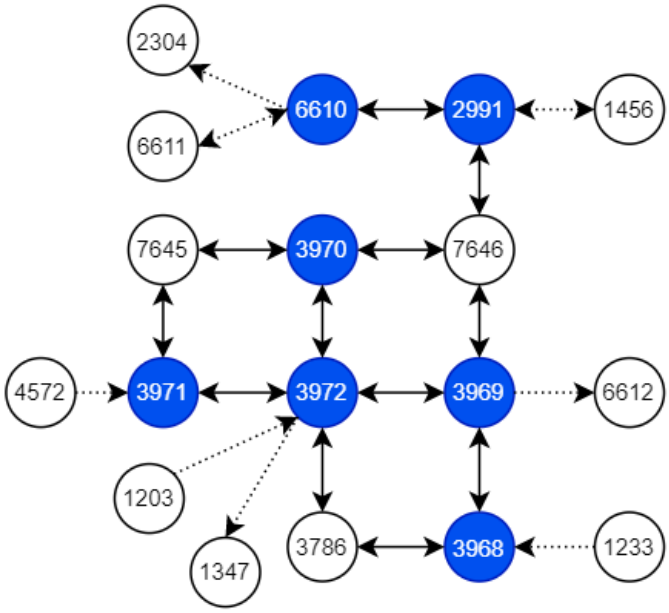


Fig. 6.7 Abstract view of UTC region with controllable junctions (Blue Vertices). The direction of arrows indicate the traffic flow.

In SimplifAI project, the historical traffic data of selected urban regions have been exploited which contain the topology of road links, vehicle capacity of all links, traffic flow rate between road sections, minimum-maximum green time of signal phase, traffic signal position (active or inter), intergreen time between phases of signals etc. These data items are encoded in the traffic model wherein the numerical constraint of occupancy level is set as goal. The ENHSP (Scala et al. 2016) PDDL+ planner is adopted to produce a signal plan P which contains the switching sequence of signal phases with time i.e. the deactivation of current phase and the activation of next phase in a junction. An industry-standard, proprietary simulator called AIMSUN (Barceló and Casas 2005)

is used to execute the plan P from a given initial state, I. At any time T assuming that the state of every junction agrees with the planning simulation of P from state I.

6.2.4 PDDL+ constructs to regulate road junctions

For regulating the traffic flow in each junction (or intersection) of a road network, the PDDL+ modelling components with functionalities are discussed in the table 6.13 below. The complete PDDL+ model with domain and problem definition is given in Appendix C.1.

Table 6.13 PDDL+ modelling components of Original UTC domain (McCluskey and Vallati 2017)

Modelling components	Functionalities
:action *switchPhase (p, i)*	The planner takes this action when a phase, *p* of a controllable junction, *i* is active (i.e. green time is on) and the green time reaches the given limit of minimum phase time. As a consequence, it enables the event *trigger-inter (p, i)*.
:event *maxgreenreached (p ,i)*	It activates when the green time of active phase, *p* in a junction, *i* exceeds the given limit of maximum phase time. Consequently, it allows the event *trigger-inter (p, i)* to execute.
:event *trigger-inter (p, i)*	It deactivates the current phase, *p* of junction, *i* by turning on the intergreen phase (i.e. amber light phase) of *p*. Besides, it resets the green time counter of junction, *i* to zero.
:process *keepinter (p, i)*	It starts counting the intertime (i.e. amber light time) when the phase, *p* of junction, *i* is in intergreen position. It stops once exceeds the given limit of intertime.

Continued on next page

Table 6.13 – continued from previous page

Modelling components	Functionalities
:event *trigger-change (p, i)*	It is turned on when the intertime of intergreen phase, *p* surpasses the inter limit. As an effect, it switches the signal phase from *p* to *p1* by deactivating the intergreen phase, *p* and activating the phase, *p1*.
:process *keepgreen (p, i)*	It starts counting the greentime (i.e. green light time) when the phase, *p* of junction, *i* is in active position. It stops once the greentime exceeds the maximum phase time or the planner deactivates the current phase, *p*.
:process *flowrun_green (p, r1, r2)*	It is executed simultaneously with the *keepgreen (p, i)* process when the phase, *p* of junction, *i* is activated (i.e. green time is on). It allows the vehicles to move from road, *r1* to road, *r2* at the given flow rate (non-zero value) till *r1* has no vehicle or *r2* exceeds its capacity (shown in figure 1.2).

6.2.5 Rationale for using PM to improve UTC domain

In the original formulation of UTC domain with PDDL+ constructs (table 6.13), the dynamic state of a road network at any instant T consists of:
A - occupancy of every road section i.e. incoming/outgoing vehicles of a road section
B - "active" or "intergreen" situation of the current phase for every signalised junction
Besides, it contains the static knowledge such as link/junction connections, capacity of road section, and assumed traffic flows. Other static knowledge that could put in PDDL+ model are missing at the moment e.g. link lengths, or number of lanes in a link. Moreover, the occupancy (mentioned in **A** above) is generally inadequate due to the coarse description of process for traffic flow between links. As well as, other assumptions that the domain model makes are unfeasible, for instance, assuming that the traffic flows across a link instantaneously. Specifically, the flow rate (or turn-rate) of ve-

hicles between two road sections is predefined as a static value which took no account of the changing features of the road network.

This thesis analyses the traffic data taken from AIMSUN in the form of variations in traffic flow over a phase of the junction. The aim is to induce the PM of the turnrate (i.e. the vehicle flow rate between road links) which is then embedded in the *flowrun_green (p, r1, r2)* process of original PDDL+ formulation. The learned model of UTC domain can regulate the turnrate value based on different learning factors such as green time, inter time and incoming/outgoing road saturation, which reflects the difference between road links. As a result, it estimates the more accurate and representative value of the turnrate automatically without having to declare it statically. At the same time, it has broaden the scope of employing other influencing factors in the UTC model. For example the speed limit, cross flow, the number of non-motor vehicles and the density of bus stops can affect the traffic flow rate in the urban roads (He and Zhao 2013).

6.2.6 Application of PMI method for the UTC Domain

This section explicates the steps of acquiring and implementing Process Models (PM) in the UTC domain.

6.2.6.1 Data Collection

The first step of PMI, for constructing the PM, is to collect traffic data from AIMSUN (Barceló and Casas 2005) simulator that has been utilised in SimplifAI project. The AIMSUN provides traffic simulation models of road junctions. A traffic model represents the road traffic state, for instance, controllable junctions (or the active connected links), signal phases (or stages) of each junction, sequence of signal phases, status (i.e. active or inter) of signal phases, green time/inter time of junctions, turnrate of road links, occupancy (i.e. number of existing vehicles) and vehicle capacity of road links etc.
To empirically analyse the UTC domain, we exemplify a set of active (also known as controllable) junctions of selected urban region as illustrated with the blue vertices in figure 6.7 . Traffic situations of that particular region are observed by AIMSUN for one hour (3600 seconds) period. The traffic observations (i.e. traffic simulation models) are recorded for every 5 seconds. From the recorded traffic models, we extract the traffic data for each signal phase of a junction.

6.2.6.2 Data Preparation

After collecting the traffic data, the PMI method prepares the data that can be analysed and exploited appropriately in order to formulate the PM. The irrelevant or inconsistent features are removed from the data, for instance, the inter time/occupancy with all zero values, the outlier value of road capacity etc. Finally, we will get the refined data that contains the green time, inter time, turnrate and incoming/outgoing road saturation for each road-link.

For each link of a signal phase, the PMI method divides the data into two parts: training and testing sets applying the K-fold Cross-Validation (CV) method. Due to the large size of sample data, the data is divided into 2 folds. In 2-Fold CV (K=2), the formulation of regression model repeats 2 times with the swapped data-sets (i.e. training and testing). Where, training set is exploited by the statistical methods and ML techniques to induce the regression model. On the other side, testing set is employed to measure the Mean Square Error (MSE) of resulting model. The MSE value quantifies that how well the model fits new data. Besides the best fitted model is selected with the lowest MSE value among resulting models.

6.2.6.3 Formulation of PM

From the data pre-processing steps e.g. organising, refining and encoding, the data is prepared for applying statistical methods and ML techniques. The aim is to induce PM of turnrate that automatically measures the vehicle flow rate of each link in the corresponding signal phase.

To demonstrate and report the empirical results of correlation and regression analysis (detailed in next chapter 7), we select the junction $n3969$ which has three signal phases $s0$, $s1$ and $s2$ (depicted in figure 6.5). Table - 6.14 lists the connected links of junction $n3969$ that are activated when the corresponding signal phase is ON. For the sake of brevity, we present the full experimental results for a particular junction $n3969$. The outcome of correlation and regression analysis are described below, along with the statistical methods and ML techniques that have been applied by PMI method.

Link number	Signal phase	Incoming road ($road1$)	Outgoing road ($road2$)
$s0\_link1$	$s3969\_s0$	$n6612\_n3969$	$n3969\_n3972$
$s0\_link2$	$s3969\_s0$	$n6612\_n3969$	$n3969\_n7646$
$s0\_link3$	$s3969\_s0$	$n3972\_n3969$	$n3969\_n6612$
$s1\_link1$	$s3969\_s1$	$n6612\_n3969$	$n3969\_n7646$
$s2\_link1$	$s3969\_s2$	$n3968\_n3969$	$n3969\_n3972$
$s2\_link2$	$s3969\_s2$	$n3968\_n3969$	$n3969\_n6612$
$s2\_link3$	$s3969\_s2$	$n3968\_n3969$	$n3969\_n7646$

Table 6.14 Active links in corresponding signal phases ($s0$, $s1$ and $s2$) of junction $n3969$

Correlation Analysis

The Pearson Correlation Coefficient (PCC) method is applied to estimate the the correlation coefficient (r) from the input data. Where the dependent variable (y) is turnrate and the independent variables (X) are greentime ($x1$), intertime ($x2$), road1 saturation ($x3$) and road2 saturation ($x4$). The PCC test results for each signal phase are summarised below with figure 6.8, 6.9 and 6.10 consecutively. Where, Pearson's correlation coefficient between y and each x variable is denoted by $r_{x1,y}$, $r_{x2,y}$, $r_{x3,y}$ and $r_{x4,y}$ consecutively. Besides, the correlation between two different x variables is signified by r_{x_i,x_j}. By following the general guidelines (defined in table 6.10), PMI method assesses the PCC values for each link to infer relationship among variables. The presence of correlation between two variables is highlighted with green colour in the PCC tables (shown in figure 6.8, 6.9 and 6.10).

Signal phase $s0$

s0_link1	Turnrate (y)	Greentime (x1)	Intertime (x2)	Road1_Sat (x3)
Greentime (x1)	-0.531443386			
Intertime (x2)	0.00353702	-0.187231537		
Road1_Sat (x3)	0.318631236	-0.76003397	-0.100831896	
Road2_Sat (x4)	0.481740059	-0.405333726	-0.064289407	0.093616259

(a) Pearson's r for $link1$ in phase $s0$

s0_link2	Turnrate (y)	Greentime (x1)	Intertime (x2)	Road1_Sat (x3)
Greentime (x1)	-0.044204225			
Intertime (x2)	-0.018497181	-0.187231537		
Road1_Sat (x3)	-0.075369387	-0.76003397	-0.100831896	
Road2_Sat (x4)	-0.251894306	-0.062018813	-0.026253295	0.161620153

(b) Pearson's r for $link2$ in phase $s0$

s0_link3	Turnrate (y)	Greentime (x1)	Intertime (x2)	Road1_Sat (x3)
Greentime (x1)	-0.275263941			
Intertime (x2)	-0.036442848	-0.187231537		
Road1_Sat (x3)	0.619691994	-0.367450258	-0.050701925	
Road2_Sat (x4)	0.390038438	0.06317072	-0.052069641	0.158197155

(c) Pearson's r for $link3$ in phase $s0$

Fig. 6.8 Result of Correlation test (PCC) for signal phase $s0$ in UTC domain

Figure 6.8 demonstrates the result of correlation (PCC) test for each link in signal phase $s0$. PMI method infers that turnrate of link1 is dependent on greentime, road1 and road2 saturation with the corresponding $r_{x,y}$ value greater than 0.3, where road1 and road2 saturation are correlated with greentime. For link2, the turnrate is only related with road2 saturation, whereas the value $r_{x1,x3} > 0.5$ indicates the strong relationship between greentime and road1 saturation. For link3, turnrate depends on greentime, road1 and road2 saturation, where greentime and road1 saturation are moderately correlated (i.e. $0.3 < r_{x1,x3} < 0.5$).

Signal phase $s1$

s1_link1	Turnrate (y)	Greentime (x1)	Intertime (x2)	Road1_Sat (x3)
Greentime (x1)	-0.129194665			
Intertime (x2)	-0.02511649	-0.305750689		
Road1_Sat (x3)	-0.2077176	0.732870621	0.267023304	
Road2_Sat (x4)	-0.325241661	0.043359352	0.004090214	0.08543857

Fig. 6.9 Result of Correlation test (PCC) for signal phase $s1$ in UTC domain

From figure 6.9, we get the PCC values for signal phase $s1$ with an active link. Here, turnrate is correlated with only road2 saturation. Road1 saturation correlates with greentime and intertime.

Signal phase $s2$

s2_link1	Turnrate (y)	Greentime (x1)	Intertime (x2)	Road1_Sat (x3)
Greentime (x1)	0.400496241			
Intertime (x2)	-0.505455458	-0.618590075		
Road1_Sat (x3)	-0.144840295	0.042738125	-0.412536626	
Road2_Sat (x4)	-0.202427998	-0.234398052	0.462131848	-0.696344236

(a) Pearson's r for $link1$ in phase $s2$

s2_link2	Turnrate (y)	Greentime (x1)	Intertime (x2)	Road1_Sat (x3)
Greentime (x1)	0.28927527			
Intertime (x2)	-0.304516447	-0.618590075		
Road1_Sat (x3)	-0.058640061	0.042738125	-0.412536626	
Road2_Sat (x4)	0.297844977	0.072867346	0.261445726	-0.200097832

(b) Pearson's r for $link2$ in phase $s2$

s2_link3	Turnrate (y)	Greentime (x1)	Intertime (x2)	Road1_Sat (x3)
Greentime (x1)	0.057801211			
Intertime (x2)	-0.277004963	-0.618590075		
Road1_Sat (x3)	0.007023965	0.042738125	-0.412536626	
Road2_Sat (x4)	-0.120019545	0.056947116	0.073572844	-0.044270309

(c) Pearson's r for $link3$ in phase $s2$

Fig. 6.10 Result of Correlation test (PCC) for signal phase $s2$ in UTC domain

Figure 6.10 illustrates the PCC values of each link for signal phase $s2$. PMI method infers that turnrate of link1 depends on greentime, intertime and road2 saturation (table in figure 6.10a), where, road1 and road2 saturation are strongly correlated (i.e. $|r_{x3,x4}| > 0.5$). For link2, turnrate has relationship with greentime, intertime and road2 saturation, where, road1 saturation correlates with intertime. The turnrate of link3 is moderately correlated with intertime, whereas, road1 saturation has relationship with intertime (i.e. $|r_{x2,x3}| > 0.3$).

In conclusion, it is clear that turnrate of each road-link is either moderately or highly dependent on one or more independent variables (or predictors), where the predictors have (medium/strong) correlation with each other.

Regression Analysis

From correlation analysis, PMI method have identified the feature associations and their interdependencies in order to nominate the appropriate regression technique. PMI

method have chosen the Multiple Linear Regression with Shrinkage method according to the Case3 (i.e. y is related to multiple x-variables with inter-correlation) discussed in section 5.3.3.1, where the outcome variable (i.e. turnrate) depends on intercorrelated predictor variables i.e. greentime, intertime, road1 saturation and road2 saturation. Such phenomenon in the multiple regression is called multicollinearity, where two or more predictor variables are linearly related. To deal with multicollinearity, PMI method have applied two different regression techniques e.g. stepwise regression and ridge regression based on the severity of multicollinearity.

At first, PMI method estimates the Variance Inflation Factor (VIF) (defined in section 5.3.3.2) for predictors that detects the strength of intercorrelation among x variables. For each link in the signal phases (i.e. $s0$, $s1$ and $s2$), the VIF values have calculated that are listed in table 6.16a, 6.16b and 6.16c respectively. To estimate the VIF for given predictors, PMI method have utilised the built-in function *"variance_inflation_factor"* from *statsmodels* python module (Perktold et al. 2021).

The strength of Multicollinearity based on VIF value
$VIF = 1$: Complete absence of collinearity
$1 < VIF \leq 5$: Moderately correlated
$VIF > 5$: Highly correlated

Table 6.15 A rule of thumb for interpreting the Variance Inflation Factor (VIF) (James et al. 2013).

Signal Phase $s0$	greentime $x1$	intertime $x2$	road1 saturation $x3$	road2 saturation $x4$
link1	4.369	1.371	3.514	1.556
link2	2.869	1.213	2.857	1.037
link3	1.235	1.055	1.221	1.044

(a) VIF values for each link: Phase $s0$

Signal Phase $s1$	greentime $x1$	intertime $x2$	road1 saturation $x3$	road2 saturation $x4$
link1	**5.228**	2.605	**5.131**	1.012

(b) VIF values for each link: Phase $s1$

Signal Phase $s2$	greentime $x1$	intertime $x2$	road1 saturation $x3$	road2 saturation $x4$
link1	1.817	2.163	2.211	2.139
link2	1.943	2.418	1.322	1.187
link3	1.808	2.175	1.322	1.024

(c) VIF values for each link: Phase $s2$

Table 6.16 VIF values for the signal phases: $s0$, $s1$ and $s2$ consecutively

Table 6.16 provides the VIF values for each link of corresponding signal phase. Those VIF values are interpreted by following the rules given in table 6.15. For signal phase $s0$ (table 6.16a), VIF values of predictors are between 1 and 5 that evinces the moderate correlation among them. Therefore we have applied the Stepwise regression for each link of signal phase $s0$. In signal phase $s1$ (table 6.16b), two predictor variables (i.e. greentime and road1 saturation) have VIF values greater than 5 that reveals the high intercorrelation between them. Hence the Ridge regression (i.e. shrinkage method) is employed for the link of signal phase $s1$. In signal phase $s2$ (table 6.16c), all predictors have VIF values between 1 and 5, thus Stepwise regression is performed for each link. In summary, the **Stepwise regression** is nominated for all links in signal phase $s0$ and $s2$. On the other side, **Ridge regression** is selected for the link of signal phase $s1$.

In this thesis, we have implemented the stepwise regression with backward elimination approach that identifies and removes the least significant predictor variables from the model based on specified criteria (detailed in section 4.5.2). To perform backward elimination, we have encoded a function named *"backward_elimination"* in the Python program (given in Appendix E). It conducts statistical tests e.g. f-statistic and t-test, that quantify the significance of predictor variables in the regression model. Based on the p-values (i.e. greater than the significance level of 0.05) from f-statistic and t-test, the insignificant predictors are removed from regression model. To exploit statistical data and build statistical models, PMI employs the *statsmodels* module (Seabold and Perktold 2010). Moreover it utilises the built-in function *sklearn.linear_model.Ridge* to perform ridge regression.

Above mentioned in section 6.2.6.2, the input dataset is divided into two parts: training set and testing set utilising the K-Fold cross validation (CV) method. To learn the regression model, the training set is exploited by the Multiple Linear Regression (MLR) techniques e.g. stepwise regression and ridge regression. The intent is to predict the value of outcome variable (i.e. turnrate) for the given values of predictor variables i.e. greentime, intertime, road1 saturation and road2 saturation. Whereas the testing set is required to measure the prediction accuracy of the model in terms of Mean Squared

Error (MSE). To calculate MSE values, PMI method have utilised the built-in function *"mean_squared_error"* from *sklearn.metrics* python module.

In 2-Fold CV, the entire procedure of formulating regression model repeats two times by swapping the training and testing sets. As a consequence, it provides two models with approximated parameters, for example intercept (β_0) and the regression coefficient of corresponding predictor variables, β_1, β_2 β_3 and β_4. For each link in signal phases (i.e. $s0$, $s1$ and $s2$), the estimated parameter values of regression model in different folds are provided below (tables 6.17a, 6.17b and 6.17c respectively).

(a) Estimated Regression Coefficients for each link in signal phase $s0$

k-fold	Phase $s0$	Intercept (β_0)	Coeff. of x_1 (β_1)	Coeff. of x_2 (β_2)	Coeff. of x_3 (β_3)	Coeff. of x_4 (β_4)	MSE
k=1	Link1	-0.062	0	0	1.019	1.539	0.196
k=2	Link1	1.475	-0.007	0	-0.629	0.666	0.117
k=1	Link2	0	0	0	0	0	N/A
k=2	Link2	0.091	-0.0004	0	-0.068	-0.063	0.0001
k=3	Link2	0.053	0	0	0	-0.0889	0.0005
k=1	Link3	0.058	-0.002	0	4.010	2.137	0.152
k=2	Link3	0.119	0	0	3.326	1.349	0.125

(b) Estimated Regression Coefficients for each link in signal phase $s1$

k-fold	Phase $s1$	Intercept (β_0)	Coeff. of x_1 (β_1)	Coeff. of x_2 (β_2)	Coeff. of x_3 (β_3)	Coeff. of x_4 (β_4)	MSE
k=1	Link1	0	0	0	0	0	N/A
k=2	Link1	0.107	0.0004	0.0109	-0.136	-0.074	0.0002
k=3	Link1	0.094	0.0002	0.0056	-0.093	-0.091	0.0002
k=4	Link1	0.0951	0.0002	0.0068	-0.1008	-0.081	0.0002
k=5	Link1	0.098	0.0001	0.005	-0.081	-0.1007	0.0005

(c) Estimated Regression Coefficients for each link in signal phase $s2$

k-fold	Phase $s2$	Intercept (β_0)	Coeff. of x_1 (β_1)	Coeff. of x_2 (β_2)	Coeff. of x_3 (β_3)	Coeff. of x_4 (β_4)	MSE
k=1	Link1	0.330	0	−0.113	0	3.612	0.830
k=2	Link1	1.496	0	−0.090	−1.366	−0.756	0.065
k=1	Link2	−0.010	0	0	0	4.499	0.080
k=2	Link2	0.364	0	−0.054	0	0	0.172
k=1	Link3	0	0	0	0	0	N/A
k=2	Link3	0.246	0	−0.0295	0	0	0.038
k=3	Link3	0	0	0	0	0	N/A
k=4	Link3	0.365	0	−0.0298	0	−0.369	0.117

Table 6.17 Estimated Regression Coefficients for signal phases $s0$, $s1$ and $s2$ respectively with K-fold cross validation

From Table - 6.17, we get the final regression model (highlighted with Gray colour in the table) with lowest MSE value for each link in corresponding signal phases $s0$, $s1$ and $s2$. Here, we have found some exceptional cases during model formulation, for instance, the turnrate has all-zero values (e.g. fold 1 in s0_link2), all predictors have p-values greater than 0.05 (e.g. fold 1 in s2_link3), multiple folds have similar (lowest) MSE values (e.g. fold 1 to 4 in s1_link1) etc. To handle those cases, we have utilised different fold sizes in the K-fold CV, e.g. k = 3 for s0_link2, k = 5 for s1_link1 and k = 4 for s2_link3. Besides, we have used different performance metrics (i.e. p-values from F-test and t-test) to analyse the models with similar (lowest) MSE values, for instance, the model in k = 2 is finalised with p-values < 0.05 for s1_link1.
With estimated parameters i.e. the intercept β_0, the regression coefficients β_1 of greentime (x_1), β_2 of intertime (x_2), β_3 of road1 saturation (x_3) and β_4 of road2 saturation (x_4), the mathematical formula of regression model can be written as follows (equation 6.4):

$$turnrate = \beta_0 + (\beta_1 * greentime) + (\beta_2 * intertime) + (\beta_3 * road1sat) + (\beta_4 * road2sat) \tag{6.4}$$

To measure the significance of predictors in the formulated regression model (equation 6.4), we conduct the t-test. Where the p-value < 0.05 for corresponding predictor (excluding the intercept) indicates at least a 95% chance of true relationship between outcome and predictor variable in the population. Also, the ANOVA test (F-test) is performed to evaluate the statistical significance of entire regression model. From F-test, p-value less than a significance level of 0.05 suggests that the regression model fits the

data. In a nutshell, the regression model with all p-values < 0.05 represents the Process Model (PM). For conducting the statistical significance tests for regression model, we have employed the built-in method *"statsmodels.formula.api"* in the Python program. Besides, significance tests for the ridge model is performed by utilising the Python module *" regressors.stats"* (Haas 2015). The output of statistical test (e.g. F-test and t-test) for each link in signal phases $s0$, $s1$ and $s2$ are demonstrated below (figure 6.11, 6.12 and 6.13 consecutively). The results exhibit that the p-values from both F-test and t-test are less than 0.05 (highlighted with red box in the figure). Hence we can conclude that the estimated regression models (specified in table 6.17) signify the PM of turnrate for corresponding links.

OLS Regression Results

Dep. Variable:	Turnrate	R-squared:	0.321
Model:	OLS	Adj. R-squared:	0.308
Method:	Least Squares	F-statistic:	25.49
Date:	Sun, 14 Mar 2021	Prob (F-statistic):	1.47e-13
Time:	14:02:39	Log-Likelihood:	-76.146
No. Observations:	166	AIC:	160.3
Df Residuals:	162	BIC:	172.7
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.4753	0.221	6.666	0.000	1.038	1.912
Greentime	-0.0079	0.001	-5.430	0.000	-0.011	-0.005
Road1_Sat	-0.6299	0.264	-2.388	0.018	-1.151	-0.109
Road2_Sat	0.6663	0.185	3.594	0.000	0.300	1.032

(a) Results of F-test and t-test for link 1

OLS Regression Results

Dep. Variable:	Turnrate	R-squared:	0.081
Model:	OLS	Adj. R-squared:	0.069
Method:	Least Squares	F-statistic:	6.399
Date:	Sun, 14 Mar 2021	Prob (F-statistic):	0.000358
Time:	19:53:10	Log-Likelihood:	342.40
No. Observations:	221	AIC:	-676.8
Df Residuals:	217	BIC:	-663.2
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0911	0.023	3.958	0.000	0.046	0.136
Greentime	-0.0004	0.000	-2.476	0.014	-0.001	-7.82e-05
Road1_Sat	-0.0681	0.030	-2.238	0.026	-0.128	-0.008
Road2_Sat	-0.0625	0.019	-3.300	0.001	-0.100	-0.025

(b) Results of F-test and t-test for link 2

OLS Regression Results

Dep. Variable:	Turnrate	R-squared:	0.360
Model:	OLS	Adj. R-squared:	0.352
Method:	Least Squares	F-statistic:	45.90
Date:	Sun, 14 Mar 2021	Prob (F-statistic):	1.55e-16
Time:	14:34:44	Log-Likelihood:	-72.981
No. Observations:	166	AIC:	152.0
Df Residuals:	163	BIC:	161.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1193	0.070	1.693	0.092	-0.020	0.258
Road1_Sat	3.3262	0.398	8.356	0.000	2.540	4.112
Road2_Sat	1.3491	0.333	4.057	0.000	0.692	2.006

(c) Results of F-test and t-test for link 3

Fig. 6.11 Statistical significance tests for regression models in signal phase $s0$

Ridge Regression Results

Model:	Ridge(alpha=0.1)
R-squared:	0.1600935920356651
Adj. R-squared:	0.14686671946929764
F-statistic:	12.103661786440865
P-value:	0.0
No. Observations:	259

	coef	std err	t	P-value
intercept	0.1073	0.0125	8.5984	0.0
Greentime	0.0004	0.0	8.115	0.0
Intertime	0.0109	0.0046	2.4001	0.0171
Road1_Sat	-0.1362	0.0437	-3.1126	0.0021
Road2_Sat	-0.0741	0.0152	-4.8732	0.0

Fig. 6.12 Statistical significance tests for the regression model in signal phase $s1$

OLS Regression Results

Dep. Variable:	Turnrate	R-squared:	0.491
Model:	OLS	Adj. R-squared:	0.439
Method:	Least Squares	F-statistic:	9.343
Date:	Sun, 14 Mar 2021	Prob (F-statistic):	0.000176
Time:	14:36:14	Log-Likelihood:	-6.5026
No. Observations:	33	AIC:	21.01
Df Residuals:	29	BIC:	26.99
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.4964	0.291	5.142	0.000	0.901	2.092
Intertime	-0.0907	0.024	-3.842	0.001	-0.139	-0.042
Road1_Sat	-1.3658	0.439	-3.109	0.004	-2.264	-0.467
Road2_Sat	-0.7563	0.358	-2.110	0.044	-1.489	-0.023

(a) Results of F-test and t-test for link 1

OLS Regression Results

Dep. Variable:	Turnrate	R-squared:	0.189
Model:	OLS	Adj. R-squared:	0.162
Method:	Least Squares	F-statistic:	6.988
Date:	Sun, 14 Mar 2021	Prob (F-statistic):	0.0129
Time:	14:37:59	Log-Likelihood:	-13.154
No. Observations:	32	AIC:	30.31
Df Residuals:	30	BIC:	33.24
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0110	0.136	-0.081	0.936	-0.288	0.266
Road2_Sat	4.4986	1.702	2.643	0.013	1.023	7.974

(b) Results of F-test and t-test for link 2

OLS Regression Results

Dep. Variable:	Turnrate	R-squared:	0.091
Model:	OLS	Adj. R-squared:	0.072
Method:	Least Squares	F-statistic:	4.701
Date:	Sun, 14 Mar 2021	Prob (F-statistic):	0.0352
Time:	20:02:30	Log-Likelihood:	-4.0986
No. Observations:	49	AIC:	12.20
Df Residuals:	47	BIC:	15.98
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.2460	0.046	5.403	0.000	0.154	0.338
Intertime	-0.0296	0.014	-2.168	0.035	-0.057	-0.002

(c) Results of F-test and t-test for link 3

Fig. 6.13 Statistical significance tests for regression models in signal phase $s2$

6.2.6.4 Integration of PM

The resulting PM represents the statistical relationship between one dependent variable (i.e. vehicle flow rate between roads) and a series of independent variables (e.g. traffic signal active/interval time, occupancy/capacity of roads). It estimates the rate of vehicle flow (defined as turnrate) in active road junctions based on different influencing factors, for instance, greentime, intertime, incoming and outgoing road saturation that affect the traffic flow rate. The mathematical formula of PM is given in equation 6.4. Where, the approximated parameter values (i.e. β_0, β_1, β_2, and β_3) for the junction $n3969$ are provided in table 6.17 (highlighted with Gray colour).

The estimated PM of turnrate is integrated and adjusted into the existing *"flowrun_green"* process of previously engineered (hybrid) planning model for controlling traffic flow (provided in Appendix D.1). Figure 6.14 below shows the process *"flowrun_green"* that has incorporated PM in its effect. For an active signal phase p in junction i, it calculates the turnrate of vehicle flow from incoming road $r1$ to outgoing road $r2$. Where, the occupancy of $r1$ decreases and the occupancy of $r2$ increases by the estimated turnrate value with time (defined by #t literal).

```
(:process flowrun_green
    :parameters (?p - stage ?r1 ?r2 - link ?i - junction)
    :precondition (and (active ?p) (contains ?i ?p)
            (> (occupancy ?r1) 0.0)
            (< (occupancy ?r2) (capacity ?r2))
    )
    :effect (and

            (assign (turnrate  ?p ?r1 ?r2 )
                (+ (beta0 ?p ?r1 ?r2)
                    (+ (* (beta1 ?p ?r1 ?r2) (greentime ?i) )
                        (+ (* (beta2 ?p ?r1 ?r2) (intertime ?i))
                            (+ (* (beta3 ?p ?r1 ?r2) (/(occupancy ?r1) (capacity ?r1)))
                                (* (beta4 ?p ?r1 ?r2) (/(occupancy ?r2) (capacity ?r2)))
                            )
                        )
                    )
                )
            )

            (increase (occupancy ?r2) (* #t (turnrate ?p ?r1 ?r2)))
            (decrease (occupancy ?r1) (* #t (turnrate ?p ?r1 ?r2)))
    )
)
```

Fig. 6.14 flowrun_green process with PM in the UTC domain

CHAPTER 7

Empirical Analysis and Evaluation

As evidence of the concept of learning by the application of our system, this chapter includes the empirical analysis and evaluation of the proposed PMI method on the running examples, i.e. Urban Traffic Control (UTC) domain and Coffee domain (elaborated in previous chapter 6). The focus of the analysis is to test the effectiveness of the learned Process Model (PM) that is embedded into a pre-engineered, original domain model, specifically to learn an accurate model of the continuously changing features in a process description. The aim is to refine/improve the process knowledge in hybrid planning domains with respect to enhance the simulation accuracy, which can ultimately lead to higher-quality plans.

7.1 Coffee Domain

Aforementioned in the earlier chapter (section 6.1), the bonafide coffee brewing requires efficiently controlled brewing time with temperature to get personalised espresso taste. In this thesis, we have reformulated the PDDL+ model of the coffee domain (provided in Appendix B.1), that is partially inspired by the original coffee domain (given in Appendix A.1), where the coffee brewing procedure is regulated by the PM for extraction yield%. In simple terms, it brews coffee according to the espresso taste (i.e. expected coffee yield%). It estimates the extracted coffee yield% in terms of variation in the brewing time with specified brewing temperature. To formulate the PM, we have exploited the coffee data that is taken from the observational study (discussed in section 6.1.4) conducted by Easthope (2015). Then the learned PM is embedded into the process specification of pre-engineered coffee domain, that is our reformulated PDDL+ coffee domain (see section 6.1.5, for details). Finally, we evaluate the learned domain with embedded PM (given in Appendix B). The intent is to assess how well it improves the simulation accuracy, and consequently the plan quality.

7.1.1 Evaluation of the Learned Domain Model

We evaluate the learned domain model with embedded PM in order to measure its performance and effectiveness in the real planning applications e.g. brewing coffee ac-

cording to personalised taste. For evaluating the learned (coffee) domain model, we deploy the domain model with embedded PM in the hybrid planning engines that can generate realistic plans. In this thesis, we have formulated the learned domain model with PDDL+ representation. Therefore, the ENHSP (Scala et al. 2016) hybrid planner has been utilised which supports PDDL+ domain models. The learned domain model with PDDL+ constructs has two parts: Domain definition and Problem definition. The domain definition contains the PM that is integrated into the *brewing* process (given in appendix B.1). The parameters value of PM are initialised in the problem definition (an example is given in appendix B.2).

Aforementioned in section 6.1, the brewing temperature and brewing time both affect the coffee extraction (i.e. yield%), which defines the flavour and aroma of espresso. The expected espresso taste with corresponding extraction percentage (i.e. yield%) is listed in the table - 7.1. During the coffee brewing process, the learned domain model can control the extraction of coffee. For a given brewing temperature (shown in table 7.1), the Process Model (PM), that is embedded in the learned domain model, estimates yield% with increment of brewing time.

In this thesis, the learned (coffee) domain model is experimented with four different problem tasks for the temperature 92°C, 94°C, 96°C and 98°C respectively. The ENHSP provides plan for each task using the same domain model with different temperature reading and corresponding expected yield%. The sample plan for each separate problem with different brew temperature is provided below in figure 7.1.

Brew Temp. (°C)	Yield (%)	Espresso taste
92	18.95	The coffee has high acidity but with lower body, sweetness and bitterness
94	19.38	With balanced acid, high levels of sweetness, good body, and low bitterness
96	19.4	With balanced acid, high levels of sweetness, good body, and low bitterness
98	19.57	It has medium acidity with lots of sweetness, bitterness, full body and with strange powdery mouthfeel

Table 7.1 Expected espresso taste with yield% at specific brew temperature (data extracted from table 6.7 and 6.8)

Plan with Brew Temperature 92 °C	Plan with Brew Temperature 94 °C
Plan: 0.00000: (heatwater coffee1 water1) ;42.00000: (stop-heating coffee1 water1) 45.00000: (brew-coffee coffee1 water1) ;75.00000: (stop-brewing coffee1 water1) 75.00000: (serve-coffee coffee1 water1)	**Plan:** 0.00000: (heatwater coffee1 water1) ;43.00000: (stop-heating coffee1 water1) 45.00000: (brew-coffee coffee1 water1) ;76.00000: (stop-brewing coffee1 water1) 80.00000: (serve-coffee coffee1 water1)
Plan Details: Plan-Length: 80 Planning Time: 307 Heuristic Time: 1 Search Time: 29 Expanded Nodes: 37 States Evaluated: 40 Duration: 75.0000000 Total Cost: 0.0	**Plan Details:** Plan-Length: 85 Planning Time: 946 Heuristic Time: 3 Search Time: 45 Expanded Nodes: 39 States Evaluated: 42 Duration: 80.0000000 Total Cost:0.0
Plan with Brew Temperature 96 °C	**Plan with Brew Temperature 98 °C**
Plan: 0.00000: (heatwater coffee1 water1) ;44.00000: (stop-heating coffee1 water1) 45.00000: (brew-coffee coffee1 water1) ;75.00000: (stop-brewing coffee1 water1) 75.00000: (serve-coffee coffee1 water1)	**Plan:** 0.00000: (heatwater coffee1 water1) ;45.00000: (stop-heating coffee1 water1) 45.00000: (brew-coffee coffee1 water1) ;75.00000: (stop-brewing coffee1 water1) 75.00000: (serve-coffee coffee1 water1)
Plan Details: Plan-Length: 80 Planning Time: 317 Heuristic Time: 0 Search Time: 28 Expanded Nodes: 37 States Evaluated: 40 Duration: 75.0000000 Total Cost: 0.0	**Plan Details:** Plan-Length: 80 Planning Time: 312 Heuristic Time: 1 Search Time: 34 Expanded Nodes: 37 States Evaluated: 40 Duration: 75.0000000 Total Cost:0.0

Fig. 7.1 Four different plans to brew coffee according to the espresso taste (defined by yield%)

Figure 7.1 exhibits four different plans where the plan is from *heatwater coffee1 water1* to *serve-coffee coffee1 water1*. On the left of each plan, the time is defined at which the actions need to be executed. For convenience, the ENHSP planner also outputs the internal events call in the semi colon separated lines.

Afterwards, the produced plans are simulated in the PSIM (PDDL+ plan simulator developed by Lindsay et al. (2020)) to explore the values of estimated yield% with time. For each temperature, table 7.2 demonstrates the simulation result of corresponding generated plan. Here, the total Brew time (sec) is taken by the planner to attain the Expected yield% (given in table 6.7) with corresponding Brew temperature (°C). Besides, the Estimated yield% represents the final outcome of extraction yield% achieved by the planner.

Table 7.2 Simulation Result of the plan for each temperature (°C)

	Brew Temperature (°C)			
	92°C	*94°C*	*96°C*	*98°C*
Total Brew Time (Sec)	31.00	32.00	31.00	31.00
Expected yield%	18.95	19.38	19.40	19.57
Estimated yield%	19.041	19.445	19.456	19.664
Error (%)	9.1%	6.5%	5.6%	9.4%

From table 7.2, we have calculated the error (%) for each temperature reading using the value of expected yield% and estimated yield%. Error (stated in equation 5.2), also known as residual, denotes the difference between the actual or expected outcome value and the corresponding estimated or predicted value. As an evaluation metric of our learned model, we have utilised the anticipated error (%) in the graphical residual analysis (discussed in section 5.3.5). Figure 7.2 illustrates the bar graph displaying the residual of yield% at each temperature, where all the residuals (%) or errors (%) are less than 10% which is acceptable.

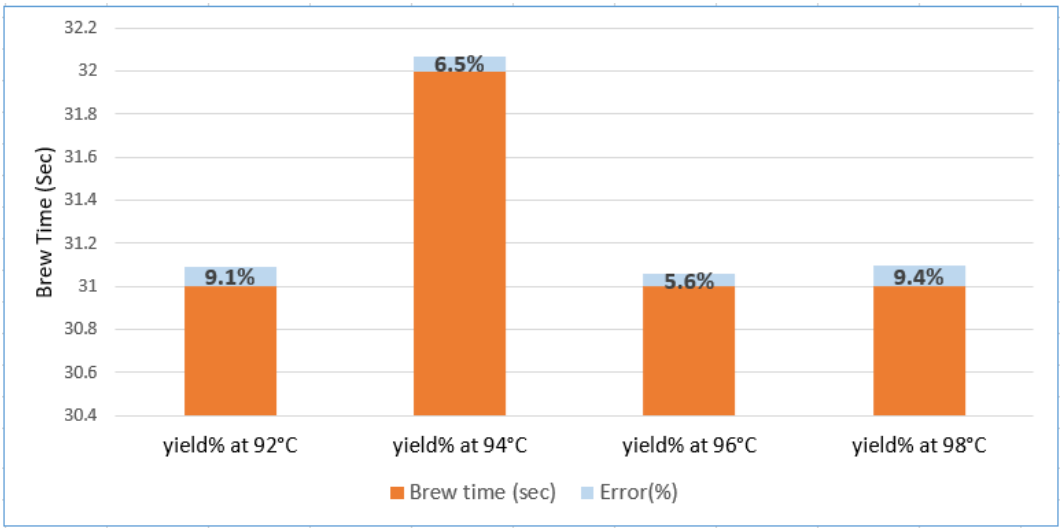


Fig. 7.2 Bar graph displaying Brew Time (sec) with corresponding yield error (%) at various temperatures

7.1.2 Discussion

By analysing the simulation result (table 7.2) and the residual bar graph (figure 7.2), we can conclude that the learned domain model approximates the more rational and pragmatic value of outcome variable (i.e. yield%), which is very close to the actual value (i.e. expected yield%). Consequently, the improved simulation output aids the

planning engine to produce high-quality plans (as stated in the Aims and Objectives, section 1.2). Bedsides, without declaring the process parameters (e.g. brewing time) statically, the PM embedded in coffee domain can adjust the brewing time automatically according to the expected yield% in the brewing process. Hence, it can control the brewing process automatically by estimating the yield% according to the changing parameters (i.e. brewing time and temperature). In that manner, it can facilitate the knowledge engineering task of modelling processes with dynamically changing parameters in the real-world hybrid (planning) domains (as mentioned in Thesis Contribution, section 1.3).

Furthermore, the Process Model (PM) is formulated by exploiting the real-time dataset, thereby updating/adding new numeric features in the learning dataset can enhance the **Dynamicity** of the planning model as well as boost its **Versatility** (as described in the Aims and Objectives, section 1.2). For example, the extraction of coffee or coffee yield% (i.e. the outcome variable), that defines the espresso taste, may depend on other factors such as coffee dosage, coffee-to-water ratio, coffee particle size (finer or coarser grind) etc in the brewing process. In addition, the values of process parameters in the learning dataset may need to modify or update over time with the changes in dynamic states. On this account, by updating the existing parameter values, adding new rows of values, or adding new feature columns in the learning data set, it can keep the process model up-to-date and consequently, improve the hybrid domain model in respect of **Dynamicity** and **Versatility**. Moreover, during the formulation of PM, it identifies the significant/ineffective process variables automatically along with their interdependencies, for instance, yield% is dependent on brewing time and brewing temperature, which are independent from each other. In that way, it can assist the knowledge engineers by choosing effective numeric features (i.e. process parameters) and removing the irrelevant ones from the process models (as stated in Thesis Contribution, section 1.3).

By interpreting the planning output for different problem tasks (given in figure 7.1), we can say that the learned domain model provides individual plan in accordance with the espresso taste, by automatically manipulating the brewing process . In a nutshell, the learned domain model with embedded PM can generate advanced plans by enhancing the simulation accuracy (as discussed above).

7.2 Urban Traffic Control (UTC) Domain

As mentioned in previous chapter (section 6.2), the traffic flow rate in a road junction fluctuates over time which can be affected by different factors e.g. incoming/outgoing road saturation, active (green) time, inter time, cross flow, peak/off-peak hour, low capacity roads with parking spaces, and number of lanes etc. Those influencing factors

vary according to the road infrastructure. In the original PDDL+ formulation of UTC domain (given in Appendix C.1), the vehicle flow rate of each road link is statically defined across a junction. Besides, the estimation of flow rate does not contemplate the dynamically changing factors of the road network. For this thesis, we aim to use PMI to improve the flow rate PM that is embedded in the existing PDDL+ model of UTC domain (given in Appendix D.1). The Process Model (PM) of flow rate estimates the vehicle turn rate of each road link in terms of variation in the corresponding influencing factors. To induce the PM, we have utilised the traffic data that has been experimented in SimplifAI (McCluskey, Vallati and Franco 2017) project (discussed in previous chapter, section 6.2). This section evaluates the learned UTC domain with embedded PM to measure the simulation accuracy of turnrate values. Specifically, it assesses the effectiveness of PMI in the hybrid planning domain with respect to improve the simulation output, and consequently the quality of generated plans.

7.2.1 Evaluation of the Learned Domain Model

In this thesis, we evaluate the learned model to measure its performance and effectiveness in the real planning applications e.g. controlling traffic flow in the urban road junctions/intersections. For evaluating the learned model, we have utilised the ENHSP (Scala et al. 2016) PDDL+ planner to generate plans. Besides, produced plans are simulated in the PSIM (PDDL+ plan simulator developed by Lindsay et al. (2020)) to explore the values of turnrate with time.

The learned model is experimented with a particular set of road junctions in the Manchester (UK) urban area (abstract view in figure 6.7), where the PM of turnrate for each link is formulated and adjusted from the originally-engineered UTC planning model. From a specified initial state, we produce signal plans for both original and learned UTC planning model to achieve a common goal. Both plans are individually simulated in the PSIM to observe turnrate values. To assess turnrate errors, the estimated turnrate values in learned model and the static turnrate values in original model are compared with the corresponding real-time turnrate value observed in AIMSUN. Finally, we calculate the mean value of squared turnrate errors for both models that is defined as MSE (Mean Squared Error) of turnrate. For Original and Learned model, the respective figures 7.3a and 7.3b provide the MSE of turnrate values observed after 200, 600, 800, 1200 and 1600 seconds of simulation. Here, MSE values are estimated from the simulation output of three different time periods (i.e. 1 hour long each). The estimated MSE values for each period (i.e. $T1$, $T2$ and $T3$) are plotted as a line graph shown in figure 7.4.

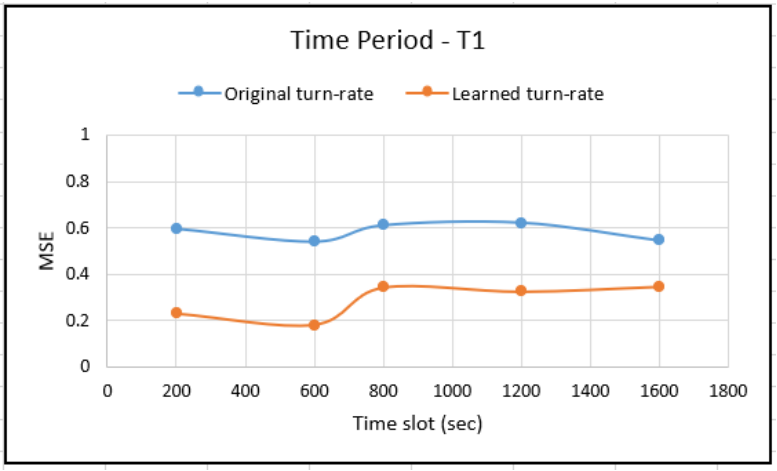
One hour Period	Time slot				
	200	600	800	1200	1600
T1	0.59664	0.54283	0.61097	0.62193	0.5478
T2	0.58874	0.51977	0.55397	0.58939	0.49744
T3	0.593	0.50844	0.6152	0.60829	0.52461

(a) MSE (Mean Squared Error) of turnrate values in Original Model

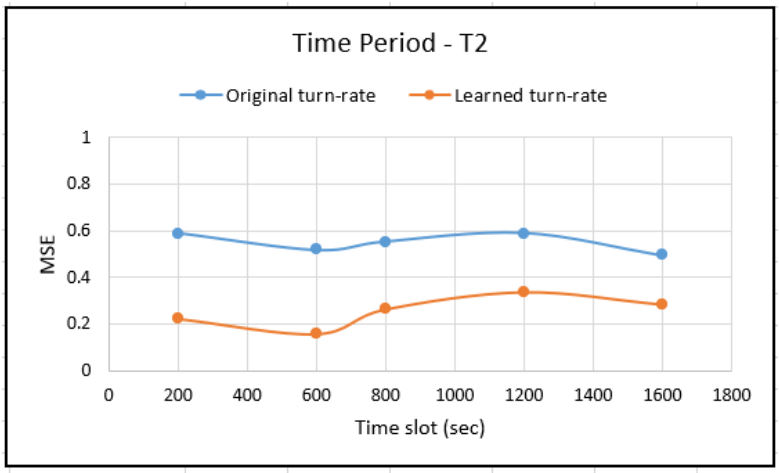
One hour Period	Time slot				
	200	600	800	1200	1600
T1	0.2302	0.18153	0.34319	0.32532	0.3467
T2	0.22309	0.15641	0.26359	0.33943	0.28516
T3	0.22336	0.19931	0.3654	0.32077	0.32109

(b) MSE (Mean Squared Error) of turnrate values in Learned Model

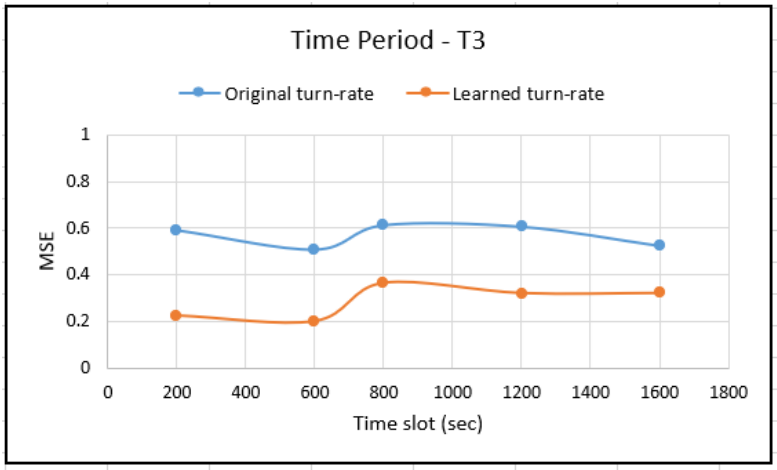
Fig. 7.3 Estimated MSE of turnrate values observed after 200, 600, 800, 1200 and 1600 seconds of simulation in $T1$, $T2$ and $T3$ period



(a) MSE (Mean Squared Error) of turnrate values observed in T1 period



(b) MSE (Mean Squared Error) of turnrate values observed in T2 period



(c) MSE (Mean Squared Error) of turnrate values observed in T3 period

Fig. 7.4 MSE comparison between original and learned UTC models with the turn-rate observed in AIMSUN after 200, 400, 600 and 800 seconds of simulation.

As an evaluation metric of our learned model, we have estimated the MSE of turn-rate values for original and learned UTC models. By analysing the MSE values (figure 7.3) with line graphs (figure 7.4), we can conclude that the learned model improves simulation accuracy and produce less error as compared to the original UTC model with static turn-rate across the road junctions.

Furthermore, we deploy the learned model in the hybrid planning engines in order to generate real-time plans. In this thesis, the learned model is formulated in PDDL+ planning language. Therefore, a PDDL+ planner (we chose ENHSP for this) is employed to produce plans that supports PDDL+ representation of planning models. As mentioned earlier, the PDDL+ (hybrid) planning model consists of two parts: Domain definition

and Problem definition. The domain definition of learned UTC model integrates the PM into the *"flowrun_green"* process effect (given in appendix D.1), whereas the parameters value (i.e. β-values) of PM are initialised in the problem definition (a sample problem is given in appendix D.2).

To demonstrate the efficiency of learned model in plan generation, we have taken a simple planning problem with a single junction $n3969$, where the goal is to reduce traffic congestion in a specific road $n3968\_n3969$ that becomes active in the signal phase $s3969\_s2$. With the same set of initial states and goal, we have produced the signal plans from original and learned UTC model. The domain and problem definition of original and learned UTC model are given in appendix C and D correspondingly. Figure 7.5a and 7.5b below provide the signal plans generated from original and learned model respectively.

Original Model

Plan:

;89.00000: (maxgreenreached s3969_s2 n3969)
;89.00000: (trigger-inter s3969_s2 n3969)
;98.00000: (trigger-change s3969_s2 s3969_s0 n3969)
120.00000: (switchPhase s3969_s0 n3969)
;120.00000: (trigger-inter s3969_s0 n3969)
;121.00000: (trigger-change s3969_s0 s3969_s1 n3969)
150.00000: (switchPhase s3969_s1 n3969)
;150.00000: (trigger-inter s3969_s1 n3969)
;155.00000: (trigger-change s3969_s1 s3969_s2 n3969)
170.00000: (switchPhase s3969_s2 n3969)
;170.00000: (trigger-inter s3969_s2 n3969)
;179.00000: (trigger-change s3969_s2 s3969_s0 n3969)

Plan Details:

Plan-Length:192
Planning Time:427
Heuristic Time:6
Search Time:77
Expanded Nodes:56
States Evaluated:82
Duration:180.0000000
Total Cost:0.0
Fixed constraint violations during search (zero-crossing):0
Number of Dead-Ends detected:0
Number of Duplicates detected:7
Number of LP invocations:0

(a) Plan generated from Original UTC model

Learned Model
Plan: 80.00000: (switchPhase s3969_s2 n3969) ;80.00000: (trigger-inter s3969_s2 n3969) ;89.00000: (trigger-change s3969_s2 s3969_s0 n3969)
Plan Details: Plan-Length:93 Planning Time:490 Heuristic Time:5 Search Time:84 Expanded Nodes:24 States Evaluated:40 Duration:90.0000000 Total Cost:0.0 Fixed constraint violations during search (zero-crossing):0 Number of Dead-Ends detected:0 Number of Duplicates detected:5 Number of LP invocations:0

(b) Plan generated from Learned UTC model

Fig. 7.5 Signal Plan for reducing traffic congestion in $n3968\_n3969$ with Original and Learned UTC model

Figure 7.5 exhibits two separate plans to reduce vehicle congestion in a specific road $n3968\_n3969$, where the plan is defined by the sequence of action *"switchPhase"* taken with time. The action *"switchPhase"* activates the next signal phase by deactivating the current phase. Besides, the internal events are indicated by the semi colon separated lines that occur during plan generation. The resulting signal plans (figure 7.5a and 7.5b) show that the learned model takes less signal cycle with time to reduce congestion in $n3968\_n3969$. On the other hand, the original model requires more than one signal cycle with additional time to achieve goal.

7.2.2 Discussion

By analysing the simulation output (given in figure 7.3) and the error graph (illustrated in figure 7.4), we can conclude that the learned domain model approximates the more rational and pragmatic value of outcome variable (i.e. turnrate), which is very close to the actual value (i.e. the real-time turnrate value observed in AIMSUN). Consequently, the improved simulation accuracy aids the planning engine to produce high-quality plans (as stated in the Aims and Objectives, section 1.2). Bedsides, without declaring the process parameters (e.g. turnrate or flow rate of vehicles) statically, the PM embedded in UTC domain can adjust the turnrate automatically according to the corresponding influencing factors (i.e. incoming/outgoing road saturation, active (green) time, inter time etc.) in the road junctions. Hence, it can automatically control the flowrun_green process (i.e. allows car to flow if the corresponding green is on) by estimating the

turnrate according to the changing parameters (i.e. active/green time, inter/amber time, incoming/outgoing road saturation etc.). In that manner, it can facilitate the knowledge engineering task of modelling processes with dynamically changing parameters in the real-world hybrid (planning) domains (as mentioned in Thesis Contribution, section 1.3).

Furthermore, the Process Model (PM) is formulated by exploiting the real-time dataset, thereby updating/adding new numeric features in the learning dataset can enhance the **Dynamicity** of the planning model as well as boost its **Versatility** (as described in the Aims and Objectives, section 1.2). For example, the vehicle flow rate on a road junction may depend on other factors such as the gradient of roads, the density of intersections, the density of bus stops, the speed limit etc when the corresponding signal phase is active. Additionally, the values of process parameters in the learning dataset may need to modify or update over time with the variations in dynamic states. On this account, by updating the existing parameter values, adding new rows of values, or adding new feature columns in the learning data set, it can keep the process model up-to-date and consequently, improve the hybrid domain model in respect of **Dynamicity** and **Versatility**.

Moreover, during the formulation of PM, the proposed PMI method identifies the significant/ineffective process variables automatically along with their interdependencies, for instance, the turnrate on the road $s0\_link1$, for the signal phase $s0$ (i.e. an active link in the signal phase $s0$ of road junction $n3969$, stated in table 6.14), is dependent on greentime, $road1$ and $road2$ saturation, where the $road1$ and $road2$ saturation are correlated with greentime. Therefore, the formulated Process Model (PM) of turnrate for $s0\_link1$ contains the corresponding effective process parameters, i.e. greentime, $road1$ and $road2$ saturation (highlighted in the table 6.17a). Besides, the turnrate on another road $s2\_link2$, for the signal phase $s2$, is influenced only by the $road2$ saturation. Hence, the formulated PM of turnrate for $s2\_link2$ contains only the $road2$ saturation and removes others (shown in table 6.17c). In that way, it can assist the knowledge engineers by choosing the effective numeric features (i.e. process parameters) and removing the irrelevant ones from the process models (as stated in Thesis Contribution, section 1.3).

By interpreting the planning outcome generated by the Original and Learned UTC model with same set of problem task, we can say that the learned domain model provides better plan for the traffic signal phase in accordance with the traffic congestion, by automatically approximating the the more rational and pragmatic value of turnrate. In a nutshell, the learned model can advance the plan quality by improving simulation accuracy in terms of adjusting the numeric features automatically. The learned UTC model improves planning by approximating the appropriate flow rate, in order to obtain the traffic signal plans accordingly.

7.3 Related Work

Bay et al. (2002) combines machine learning methods with human expert knowledge for revising the initial models of engineering domains. The revised models in the form of mathematical equations represent the physical devices, for instance, a battery model on the International Space Station. An equation discovery program named Lagramge (Todorovski and Dzeroski 1997) is implemented to search mathematical models, that estimates parameters and score models. The engineer's knowledge is utilised that constrains the search space in order to make the equation discovery feasible. The system inputs a set of initial equations that describes the system behaviour, data on the observable variables in the equation, and the knowledge about equations i.e. all plausible independent variables, functional forms, parameter values, and their dependencies. After taking inputs from user, the problem is transformed into equation discovery task utilising the Lagramge program. Lagramge searches through the space of equations and evaluates candidate models according to the MDL (Minimum Description Length) score function. Finally it returns the best model that better explains the data. In contrast to this work, our proposed PMI method implements the regression techniques instead of Lagramge program to find the best fitted model. Besides, it involves different statistical tests (i.e. the t-test and ANOVA) to evaluate the significance of regression model. Moreover, the proposed PMI method formulates the process models (i.e. regression model) by exploiting the real-time data without inputting the initial models/equations or parameter values. Also, it automatically infers the dependencies among independent variables by performing the statistical test, i.e. Pearson's correlation coefficient. Learning models in the form of mathematical functions, as proposed by Bay et al. (2002), is complementary to our work, where we learn the process models that represent the continuous effects in the processes of hybrid planning domains.

Fine-Morris et al. (2020) presents an algorithm that learns planning actions with continuous effects and symbolic literals of operators for waypoint navigation simulations. As input, it takes a set of preprocessed positive and negative examples generated by simulation. It employs the MAX-SAT constraint solver for identifying the symbolic preconditions. Besides, it exploits the logistic regression to learn the continuous effects of numeric state variables with preconditions. This algorithm is similar to our proposed approach in the context of learning continuous effects by fitting regression models. However, the basic difference is that Fine-Morris et al. (2020) learns the continuous effects with preconditions of durative actions for classical/temporal domains that are represented in STRIPS (Fikes and Nilsson 1971) notation. On the contrary, we focus entirely on learning the continuous effects of processes for hybrid domains with PDDL+ representation. Additionally, our proposed PMI method applies different types

of regression techniques according to the inferred relationship among numeric input variables (i.e. process parameters).

Segura-Muros et al. (2021) develops a domain learner named PlanMiner that learns planning domains (i.e. a set of numeric action models) from plan traces with partial knowledge of intermediate states. The planning domains are learned with numeric predicates and arithmetic/logical relations between predicates which are represented in PDDL modelling language. PlanMiner applies a selection of preprocessing, classification, and symbolic regression techniques to extract information from plan traces, prepare the planning data, and infer arithmetic/relational expressions that define the precondition and effects of action models. Unlike our PMI method, PlanMiner learns arithmetic/relational expressions in action models for classical domains with PDDL formulation. The PMI method learns the process models (i.e. regression models), in the form of mathematical equation of numeric fluent, for hybrid domains with PDDL+ representation. The major difference is that PlanMiner utilises symbolic regression technique to search the best fitted arithmetic expression. On the contrary, our PMI approach implements different linear regression techniques to formulate the regression models that are represented by process models. Besides, PlanMiner learns the entire action model with preconditions and effects from plan traces, whereas we exploit the real-time data to construct the process models that are embedded/adjusted in the process effect of existing hybrid domain models. Similar with our work, PlanMiner does not support non-liner relationship between numeric fluent.

The most recent work on implementing ML techniques in APHD (Automated Planning with Hybrid Domains) with regard to refine the domain models is presented by Lindsay et al. (2020) (discussed in section 3.3). This work is relevant to our work in terms of learning/improving the process knowledge in hybrid planning domains. The main difference is that Lindsay et al. (2020) refines the process effects with preconditions by exploiting the decision tree learning approach, whereas our PMI method learns/formulates the process models by utilising different linear regression techniques.

A comparison table of related research illustrating how they meet certain desirable requirements and strengthening the case for the introduction of the proposed PMI method is given below (table 7.3):

Author	Bay et al. (2002)
Algorithm used	Lagramge (Todorovski and Dzeroski 1997), an equation discovery program

Input	A set of initial equations along with the variable data, theoretical knowledge about equations i.e. all plausible independent variables, functional forms, parameter values, and their dependencies etc.
Output	The mathematical models that better explain the data
Environment	Engineering domains, for instance, a battery model on the International Space Station
Merits	Can reduce model development time and improve model quality that represents physical devices
Limitations	Structure of the model (i.e. initial equation) along with the parameter values and their dependencies has to be provided explicitly by the human expert; Requires the engineer's knowledge to constrain the search
Author	Fine-Morris et al. (2020)
Algorithm used	MAX-SAT constraint solver that identifies the symbolic preconditions, and Logistic regression that learn the continuous effects of durative actions
Input	a set of pre-processed positive and negative examples generated by waypoint navigation simulation
Output	the continuous effects with preconditions of durative actions
Environment	Classical/Temporal planning domains represented in STRIPS (Fikes and Nilsson 1971) notation
Merits	Capable of learning action models combining symbolic literals and continuous effects under noisy training data
Limitations	Requires background knowledge for computing complex numeric literals, such as durative effects; Does not address temporal considerations
Author	Segura-Muros et al. (2021)
Algorithm used	Classification and Symbolic regression techniques that discover relational/numeric expressions, or extract the preconditions and effects of the actions
Input	Plan traces, and partial knowledge of intermediate states
Output	A set of numeric action models with numeric predicates, and arithmetic/logical relations between predicates
Environment	Classical planning domains with PDDL representation
Merits	Able to learn valid planning domains (i.e. a set of numeric action models with preconditions and effects), even with high levels of incompleteness in the input states

Limitations	Does not support the planning domains with durative actions; Unable to learn the non-liner relationship between numeric fluent
Author	Lindsay et al. (2020)
Algorithm used	Supervised Machine Learning approach (i.e. decision tree)
Input	the pre-engineered hybrid planning domains with observational data
Output	the significant temporal features and states of original planning domains
Environment	Hybrid planning domains with PDDL+ representation
Merits	Can reduce the knowledge engineering effort in producing a detailed process model by the automated refinement of hybrid (planning) domain models
Limitations	Does not consider the associations/dependencies between numeric features (i.e. process parameters) in the processes; Can not identify the effective/insignificant numeric features automatically in the underlying processes; Unable to capture important phenomena in the running processes, and consequently inadequate to capture the dynamic fluctuations in the numeric variables
Method	Process Model Improvement (PMI)
Algorithm used	Different linear regression techniques (i.e. Simple linear regression, Stepwise Regression, Ridge regression etc.) with corresponding statistical tests (i.e. VIF, ANOVA, t-test etc.), and Statistical methods such as Cross Validation, Pearson Correlation Coefficient etc.
Input	Quantitative (time-series) data along with the pre-engineered, original hybrid domain model (i.e. PDDL+ domain and problem definition)
Output	The learned domain model with embedded Process Models (i.e. The PDDL+ domain definition integrated with the mathematical Process Model, and The PDDL+ problem definition that initialises the coefficient values of Process Model)
Environment	Hybrid planning domains with PDDL+ representation

Merits	Upgrades the simulation accuracy by approximating the more rational and pragmatic value of outcome variable in the running process; Improved simulation output with the refined domain models can consequently lead to higher quality plans; Facilitates the knowledge engineering task by automatically learns the dynamic values of numeric variables, without declaring them statically; Assists the knowledge engineers by automatically identifies the significant/ineffective process variables (i.e. numeric features) along with their interdependencies
Limitations	Cannot address non-linear relationships among numeric variables (i.e. process parameters); Unable to formulate Process Models in the absence of relevant data, or the relationship between outcome and any predictor variables

Table 7.3 Comparison table of Related research

CHAPTER 8

Conclusion and Future Work

A hybrid planner requires adequate modelling of numeric features and their continuously changing effects in the processes of real-time (hybrid) planning domains (Fox and Long 2006). Besides, the accurate measurement of such continuous variables are essential to effectively control a process in hybrid domains. In this thesis, we propose a method, named PMI, to automatically learn accurate and run-time representative estimation of continuous changes in the processes of hybrid planning domains. Specifically, PMI method estimates the fluctuating quantities of numeric variables utilising the process models. The process models are formulated from real-time observational/simulation data by exploiting different linear regression techniques. The learned process models are then integrated into the process effects of pre-engineered hybrid domain models with PDDL+ representation, whereas the parameter values of process models are initialised in the problem definition of PDDL+ domain model.

To illustrate the feasibility of PMI method and evaluate on realistic planning applications, we perform empirical analysis with the PDDL+ model of Urban Traffic Control (UTC) domain and Coffee domain. The results demonstrate that the learned process model integrated with an engineered hybrid domain model provides more accurate plan simulation which is more rational and closer to the actual process model. Besides, the Learned domain model is more rational and pragmatic by automatically estimating/adjusting the continuously changing quantities of numeric fluent without declaring them statically. In addition, PMI method can assist knowledge engineering task by choosing effective process parameters (i.e. numeric features) and identifying/removing the irrelevant ones from the process models. In a nutshell, the learned process models integrated into hybrid planning domains can lead to higher-quality plans by enhancing the simulation accuracy.

8.1 Limitations

We demonstrate with empirical analysis that the induced process models can identify/-choose the effective numeric fluent and efficiently approximate their changing quanti-

ties in the processes of hybrid planning domains. However, the proposed PMI method has three main limitations that we discuss in this section.

First, the PMI method formulates Process Models (PM) by exploiting real-world data collected from actual execution of the underlying process in (hybrid) planning domains. In other words, the PMI method entirely relies on the relevant data with respect to formulate process models for corresponding planning domains. Therefore, the performance of PMI method depends on the availability of relevant data. However, ensuring or managing the data availability emerges some challenges such as data relevance, data quality, data accessibility, reliability of access, timeliness, and continuity of information etc.

Second, the PMI method focuses on learning the continuous linear changes in the process parameters. Therefore the formulated process models represent the linear relationship between outcome and predictor variables. In other words, the PMI method cannot address non-linear relationships among numeric variables.

Third, an exception can happen where the outcome variable (y) may not be dependent on any predictor variables (X). In short, y may not related with any X variables in the process. Though, in AI planning context, such type of exception is quite impossible/rare because knowledge engineers/domain experts build hybrid domain models with processes based on some relationship assumptions among numeric variables.

8.2 Future Work

There are a number of possible future directions to enhance and improve our work further. First, the PMI method only learns the Process Model (PM) that automatically capture and adjust process effects, or continuous changes in the numeric features for hybrid planning domains. In future work we will expand our method within a framework for learning/inferring arithmetic and relational (i.e. logical) expressions that define the precondition and effects of processes. Second, we will reduce the manual efforts of modelling processes by automating the entire step of PM integration, for instance, it will automatically learn/adjust the learned PM along with the objects, predicates, parameter types, preconditions, and other numeric expressions etc in the process specification accordingly. Third, we will consider non-linear relationship among numeric variables in order to capture the non-linear effects of processes in the hybrid (planning) domains. Thereby various non-linear regression techniques will be experimented with hybrid domains and will be utilised accordingly. Fourth, Lasso regression can be a better shrinkage method than ridge with respect to handle Multicollinearity with highly correlated predictors (i.e. independent variables). However, ridge works well with large set of predictors and have most impact on outcome, for example, traffic flow rate is affected by several influencing factors in the UTC domain. In our future work we will

apply both of the shrinkage methods such as Lasso and Ridge regression according to the (planning) application domain. Fifth, the significance test of ridge regression models developed by Cule et al. (2011) can be useful, where the t-test may fail to provide satisfactory results. Therefore, it will be implemented along with other statistical tests.

8.2.1 Potential Application Area

For future work, we aim to explore some potential application areas where the need of learning/refining domain knowledge with processes is crucial. For instance, Autonomous manufacturing cell for creating Ultra-Precision Surfaces (UPS) (Walker et al. 2019), Urban Traffic Management (UTM) for autonomous management of traffic flows (Antoniou et al. 2019), and petroleum refinery production for getting more cost-effective refinery (Fox and Long 2006) etc.

An example of a *potential* application of our process improvement method from a different but no less important area, is one in the domain of Ultra Precision Manufacturing (UPM), where an automated planning process is needed to control the path and forces of a polishing tool within polishing processes (Walker et al. 2019, 1998, Kim et al. 1996). After a grinding process in surface manufacture, we need to polish the surface of the work-piece. For bonnet polishing, the surface material is removed by pressure of the pad, friction of the abrasive-slurry and the velocity of work-piece/bonnet-pad. The Material Removal Rate (MRR) is normally calculated using **Preston's equation**:

$$MRR = K \times P \times V$$

where P = pressure, V = Velocity and K = Preston coefficient which takes into account the work-piece material, polishing pads, abrasive size and material etc. Then the estimated MRR is used to predict the surface-form (Roughness of the surface in RMS or PV). Within a process model formulating the action of polishing, for input to an automated planner, Preston's equation would be used as the first approximation to determine the removal of surface material over time. However, it is known in the UPM community that it is not possible to pre-engineer an accurate process model formulation by hand, given the range of features that the removal process depends on, for varying materials and tools. Given training data that measures the removal rate for a particular scenario, it is clear that a method (such as the one we propose in this thesis) to learn a more accurate removal rate is feasible. While we studied this domain with the intention of using our approach on it and investigate with real-time data, at the time of this thesis work the worldwide COVID-19 pandemic in the United Kingdom started in late January 2020. Due to the faster-spreading of Coronavirus disease (COVID-19) across the country, the UK government imposed an immediate national lockdown. During the

lockdown situation, all schools, universities, laboratories, non-essential shops etc were closed except grocery stores and pharmacies. Besides, all household mixing as well as official/educational meeting were banned. Consequently, it was not possible to collect training data to feed the machine learning algorithms, hence we see this as future work.

REFERENCES

Allen, M. P. (2004), *Understanding regression analysis*, Springer Science and Business Media.

Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C. and Norman, M. K. (2010), *How learning works: Seven research-based principles for smart teaching*, John Wiley and Sons.

Andueza, S., Maeztu, L., Pascual, L., Ibáñez, C., de Peña, M. P. and Cid, C. (2003), 'Influence of extraction temperature on the final quality of espresso coffee', *Journal of the Science of Food and Agriculture* **83**(3), 240–248.

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L. and Ridella, S. (2012), The'k'in k-fold cross validation., *in* 'ESANN'.

Antoniou, G., Batsakis, S., Davies, J., Duke, A., McCluskey, T. L., Peytchev, E., Tachmazidis, I. and Vallati, M. (2019), 'Enabling the use of a planning agent for urban traffic management via enriched and integrated urban data', *Transportation Research Part C: Emerging Technologies* **98**, 284–297.

Arora, A., Fiorino, H., Pellier, D., Métivier, M. and Pesty, S. (2018), 'A review of learning planning action models', *The Knowledge Engineering Review* **33**.

Arvay, A. and Langley, P. (2016), Selective induction of rate-based process models, *in* 'Proceedings of the Fourth Annual Conference on Cognitive Systems'.

Barceló, J. and Casas, J. (2005), Dynamic network simulation with aimsun, *in* 'Simulation approaches in transportation analysis', Springer, pp. 57–98.

Bay, S. D., Shapiro, D. G. and Langley, P. (2002), Revising engineering models: Combining computational discovery with knowledge, *in* 'European Conference on Machine Learning', Springer, pp. 10–22.

Bell, K., Coles, A., Coles, A., Fox, M. and Long, D. (2009), 'The role of ai planning as a decision support tool in power substation management', *Ai Communications* **22**(1), 37–57.

Berry, W. D., Feldman, S. and Stanley Feldman, D. (1985), *Multiple regression in practice*, number 50, Sage.

Bhatti, F., Kitchin, D. and Vallati, M. (2019), A general approach to exploit model predictive control for guiding automated planning search in hybrid domains, *in* 'International Conference on Innovative Techniques and Applications of Artificial Intelligence', Springer, pp. 139–145.

Bretherton, R. (1990), 'Scoot urban traffic control system—philosophy and evaluation', *IFAC Proceedings Volumes* **23**(2), 237–239.

Bronshtein, A. (2017), 'Train/test split and cross validation in python'.
URL: *https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6*

Bruce, P., Bruce, A. and Gedeck, P. (2020), *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python*, O'Reilly Media.

Brushett, D. (2014), 'The perfect cup of coffee boils down to four factors'.
URL: *https://theconversation.com/the-perfect-cup-of-coffee-boils-down-to-four-factors-30208*

Bryce, D., Benton, J. and Boldt, M. W. (2016), Maintaining evolving domain models, *in* 'Proceedings of the twenty-fifth international joint conference on artificial intelligence', pp. 3053–3059.

Bryce, D., Bogomolov, S., Heinz, A. and Schilling, C. (2016), 'Instrumenting an smt solver to solve hybrid network reachability problems', *arXiv preprint arXiv:1609.03847* .

Bryce, D., Gao, S., Musliner, D. and Goldman, R. (2015), Smt-based nonlinear pddl+ planning, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 29.

Burnham, K. P. and Anderson, D. R. (2004), 'Multimodel inference: understanding aic and bic in model selection', *Sociological methods and research* **33**(2), 261–304.

Carbonell, J. G., Michalski, R. S. and Mitchell, T. M. (1983), An overview of machine learning, *in* 'Machine learning', Elsevier, pp. 3–23.

Cashmore, M., Fox, M., Long, D. and Magazzeni, D. (2016), A compilation of the full pddl+ language into smt, *in* 'Proceedings of the International Conference on Automated Planning and Scheduling', Vol. 26.

Cashmore, M., Fox, M., Long, D., Magazzeni, D. and Ridder, B. (2017), 'Opportunistic planning in autonomous underwater missions', *IEEE Transactions on Automation Science and Engineering* **15**(2), 519–530.

Cashmore, M., Magazzeni, D. and Zehtabi, P. (2020), 'Planning for hybrid systems via satisfiability modulo theories', *Journal of Artificial Intelligence Research* **67**, 235–283.

Cenamor, I., Chrpa, L., Jimoh, F., McCluskey, T. L. and Vallati, M. (2014), 'Planning and scheduling applications in urban traffic management'.

Chai, T. and Draxler, R. R. (2014), 'Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature', *Geoscientific model development* **7**(3), 1247–1250.

Chrpa, L., Magazzeni, D., McCabe, K., McCluskey, T. L. and Vallati, M. (2016), 'Automated planning for urban traffic control: Strategic vehicle routing to respect air quality limitations', *Intelligenza Artificiale* **10**(2), 113–128.

Claus, C. and Boutilier, C. (1998), 'The dynamics of reinforcement learning in cooperative multiagent systems', *AAAI/IAAI* **1998**(746-752), 2.

Cohen, J. (2013), *Statistical power analysis for the behavioral sciences*, Academic press.

Coles, A. J. and Coles, A. I. (2014), Pddl+ planning with events and linear processes, *in* 'Twenty-Fourth International Conference on Automated Planning and Scheduling'.

Coles, A. J., Coles, A. I., Fox, M. and Long, D. (2012), 'Colin: Planning with continuous linear numeric change', *Journal of Artificial Intelligence Research* **44**, 1–96.

Craven, B. and Islam, S. M. (2011), *Ordinary least-squares regression*, Sage Publications.

Cresswell, S. and Gregory, P. (2011), Generalised domain model acquisition from action traces, *in* 'Proceedings of the International Conference on Automated Planning and Scheduling', Vol. 21.

Cresswell, S. N., McCluskey, T. L. and West, M. M. (2013), 'Acquiring planning domain models using locm', *Knowledge Engineering Review* **28**(2), 195–213.

Cule, E., Vineis, P. and De Iorio, M. (2011), 'Significance testing in ridge regression for genetic data', *BMC bioinformatics* **12**(1), 1–15.

Dalton, J. (1802), 'On the constitution of mixed gases, on the force of steam of vapour from water and other liquids in different temperatures, both in a torricellia vacuum and in air; on evaporation; and on the expansion of gases by heat', *Memoirs, Literary and Philosophical Society of Manchester* **5**(2), 536–602.

Danubianu, M. (2015), Step by step data preprocessing for data mining. a case study, *in* 'Proc. Of the International Conference on Information Technologies (InfoTech-2015)', pp. 117–124.

De La Rosa, T., Celorrio, S. J. and Borrajo, D. (2008), Learning relational decision trees for guiding heuristic planning., *in* 'ICAPS', pp. 60–67.

De la Rosa, T., Jiménez, S., Fuentetaja, R. and Borrajo, D. (2011), 'Scaling up heuristic planning with relational decision trees', *Journal of Artificial Intelligence Research* **40**, 767–813.

Della Penna, G., Intrigila, B., Magazzeni, D. and Mercorio, F. (2010), A pddl+ benchmark problem: The batch chemical plant, *in* 'Twentieth International Conference on Automated Planning and Scheduling', Citeseer.

Della Penna, G., Magazzeni, D. and Mercorio, F. (2012), 'A universal planning system for hybrid domains', *Applied intelligence* **36**(4), 932–959.

Della Penna, G., Magazzeni, D., Mercorio, F. and Intrigila, B. (2009), Upmurphi: A tool for universal planning on pddl+ problems, *in* 'Proceedings of the International Conference on Automated Planning and Scheduling', Vol. 19.

Easthope, A. (2015), 'Brew temperature and its effects on espresso'.
URL: *https://www.fivesenses.com.au/blog/brew-temperature-and-its-effects-on-espresso*

Edelkamp, S. and Hoffmann, J. (2004), Pddl2. 2: The language for the classical part of the 4th international planning competition, Technical report, Technical Report 195, University of Freiburg.

Erol, K., Hendler, J. A. and Nau, D. S. (1994), Umcp: A sound and complete procedure for hierarchical task-network planning., *in* 'Aips', Vol. 94, pp. 249–254.

Faraway, J. J. (2002), *Practical regression and ANOVA using R.*, Vol. 168, University of Bath Bath.

Fikes, R. E. and Nilsson, N. J. (1971), 'Strips: A new approach to the application of theorem proving to problem solving', *Artificial intelligence* **2**(3-4), 189–208.

Fine-Morris, M., Auslander, B., Muños-Avila, H. and Gupta, K. (2020), Learning actions with symbolic literals and continuous effects for a waypoint navigation simulation, *in* 'Proceedings of SAI Intelligent Systems Conference', Springer, pp. 86–96.

Fox, M. and Long, D. (2002), Pddl+: Modeling continuous time dependent effects, *in* 'Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space', Vol. 4, p. 34.

Fox, M. and Long, D. (2003), 'Pddl2.1: An extension to pddl for expressing temporal planning domains', *Journal of artificial intelligence research* **20**, 61–124.

Fox, M. and Long, D. (2006), 'Modelling mixed discrete-continuous domains for planning', *Journal of Artificial Intelligence Research* **27**, 235–297.

Fox, M., Long, D. and Magazzeni, D. (2012), 'Plan-based policies for efficient multiple battery load management', *Journal of Artificial Intelligence Research* **44**, 335–382.

Freund, R. J., Wilson, W. J. and Sa, P. (2006), *Regression analysis*, Elsevier.

Gerevini, A. and Long, D. (2005), 'Plan constraints and preferences in pddl3-the language of the fifth international planning competition'.

Ghallab, M., Nau, D. and Traverso, P. (2004), *Automated Planning: theory and practice*, Elsevier.

Ghallab, M., Nau, D. and Traverso, P. (2016), *Automated planning and acting*, Cambridge University Press.

Gogtay, N., Deshpande, S. and Thatte, U. (2017), 'Principles of regression analysis', *Journal of the Association of Physicians of India* **65**, 48–52.

Gogtay, N. and Thatte, U. (2017), 'Principles of correlation analysis', *Journal of the Association of Physicians of India* **65**(3), 78–81.

Golub, G. H., Heath, M. and Wahba, G. (1979), 'Generalized cross-validation as a method for choosing a good ridge parameter', *Technometrics* **21**(2), 215–223.

Gomoluch, P. K. (2020), Learning heuristic functions and search policies for classical planning, PhD thesis, Imperial College London.

Gregory, P. and Cresswell, S. (2015), Domain model acquisition in the presence of static relations in the lop system, *in* 'Proceedings of the International Conference on Automated Planning and Scheduling', Vol. 25.

Gregory, P. and Lindsay, A. (2016), Domain model acquisition in domains with action costs, *in* 'Proceedings of the International Conference on Automated Planning and Scheduling', Vol. 26.

Groebner, D. F., Shannon, P. W. and Fry, P. C. (2018), *Business statistics: a decision-making approach*, tenth;global; edn, Pearson, Harlow.

Gulić, M., Olivares, R. and Borrajo, D. (2016), 'Using automated planning for traffic signals control', *PROMET-TrafficandTransportation* **28**(4), 383–391.

Haas, N. (2015), 'Modules (regressors.stats)'.
URL: *https://regressors.readthedocs.io/en/latest/modules.html*

He, N. and Zhao, S. (2013), 'Discussion on influencing factors of free-flow travel time in road traffic impedance function', *Procedia-Social and Behavioral Sciences* **96**, 90–97.

Heckert, N. A., Filliben, J. J., Croarkin, C. M., Hembree, B., Guthrie, W. F., Tobias, P. and Prinz, J. (2002), 'Handbook 151: Nist/sematech e-handbook of statistical methods'.

Henzinger, T. (1996), 'The theory of hybrid automata, lics'96: Proceedings of the 11th annual ieee symposium on logic in computer science (washington, dc, usa)', *IEEE Computer Society* p. 278.

Hocking, R. R. (1976), 'A biometrics invited paper. the analysis and selection of variables in linear regression', *Biometrics* **32**(1), 1–49.

Hoerl, A. E. and Kennard, R. W. (2000), 'Ridge regression: Biased estimation for nonorthogonal problems', *TECHNOMETRICS* **42**(1), 80.

Jabbar, H. and Khan, R. Z. (2015), 'Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)', *Computer Science, Communication and Instrumentation Devices* pp. 163–172.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Jilani, R., Crampton, A., Kitchin, D. E. and Vallati, M. (2014), 'Automated knowledge engineering tools in planning: state-of-the-art and future challenges'.

Jilani, R., Crampton, A., Kitchin, D. and Vallati, M. (2015), Ascol: A tool for improving automatic planning domain model acquisition, *in* 'Congress of the Italian association for artificial intelligence', Springer, pp. 438–451.

Jiménez, S., De La Rosa, T., Fernández, S., Fernández, F. and Borrajo, D. (2012), 'A review of machine learning for automated planning', *The Knowledge Engineering Review* **27**(4), 433–467.

KCL-Planning (2019), 'Kcl-planning/val'.
URL: *https://github.com/KCL-Planning/VAL/tree/master/samples/domainsWithEventsAndProcesses*

Kim, S.-W., Rees, D. J., Walker, D. D., Bingham, R. G., Brooks, D., Humm, B., Jamshidi, H. O., Kim, D.-H., Yang, H.-S. and Nixon, G. (1996), Oglp-400: an innovative computer-controlled polishing machine, *in* 'Specification, Production, and Testing of Optical Components and Systems', Vol. 2775, International Society for Optics and Photonics, pp. 491–496.

Kocsis, L. and Szepesvári, C. (2006), Bandit based monte-carlo planning, *in* 'European conference on machine learning', Springer, pp. 282–293.

Korhonen, J. (2019), 'Creating the perfect espresso recipe'.
URL: *https://www.baristainstitute.com/blog/jori-korhonen/july-2019/creating-perfect-espresso-recipe*

Kovacs, D. L. (2011), 'Complete bnf description of pddl 3.1', *Dept. Meas. Inf. Syst., Budapest Univ. Technol. Econ., Budapest, Hungary* .

Krishni (2018), 'K-fold cross validation'.
URL: *https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833*

Lanchas, J., Jiménez, S., Fernández, F. and Borrajo, D. (2007), Learning action durations from executions, *in* 'Proceedings of the ICAPS', Vol. 7, Citeseer.

Langley, P. and Arvay, A. (2017), Flexible model induction through heuristic process discovery, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 31.

Langley, P., George, D., Bay, S. D. and Saito, K. (2003), Robust induction of process models from time-series data, *in* 'Proceedings of the 20th International Conference on Machine Learning (ICML-03)', pp. 432–439.

Langley, P., Sanchez, J. N., Todorovski, L. and Dzeroski, S. (2002), 'Inducing process models from continuous data'.

Leckie, C. and Zukerman, I. (1998), 'Inductive learning of search control rules for planning', *Artificial Intelligence* **101**(1-2), 63–98.

Leonetti, M., Iocchi, L. and Stone, P. (2016), 'A synthesis of automated planning and reinforcement learning for efficient, robust decision-making', *Artificial Intelligence* **241**, 103–130.

Lin, L.-J. (1992), 'Self-improving reactive agents based on reinforcement learning, planning and teaching', *Machine learning* **8**(3-4), 293–321.

Lindsay, A., Franco, S., Reba, R. and McCluskey, T. L. (2020), Refining process descriptions from execution data in hybrid planning domain models, *in* 'Proceedings of the International Conference on Automated Planning and Scheduling', Vol. 30, pp. 469–477.

Little, J. D., Kelson, M. D. and Gartner, N. H. (1981), 'Maxband: A versatile program for setting signals on arteries and triangular networks'.

Lowrie, P. and PR, L. (1982), 'The sydney co-ordinated adaptive traffic system: Principles, methodology, algorithms'.

Ludwig, I. A., Sanchez, L., Caemmerer, B., Kroh, L. W., De Peña, M. P. and Cid, C. (2012), 'Extraction of coffee antioxidants: Impact of brewing time and method', *Food Research International* **48**(1), 57–64.

Magazzeni, D., Py, F., Fox, M., Long, D. and Rajan, K. (2014), 'Policy learning for autonomous feature tracking', *Autonomous Robots* **37**(1), 47–69.

Maurelli, F., Carreras, M., Salvi, J., Lane, D., Kyriakopoulos, K., Karras, G., Fox, M., Long, D., Kormushev, P. and Caldwell, D. (2016), The pandora project: A success story in auv autonomy, *in* 'OCEANS 2016-Shanghai', IEEE, pp. 1–8.

McCluskey, L. (2012), 'Coursework'.
URL: *https://selene.hud.ac.uk/scomtlm/cha2555/*

McCluskey, T. L., Cresswell, S., Richardson, N. E. and West, M. M. (2009), Action knowledge acquisition with opmaker2, *in* 'International conference on agents and artificial intelligence', Springer, pp. 137–150.

McCluskey, T. L., Richardson, N. E. and Simpson, R. M. (2002), An interactive method for inducing operator descriptions., *in* 'AIPS', pp. 121–130.

McCluskey, T. L., Vallati, M. and Franco, S. (2017), Automated planning for urban traffic management., *in* 'IJCAI', pp. 5238–5240.

McCluskey, T. L., Vaquero, T. S. and Vallati, M. (2017), Engineering knowledge for automated planning: Towards a notion of quality, *in* 'Proceedings of the Knowledge Capture Conference', pp. 1–8.

McCluskey, T. L., Vaquero, T. and Vallati, M. (2016), 'Issues in planning domain model engineering'.

McCluskey, T. and Vallati, M. (2017), Embedding automated planning within urban traffic management operations, *in* '27th International Conference on Automated Planning and Scheduling', Association for the Advancement of Artificial Intelligence.

McDermott, D. (2003), 'Pddl2. 1–the art of the possible? commentary on fox and long', *Journal of Artificial Intelligence Research* **20**, 145–148.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D. and Wilkins, D. (1998), 'PDDL - The Planning Domain Definition Language'.

McDermott, D. M. (2000), 'The 1998 ai planning systems competition', *AI magazine* **21**(2), 35–35.

Melrose, J., Roman-Corrochano, B., Montoya-Guerra, M. and Bakalis, S. (2018), 'Toward a new brewing control chart for the 21st century', *Journal of agricultural and food chemistry* **66**(21), 5301–5309.

Mitchell, R., Michalski, J. and Carbonell, T. (2013), *An artificial intelligence approach*, Springer.

Palomeras, N., Carrera, A., Hurtós, N., Karras, G. C., Bechlioulis, C. P., Cashmore, M., Magazzeni, D., Long, D., Fox, M., Kyriakopoulos, K. J. et al. (2016), 'Toward persistent autonomous intervention in a subsea panel', *Autonomous Robots* **40**(7), 1279–1306.

Papageorgiou, M., Ben-Akiva, M., Bottom, J., Bovy, P. H., Hoogendoorn, S. P., Hounsell, N. B., Kotsialos, A. and McDonald, M. (2007), 'Its and traffic management', *Handbooks in operations research and management science* **14**, 715–774.

Paul, R. K. (2006), 'Multicollinearity: Causes, effects and remedies', *IASRI, New Delhi* **1**(1), 58–65.

Pednault, E. P. (1989), 'Adl: Exploring the middle ground between strips and the situation calculus.', *Kr* **89**, 324–332.

Penna, G. D., Magazzeni, D., Mercorio, F. and Intrigila, B. (2009), Upmurphi: A tool for universal planning on pddl+ problems, *in* 'Proceedings of the Nineteenth International Conference on International Conference on Automated Planning and Scheduling', pp. 106–113.

Percassi, F., Gerevini, A. E., Scala, E., Serina, I. and Vallati, M. (2020), Generating and exploiting cost predictions in heuristic state-space planning, *in* 'Proceedings of the International Conference on Automated Planning and Scheduling', Vol. 30, pp. 569–573.

Perktold, J., Seabold, S. and Taylor, J. (2021), 'statsmodels.stats.outliers_influence.variance_inflation_factor'.
URL: *https://www.statsmodels.org/stable/user-guide.html*

Piacentini, C., Magazzeni, D., Long, D., Fox, M. and Dent, C. (2016), Solving realistic unit commitment problems using temporal planning: challenges and solutions, *in* 'Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling', pp. 421–430.

Piotrowski, W. M., Fox, M., Long, D., Magazzeni, D. and Mercorio, F. (2016), Heuristic planning for pddl+ domains., *in* 'AAAI Workshop: Planning for Hybrid Systems', Vol. 16, p. 12.

Plch, T., Chomut, M., Brom, C. and Barták, R. (2012), 'Inspect, edit and debug pddl documents: Simply and efficiently with pddl studio', *System Demonstrations and Exhibits at ICAPS* pp. 15–18.

Pozanco, A., Fernández, S. and Borrajo, D. (2018), Distributed planning and model learning for urban traffic control, *in* 'KEPS', p. 20.

Prechelt, L. (1998), 'Automatic early stopping using cross validation: quantifying the criteria', *Neural Networks* **11**(4), 761–767.

Preece, J., Sharp, H. and Rogers, Y. (2015), *Interaction design: beyond human-computer interaction*, John Wiley and Sons.

Pyle, D. (1999), *Data preparation for data mining*, morgan kaufmann.

Ramirez, M., Scala, E., Haslum, P. and Thiebaux, S. (2017), 'Numerical integration and dynamic discretization in heuristic search planning over hybrid domains', *arXiv preprint arXiv:1703.04232* .

Rogers, Y., Sharp, H. and Preece, J. (2011), *Interaction design: beyond human-computer interaction*, John Wiley and Sons.

Roman Corrochano, B. (2017), Advancing the engineering understanding of coffee extraction, PhD thesis, University of Birmingham.

Russell, S. and Norvig, P. (1995), 'Solving problems by searching', *Artificial intelligence: a modern approach* pp. 64–119.

Saxena, S. (2019), 'What's the difference between rmse and rmsle?'.
URL: *https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a*

Say, B., Wu, G., Zhou, Y. Q. and Sanner, S. (2017), Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming., *in* 'IJCAI', pp. 750–756.

SCA (2018), 'Coffee standards'.
URL: *https://sca.coffee/research/coffee-standards*

Scala, E., Haslum, P. and Thiébaux, S. (2016), Heuristics for numeric planning via subgoaling, *in* 'Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence', pp. 3228–3234.

Seabold, S. and Perktold, J. (2010), statsmodels: Econometric and statistical modeling with python, *in* '9th Python in Science Conference'.

Segura-Muros, J. Á., Pérez, R. and Fernández-Olivares, J. (2018), 'Learning numerical action models from noisy and partially observable states by means of inductive rule learning techniques', *KEPS* **46**, 2018.

Segura-Muros, J. Á., Pérez, R. and Fernández-Olivares, J. (2021), 'Discovering relational and numerical expressions from plan traces for learning action models', *Applied Intelligence* pp. 1–17.

Shah, M., Chrpa, L., Jimoh, F., Kitchin, D., McCluskey, T., Parkinson, S. and Vallati, M. (2013), 'Knowledge engineering tools in planning: State-of-the-art and future challenges', *Knowledge engineering for planning and scheduling* **53**, 53.

Shin, J.-A. and Davis, E. (2005), 'Processes and continuous change in a sat-based planner', *Artificial Intelligence* **166**(1-2), 194–253.

Simpson, R. M., Kitchin, D. E. and McCluskey, T. L. (2007), 'Planning domain definition using gipo', *The Knowledge Engineering Review* **22**(2), 117–134.

Smith, D. E., Frank, J. and Cushing, W. (2008), The anml language, *in* 'The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)'.

Sutton, R. S. (1991), 'Dyna, an integrated architecture for learning, planning, and reacting', *ACM Sigart Bulletin* **2**(4), 160–163.

Tate, A. and Wickler, G. (2015), 'Machine learning and adaptation of domain models to support real time planning in autonomous systems–final summary report'.

Tate, A., Wickler, G., McCluskey, L. and Chrpa, L. (2012), 'Machine learning and adaptation of domain models to support real time planning in autonomous systems', *HEdLAMP–Huddersfield+ Edinburgh: Learning and Adaptation of Models for Planning, University of Edinburgh* .

Thomas, A. and Henzinger, Z. (1996), The theory of hybrid automata, *in* '11th Annual IEEE Symposium on Logic in Computer Science', pp. 278–292.

Todorovski, L., Bridewell, W., Shiran, O. and Langley, P. (2005), Inducing hierarchical process models in dynamic domains, *in* 'AAAI', pp. 892–897.

Todorovski, L. and Dzeroski, S. (1997), Declarative bias in equation discovery, *in* 'Proceedings of the Fourteenth International Conference on Machine Learning', Morgan Kaufmann, pp. 376–384.

Turner, J. R. and Thayer, J. (2001), *Introduction to analysis of variance: design, analyis and interpretation*, Sage Publications.

Vallati, M., Hutter, F., Chrpa, L. and McCluskey, T. L. (2015), On the effective configuration of planning domain models, *in* 'International Joint Conference on Artificial Intelligence (IJCAI)', AAAI press.

Vallati, M., Magazzeni, D., De Schutter, B., Chrpa, L. and McCluskey, T. L. (2016), Efficient macroscopic urban traffic models for reducing congestion: a pddl+ planning approach, AAAI press.

Vaquero, T. S., Romero, V., Tonidandel, F. and Silva, J. R. (2007), itsimple 2.0: An integrated tool for designing planning domains., *in* 'ICAPS', pp. 336–343.

Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E. and Blythe, J. (1995), 'Integrating planning and learning: The prodigy architecture', *Journal of Experimental and Theoretical Artificial Intelligence* **7**(1), 81–120.

Walker, D. D., Kim, S.-W., Bingham, R. G., Brooks, D., Kim, D.-H. and Thirtle, J. (1998), Computer-controlled polishing of moderate-sized general aspherics for instrumentation, *in* 'Optical Astronomical Instrumentation', Vol. 3355, International Society for Optics and Photonics, pp. 947–954.

Walker, D. D., McCluskey, T. L., Yu, G., Petrovic, S. and Li, H. (2019), 'Fully automating fine-optics manufacture-why so tough, and what are we doing?', *Journal of the European Optical Society-Rapid Publications* **15**(1), 1–10.

Wang, X. (1996), Learning planning operators by observation and practice, PhD thesis, Carnegie Mellon University.

Watkins, C. J. and Dayan, P. (1992), 'Q-learning', *Machine learning* **8**(3-4), 279–292.

Wei, H., Zheng, G., Gayah, V. and Li, Z. (2019), 'A survey on traffic signal control methods', *arXiv preprint arXiv:1904.08117* .

Weld, D. and Penberthy, J. (1992), Ucpop: A sound, complete, partial order planner for adl, *in* '1992 International Conference on Principles of Knowledge Representation and Reasoning', pp. 103–114.

Wu, G., Say, B. and Sanner, S. (2017), 'Scalable planning with tensorflow for hybrid nonlinear domains', *arXiv preprint arXiv:1704.07511* .

Zaiontz, C. (2014*a*), 'Method of least squares for multiple regression'.
URL: *https://www.real-statistics.com/multiple-regression/least-squares-method-multiple-regression/*

Zaiontz, C. (2014*b*), 'Method of least squares for multiple regression detailed'.
URL: *https://www.real-statistics.com/multiple-regression/least-squares-method-multiple-regression/least-squares-method-multiple-regression-detailed/*

Zaiontz, C. (2015*a*), 'Basic concepts of correlation'.
URL: *https://www.real-statistics.com/correlation/basic-concepts-correlation/*

Zaiontz, C. (2015*b*), 'Measures of variability'.
URL: *https://www.real-statistics.com/descriptive-statistics/measures-variability/*

Zhuo, H. H. and Kambhampati, S. (2013), Action-model acquisition from noisy plan traces, *in* 'Twenty-Third International Joint Conference on Artificial Intelligence'.

Zhuo, H. H., Nguyen, T. A. and Kambhampati, S. (2013), Refining incomplete planning domain models through plan traces., *in* 'IJCAI', pp. 2451–2458.

Zimmerman, T. and Kambhampati, S. (2003), 'Learning-assisted automated planning: looking back, taking stock, going forward', *AI Magazine* **24**(2), 73–73.

Appendix A

PDDL+ representation of Coffee Domain (Original Model)

A.1 Domain Definition

```
(define (domain COFFEE)
  (:requirements :typing :durative-actions :fluents :timed-initial-
   literals :time
    :duration-inequalities)

  (:types coffee water)

  (:predicates
    (havecoffee ?c - coffee)
    (hot ?w - water)
    (cold ?w - water)
    (madecoffee ?c - coffee ?w - water)
    (heating ?w - water)
    (cooling ?w - water)
    (boiled ?w - water)
  )

  (:functions
    (temperature ?w - water)
  )


  (:durative-action makecoffee
  :parameters (?c - coffee ?w - water)
  :duration (>= ?duration 1)
  :condition (and
          (at start (boiled ?w))
          (over all (>= (temperature ?w) 60))
          (over all (<= (temperature ?w) 80))
          (at start (havecoffee ?c))
        )
  :effect (and (at end (madecoffee ?c ?w)) )
  )
```

```
33
34
35  (:action heatwater
36  :parameters (?w - water)
37  :precondition (and (cold ?w))
38  :effect (and (not (cold ?w))
39            (heating ?w)
40      )
41  )
42
43
44  (:process heating
45  :parameters (?w - water)
46  :precondition (and (heating ?w) (<= (temperature ?w) 100))
47  :effect (increase (temperature ?w) (* #t 2))
48  )
49
50
51  (:event stop-heating
52  :parameters (?w - water)
53  :precondition (and (>= (temperature ?w) 100) (heating ?w))
54  :effect (and (not (heating ?w)))
55  )
56
57
58  (:event boil
59  :parameters (?w - water)
60  :precondition (and (>= (temperature ?w) 100)
61                (not (boiled ?w))
62          )
63  :effect (and (boiled ?w)) ;; boiled used in the durative action
64  )
65
66
67  (:process cooling
68  :parameters (?w - water)
69  :precondition (>= (temperature ?w) 18)
70  :effect (decrease (temperature ?w) (* #t 0.5))
71  )
72
73
74  (:event stop-cooling
75  :parameters (?w - water)
76  :precondition (and (<= (temperature ?w) 18) (cooling ?w))
77    ;; Assume we are working with a room temperature of
78    ;; 18 degrees
79  :effect (and (not (cooling ?w)))
```

```
80   )
81
82 )
```

A.2 Problem Definition

```
1 (define (problem COFFEE1)
2   (:domain COFFEE)
3   (:objects
4     water1 - water
5     coffee1 - coffee
6   )
7
8   (:init
9     (havecoffee coffee1)
10    (cold water1)
11    (= (temperature water1) 7) ;; Cold tap water at 7 degrees
12  )
13
14  (:goal (and (madecoffee coffee1 water1)))
15 )
```

Appendix B

PDDL+ formulation of Coffee Domain (Learned Model)

B.1 Domain Definition

```
(define (domain coffee)
  ;;(:requirements :typing :durative-actions :fluents :timed-initial-
    literals :time :duration-inequalities)
  (:types coffee water)

  (:predicates
    (havecoffee ?c - coffee)
    (hot ?w - water)
    (cold ?w - water)
    (madecoffee ?c - coffee ?w - water)
    (heating ?w - water)
    (brewing ?c - coffee ?w - water)
    (brewed ?c - coffee)
  )
  (:functions
    (temperature ?w - water)
    (brew-temperature ?w - water)
    (brew-time ?c - coffee ?w - water) ;brew-time or shot time
    (eyield ?c - coffee ?w - water) ;eyield = expected yield
    (yield ?c - coffee ?w - water)

    ;;coefficients
    (beta0 ?c - coffee ?w - water)
    (beta1 ?c - coffee ?w - water)
    (beta2 ?c - coffee ?w - water)
  )

  (:action heatwater
    :parameters (?c - coffee ?w - water)
    :precondition (and (havecoffee ?c) (cold ?w) )
    :effect (and (not (cold ?w)) (heating ?w) )
  )

```

```
33  (:process heating
34    :parameters (?c - coffee ?w - water)
35    :precondition (and (heating ?w))
36    :effect (and (increase (temperature ?w) (* #t 2)) )
37  )
38
39  (:event stop-heating
40    :parameters (?c - coffee ?w - water)
41    :precondition (and (heating ?w)
42                (>= (temperature ?w) (brew-temperature ?w))
43             )
44    :effect (and (not (heating ?w)) (hot ?w) )
45  )
46
47  (:action brew-coffee
48    :parameters (?c - coffee ?w - water)
49    :precondition (and (hot ?w) (<= ( yield ?c ?w) 0) )
50    :effect (and (brewing ?c ?w) )
51  )
52
53  (:process brewing
54    :parameters (?c - coffee ?w - water)
55    :precondition (and (brewing ?c ?w))
56    :effect (and
57          ;;**** brewing process with PM of extraction yield *****
58          ;;Assume, the temperature is constant (multiple regression)
59           (assign (yield ?c ?w)
60               (+ (beta0 ?c ?w)
61                (+ (* (beta1 ?c ?w) (temperature ?w))
62                   (* (beta2 ?c ?w) (brew-time ?c ?w))
63                )
64               )
65           )
66
67           (increase (brew-time ?c ?w) (* #t 1))
68        )
69  )
70
71  (:event stop-brewing
72    :parameters (?c - coffee ?w - water)
73    :precondition (and (brewing ?c ?w) (>= (yield ?c ?w) (eyield ?c ?
     w)) )
74    :effect (and (not (brewing ?c ?w))
75           (brewed ?c)
76        )
77  )
78
```

```
79   (:action serve-coffee
80     :parameters (?c - coffee ?w - water)
81     :precondition (and (brewed ?c))
82     :effect (and (madecoffee ?c ?w))
83   )
84
85 )
```

B.2 Problem Definition (For Temperature $96°C$)

```
1  (define (problem coffee-maker)
2    (:domain coffee)
3    (:objects
4      water1 - water
5      coffee1 - coffee
6    )
7
8    (:init
9      (havecoffee coffee1)
10     (cold water1)
11
12     ;; Cold tap water at 8 degrees or 7 degrees
13     (= (temperature water1) 8)
14     (= (brew-temperature water1) 96)
15
16     ;;expected yield for brewing temperature 96 is 19.40
17     (= (eyield coffee1 water1) 19.40)
18     (= (yield coffee1 water1) 0)
19     (= (brew-time coffee1 water1) 1)
20
21     ;; intercept
22     (= (beta0 coffee1 water1)  3.597629)
23     ;; regression coefficient of brew-temperature
24     (= (beta1 coffee1 water1) 0.103934 )
25     ;; regression coefficient of brew-time
26     (= (beta2 coffee1 water1) 0.196035 )
27
28   )
29
30   (:goal (and (madecoffee coffee1 water1)) )
31
32 )
```

Appendix C

PDDL+ representation of UTC Domain (Original Model)

C.1 Domain Definition

```
(define (domain urbantraffic)
  ;;(:requirements :typing :fluents :time :timed-initial-literals :
   duration-inequalities :adl)

  (:types junction link stage)

  (:predicates
    (controllable ?i - junction)
    (inter ?p - stage)
    (active ?p - stage)
    (next ?p ?p1 - stage)
    (trigger ?i - junction)
    (contains ?i - junction ?p - stage)
  )

  (:functions
    (turnrate ?p - stage ?r1 - link  ?r2 - link)
    (interlimit ?p - stage)
    (occupancy ?r - link)
    (capacity ?r - link)
    (maxgreentime  ?p - stage )
    (mingreentime ?p - stage )
    (greentime ?i - junction)
    (intertime ?i - junction)
  )

  ;; the maximum time limit for green has been reached, but no need
   to restart token!
  (:event maxgreenreached
   :parameters (?p - stage ?i - junction)
   :precondition (and (active ?p)
                      (contains ?i ?p)
                    (>= (greentime ?i)
```

```
                    (maxgreentime ?p))
   )
   :effect (and  (trigger ?i) )
)

;; process that keeps the green/intergreen on, and updates the
 greentime value
(:process keepgreen
:parameters (?p - stage ?i - junction)
:precondition (and (active ?p) (contains ?i ?p)
            (< (greentime ?i) (maxgreentime ?p))
    )
:effect (and (increase (greentime ?i) (* #t 1 ) ) )
)


;;allows car to flow if the corresponding green is on
(:process flowrun_green
:parameters (?p - stage ?r1 ?r2 - link)
:precondition (and
      (active ?p)
      (> (occupancy ?r1) 0.0)
      (> (turnrate ?p ?r1 ?r2) 0.0)
      (< (occupancy ?r2) (capacity ?r2))
    )
:effect (and
        (increase (occupancy ?r2) (* #t (turnrate ?p ?r1 ?r2)))
        (decrease (occupancy ?r1) (* #t (turnrate ?p ?r1 ?r2)))
  )
)

;; let the planner in control to stop the green before maxgreen
(:action switchPhase
:parameters (?p - stage ?i - junction)
:precondition (and (controllable ?i)
             (active ?p) (contains ?i ?p)
             (> (greentime ?i) (mingreentime ?p) )
    )
:effect (and (trigger ?i) )
)

(:event trigger-inter
 :parameters (?p - stage ?i - junction)
 :precondition (and  (trigger ?i)
                  (active ?p)
                  (contains ?i ?p)
             )
```

```
78      :effect (and  (not (trigger ?i))
79                    (not (active ?p))
80                    (inter ?p)
81                    (assign (greentime ?i) 0)
82       )
83    )
84
85      (:process keepinter
86        :parameters (?p - stage ?i - junction)
87        :precondition (and
88            (inter ?p) (contains ?i ?p)
89            (< (intertime ?i) (interlimit ?p)  )
90         )
91         :effect (and  (increase (intertime ?i) (* #t 1 ) ) )
92      )
93
94      (:event trigger-change
95      :parameters (?p ?p1 - stage ?i - junction)
96       :precondition (and (inter ?p) (contains ?i ?p)
97                          (next ?p ?p1)
98                          (>= (intertime ?i) (- (interlimit ?p) 0.1)  )
99        )
100       :effect (and  (not (inter ?p))
101                     (active ?p1)
102                   (assign (intertime ?i) 0)
103             )
104     )
105
106 )
```

C.2 Problem Definition (for junction $n3969$)

```
(define (problem manchester)
  (:domain urbantraffic)
  (:objects
      n3969 - junction
      n3969_n6612 n6612_n3969 n3972_n3969 n3969_n3972 n3969_n7646
   n3968_n3969  - link
      s3969_s2 s3969_s0 s3969_s1 - stage
  )
  (:init

      (controllable n3969)
      (active s3969_s2)

      (next s3969_s0 s3969_s1)
      (next s3969_s1 s3969_s2)
      (next s3969_s2 s3969_s0)

      (contains n3969 s3969_s0)
      (contains n3969 s3969_s1)
      (contains n3969 s3969_s2)

      (= (greentime n3969) 0)
      (= (intertime n3969) 0.0)

      (= (interlimit s3969_s0) 1.0)
      (= (interlimit s3969_s1) 5.0)
      (= (interlimit s3969_s2) 9.0)

      (= (mingreentime s3969_s0) 18)
      (= (maxgreentime s3969_s0) 117)
      (= (defaultgreentime s3969_s0) 28)
      (= (mingreentime s3969_s1) 26)
      (= (maxgreentime s3969_s1) 115)
      (= (defaultgreentime s3969_s1) 35)
      (= (mingreentime s3969_s2) 10)
      (= (maxgreentime s3969_s2) 89)
      (= (defaultgreentime s3969_s2) 44)

      ;;capacity
      (= (capacity n6612_n3969) 102.55667774)
      (= (capacity n3969_n6612) 76.1414616636)
      (= (capacity n3972_n3969) 58.5186831)
      (= (capacity n3969_n3972) 58.7995689051)
      (= (capacity n3969_n7646) 8.12771104258)
```

```
44        (= (capacity n3968_n3969) 38.116362703)
45
46        ;;occupancy
47        (= (occupancy n6612_n3969) 83.0)
48        (= (occupancy n3969_n6612) 0)
49        (= (occupancy n3972_n3969) 17.0)
50        (= (occupancy n3969_n3972) 50.0)
51        (= (occupancy n3969_n7646) 1.0)
52        (= (occupancy n3968_n3969) 37.0)
53
54        ;;=============== turnrate ======================
55        ;;s3969_s0
56        ;;---------- link1 ---------------
57        (= (turnrate s3969_s0 n6612_n3969 n3969_n3972) 1.030303)
58        ;;---------- link2 ---------------
59        (= (turnrate s3969_s0 n6612_n3969 n3969_n7646) 0.30303)
60        ;;--------- link3 ----------------
61        (= (turnrate s3969_s0 n3972_n3969 n3969_n6612) 1.583333)
62
63        ;;s3969_s1
64        ;;-------- link1 ----------------
65        (= (turnrate s3969_s1 n6612_n3969 n3969_n7646) 0.2)
66
67        ;;s3969_s2
68        ;;------- link1 ----------------
69        (= (turnrate s3969_s2 n3968_n3969 n3969_n3972) 0.741379)
70        ;;------- link3 ----------------
71        (= (turnrate s3969_s2 n3968_n3969 n3969_n7646) 0.465517)
72        ;;------ link2 -----------------
73        (= (turnrate s3969_s2 n3968_n3969 n3969_n6612) 0.2)
74    )
75
76    (:goal
77      (and
78        (< (occupancy n3968_n3969) 1)
79        (active s3969_s0)
80      )
81    )
82 )
```

Appendix D

PDDL+ formulation of UTC Domain (Learned Model)

D.1 Domain Definition

```
(define (domain urbantraffic)
    ;;(:requirements :typing :fluents :time :timed-initial-literals :
    duration-inequalities :adl)

    (:types junction link stage)

  (:predicates
    (controllable ?i - junction)
    (inter ?p - stage)
    (active ?p - stage)
    (next ?p ?p1 - stage)
    (trigger ?i - junction)
    (contains ?i - junction ?p - stage)
  )

  (:functions
    (turnrate ?p - stage ?r1 - link  ?r2 - link)
    (interlimit ?p - stage)
    (occupancy ?r - link)
    (capacity ?r - link)
    (maxgreentime  ?p - stage )
    (mingreentime ?p - stage )
    (greentime ?i - junction)
    (intertime ?i - junction)

    ;;intercept
    (beta0 ?p - stage ?r1 - link  ?r2 - link)
    ;;coeffcient of greentime
    (beta1 ?p - stage ?r1 - link  ?r2 - link)
    ;;coeffcient of intertime
    (beta2 ?p - stage ?r1 - link  ?r2 - link)
    ;;coeffcient of road1 saturation (occ/cap)
    (beta3 ?p - stage ?r1 - link  ?r2 - link)
```

```
33     ;;coeffcient of road2 saturation (occ/cap)
34     (beta4 ?p - stage ?r1 - link  ?r2 - link)
35   )
36
37
38   ;; the maximum time limit for green has been reached, but no need
       to restart token!
39   (:event maxgreenreached
40    :parameters (?p - stage ?i - junction)
41    :precondition (and (active ?p) (contains ?i ?p)
42                       (>= (greentime ?i) (maxgreentime ?p))
43                )
44     :effect (and  (trigger ?i) )
45   )
46
47
48   ;; process that keeps the green/intergreen on, and updates the
       greentime value
49   (:process keepgreen
50   :parameters (?p - stage ?i - junction)
51   :precondition (and (active ?p) (contains ?i ?p)
52                      (< (greentime ?i) (maxgreentime ?p))
53              )
54   :effect (and (increase (greentime ?i) (* #t 1 )) )
55   )
56
57
58   ;; let the planner in control to stop the green before maxgreen
59   (:action switchPhase
60   :parameters (?p - stage ?i - junction)
61   :precondition (and (controllable ?i)
62                      (active ?p) (contains ?i ?p)
63                   (> (greentime ?i) (mingreentime ?p) )
64              )
65   :effect (and (trigger ?i) )
66   )
67
68
69   (:event trigger-inter
70    :parameters (?p - stage ?i - junction)
71    :precondition (and  (trigger ?i)
72                        (active ?p)
73                        (contains ?i ?p)
74                )
75     :effect (and  (not (trigger ?i))
76                   (not (active ?p))
77                   (inter ?p)
```

```
                (assign (greentime ?i) 0)
        )
)


  ;;********* flowrun_green process with PM of turnrate *****
(:process flowrun_green
:parameters (?p - stage ?r1 ?r2 - link ?i - junction)
:precondition (and (active ?p) (contains ?i ?p)
              (> (occupancy ?r1) 0.0)
              (< (occupancy ?r2) (capacity ?r2))
         )
:effect (and
    ;;----- assign turnrate value ------
  (assign (turnrate  ?p ?r1 ?r2 )
    (+ (beta0 ?p ?r1 ?r2)
     (+ (* (beta1 ?p ?r1 ?r2) (greentime ?i) )
      (+ (* (beta2 ?p ?r1 ?r2) (intertime ?i))
        (+ (* (beta3 ?p ?r1 ?r2) (/(occupancy ?r1) (capacity ?r1)))
           (* (beta4 ?p ?r1 ?r2) (/(occupancy ?r2) (capacity ?r2)))
        )
       )
     )
     )
    )

    (increase (occupancy ?r2) (* #t (turnrate ?p ?r1 ?r2)))
    (decrease (occupancy ?r1) (* #t (turnrate ?p ?r1 ?r2)))
  )
)



  (:process keepinter
    :parameters (?p - stage ?i - junction)
    :precondition (and (inter ?p) (contains ?i ?p)
                       (< (intertime ?i) (interlimit ?p)  )
                  )
     :effect (and (increase (intertime ?i) (* #t 1 ) ) )
  )

  (:event trigger-change
  :parameters (?p ?p1 - stage ?i - junction)
   :precondition (and (inter ?p) (contains ?i ?p)
                      (next ?p ?p1)
                      (>= (intertime ?i) (- (interlimit ?p) 0.1)  )
                )
```

```
125       :effect (and  (not (inter ?p)) (active ?p1)
126                     (assign (intertime ?i) 0)
127             )
128     )
129 )
```

D.2 Problem Definition (for junction $n3969$)

```
1 (define (problem manchester)
2   (:domain urbantraffic)
3   (:objects
4       n3969 - junction
5       n3969_n6612 n6612_n3969 n3972_n3969 n3969_n3972 n3969_n7646
     n3968_n3969  - link
6       s3969_s2 s3969_s0 s3969_s1 - stage
7   )
8   (:init
9       (controllable n3969)
10      (active s3969_s2)
11
12      (next s3969_s0 s3969_s1)
13      (next s3969_s1 s3969_s2)
14      (next s3969_s2 s3969_s0)
15
16      (contains n3969 s3969_s0)
17      (contains n3969 s3969_s1)
18      (contains n3969 s3969_s2)
19
20      (= (greentime n3969) 0)
21      (= (intertime n3969) 0.0)
22
23      (= (interlimit s3969_s0) 1.0)
24      (= (interlimit s3969_s1) 5.0)
25      (= (interlimit s3969_s2) 9.0)
26
27      (= (mingreentime s3969_s0) 18)
28      (= (maxgreentime s3969_s0) 117)
29      (= (defaultgreentime s3969_s0) 28)
30      (= (mingreentime s3969_s1) 26)
31      (= (maxgreentime s3969_s1) 115)
32      (= (defaultgreentime s3969_s1) 35)
33      (= (mingreentime s3969_s2) 10)
34      (= (maxgreentime s3969_s2) 89)
35      (= (defaultgreentime s3969_s2) 44)
36
37      ;;capacity
38      (= (capacity n6612_n3969) 102.55667774)
39      (= (capacity n3969_n6612) 76.1414616636)
40      (= (capacity n3972_n3969) 58.5186831)
41      (= (capacity n3969_n3972) 58.7995689051)
42      (= (capacity n3969_n7646) 8.12771104258)
43      (= (capacity n3968_n3969) 38.116362703)
```

```
44
45      ;;occupancy
46      (= (occupancy n6612_n3969) 83.0)
47      (= (occupancy n3969_n6612) 0);;0
48      (= (occupancy n3972_n3969) 17.0)
49      (= (occupancy n3969_n3972) 50.0)
50      (= (occupancy n3969_n7646) 1.0)
51      (= (occupancy n3968_n3969) 37.0)
52
53      ;;============== turnrate ============================
54      ;;s3969_s0
55      ;;---------- link1 ------------------
56      (= (turnrate s3969_s0 n6612_n3969 n3969_n3972) 0)
57      (= (beta0 s3969_s0 n6612_n3969 n3969_n3972) 0.091)
58      (= (beta1 s3969_s0 n6612_n3969 n3969_n3972) -0.0004)
59      (= (beta2 s3969_s0 n6612_n3969 n3969_n3972) 0)
60      (= (beta3 s3969_s0 n6612_n3969 n3969_n3972) -0.068)
61      (= (beta4 s3969_s0 n6612_n3969 n3969_n3972) -0.063)
62
63      ;;------------ link2 -------------------
64      (= (turnrate s3969_s0 n6612_n3969 n3969_n7646) 0)
65      (= (beta0 s3969_s0 n6612_n3969 n3969_n7646) 1.475)
66      (= (beta1 s3969_s0 n6612_n3969 n3969_n7646) -0.007)
67      (= (beta2 s3969_s0 n6612_n3969 n3969_n7646) 0)
68      (= (beta3 s3969_s0 n6612_n3969 n3969_n7646) -0.629)
69      (= (beta4 s3969_s0 n6612_n3969 n3969_n7646) 0.666)
70
71      ;;----------- link3 ----------------------
72      (= (turnrate s3969_s0 n3972_n3969 n3969_n6612) 0)
73      (= (beta0 s3969_s0 n3972_n3969 n3969_n6612) 0.119)
74      (= (beta1 s3969_s0 n3972_n3969 n3969_n6612) 0)
75      (= (beta2 s3969_s0 n3972_n3969 n3969_n6612) 0)
76      (= (beta3 s3969_s0 n3972_n3969 n3969_n6612) 3.326)
77      (= (beta4 s3969_s0 n3972_n3969 n3969_n6612) 1.349)
78
79      ;;s3969_s1
80      ;;------------- link1 --------------
81      (= (turnrate s3969_s1 n6612_n3969 n3969_n7646) 0)
82      (= (beta0 s3969_s1 n6612_n3969 n3969_n7646) 0.070)
83      (= (beta1 s3969_s1 n6612_n3969 n3969_n7646) 0.000)
84      (= (beta2 s3969_s1 n6612_n3969 n3969_n7646) 0.003)
85      (= (beta3 s3969_s1 n6612_n3969 n3969_n7646) -0.055)
86      (= (beta4 s3969_s1 n6612_n3969 n3969_n7646) -0.072)
87
88      ;;s3969_s2
89      ;;----------- link1 ---------
90      (= (turnrate s3969_s2 n3968_n3969 n3969_n3972) 0)
```

```
      (= (beta0 s3969_s2 n3968_n3969 n3969_n3972) 1.496)
      (= (beta1 s3969_s2 n3968_n3969 n3969_n3972) 0)
      (= (beta2 s3969_s2 n3968_n3969 n3969_n3972) -0.090)
      (= (beta3 s3969_s2 n3968_n3969 n3969_n3972) -1.366)
      (= (beta4 s3969_s2 n3968_n3969 n3969_n3972) -0.756)

      ;;--------- link3 --------------
      (= (turnrate s3969_s2 n3968_n3969 n3969_n7646) 0)
      (= (beta0 s3969_s2 n3968_n3969 n3969_n7646) -0.010)
      (= (beta1 s3969_s2 n3968_n3969 n3969_n7646) 0)
      (= (beta2 s3969_s2 n3968_n3969 n3969_n7646) 0)
      (= (beta3 s3969_s2 n3968_n3969 n3969_n7646) 0)
      (= (beta4 s3969_s2 n3968_n3969 n3969_n7646) 4.499)

      ;;----------- link2 ----------------
      (= (turnrate s3969_s2 n3968_n3969 n3969_n6612) 0)
      (= (beta0 s3969_s2 n3968_n3969 n3969_n6612) 0.246)
      (= (beta1 s3969_s2 n3968_n3969 n3969_n6612) 0)
      (= (beta2 s3969_s2 n3968_n3969 n3969_n6612) -0.0295)
      (= (beta3 s3969_s2 n3968_n3969 n3969_n6612) 0)
      (= (beta4 s3969_s2 n3968_n3969 n3969_n6612) 0)

  )

  (:goal
    (and
      (< (occupancy n3968_n3969) 1)
      (active s3969_s0)
    )
  )
)
```

Appendix E

Python Function (Backward Elimination)

```
#*********************************************************************
# Function to apply stepwise regression with Backward Elimination
# Based on p-value of f-statisitc and t-test
#*********************************************************************

import pandas as pd
import statsmodels.formula.api as smf


def backward_elimination(data, outcome, predictors):

    # == verify all-zero values of outcome variable ====
    # -- if true then stop and return "None" ----
    if all(value == 0 for value in data[outcome]):
        print(outcome, "has all-zero values")
        return "None"

    # === drop predictor with all-zero values ===
    for x in predictors:
        if all(value == 0 for value in data[x]):
            predictors.remove(x)

    # iterate loop till no predictor left in the model,
  # otherwise break loop
    no_of_predictors = len(predictors)
    while no_of_predictors != 0:

        # == formulate model ===
        formula = "{} ~ {} + 1".format(outcome, ' + '.join(predictors
    ))
        model = smf.ols(formula, data).fit()

        # == check p-value of t-test (should be < 0.05) ====
        # --- if all p-values of t-test less than 0.05,
    #   then check p-value of f-statistic ---
```

```
35          if (all(model.pvalues.loc[i] < 0.05 for i in predictors)):
36          #= check p-value of f-statistics (should be < 0.05) =
37              if model.f_pvalue < 0.05:
38                  print("p-value of f statistics is (< 0.05) : ", model
    .f_pvalue)
39                  print("***** Get the final model *****")
40                  break
41              else:
42                  print("EXCEPTION: p-value of f-statistic > 0.05,
    Though All p-values of t-test < 0.05")
43
44          # otherwise, check which predictor has p-value > 0.05
45          else:
46              for x in predictors:
47                  # Eliminates the least significant predictor
48          #i.e. p-value > 0.05
49                  if model.pvalues.loc[x] > 0.05:
50                      predictors.remove(x)
51                      print(x, " is eliminated from the model with p-
    value (>0.05) : ", model.pvalues.loc[x])
52
53                      # --- update the length size of predictor list
54                      no_of_predictors = len(predictors)
55
56      # === if there all predictors are eliminated due to
57      # all-zero values OR p-value > 0.05 ===
58      if len(predictors) == 0:
59          return "None"
60      else:
61          return model
```

Appendix F

An initial Process Specification For Polishing Domain

Goal: the target specification (form and texture)

Process	Properties	Precondition	Effect
Pre-polishing (coarse step)	Uniform and "fast/larger removal" over the part Removes **S**ubsurface **D**amage (SSD) caused by the grinding phase Reduces the surface roughness (enabling interferometry) Pre-polishing was performed using a bonnet tool, with the tool-path programmed for uniform stock removal.	Require bonnet tool Require tool-path programme	- Achieve a specular ('shiny') surface good enough for interferometry - To generate an error-map
Corrective polishing (iterative step)	A finer series of polishing-runs with "lower removal rates" get form accuracy < 1µm RMS, roughness @ nm level form correction is an iterative process using data obtained from the Optical Test Tower via null optics, 4D interferometer and 4Sight software.	Require error-map Require metrology data	- Reducing the form-error in stages using metrology data at each stage - the dwell time of the tool at each position is set according to the removal required to correct the measured local form error - Stop when **form-specification met**
Finishing step (a very fine step)	"Very low removal rates" which don't materially alter the form	Already met/achieved the form-specification	- To clean up the surface - To meet **texture specification**

Fig. .1 An initial process specification for Polishing Domain