AN IMPROVED ROBUST OPTIMIZATION APPROACH FOR SCHEDULING UNDER UNCERTAINTY

A Thesis

by

UTKARSH DINESH SHAH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2017

Major Subject: Chemical Engineering

ABSTRACT

In practice, the uncertainty in processing time data frequently affects the feasibility of optimal solution of the nominal production scheduling problem. Using the unit-specific event-based continuous time model for scheduling, we develop a novel multi-stage robust approach with corrective action to ensure robust feasibility of the worst case solution while reducing the conservatism arising from traditional robust optimization approaches. We quantify the probability of constraint satisfaction by using *a priori* and *a posteriori* probabilistic bounds for known and unknown uncertainty distributions, consequently, improving the objective value for a given risk scenario. Computational experiments on several examples were carried out to measure the effectiveness of the proposed method. For a given constraint satisfaction probability, the proposed method improves the objective value compared to the traditional robust optimization approaches.

DEDICATION

In memory of Prof. Christodoulos A. Floudas.

ACKNOWLEDGMENTS

I would like to thank Prof. Christodoulos A. Floudas under whose guidance I initiated my research project at Texas A&M University. It was his charismatic personality and passion that instill in me the basics of computational thinking.

I would like to thank my thesis advisor, Prof. Efstratios N. Pistikopoulos for his guidance in helping me finish my thesis successfully. I would also like to thank people on my committee, Prof. Costas Kravaris and Dr. Phanourios Tamamis. This work would not have been possible without academic supports from my lab mates, Dr Yannis Guzman, Logan Matthews and Wenjie Xu. I cannot imagine my time at Texas A&M University without the company of Justin Katz, William Tso, Melis Onel, Doga, Barris and Burcu Bekyal. Finally, I would like to acknowledge my friends and family who provided me with constant motivation and emotional support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Prof. Pistikopoulos and Prof. Kravaris of the Department of Chemical Engineering and Dr. Tamamis of the Energy Institute.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was self-supported.

NOMENCLATURE

Indices

i, i'	Tasks
j, j'	Units
n, n', n''	Events
s	State
u	Utilities

Sets

$\mathbf{I}$	tasks
$\mathbf{I_j}$	task that can be performed in unit j
$\mathbf{I_s}$	task that can process state s and either consume or produce it
$\mathbf{I_s^c}$	task that consume state s
$\mathbf{I_s^p}$	task that produce state s
$\mathbf{J}$	units
$\mathbf{J_i}$	units suitable for performing task i
$\mathbf{N}$	event points within the scheduling horizon
$\mathbf{S}$	states
$\mathbf{S^P}$	states that are final product
$\mathbf{S_R}$	states that are raw materials

Parameters

B_i^L	minimum capacity (batch size) of task i
B_i^U	maximum capacity (batch size) of task i
D_s	demand for state s

H	short-term time horizon
M	large positive number in big-M
N	number of event points
P_s	price of state s
ST_s^0	initial amount of state s available
ST_s^{max}	maximum amount of state s
α_i	coefficient of constant term of processing time of task i
β_i	coefficient of variable term of processing time of task i
Δn	maximum number of events over which task i is allowed to continue
ρ_{is}	proportion of state s produced ($\rho_{is} \geq 0$) or consumed ($\rho_{is} \leq 0$) by task i
Binary Variable	
$w_{inn'}$	binary variable for assignment of task i that starts at event n and ends at event n'
$\omega_{ini'n'}$	binary variable for assignment of weight for task i' finished in event n' on task i about to occur in n ($n > n'$)
Positive Variable	
$b_{inn'}$	amount of material undertaking task i that starts at event n and ends at event n' ($n' \geq n$)
ST_{0s}	initial amount of state s that is required from external resources
ST_{sn}	excess amount of state s that needs to be stored at event n
T_{in}^f	time at which task i ends at event n
T_{in}^s	time at which task i starts at event n
MIP	Mixed Integer Program
LP	Linear Program

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

1. INTRODUCTION

Production scheduling is a decision-making process to determine what to produce, when to produce and how much to produce. Traditionally, these decisions are carried out by trained individuals without mathematical optimization using spreadsheets and gantt charts [1]. The increase in production volumes, product portfolios, alternative production recipes and energy cost has raised the complexity of manual scheduling. The later coupled with growing global competition and complexity of manufacturing facilities has made it essential to deploy effective optimization tools for generating most profitable and effortless schedules.

In spite of the literary advances in process scheduling, the gap between academic research and industrial application of these optimization tools remains wide open. One of the primary cause can be attributed to accounting for fundamental reality of uncertainty in processing parameters. The uncertainty in processing parameters can not only lead to delays in schedules but also lead to infeasibility for otherwise optimal solution [2] leading to decrease in operators confidence in the optimal schedule. In order to handle the critical reality of uncertainty in scheduling, we propose a multi-stage robust optimization approach with corrective action to ensure feasibility of the worst case solution while reducing the conservatism arising from traditional robust optimization.

1.1 Statement of Problem

The scheduling problem of chemical processes is defined as follows. Given:

i. production recipes (i.e. the processing times for each task at the suitable units, and the amount of the materials required for the production of each product),

ii. available equipment and the ranges of their capacities,

iii. material storage policy,

iv. production requirement, and

v. time horizon under consideration,

Determine

i. the optimal sequence of tasks taking place in each unit,

ii. the amount of material being processed at each time in each unit,

iii. the processing time of each task in each unit,

so as to optimize a performance criterion, for example, to minimize the makespan or to maximize the overall profit. The most common sources of uncertainty in the aforementioned scheduling problem are:

i. the processing times of tasks,

ii. the market demands for products, and

iii. the prices of products and/or raw materials.

The uncertainty in market prices and product demands manifest on the scale of 24-48 hrs, while the uncertainty in processing time parameters manifest on scale of few minutes to hours. Hence, from a context of short term scheduling with a typical time horizon of 8-16 hrs, processing time uncertainty has the maximum detrimental effect on operations. The uncertainty in processing time can be described using known or unknown, symmetric or asymmetric, continuous or discrete distributions. Note that, with increased knowledge about uncertain parameters distribution, we gain improved probabilistic guarantees on robust solutions feasibility.

1.2 Outline of Thesis

The rest of this book is organized as follows. In Chapter 2, we briefly review the relevant literature in the area of short-term process scheduling and describe in details the model used in this work. Chapter 3 provides a background on theory of robust optimization including the use of probabilistic bounds. Chapter 4 address the issue of scheduling under uncertainty and proposes a improved robust optimization formulation for scheduling under uncertainty.

Finally, in the last chapter, we summarize the contribution of the developments and point out avenues for future research.

2. SHORT-TERM BATCH SCHEDULING FRAMEWORK

Scheduling is a important tool for production facilities, affecting the productivity and profitability of a facility. The problem of scheduling has received considerable attention in recent years from both academic and industrial research communities [1, 3] as it arises in almost any type of industrial production facilities (Pulp and paper, Metals, Oil and Gas, Pharmaceuticals, Food and Beverages, etc.). The objective of scheduling problem is to allocate optimal sequences and sizes of task to suitable units subject to availability of resource such as materials and utilities to meet market demands. Due to increase in complexity and flexibility of the production facilities, scheduling problem sizes vary from simple, single-stage processes to highly complicated, multi-product, multi-purpose processes. More thorough reviews on scheduling are presented by Floudas and Lin [3], Mendez et al. [4], Phanden et al. [5] and Maravelias [6].

2.1 Time Representations

Scheduling models can be broadly classified into two main categories based on time representations, namely, discrete-time and continuous-time models. Early attempts at process scheduling were based on discretization of time horizon into a finite number of time interval [7]. Since, the beginning and ending of tasks are associated with boundaries of these intervals, one needs to select intervals as small as greatest common factor of arbitrary processing times. Selecting such small interval often leads to extremely large problem size, especially for real world problems. Specific solution techniques and reformulation are employed to reduce the problem size and improve computational efficiency [8, 9]. Discrete-time methods are a relatively straightforward method to model timing constraints as they provide exact location of every time t in grid. This simplicity comes in handy when modeling the change in electricity prices and power availability [10, 11, 12]. However, this simplicity comes at the cost of approximation of continuous nature of processing task durations and large number of binary variables corresponding to each discrete time interval.

To alleviate the inherent limitations of discrete-time models, continuous-time representation methods have been developed. Unlike discrete-time models where events are allowed to begin only at certain time intervals, continuous-time models allow event to begin at almost any time point. Continuous representation of time is captured by variable event times, or event points, which can be defined globally or on a unit-specific basis. A large number of inactive intervals of discrete-time models are eliminated by using variable time event points, thus, significantly reducing number of integer variables. However, additional variables and constraints has been defined to accurately model the timing and sequencing constraints, resulting in models with more challenging structures. As a result, a significant amount of research has been dedicated to the development of efficient continuous-time formulations in past two decades.

Continuous-time models can be classified based on type of process representation, namely, sequential processes and general network-represented process. While sequential processes do explicitly consider mass balances as they are order- or batch-oriented, general network-represented processes can handle more general cases of mass splitting and merging balances. Two groups of general network-represented models have been developed, slot based model and event based model. Ordered blocks or slot of unknown, variable lengths represents time horizon in slot based models. While on the other hand, continuous variables are directly defined to represent timings of tasks in event based models. Event based models can be further classified as global event-point and unit-specific event-point model. Global event based models [13] represent time horizon using a set of event (or time) points that are common for all tasks and in all units. In contrast, event points on a unit basis are defined in unit-specific models [14].

The unit-specific event based continuous-time formulation for short-term was originally proposed by Floudas and coworkers [14, 15, 16, 17, 18, 19]. Event points are defined as a sequence of time instances located along time axis of each resource or unit, each representing the beginning of a task or utilization of a resource. Since the location of event points are different for different units, the tasks assigned to same event are allowed to start at different moments in different units. Due to this additional decomposition of event points for different units, for the same scheduling problem, the number of event points required in the unit-specific event based formulation is smaller than the number of events in global event based models.For a scheduling problem, reduced number of binary variables that results from unit-specific event-based formulation, leads to smaller model sizes and efficient solutions for large-scale models.

2.2 Mathematical Model

Due to established advantages of unit-specific event-based models [20, 21], we use them as they lead to smaller number of binary variables and computationally efficient models. The work presented in this thesis is based on the continuous time unit-specific event-based deterministic model proposed by Floudas and co-workers. Note that the formulation presented below may differ from original publication, but in spirit, it is mathematical equivalent and identical with one found in Li and Floudas [22].

Allocation Constraints

Based on the original formulation, a three-index binary allocation variable $w_{inn'}$ is defined, to determine assignment of a task i that starts in event n and ends in event n' ($n \leq n'$).

$$\begin{aligned} &\sum_{i \in \mathbf{I}_j} \sum_{n' \in \mathbf{N}} w_{inn'} \leq 1 \,\forall\, j \in \mathbf{J},\ n \in \mathbf{N} \\ &\sum_{i \in \mathbf{I}_j} \sum_{n \in \mathbf{N}} w_{inn'} \leq 1 \,\forall\, j \in \mathbf{J},\ n' \in \mathbf{N} \end{aligned} \tag{2.1}$$

Constraint 2.1 allows at most one task to begin (end) at an event n (n') in unit j. If a given task can be performed in multiple units, then the task is split into multiple tasks, each suitable in only one unit.

$$\sum_{n' \in \mathbf{N}} w_{inn'} + \sum_{\substack{i' \in \mathbf{I}_j \\ i' \neq i}} \sum_{n' \in \mathbf{N}} w_{i'n'n} \leq 1 \, \forall \, i \in \mathbf{I}_j, \; j \in \mathbf{J}, \; n \in \mathbf{N} \tag{2.2}$$

Constraint 2.2 states that if a task starts in unit j at event n, no other task in unit j can end at event n. Only the task that begins at event n can end at event n.

$$\sum_{n' \in \mathbf{N}} w_{inn'} \leq 1 - \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} \sum_{n'' \in \mathbf{N}} \sum_{i' \in \mathbf{I}_j} w_{i'n'n''} + \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} \sum_{n'' \in \mathbf{N}} \sum_{i' \in \mathbf{I}_j} w_{i'n''n'} \forall \, j \in \mathbf{J}, i \in \mathbf{I}_j, n \in \mathbf{N}, n > 1 \tag{2.3}$$

Constraint 2.3 states that a task i in unit j can start at event n only if unit j is idle i.e. all tasks i' that started before event n have ended before event n.

$$\sum_{n' \in \mathbf{N}} w_{in'n} \leq \sum_{\substack{n' \in \mathbf{N} \\ n' \leq n}} \sum_{n'' \in \mathbf{N}} w_{in'n''} - \sum_{n'' \in \mathbf{N}} \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} w_{in''n'} \forall i \in \mathbf{I}, \; n \in \mathbf{N} \tag{2.4}$$

Constraint 2.4 allows a task to end only if it had started earlier.

Capacity Constraints

$$B_i^{min} w_{inn'} \leq b_{inn'} \leq B_i^{max} w_{inn'} \forall i \in \mathbf{I}, \; n, n' \in \mathbf{N} \tag{2.5}$$

Batch size limitations are enforced by constraint 2.5

Material Balances

$$ST_{sn} = ST_{s(n-1)} + \sum_{i \in \mathbf{I}_s^p} \rho_{is} \sum_{n' \in \mathbf{N}} b_{in'(n-1)} + \sum_{i \in \mathbf{I}_s^c} \rho_{is} \sum_{n' \in \mathbf{N}} b_{inn'} \forall\ s \in \mathbf{S}, n \in \mathbf{N}, n > 1 \quad (2.6)$$

In Constraint 2.6, the inventory of a state s at event n is adjusted by considering the inventory in previous event, amount generated in previous event and amount consumed starting from event n.

$$ST_{sn} = ST_0 s + \sum_{i \in \mathbf{I}_s^c} \rho_{is} \sum_{n' \in \mathbf{N}} b_{inn'} \forall\ s \in \mathbf{S},\ n = 1 \quad (2.7)$$

Constraint 2.7 accounts for initial inventory of the state s.

Duration Constraints

$$T_{in}^f \geq T_{in}^s + \alpha_i w_{inn} + \beta_i b_{inn} \forall\ i \in \mathbf{I},\ n \in \mathbf{N} \quad (2.8)$$

Constraint 2.8 ensures that the finish time of a task is later than the sum of its start time and processing time.

$$T_{in}^f \geq T_{in}^s\ \forall i \in \mathbf{I},\ n \in \mathbf{N} \quad (2.9)$$

Constraint 2.9 states that finish time of a task is later than its start time.

$$T_{in'}^f \geq T_{in}^s + \alpha_i w_{inn'} + \beta_i b_{inn'} - M(1 - w_{inn'}) \forall i \in \mathbf{I},\ n, n' \in \mathbf{N},\ n < n' \quad (2.10)$$

Constraint 2.10 adjusts the finish time of a task if the task is processed across multiple event points.

Sequencing Constraints

$$T^s_{i(n+1)} \geq T^f_{in} \forall i \in \mathbf{I},\ n \in \mathbf{N},\ n < N \tag{2.11}$$

$$\begin{aligned} T^s_{i(n+1)} \leq T^f_{in} + M\Big[1 - (\sum_{\substack{n' \in \mathbf{N} \\ n' \leq n}} \sum_{n'' \in \mathbf{N}} w_{in'n''} - \sum_{n'' \in \mathbf{N}} \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} w_{in''n'})\Big] \\ + M \sum_{n' \in \mathbf{N}} w_{in'n} \forall i \in \mathbf{I},\ n \in \mathbf{N},\ n < N \end{aligned} \tag{2.12}$$

Constraint 2.11 states that start time of task in an event is later than finish time of task in previous event. If a task spans across multiple event points, a zero-wait condition is applied when the task ends in event later than n.

$$\begin{aligned} T^s_{i(n+1)} \geq T^f_{i'n} - M\Big[1 - (\sum_{\substack{n' \in \mathbf{N} \\ n' \leq n}} \sum_{n'' \in \mathbf{N}} w_{i'n'n''} - \\ \sum_{n'' \in \mathbf{N}} \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} w_{i'n''n'})\Big] \forall i, i' \in \mathbf{I}_j,\ i \neq i',\ j \in \mathbf{J},\ n \in \mathbf{N},\ n < N \end{aligned} \tag{2.13}$$

Constraint 2.13 states that start time of a task in an unit has to be later than finish time of task that occurs in the same unit in previous event.

$$\begin{aligned} T^s_{i(n+1)} \geq T^f_{i'n} - M(1 - \sum_{n' \in \mathbf{N}} w_{i'n'n}) \\ \forall s \in \mathbf{S},\ i \in \mathbf{I}^c_s,\ i' \in \mathbf{I}^p_s,\ i \in \mathbf{I}_j,\ i' \in \mathbf{I}'_j,\ i \neq i',\ j \neq j',\ n \in \mathbf{N},\ n < N \end{aligned} \tag{2.14}$$

For tasks that consume and produce the same state in different units, Constraint 2.14 enforces the start time of consumption task to be later than finish time of production task.

Tightening Constraint

$$\sum_{i\in\mathbf{I}_j}\sum_{n\in\mathbf{N}}\sum_{n'\in\mathbf{N}}(\alpha_i w_{inn'} + \beta_i b_{inn'}) \leq H \forall j \in \mathbf{J} \tag{2.15}$$

Sum of processing times for all the tasks that take place in a unit should be less than the scheduling horizon.

Bounds on Variable

$$T^s_{in} \leq H,\ T^f_{in} \leq H \forall i \in \mathbf{I},\ n \in \mathbf{N} \tag{2.16}$$

$$w_{inn'} = 0, b_{inn'} = 0 \forall i \in \mathbf{I},\ n, n' \in \mathbf{N}, n' < n \tag{2.17}$$

Objective Function

Several different objective functions can be employed for sort-term scheduling problems. Two most common types are reviewed below.

Maximization of Profit

$$\max \text{ Profit } = \sum_{s\in\mathbf{S}} P_s \sum_{n=N}(ST_{sn} + \sum_{i\in\mathbf{I}^p_s}\rho_{is}\sum_{n'\in\mathbf{N}} b_{in'n}) \tag{2.18}$$

Alternatively, one can optimize the time taken by a schedule (a.k.a. makespan) to meet given demands of products. *Minimization of Makespan*

$$\min \ \mathbf{MS}$$

$$\sum_{n=N}(ST_{sn} + \sum_{i \in \mathbf{I}_s^p} \rho_{is} \sum_{n' \in \mathbf{N}} b_{in'n}) \geq D_s \forall s \in \mathbf{S} \tag{2.19}$$

$$T_{iN}^f \leq MS \forall i \in \mathbf{I}$$

Note that, if using minimization of makespan as objective, replace time horizon H with makespan MS in the model. Also, additional constraints like utility balance, intermediate due dates, storage considerations, etc. can be added to the model.

2.3 Motivating Example

The study was performed on the standard benchmark example originally introduced by Kondili et al [7]. In figure 2.1, states $s1$, $s2$ and $s3$ represent the raw materials. While $s2$ and $s3$ can be directly used in reaction 2 $i2, i3$, $s1$ needs to be under go heating $i1$ in the heater $j1$. States $s4$, $s5$, $s6$ and $s7$ are the intermediate products of the reaction scheme that produces final products as states $s8$ and $s9$. All the reactions can take place in two reactor units $j2$ and $j3$, hence mathematically reaction in one reactor is treated differently than that in other reactor. Eg., reaction 1 is broken into two tasks $i2$ and $i3$ that take place in reactor 1 $j2$ and reactor 2 $j3$ respectively. The data for parameters can be found in appendix. Nominal schedule for profit maximization problem with 8 hrs as its time horizon is showed below. Note that for above problem, we use 4 events points as determined by Li and Floudas [22].

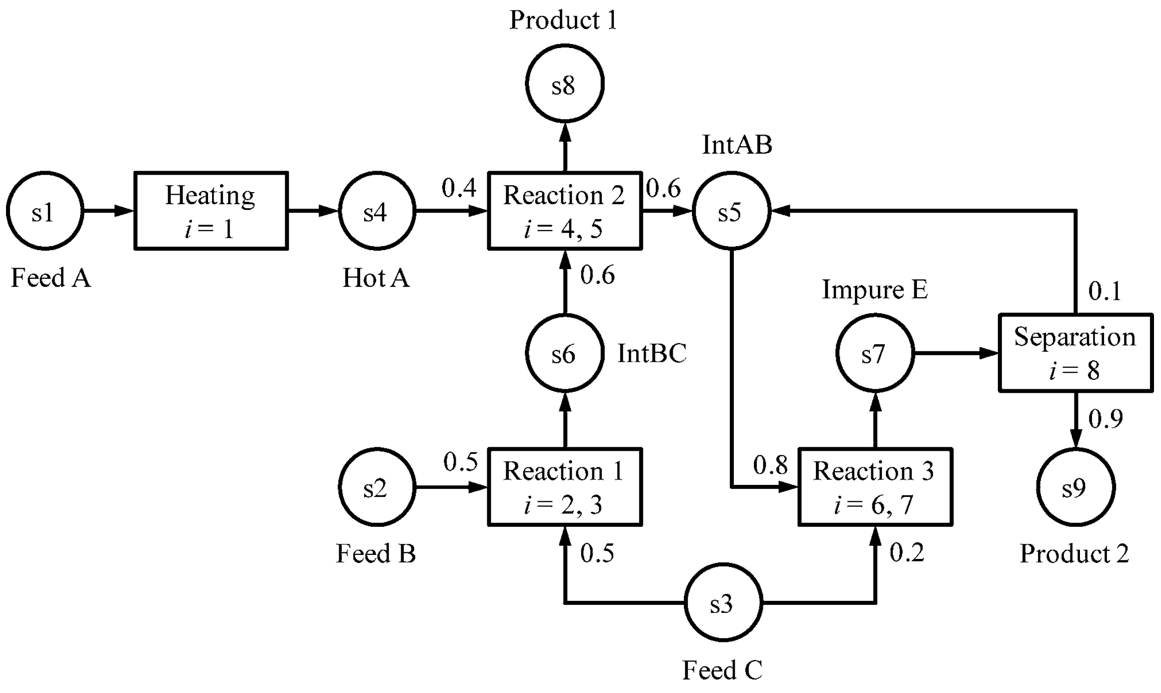


Figure 2.1: State-Task Network (STN) for motivating example. Reprinted with permission from [22]. Copyright ©2010 American Chemical Society.

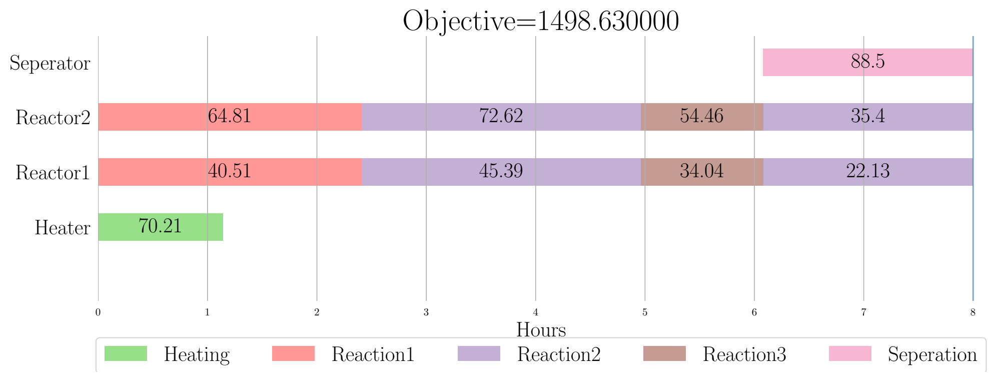


Figure 2.2: Nominal Schedule for motivating example (H =8 hrs)

3. ROBUST OPTIMIZATION

The efficacy of a mathematical model relies on the accuracy of information fed to the model in form of parameters. Typically, these parameter values exhibit uncertainty in data due to limited information or measurement error. The solution of such a mathematical model with uncertain parameter values would vary greatly based on which values are realized by these uncertain parameter. One of the methods to immunize the mathematical models against uncertainty is Robust Optimization. These is done by ensuring feasibility of the constraints for any possible realization of parameter points in uncertainty sets. These uncertainty sets are defined by deterministic data of parameter realization. The problem that corresponds to feasible solution for uncertainty set is also called robust counterpart.

Earliest work in robust optimization was published by Soyster [23], formulating the so-called "worst-case" robust counterpart, where the problem is immunized against all possible perturbations of data. The Soyster solution to a problem would ensure that no possible realization of uncertain parameter would render the solution infeasible. However, if the uncertain parameters take unbounded distributions, it is impossible to generate a Soyster solution for which probability of constraint violation is zero for any realization of uncertain parameter. Thus, it is desirable to generate solutions and quantify the trade-off between robustness and performance. The interval + ellipsoidal uncertainty set and a method to provide upper bound on constraint violation probability for given set was proposed by Ben-Tal and Nemirovski [24, 25]. A linear robust counterpart method was introduced by Bertsimas and Sim [26], as they proposed and characterized interval + polyhedral uncertainty set. The robust optimization framework was extended to mixed-integer linear programs(MILPs) by Floudas and coworkers [2, **?**, 27, 28] with parameters subject to uncertainty with known or unknown distributions, including unbounded probability distributions.

Traditionally, the quality of a robust solution relies on strength of *a priori* methods that define uncertainty sets which satisfy a upper bound on probability of constraint violation or a lower bound on probability of constraint satisfaction. A tighter probabilistic bound would allow for improvement in objective value for a given risk, thus leading to a less conservative solution. *A priori* methods were characterized by Ben-Tal and Nemirovski [25] and Bertsimas and Sim [29] for interval + ellipsoidal and interval + polyhedral uncertainty sets, respectively. These bounds were extended to apply to other variety of uncertainty sets and various distributions of uncertainty by Floudas and coworkers [30, 31, 32, 33, 34]. Alternatively, one can characterize the probability of constraint violation of a solution, namely, *a posteriori* bounds on constraint violation, which are stronger than *a priori* bound of equivalent structure. A nonconvex optimization problem is generated when *a posteriori* bounds are incorporated in robust counterparts instead of using uncertainty sets. To avoid solving a nonconvex optimization problem and to take advantage of tighter *a posteriori* bounds, Li and Floudas [31] proposed an iterative method to obtain improved solution for a particular risk tolerance. Compared to both worst-case solution and traditional one-pass robust optimization framework, a dramatic improvement in solutions can be obtained by utilizing tighter *a priori* and *a posteriori* bounds along with the iterative framework.

3.1 Uncertain Inequality Constraints

Methods for generating robust counterparts of deterministic models have been developed to apply on cases, where uncertain parameters are involved in linear or mixed-integer linear inequality constraints. In practice, these forms can often be achieved with simple substitutions or reformulations. Given arbitrary function $f_i(x, y)$ participating in constraint

i, where x and y are continuous and integer variables respectively:

$$f_i(x,y) + \sum_k a_{ik}x_k + \sum_l b_{il}y_l + \sum_m p_{im} \leq 0 \tag{3.1}$$

Assume that the exact values of some or all of parameters $a_{ik},\ \forall k$, $b_{il},\ \forall l$ and $p_{im},\ \forall m$ are uncertain. An equivalent reformulation of constraint 3.1 is:

$$\begin{gathered} f_i(x,y) + t_i \leq 0 \\ -t_i + \sum_k a_{ik}x_k + \sum_l b_{il}y_l + \sum_m p_{im} \leq 0 \end{gathered} \tag{3.2}$$

A similar reformulation can be applied to an objective function with uncertain parameters. The general form of LP or MIP under uncertainty is as follows:

$$\begin{aligned} \max_{x,y} \quad & \sum_k \tilde{c}_k x_k + \sum_l \tilde{d}_k y_k & \\ \text{s.t.} \quad & \sum_k \tilde{a}_{ik}x_k + \sum_l \tilde{c}_{il}y_l \leq \tilde{p}_i & \forall i \\ & y_l \in \{0,\ 1\} & \forall l \end{aligned} \tag{3.3}$$

Any parameter denoted with tilde is a parameter subject to uncertainty. The solutions and objective value of Model 3.3 changes for different realization of uncertain parameters.

Similar to $3.1 \rightarrow\ 3.2$ reformulation, we can equivalently express model 3.3 as follows:

$$\begin{aligned}
\max_{x,y,z} \quad & z \\
\text{s.t.} \quad & z - \sum_k \tilde{c}_k x_k - \sum_l \tilde{d}_k y_k \leq 0 \\
& \tilde{p}_i x_0 + \sum_k \tilde{a}_{ik} x_k + \sum_l \tilde{c}_{il} y_l \leq 0 \qquad \forall i \\
& x_0 = -1 \\
& y_l \in \{0,\ 1\} \qquad \forall l
\end{aligned} \tag{3.4}$$

Generically, the uncertain inequality constraint i can thus be represented as:

$$\sum_{j \notin \mathbf{J}_i} a_{ij} x_j + \sum_{j \in \mathbf{J}_i} \tilde{a}_{ij} x_j \leq b_i \tag{3.5}$$

where, $\tilde{a}_{ij}$ represents uncertain parameters whose indices j are in set $\mathbf{J}_i$. x_j can be continuous, integer or fixed variable to accommodate right hand side parameter uncertainty.

3.2 Uncertainty Sets

Selection of the uncertainty sets are central to formulating the robust counterpart. Uncertainty set, U_i is a deterministic set of multiple possible parameter realizations that is going to be imposed on the constraint i. Similar to behavior of guaranteed constraint feasibility when parameters realize their fixed values, constraint feasibility can be guaranteed if the *true* realization of parameter values is contained within the uncertainty set. An uncertain parameter a_{ij} can be rewritten as function of random variable ξ_{ij}:

$$\tilde{a}_{ij} = \bar{a}_{ij} + \xi_{ij} \hat{a}_{ij} \tag{3.6}$$

where, $\bar{a}_{ij}$ is the constant nominal or expected value of $\tilde{a}_{ij}$ and $\hat{a}_{ij}$ is a positive constant. Random variable ξ_{ij} captures the random realizations of $\tilde{a}_{ij}$. For a bounded uncertainty, value of $\hat{a}_{ij}$ should be chosen such that $\tilde{a}_{ij} \in [\bar{a}_{ij} - \hat{a}_{ij},\ \bar{a}_{ij} + \hat{a}_{ij}]$, in other words, $\xi_{ij} \in [-1,\ 1]$. The uncertainty set U_i can be defined as set of realizations of ξ_i that meet some criteria, where ξ_i is a vector of all random perturbations of parameters $j \in J_i$.

Various kinds of norm-based uncertainty sets have been defined in the literature. These norm-based uncertainty sets are conic convex and will help develop robust counterparts as seen further in this chapter.

A box uncertainty set can be defined using a ∞-norm distance from $\xi_i = \mathbf{0}$.

$$U_i^{\infty} = \left\{ \xi_i : \|\xi_{ij}\|_{\infty} = \max_{j \in J_i} |\xi_{ij}| \leq \Psi_i \right\} \tag{3.7}$$

where the size of U_i^{∞} is controlled by parameter Ψ_i. Geometrically, U_i^{∞} represents a hypercube. If the uncertainty is bounded, selecting $\Psi_i = 1$ will include all possible realization of uncertainty, thus representing the worst-case uncertainty set. Typically, when $\Psi_i = 1$, U_i^{∞} is also referred as interval uncertainty set.

A polyhedral uncertainty set can be defined using a 1-norm distance from $\xi_i = \mathbf{0}$.

$$U_i^{1} = \left\{ \xi_i : \|\xi_{ij}\|_{1} = \sum_{j \in J_i} |\xi_{ij}| \leq \Gamma_i \right\} \tag{3.8}$$

where the size of U_i^1 is controlled by parameter Γ_i. Geometrically, U_i^1 represents a polyhedron. For bounded uncertainty, selecting $\Gamma_i = |J_i|$ will include all possible realizations of uncertainty as well as some spurious ones (when $|J_i| > 1$) in U_i^1.

A ellipsoidal uncertainty set can be defined using a 2-norm distance from $\xi_i = \mathbf{0}$.

$$U_i^{2} = \left\{ \xi_i : \|\xi_{ij}\|_{2} = \sum_{j \in J_i} |\sqrt{\xi_{ij}^2}| \leq \Omega_i \right\} \tag{3.9}$$

where the size of U_i^2 is controlled by parameter Ω_i. Geometrically, U_i^2 represents a ellipsoid. For bounded uncertainty, selecting $\Omega_i = \sqrt{|J_i|}$ will include all possible realizations of uncertainty as well as some spurious ones (when $|J_i|> 1$) in U_i^2.

As observed, for bounded uncertainty, utilizing 1-norm or 2-norm based uncertainty sets, can lead to inclusion of realizations of uncertainty which have zero probability of occurrence due to the uncertainty set geometries. In order to alleviate this, interval uncertainty set is intersected with other norm-based criteria. Intersecting 1-norm and interval set leads to a interval+polyhedral set:

$$U_i^{1\cap\infty} = \left\{ \xi_i : \|\xi_{ij}\|_1 = \sum_{j\in J_i} |\xi_{ij}| \leq \Gamma_i, \|\xi_i\|_\infty \leq 1 \right\} \tag{3.10}$$

where, the size of set is controlled by Γ_i. For bounded uncertainty set, setting $\Gamma_i = |J_i|$ will ensure that all realization of uncertainty are included in $U_i^{1\cap\infty}$. Similarly, intersecting 2-norm and interval set leads to a interval+ellipsoidal set:

$$U_i^{2\cap\infty} = \left\{ \xi_i : \|\xi_{ij}\|_2 = \sum_{j\in J_i} |\sqrt{\xi_{ij}^2}| \leq \Omega_i, \|\xi_i\|_\infty \leq 1 \right\} \tag{3.11}$$

where, the size of set is controlled by Ω_i. For bounded uncertainty set, setting $\Omega_i = \sqrt{|J_i|}$ will ensure that all realization of uncertainty are included in $U_i^{2\cap\infty}$.

In order, to utilize a given uncertainty set U_i onto constraint i, we reformulate constraint 3.5 using 3.6,

$$\sum_j \bar{a}_{ij} x_j + \sum_{j\in\mathbf{J}_i} \xi_{ij} \hat{a}_{ij} x_j \leq b_i \tag{3.12}$$

and now ensure maximal feasibility of the constraint for all realizations of set U_i,

$$\sum_j \bar{a}_{ij} x_j + \max_{\xi_i \in U_i} \left\{ \sum_{j\in\mathbf{J}_i} \xi_{ij} \hat{a}_{ij} x_j \right\} \leq b_i \tag{3.13}$$

The inner maximization problem is conic convex and exhibits strong duality. Utilizing duality theory, inner maximization problem can be replace by a deterministically equivalent minimization problem, thus generating a robust counterpart of constraint 3.5. The robust counterpart of constraint 3.5 subject to a box uncertainty set is:

$$\sum_j \bar{a}_{ij}x_j + \Psi_i \sum_{j \in J_i} \hat{a}_{ij}|x_j| \leq b_i \tag{3.14}$$

The robust counterpart of constraint 3.5 subject to a polyhedral uncertainty set is:

$$\begin{aligned} \sum_j \bar{a}_{ij}x_j + \Gamma_i z_i \leq b_i \\ z_i \geq \hat{a}_{ij}|x_j| \quad \forall j \in J_i \end{aligned} \tag{3.15}$$

The robust counterpart of constraint 3.5 subject to a ellipsoidal uncertainty set is:

$$\sum_j \bar{a}_{ij}x_j + \Omega_i \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 x_j^2} \leq b_i \tag{3.16}$$

The robust counterpart of constraint 3.5 subject to a interval + polyhedral uncertainty set is:

$$\begin{aligned} \sum_j \bar{a}_{ij}x_j + \sum_{j \in J_i} p_{ij} + \Gamma_i z_i \leq b_i \\ z_i + p_{ij} \geq \hat{a}_{ij}|x_j| \quad \forall j \in J_i \\ p_{ij} \geq 0 \quad \forall j \in J_i \\ z_i \geq 0 \end{aligned} \tag{3.17}$$

The robust counterpart of constraint 3.5 subject to a interval + ellipsoidal uncertainty set is:

$$\sum_j \bar{a}_{ij}x_j + \sum_{j \in J_i} \hat{a}_{ij}|x_j - z_{ij}| + \Omega_i \sqrt{\sum_{j \in J_i} \hat{a}_{ij}^2 z_{ij}^2} \leq b_i \tag{3.18}$$

3.3 Probabilistic Robust Optimization

The size of an uncertainty set can be controlled using parameters $\Psi_i,\ \Gamma_i,$ or Ω_i, which can be generically referred to as Δ_i. For bounded uncertainty case, it is possible to include all realizations of ξ_i in U_i by setting Δ_i value to its maximum values ($\Psi_i = 1,\ \Gamma_i = |J_i|,$ or $\Omega_i = \sqrt{|J_i|}$), thus rendering the probability of constraint i's violation to zero for any realization of ξ_i.

$$\Pr\left\{\sum_j \bar{a}_{ij}x_j + \sum_{j\in J_i}\xi_{ij}\hat{a}_{ij}x_j > b_i\ :\ \xi_{ij}\in[-1,\ 1],\ \Delta_i = \Delta_i^{(max)}\right\} = 0 \qquad (3.19)$$

Alternatively for unbounded uncertainty case, to include every possible realization, the value of parameter Δ_i must approach infinity. Also, for either bounded or unbounded case, imposing every possible realization leads to extremely conservative solutions. A reduction in Δ_i value leads to less conservative solution at an expense of probability of constraint violation being non-zero.

$$\Pr\left\{\sum_j \bar{a}_{ij}x_j + \sum_{j\in J_i}\xi_{ij}\hat{a}_{ij}x_j > b_i\ :\ \xi_{ij}\in[-1,\ 1],\ \Delta_i < \Delta_i^{(max)}\right\} > 0 \qquad (3.20)$$

Say that we have acceptable risk appetite of constraint i's violation ϵ_i^{prio}. We can set Δ_i *a priori* such that regardless of optimal solution x^*, we can guarantee that probability of constraint violation is utmost ϵ_i^{prio}. *A priori* probabilistic bounds are the probabilistic expression $B(\Delta_i)$ that relate probability of constraint violation to Δ_i. As this bounds must work regardless of the solution, they provide upper bounds on probability of constraint violation:

$$\Pr\left\{\sum_j \bar{a}_{ij}x_j + \sum_{j\in J_i}\xi_{ij}\hat{a}_{ij}x_j > b_i\right\} \le B(\Delta_i) = \epsilon_i^{prio} \qquad (3.21)$$

Solutions become less conservative for smaller values of Δ_i, hence minimum value of Δ_i should be selected such that $B(\Delta_i) \leq \epsilon_i^{prio}$. Formulating an optimization problem:

$$\begin{aligned} \min_{\Delta_i} \quad & \Delta_i \\ \text{s.t.} \quad & B(\Delta_i) \leq \epsilon_i^{prio} \\ & \Delta_i \geq 0 \end{aligned} \tag{3.22}$$

In order to obtain tightest possible solution of model 3.22, one needs tight *a priori* bound $B(\Delta_i)$. A traditional robust optimization approach can be derived utilizing *a priori* bounds. Alternatively, one can quantify the probability of constraint violation *a posteriori* for a

Algorithm 3.1 Traditional Robust Optimization

```
Require: Provide Deterministic model $M$, select uncertainty set type $U^?$ and A priori
    probability of constraint violation $\epsilon_i^{prio} \forall\, i \in I$
  procedure APRIORIROBUSTOPTIM($M,\ U^?,\ \epsilon^{prio}$)
      $M_{RC} \leftarrow M$
      for all $i \in I$ do                          ▷ For each constraint with uncertain parameters
          $M_{RC} \leftarrow$robustCounterpartGen($M_{RC},\ U_i^?$)      ▷ Generates Robust Counterpart
          $\Delta_i \leftarrow$ aPrioriBound($\epsilon_i^{prio}$)                      ▷ Solution of model 3.22
      end for
      $x^* \leftarrow$ optimizationSolver($M_{RC},\ \Delta$)
      return $x^*, M_{RC},\ \Delta$      ▷ Retrieve Robust Optimal Solution and robust counterpart
  model
  end procedure
```

optimal solution x^*. Similar to *a priori* bounds, one can define *a posteriori* bounds as probabilistic expression $B(x^*)$ which relate probability of constraint violation to $x*$:

$$\Pr\left\{ \sum_j a_{ij} x_j^* + \sum_{j \in J_i} \xi_{ij} \hat{a}_{ij} x_j^* > b_i \right\} \leq B(x^*) = \epsilon_i^{post} \tag{3.23}$$

For a given x^*, a tighter *a posteriori* bound $B(x^*)$ will assign lower probability of constraint violation ϵ_i^{post}. Utilizing *a posteriori* bounds, Li and Floudas [31] proposed alternative method to calculate robust solution. They proposed a following approximate model:

$$\begin{aligned}
\min_{x,\ \Delta_i} \quad & cx \\
\text{s.t.} \quad & -\Delta_i(b_i - \sum_j \bar{a}_{ij}x_j) + \sum_{j\in J_i} \ln E[e^{\Delta_i \xi_{ij}\hat{a}_{ij}x_j}] \le \ln \epsilon_i^{post} && \forall\, i \\
& \Delta_i \ge 0 && \forall\, i
\end{aligned} \tag{3.24}$$

As observed, model 3.24 is non-convex optimization problem and requires deterministic global optimization approach to obtain the optimal solution. The increased objective value from using *a posteriori* bounds comes at an expense of computational power. To mitigate this increase in computational expense, they proposed an alternative method for utilizing *a posteriori* bounds: Needless to mention, tight *a priori* and *a posteriori* bounds help reduce the number of iterations for alg. 3.2 and improve the objective value of the solution. Recently, Floudas and coworkers [32, 33, 34] proposed novel *a priori* and *a posteriori* bounds and characterized them to be the strongest bounds proposed in literature to date. In this work, the bounds are generated by PROTO [35], which relies on these recently proposed novel bounds. Since the bounds generated by PROTO are for independent random parameters, we assume the parameters follow independent random distributions.

Algorithm 3.2 Iterative Approach for Robust Optimization

```
procedure ITERATIVEROBUSTOPTIM($M,\ U^?,\ \epsilon$)
    Set tolerance parameter $\delta$ (e.g. 0.01)
    $x*,\ M_{RC},\ \Delta^{satisfy} \leftarrow$ AprioriRobustOptim($M,\ U^?,\ \epsilon$)        ▷ From alg. 3.1
    $\Delta = \Delta^{satisfy}$
    for all $i \in I$ do                    ▷ For each constraint with uncertain parameters
        $\epsilon_i^{post} \leftarrow$posterioriProb($M,\ x^*$)                    ▷ Using equation 3.23
        while $|\epsilon_i^{post} - \epsilon_i| > \delta$ do
            if $\epsilon_i^{post} \leq \epsilon_i$ then
                $\Delta_i^{satisfy} = \Delta_i$
            else
                $\Delta_i^{violate} = \Delta_i$
            end if
            $\Delta_i \leftarrow 0.5(\Delta_i^{violate} + \Delta_i^{satisfy})$
            $x^* \leftarrow$ optimizationSolver($M_{RC},\ \Delta_i$)
            $\epsilon_i^{post} \leftarrow$posterioriProb($M,\ x^*$)                ▷ Using equation 3.23
        end while
    end for
    return $x^*$        ▷ Retrieve Robust Optimal Solution with *a posteriori* probability of
constraint voilation $\epsilon$
end procedure
```

4. SCHEDULING UNDER UNCERTAINTY

Scheduling problems are inherently plagued with fundamental reality of uncertainty in processing time parameters, market prices, resources and unit availability, or product demands. Most of the literary advances in area of process scheduling has ignored this fundamental reality and proposed nominal-case schedules. Upon realization of uncertain parameters during operation, these nominal schedules often cause delay or infeasibility of schedules, leading to confusion on the operation floor.

Typically, there are two approaches for scheduling under uncertainty: reactive approach and preventive approach [36]. In reactive scheduling, nominal schedules are generated and updated upon the realization of uncertainty. Generation of new schedules is based on feedback of realized states to the scheduler. These approaches tend to be computationally expensive, since uncertainty can occur frequently and optimization problems have to be solved repetitively. Since reactive scheduling relies on rescheduling of nominal problem upon realization of uncertainty, feasibility of solution cannot be guaranteed. In order to alleviate computational issues and improve feasibility of the solution, rescheduling algorithms often employ heuristics or decomposition approach.

Earliest approaches in reactive scheduling were based on decision tree analysis [37]. The decision trees were generated by introducing artificial errors to the execution and then the heuristic chose a reschedule decision such that impact on original schedule is minimal. To avoid future infeasibility, look-ahead procedures were introduced by Rodrigues et al [38]. Mendez and Cerda proposed general MILP reactive scheduling approach for multi-purpose batch plants with limited changes in batch sequencing and units for smooth rescheduling. Penalty incurred by rescheduling actions were incorporated in objective value by Kopanos et al [39]. Similar methods for reactive scheduling were proposed in the literature [40, 41]. Reactive scheduling models often employ novel techniques to enhance feasibility, minimize disruption and reduce computational efforts due to rescheduling.

In contrast, proactive scheduling approaches generate schedules prior to realization of parameters by incorporating a deterministic model of uncertainty in the optimization problem. Proactive scheduling ensures that the solution remains feasible for all possible realizations of uncertainty at the cost of detriment in objective value. One of the prominent approaches for proactive or preventive scheduling is robust optimization. Floudas and coworkers [2, 42] extended robust optimization framework to MILP problem and developed robust scheduling approach. As described in previous chapter, robust optimization does not require explicit knowledge of probabilistic models for uncertainty.

Robust scheduling usually suffers from conservatism and large deterioration in objective value. To overcome this problem, adjustable robust optimization (ARO) framework was proposed by Ben-Tal et. al. [43] Since only a subset of decision have to take place "here-and-now", many decisions can be delayed until later point. This holds true especially for scheduling problem as, only the batch-size and sequence of task needs to decided here-and-now, while the time to start a task can be decided later on realization of uncertainty. Instead of obtaining a single, static optimal solution, ARO framework aims at obtaining an optimal policy that is parameterized in realizations of uncertainty. Effectively, with respect to process scheduling literature, ARO approach can be designated as a cross between reactive and proactive scheduling. Often, these decision based on parameter realizations are assigned using heuristic. Shi and You [44] applied ARO framework to batch scheduling by formulating a 2-stage problem from deterministic MILP model. This model requires computationally expensive techniques to obtain solution and is limited to cases where all uncertain information is revealed before any second stage decision is taken. A multi-stage ARO approach for scheduling problem was proposed by Lappas and Gounaris [45]. They generated a robust counterpart of continuous-time global event point model for scheduling by incorporating a decision-dependent uncertainty set. In interest of numerical tractability, they used a heuristic affine relationship for event time decision rules:

$$T_n \leftarrow [T_n]_0 + \sum_{i \in \mathbf{I}} \sum_{n' < n} [T_n]_{in'} \alpha_{in'} \tag{4.1}$$

where, $\alpha_{in'}$ is an uncertain parameter. Through case studies, they demonstrated an improvement in objective value from ARO method when compared to traditional robust optimization method for an arbitrary size of uncertainty set. The disadvantages of ARO are use of heuristic decision rules, increase in computational size of model and absence of any probabilistic bounds on the solution.

More thorough reviews on scheduling under uncertainty are presented by Li and Ierapetritou [36], Verderame et al. [46] and, Dias and Ierapetritou [47].

4.1 Traditional Robust Scheduling Approach

As demonstrated by ARO approaches, a combination of proactive and reactive solution strategies would lead to improved objective values while ensuring feasibility of schedule. As describe in state of problem, for short-term scheduling problem we consider uncertainty in processing time parameters. Also, for demonstration purposes we consider uncertainty to be uniform and bound $\tilde{\alpha}_i = [0.7\bar{\alpha}_i, 1.3\bar{\alpha}_i]$. Note that, one can handle uncertainty in raw material or resource availability, market prices or demand as well, but for demonstration purposes we consider processing time parameters as the only uncertainty. We assume that uncertain parameters are randomly independent.

Now for deterministic unit-specific event-point model described in chapter 3, only constraints 2.8 and 2.10 contain the uncertain parameter α_i[1]:

$$T^f_{in} \geq T^s_{in} + \tilde{\alpha}_i w_{inn} + \beta_i b_{inn} \forall\, i \in \mathbf{I},\ n \in \mathbf{N}$$

$$T^f_{in'} \geq T^s_{in} + \tilde{\alpha}_i w_{inn'} + \beta_i b_{inn'} - M(1 - w_{inn'}) \forall i \in \mathbf{I},\ n, n' \in \mathbf{N},\ n < n'$$

Substituting $\tilde{\alpha}_i = \bar{\alpha}_i + \xi_i \hat{\alpha}_i$ in constraint 2.8 and 2.10 and maximizing the effect of uncertainty for a uncertainty set.

$$T^f_{in} \geq T^s_{in} + \bar{\alpha}_i w_{inn} + \beta_i b_{inn} + \max_{\xi_{in} \in U_{in}} \{\hat{\alpha}_i \xi_{in} w_{inn}\} \forall\, i \in \mathbf{I},\ n \in \mathbf{N} \tag{4.2}$$

$$T^f_{in'} \geq T^s_{in} + \bar{\alpha}_i w_{inn'} + \beta_i b_{inn'} - M(1 - w_{inn'}) + \max_{\xi_{in} \in U_{in}} \{\hat{\alpha}_i \xi_{in} w_{inn'}\} \forall i \in \mathbf{I},\ n, n' \in \mathbf{N},\ n < n' \tag{4.3}$$

Where, $\bar{\alpha}_i$ is the expected or nominal value of parameter $\tilde{\alpha}_i$ and $\hat{\alpha}$ is a positive constant. Since, cardinality of uncertainty set U_{in} is 1, choosing either box, polyhedral or ellipsoidal sets would make no difference. The robust counterpart can be written as:

$$T^f_{in} \geq T^s_{in} + \bar{\alpha}_i w_{inn} + \beta_i b_{inn} + \Delta_{in} \{\hat{\alpha}_i w_{inn}\} \forall\, i \in \mathbf{I},\ n \in \mathbf{N} \tag{4.4}$$

$$T^f_{in'} \geq T^s_{in} + \bar{\alpha}_i w_{inn'} + \beta_i b_{inn'} - M(1 - w_{inn'}) + \Delta_{in} \{\hat{\alpha}_i w_{inn'}\} \forall i \in \mathbf{I},\ n, n' \in \mathbf{N},\ n < n' \tag{4.5}$$

For bounded uncertainty case, setting $\Delta_{in} = 1$ would lead to worst-case (a.k.a. Soyster) solution. Alternatively, nominal scheduled can be achieved by setting $\Delta_{in} = 0$. Any intermediate value of Δ_{in} would lead to a solution with non-zero probability of constraint violation.

[1]Note that tightening constraint 2.15 contains uncertain parameter α_i, but for demonstration purposes we drop that constraint from the model. Since the constraint was redundant the model feasibility and solution is unaffected.

Now replacing, constraints 2.8 and 2.10 with their robust counterparts 4.4 and 4.5 in the model from chapter 2 and solving the optimization problem for motivating example, we obtain: where, the transparent shade is the extra time allotted to a task to account for

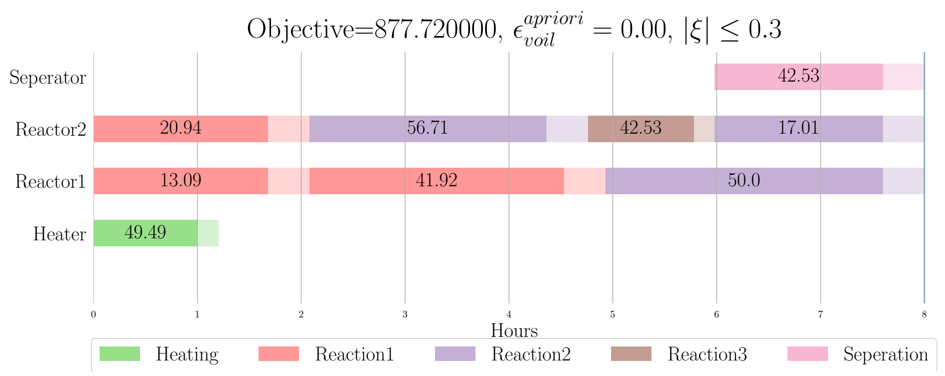


Figure 4.1: Soyster Solution for bounded uncertainty case

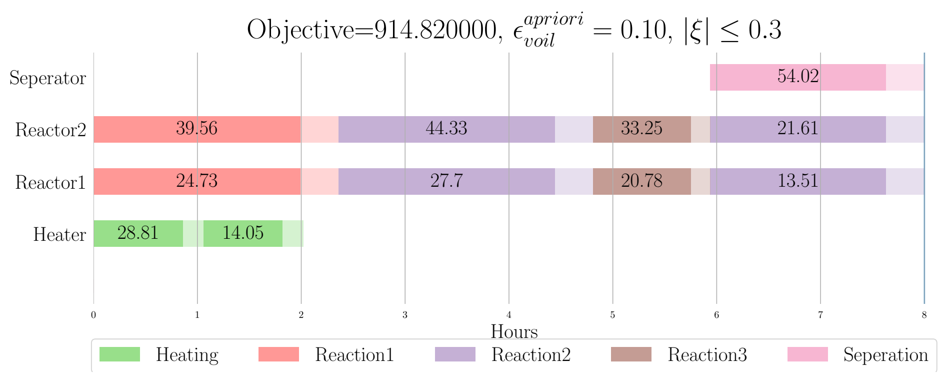


Figure 4.2: Solution with probability of constraint violation = 0.1

uncertainty. Now consider the unit reactor 1, tasks occurring in this unit are independent to tasks in other units.

Since, the feasibility of each task in Reactor 1 i.e. probability of constraint satisfaction of each task in Reactor 1 is $1 - \epsilon^{prio} = 0.9$, the overall probability of feasible operation of Reactor 1 is $(1 - \epsilon^{prio})^4 = 0.65$. Thus the actual probability of feasible operation of a single unit is much less than that desired. Extending this observation to multiple units and the actual feasibility of complete schedule would exponentially reduce, in practice, such a schedule with reduced feasibility is unacceptable. Although, it is worth noting that robust scheduling model has not added any additional variables or constraint to the problem and has not lead to any increase in problem size.

4.2 Reactive Scheduling: Improvements

On the other hand, if we were to resort to reactive scheduling approach and reschedule after realization of uncertainty at every event point. To achieve this reactive scheduling behavior, we created following algorithm.

Algorithm 4.1 Algorithm for Reactive Scheduling

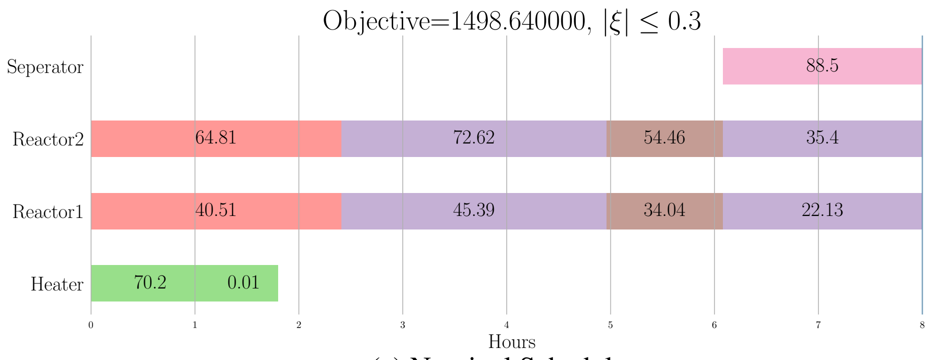
```
procedure ReactiveScheduling($M$, $\xi^{UB}$)
    $x* \leftarrow$ NominalScheduler($M$)        ▷ Using deterministic model of Chapter 3
    for all $n \in$ range$(1, N-1)$ do        ▷ e.g. 4 event points in motivating example
        $w^{fx}_{inn'} \leftarrow w^*_{inn'} \forall\, n' \geq n$        ▷ Fix task assignment of event $n$
        $b^{fx}_{inn'} \leftarrow b^*_{inn'} \forall\, n' \geq n$        ▷ Fix batch size of event $n$
        $\alpha_{in} \leftarrow$ randUniform$((1-\xi^{UB})\alpha_{in}, (1+\xi^{UB})\alpha_{in})$        ▷ A random value is assigned to processing time of tasks in event n
        $x^*_n \leftarrow x^*$        ▷ Store intermediate solution
        $x^* \leftarrow$ NominalScheduler$(M)$        ▷ Updated model is re-optimized, $w_{inn'}$ and $b_{inn'}$ are fixed
    end for
    return $x^*_n$        ▷ Retrieve all the solutions
end procedure
```

As observed from example simulation (fig. 4.3), the objective value of the overall schedule over the horizon is not guaranteed. Due to constant rescheduling, it leads to confusion on production floor. In order to take advantage of improvement in objective value that comes from using algorithm 4.1, while maintaining schedule feasibility we propose a proactive-reactive scheduling approach. In this approach, we modify algorithm 4.1 to generate robust solution for each stage instead of nominal schedule.

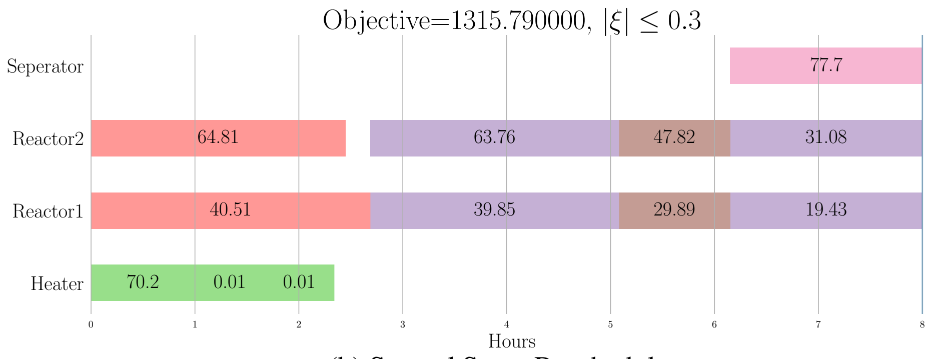
Algorithm 4.2 Algorithm for Proactive-Reactive Scheduling

```
procedure ProReactiveScheduling($M_{RC}$, $\xi^{UB}$, $\epsilon^{prio}$)    ▷ Using deterministic model of Chapter 3 coupled with constraints 4.4 and 4.5
    $x^* \leftarrow$ AprioriRobustOptim($M_{RC}$, $\xi^{UB}$, $\epsilon^{prio}$)    ▷ Robust solver from algorithm 3.1
    for all $n \in$ range$(1, N-1)$ do    ▷ e.g. $N = 4$ event points in motivating example
        $w^{fx}_{inn'} \leftarrow w^*_{inn'} \forall\, n' \geq n$    ▷ Fix task assignment of event $n$
        $b^{fx}_{inn'} \leftarrow b^*_{inn'} \forall\, n' \geq n$    ▷ Fix batch size of event $n$
        $\alpha_{in} \leftarrow$ randUniform$((1-\xi^{UB})\alpha_{in}, (1+\xi^{UB})\alpha_{in})$    ▷ A random value is assigned to processing time of tasks in event n
        $x^*_n \leftarrow x^*$    ▷ Store intermediate solution
        $x* \leftarrow$ AprioriRobustOptim($M_{RC}$)    ▷ Updated model is re-optimized, $w_{inn'}$ and $b_{inn'}$ are fixed
    end for
    return $x^*_n$    ▷ Retrieve all the solutions
end procedure
```

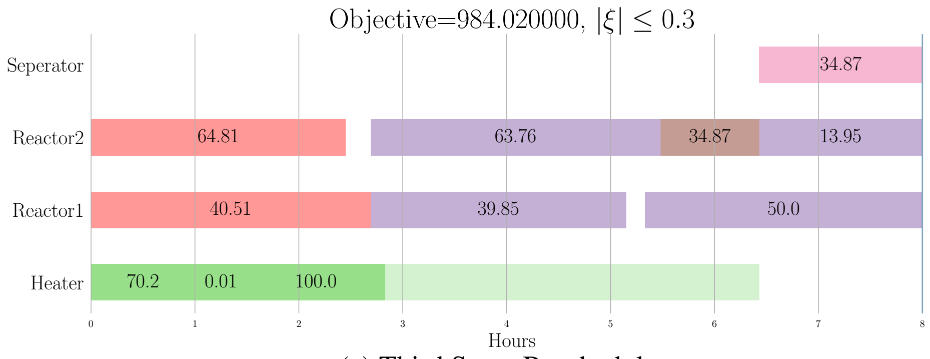
Observing the simulation example (fig. 4.4) for algorithm 4.2, the rescheduling decision are not drastic as compared to algorithm 4.1. This modification results in improved feasibility and practical applicability of method at cost of objective value deterioration.



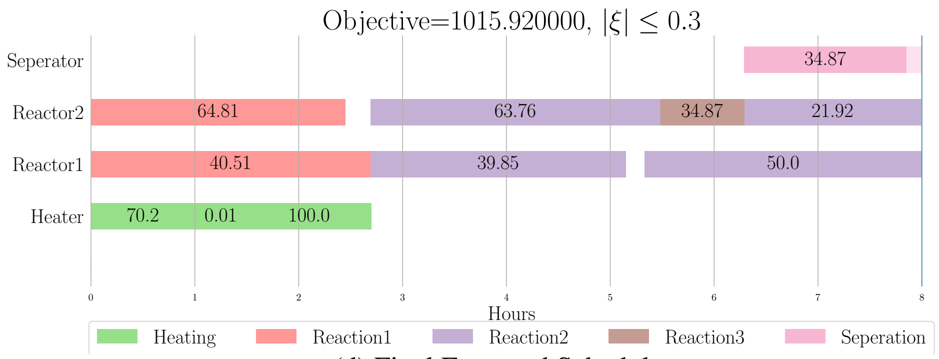
(a) Nominal Schedule



(b) Second Stage Reschedule

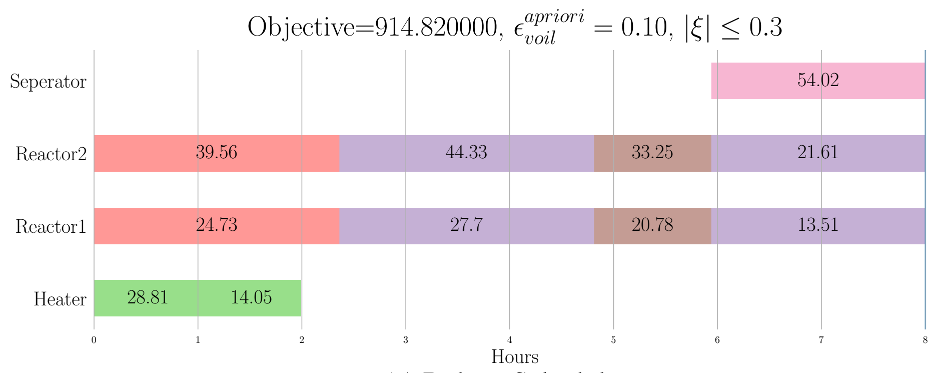


(c) Third Stage Reschedule

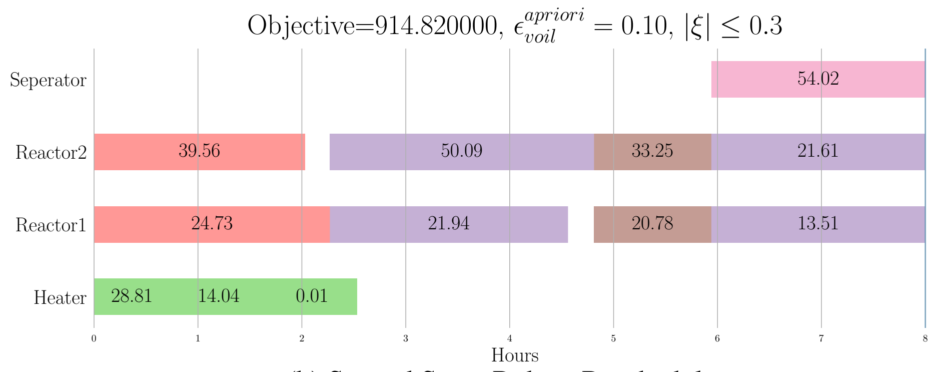


(d) Final Executed Schedule

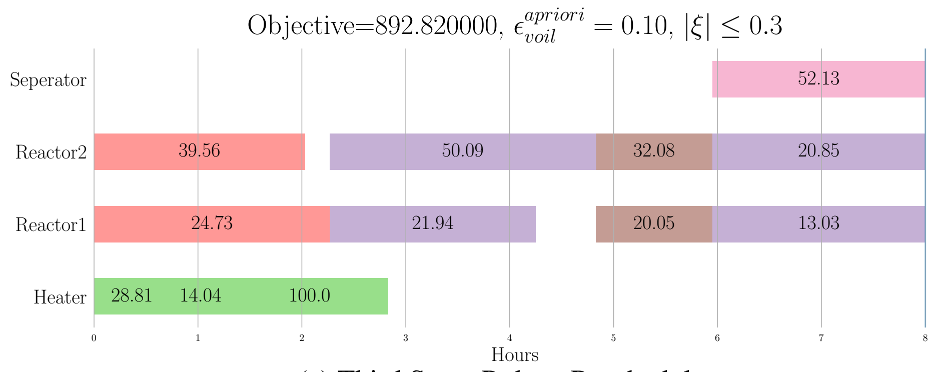
Figure 4.3: Reactive Scheduling Simulation



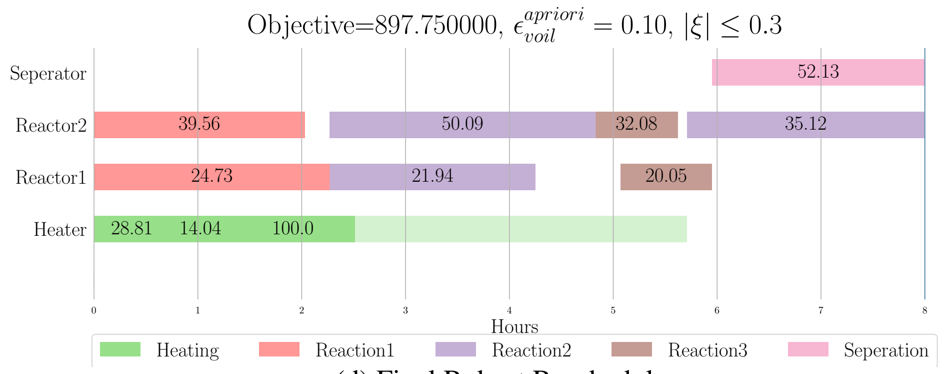
(a) Robust Schedule



(b) Second Stage Robust Reschedule



(c) Third Stage Robust Reschedule



(d) Final Robust Reschedule

Figure 4.4: Proactive-Reactive Scheduling Approach

4.3 An Improved Robust Scheduling Approach

Although the feasibility of solution has improved, the resources and skill to re-optimize a schedule at every occurrence of uncertainty is not feasible on a complex production floor. We improve algorithm 4.2, by introducing new constraints in the model at every stage in rescheduling. Referencing to algorithms 4.1 and 4.2, we define each event point of information collection from simulation or feedback as the active event point.

$$T^f_{in} \geq \tilde{T}^s_{in} + \tilde{\alpha}_{in} w_{inn} + \beta_i b_{inn} \forall\ i \in \mathbf{I},\ n \text{ is the active event point} \tag{4.6}$$

$$T^f_{in'} \geq \tilde{T}^s_{in} + \tilde{\alpha}_{in} w_{inn'} + \beta_i b_{inn'} - M(1 - w_{inn'}) \forall i \in \mathbf{I},\ n' \in \mathbf{N},\ n < n',\ n \text{ is the active event point} \tag{4.7}$$

Where, $\tilde{T}^s_{in}$ is the uncertain start time of a task at event n. This constraints are only introduced for an active event point n. Uncertainty in start time of task is induced from uncertainty in processing parameters of task that occurred in previous events ($\tilde{\alpha}_{in''} \forall\ n'' < n$). Hence we can derive an expression for $\tilde{T}^s_{in}$ as a function of $\tilde{\alpha}_{in''}$.

$$\tilde{T}^s_{in} = T^s_i n + \sum_{i' \in \mathbf{I}} \sum_{n' < n} |\omega_{in}|_{i'n'} \hat{\alpha}_{i'n'} \xi_{i'n'} \tag{4.8}$$

where, T^s_{in} is the value of start time if all parameters in the problem realize their nominal values. $|\omega_{in}|_{i'n'}$ is a binary variable which decides whether $\alpha_{i'n'}$ has any influence on the next event or not. Combining constraint 4.6 and 4.7, substituting 4.8 and generating robust counterpart with Interval + Polyhedral uncertainty set using equation 3.17:

$$\begin{aligned} T^f_{in'} \geq T^s_{in} + \alpha_{in} w_{inn'} + \beta_i b_{inn'} + \Gamma_{in} z_{in} + \sum_{i' \in \mathbf{I}} \sum_{\substack{n'' \in \mathbf{N} \\ n'' < n}} |p_{in}|_{i'n'} + p'_{in} \\ -M(1 - w_{inn'}) \; \forall \, i \in \mathbf{I}, \; n' \in \mathbf{N}, \; n \leq n', \; n \text{ is the active event point} \\ z_{in} + p'_{in} \geq \hat{\alpha}_{in} w_{inn'} \; \forall \, i \in \mathbf{I}, \; n, n' \in \mathbf{N}, \; n \leq n' \\ z_{in} + |p_{in}|_{i'n'} \geq \hat{\alpha}_{i'n'} |\omega_{in}|_{i'n'} \; \forall \, i, i' \in \mathbf{I}, \; n' \in \mathbf{N}, \; n' < n, \; n \text{ is the active event point} \end{aligned} \tag{4.9}$$

where, z_{in}, p'_{in} and $|p_{in}|_{i'n'}$ are auxiliary positive variables introduced as a part of robust counterpart formulation. The big-M condition in constraint 4.9 will ensure that the constraint is redundant if $w_{inn'}$ is inactive.

$$|\omega_{in}|_{i'n'} \leq w_{i'n''n'} \forall \, i, i' \in \mathbf{I}, \; n', n'' \in \mathbf{N}, \; n'' \leq n', \; n' < n, \; n \text{ is the active event point} \tag{4.10}$$

$$\begin{aligned} |\omega_{in}|_{i'n'} = 0 \; \forall \, i, i' \in \mathbf{I}, \; n', n'' \in \mathbf{N}, \; n' \geq n, \; n \text{ is the active event point} \\ \sum_{i' \in \mathbf{I}} \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} |\omega_{in}|_{i'n'} \leq n - 1 \; \forall \, i \in \mathbf{I}, \; n \text{ is the active event point} \\ \sum_{i' \in \mathbf{I}} \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} |\omega_{in}|_{i'n'} \leq M \sum_{\substack{n' \in \mathbf{N} \\ n' \leq n}} w_{in'n} \; \forall \, i \in \mathbf{I}, \; n \text{ is the active event point} \end{aligned} \tag{4.11}$$

Constraint 4.10 states that uncertainty in task that did not occur cannot influence start time of any task in future. Constraint 4.11 ensures that the uncertainty in future task has no influence on the start time of current task. Also, the maximum number of events that can affect the start time of a task cannot be more than the number of event points that have elapsed in the schedule so far. If a task is not suppose to start, then no past event can have any influence on its start time.

$$\sum_{i'\in\mathbf{I}}\sum_{\substack{n'\in\mathbf{N}\\n'<n}}|\omega_{in}|_{i'n'}\hat{\alpha}_{i'n'} \geq \sum_{i'\in\mathbf{I}_j}\sum_{\substack{n'\in\mathbf{N}\\n'<n}}\sum_{\substack{n''\in\mathbf{N}\\n''\leq n'}}\hat{\alpha}_{i'n'}w_{i'n''n'} - M(1-\sum_{\substack{n'\in\mathbf{N}\\n'\leq n}}w_{in'n})\;\forall\, i\in\mathbf{I}_j,\; j\in\mathbf{J},\; n \text{ is the active event point} \quad (4.12)$$

Constraint 4.12 asserts that the uncertainty in start time of a task is contributed from uncertainty in processing time of previous tasks in the same unit. Using inequality along with big-M constraint instead of equality, helps us ensure that there is no influence of uncertainty of tasks in previous event, if the current task is not performed.

$$\sum_{i'\in\mathbf{I}}\sum_{\substack{n'\in\mathbf{N}\\n'<n}}|\omega_{in}|_{i'n'}\hat{\alpha}_{i'n'} \geq \sum_{i'\in\mathbf{I}_{j'}}\sum_{\substack{n'\in\mathbf{N}\\n'<n}}\sum_{\substack{n''\in\mathbf{N}\\n''\leq n'}}\hat{\alpha}_{i'n'}w_{i'n''n'} - M(2-\sum_{\substack{n'\in\mathbf{N}\\n'\leq n}}w_{in'n} - \sum_{\substack{i'\in\mathbf{I}_s^p\\i'\in\mathbf{I}_{j'}}}\sum_{\substack{n'\in\mathbf{N}\\n'\leq n}}w_{i'n'(n-1)})\;\forall\, s\in\mathbf{S},\; i\in\mathbf{I}_j,\; i\in\mathbf{I}_s^c,\; j,j'\in\mathbf{J},\; j'\neq j,\; n \text{ is the active event point} \quad (4.13)$$

Constraint 4.13 states that uncertainty in start time of a task is also contributed from uncertainty in processing time of tasks in a unit where, the raw material for current task was produced. Together, constraint 4.12 and 4.13 ascertain that uncertainty in start time of a task, is maximum of sum of processing time uncertainty in the same unit or in a dependent different unit. Grouping constraints 4.9, 4.10, 4.11, 4.12 and 4.13 into M_n^{IR} group of constraints for every active event point n.

Compared to algorithm 4.2, here in algorithm 4.3, we fix α_{in} to its nominal value instead of a random value. We add the M_n^{IR} for each active event point. In comparison to previous reactive scheduling algorithms, this algorithm does not require any form of

Algorithm 4.3 Algorithm for Improved Robust Scheduling

```
procedure ImpRobustScheduling($M_{RC}$, $\xi^{UB}$, $\epsilon^{prio}$)    ▷ Using deterministic model of Chapter 3 coupled with constraints 4.4 and 4.5
    $x^* \leftarrow$ AprioriRobustOptim($M_{RC}$, $\xi^{UB}$, $\epsilon^{prio}$)    ▷ Robust solver from algorithm 3.1
    for all $n \in$ range$(2, N)$ do    ▷ e.g. $N = 4$ event points in motivating example
        $w^{fx}_{inn'} \leftarrow w^*_{inn'} \forall\, n' \geq n$    ▷ Fix task assignment of event $n$
        $\alpha_{i(n-1)} \leftarrow \bar{\alpha}_{i(n-1)}$    ▷ Processing time values of elapsed events is fixed to their nominal value i.e. they behavior is changed from uncertainty parameter values to fixed values
        $\Delta_{i(n-1)\leftarrow 0}$    ▷ Since the parameters are not uncertain anymore
        $M_{RC} \leftarrow M_{RC} + M^{IR}_n$    ▷ Adding group of constraints from improved robust formulation
        $x^*_n \leftarrow x^*$    ▷ Store intermediate solution
        $x* \leftarrow$ AprioriRobustOptim($M_{RC}$)    ▷ Updated model is re-optimized, $w_{inn'}$ and $b_{inn'}$ are fixed
    end for
    return $x^*_n$    ▷ Retrieve all the solutions
end procedure
```

on-line rescheduling. Once the schedule is generated, task assignment, sequencing and batch sizes remain unchanged, only the start times of tasks are affected by the uncertainty in processing time parameter.

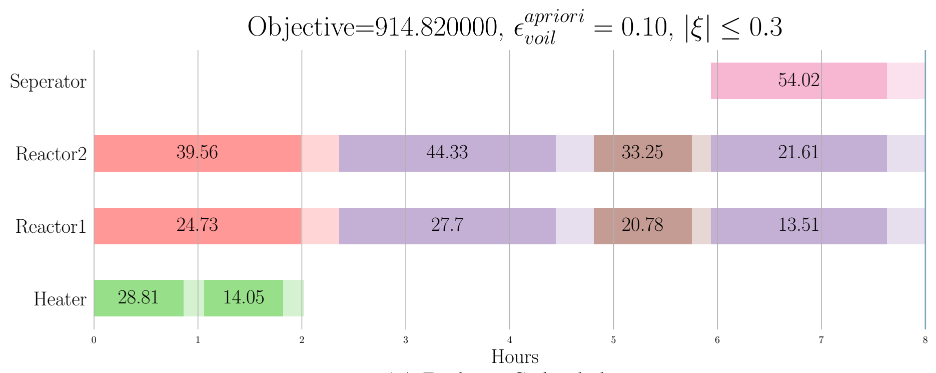
Although, compared to traditional robust optimization problem, we need to solve N instances of the scheduling problem. It must be noted that, every subsequent problem has lower number of binary decision variables as well as they start with a feasible solution derived from previous solution instance. Since, the structure of nominal scheduling model was unaltered by these robust counterpart formulation, any objective function or constraints such as makespan minimization, resource balance, intermediate due date, etc. can be applied on the model.

Figure 4.5 and 4.6 are two example instances solved using improved robust scheduling approach. As observed, the objective value has improved compared to both traditional robust scheduling and proactive-reactive scheduling for case with $\epsilon^{prio} \neq 0$. While for worst case, there is no improvement in objective value. If one were to solve with $\epsilon^{prio} = 1$, the solution would be same as nominal schedule. Instead of using heuristic to determine start time of each task, we can logically determine the start time of each task as finish time of previous dependent task.

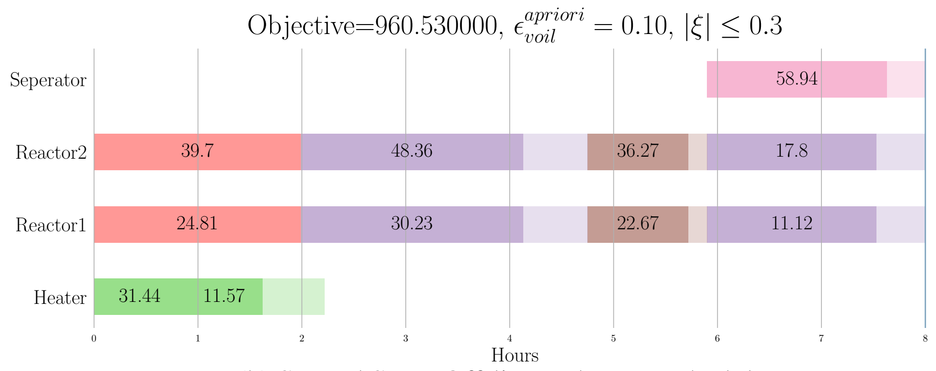
Alternatively, one can generate a robust counterpart for constraint 4.6 and 4.7 using Interval + Ellipsoidal Set. Also, for unbounded uncertainty case, one can generate a robust counterpart using polyhedral or ellipsoidal uncertainty sets.

$$
\begin{gathered}
T^f_{in'} \geq T^s_{in} + \alpha_{in} w_{inn'} + \beta_i b_{inn'} + \Omega_{in} t_{in} + \sum_{i' \in \mathbf{I}} \sum_{\substack{n'' \in \mathbf{N} \\ n'' < n}} |u_{in}|_{i'n'} + u'_{in} \\
-M(1 - w_{inn'}) \; \forall \, i \in \mathbf{I}, \; n' \in \mathbf{N}, \; n \leq n', \; n \text{ is the active event point} \\
|u_{in}|_{i'n'} \geq \hat{\alpha}_{i'n'} |\omega_{in}|_{i'n'} - |z_{in}|_{i'n'} \; \forall \, i, i' \in \mathbf{I}, \; n' \in \mathbf{N}, \; n' < n, \; n \text{ is the active event point} \\
-|u_{in}|_{i'n'} \leq \hat{\alpha}_{i'n'} |\omega_{in}|_{i'n'} - |z_{in}|_{i'n'} \; \forall \, i, i' \in \mathbf{I}, \; n' \in \mathbf{N}, \; n' < n, \; n \text{ is the active event point} \\
u'_{in} \geq \hat{\alpha}_{in} w_{inn'} - z'_{in} \; \forall \, i \in \mathbf{I}, \; n, n' \in \mathbf{N}, \; n \leq n', \; n \text{ is the active event point} \\
-u'_{in} \leq \hat{\alpha}_{i'n'} w_{inn'} - z'_{in} \; \forall \, i \in \mathbf{I}, \; n, n' \in \mathbf{N}, \; n \leq n', \; n \text{ is the active event point} \\
t^2_{in} \geq (z'_{in})^2 + \sum_{i' \in \mathbf{I}} \sum_{\substack{n' \in \mathbf{N} \\ n' < n}} (|z_{in}|_{i'n'})^2 \forall \, i \in \mathbf{I}, \; n \in \mathbf{N}, \; n \text{ is the active event point}
\end{gathered}
\tag{4.14}
$$

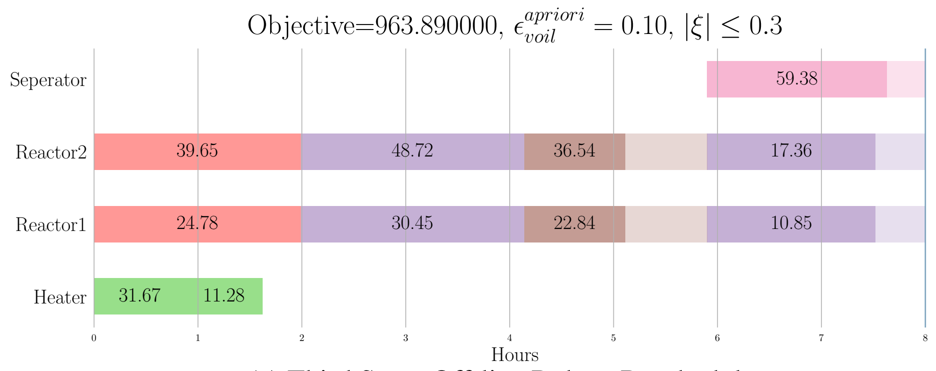
where, t_{in}, z'_{in}, u'_{in}, $|z_{in}|_{i'n'}$ and $|u_{in}|_{i'n'}$ are introduced as positive auxiliary variables. The robust counterpart 4.14 is structured differently than one presented in constraint 3.18 but in spirit they are the same. These different reformulation follows a second order conic programming (SOCP) structure and can be solved using commercial MIP solvers like CPLEX and Gurobi [48] to global optimality (convex problem).



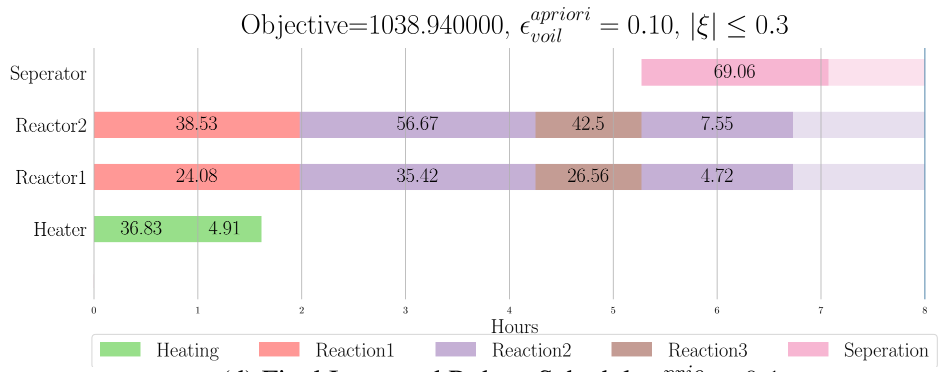
(a) Robust Schedule



(b) Second Stage Off-line Robust Reschedule



(c) Third Stage Off-line Robust Reschedule



(d) Final Improved Robust Schedule $\epsilon^{prio} = 0.1$

Figure 4.5: Improved Robust Scheduling Approach

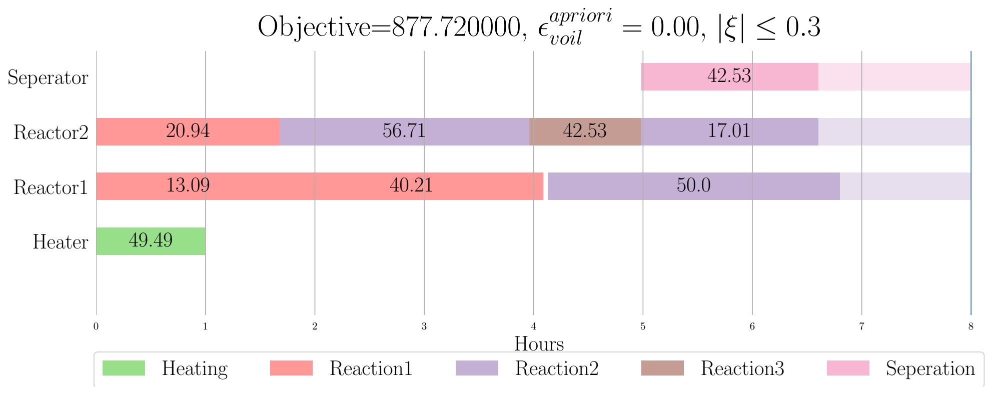


Figure 4.6: Improved Robust Soyster Solution

4.4 Improving the quality of solution

Instead of using *a priori* bounds, we can use *a posteriori* bounds to characterize the quality of robust solution. One can use iterative algorithm 3.2, for characterizing quality of solution using *a posteriori* bounds. For purpose of demonstration, we generate solutions for various *a priori* probability of constraint violation and then characterize the *a posteriori* probability of constraint violation for each individual solution. Since the constraint on which bounds are applied at is changed at every stage of algorithm, the final constraint on which *a posteriori* bounds are applied is:

$$\tilde{T}^s_{in} + \tilde{\alpha}_{in} w_{inn} + \beta_i b_{inn} \leq H \; \forall \, i \in \mathbf{I}, \; n = card(n) \tag{4.15}$$

Constraint 4.15 states that, the finish time of last event should be less than horizon. Hence, *a posteriori* probability of constraint violation would characterize the probability of delay or overtime in the task.

ϵ^{prio}	ϵ^{post}			Objective Value
	Reactor 1	Reactor 2	Separator	
0	0	0	0	877.72
0.10	0.000058	0.001206	0.020499	1038.94
0.20	0.000396	0.000396	0.027774	1092.86
0.30	0.006358	0.001648	0.047201	1137.73
0.40	0.004707	0.004707	0.074537	1175.53
0.50	0.009923	0.009923	0.106614	1209.93
0.60	0.017221	0.030504	0.141310	1242.99
0.70	0.048277	0.048277	0.182355	1278.02
0.80	0.074537	0.097187	0.236754	1323.55
0.90	0.163992	0.163992	0.314623	1377.97
1.00	0.500000	0.500000	0.500000	1498.63

Table 4.1: Improved Robust Approach for motivating example with interval-polyhedral set

ϵ^{prio}	ϵ^{post}			Objective Value
	Reactor 1	Reactor 2	Separator	
0	0	0	0	877.72
0.10	0.001359	0.001359	0.001439	981.32
0.20	0.002875	0.002875	0.006955	1036.12
0.30	0.006882	0.006882	0.017221	1084.46
0.40	0.013184	0.013184	0.033458	1126.54
0.50	0.020499	0.020499	0.055364	1165.65
0.60	0.032187	0.032187	0.085654	1203.83
0.70	0.053099	0.035522	0.125663	1244.50
0.80	0.105064	0.101074	0.182355	1296.49
0.90	0.194580	0.188738	0.266706	1359.05
1.00	0.500000	0.500000	0.500000	1498.63

Table 4.2: Improved Robust Approach for motivating example with interval-ellipsoidal set

Table 4.1 and 4.2 represent the application of improved robust approach on motivating example using interval + polyhedral and interval + ellipsoidal sets respectively. All these experiments were conducted using GAMS/Gurobi [49, 48] and the probabilistic bounds were generated using PROTO [35]. It should be noted that, robust counterpart with interval polyhedral sets are solved using mixed-integer quadratically constrained programming (MIQCP), which increases the computational expense as compared to polyhedral sets. Since, interval + ellipsoidal uncertainty set have tighter probabilistic bounds, it offsets the increase in computational cost by improving the objective value for a given *a posterori* probability of constraint violation ϵ^{post}. Thus using the probabilistic bounds one can quantify the quality of robust solution and choose an appropriate schedule that matches their risk appetite.

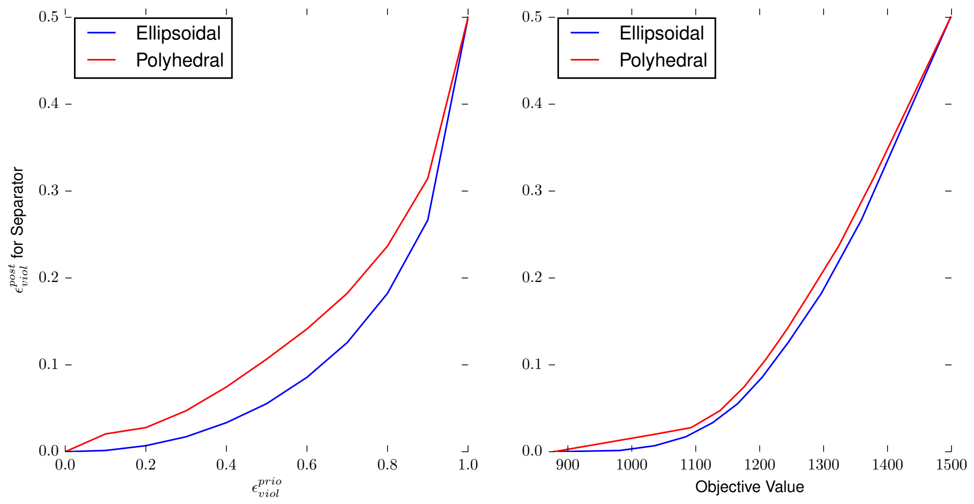


Figure 4.7: Quality of Robust Solution

4.5 Computational Studies

As observed from figure 4.6, for worst-case solution, improved robust scheduling and traditional robust scheduling lead to same objective value, batch sequencing and batch size. Hence, by using any type of proactive solution technique, the Soyster solution cannot be improved upon and it should match the solution of traditional robust approach. This fact can be utilized to verify the integrity of our approach. We apply our improved robust optimization framework on benchmark instances from Li and Floudas [22] at $\epsilon^{prio} = 0$ and $\epsilon^{prio} = 1$ to verify the integrity of our approach.

Example	Horizon (hr) H	Event points n	Objective Value			
			Traditional Robust Approach		Improved Robust Approach	
			$\epsilon^{prio} = 0$	$\epsilon^{prio} = 1$	$\epsilon^{prio} = 0$	$\epsilon^{prio} = 1$
ME	8	4	877.71	1498.63	877.71	1498.63
P2	8	6	1150.00	1583.44	1150.00	1583.44
P3	9	5	60.00	210.00	60.00	210.00
P4	10	6	160.07	400.00	160.07	400.00

Table 4.3: Computational Study on Literature Benchmark

5. SUMMARY AND CONCLUSIONS

In this chapter, we summarize the major contribution of the proposed improved robust optimization method and suggest the possible avenues for future research.

5.1 Objectives Achieved

In this work, we proposed a novel multi-stage robust optimization framework for process scheduling application without use of any heuristics [44, 45]. For a given level of risk, we demonstrated that our approach yields better objective values as compared to traditional robust optimization approaches [2, 42]. In our iterative scheme of schedule generation, we assimilate the delay in tasks for a unit in an uncertainty set and provide probabilistic bounds on the delay of particular units' processing time. To best of our knowledge, for the first time in literature, we provide a tool to study delay risk versus objective value trade-off using these strong probabilistic bounds. Moreover, our method does not require explicit information regarding uncertainty of the parameter, but can take advantage of available information [35]. As the structure of underlying deterministic model is unaltered, our approach can benefit from its computational efficiency as well as its ability to handle real-life scenarios like utility limitations, storage policies, changeover times, etc. Avoiding heuristics helps us obtain rescheduling decisions, which can be implemented by the operators without use of a computational tool. It should be highlighted that, since we are using an unaltered deterministic model, our problem size is much smaller compared with alternative methods proposed in the literature to date.

5.2 Further Study

There are several avenues of research that extend quite naturally from the work presented in this thesis. They can be summarized as follows.

- The area of simultaneous planning and scheduling is important to production scheduling community. Planning problem aims at quantifying long-term production goals, while scheduling problem seeks to determine the specifies of the production schedule itself. While the method developed in these work can handle uncertainty in lowest level of decision making, integrating it with uncertainty aware planning problem would improve the reliability of this method.
- The problem of demand side management (DSM) has received significant interest in production operations literature. The rise in electricity demand coupled with increasing penetration of intermittent renewable energy into power supply mix has increased the level of uncertainty tremendously. It would be beneficial for the planning and scheduling models for industrial DSM to account for the uncertainty.
- The area of plant design and synthesis is plagued with uncertainty in market prices and demand. A implementation of multistage proactive-reactive algorithm would greatly enhance the productivity and profitability of the plant.

REFERENCES

[1] I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand, and J. Wassick, “Scope for industrial applications of production scheduling models and solution methods,” *Computers & Chemical Engineering*, vol. 62, pp. 161–193, 2014.

[2] X. Lin, S. L. Janak, and C. A. Floudas, “A new robust optimization approach for scheduling under uncertainty:: I. Bounded uncertainty,” *Computers & Chemical Engineering*, vol. 28, no. 6, pp. 1069–1085, 2004.

[3] C. A. Floudas and X. Lin, “Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review,” 2004.

[4] C. A. Méndez, J. Cerdá, I. E. Grossmann, I. Harjunkoski, and M. Fahl, “State-of-the-art review of optimization methods for short-term scheduling of batch processes,” *Computers & Chemical Engineering*, vol. 30, no. 6, pp. 913–946, 2006.

[5] R. K. Phanden, A. Jain, and R. Verma, “Integration of process planning and scheduling: a state-of-the-art review,” *International Journal of Computer Integrated Manufacturing*, vol. 24, pp. 517–534, jun 2011.

[6] C. T. Maravelias, “General framework and modeling approach classification for chemical production scheduling,” *AIChE Journal*, vol. 58, pp. 1812–1828, jun 2012.

[7] E. Kondili, C. Pantelides, and R. Sargent, “A general algorithm for short-term scheduling of batch operationsI. MILP formulation,” *Computers & Chemical Engineering*, vol. 17, pp. 211–227, feb 1993.

[8] C. T. Maravelias, “On the combinatorial structure of discrete-time MIP formulations for chemical production scheduling,” *Computers and Chemical Engineering*, vol. 38,

pp. 204–212, 2012.

[9] S. Velez and C. T. Maravelias, “Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling,” *Computers and Chemical Engineering*, vol. 53, pp. 70–85, 2013.

[10] P. M. Castro, A. P. Barbosa-po, H. a. Matos, and A. Q. Novais, “Simple Continuous-Time Formulation for Short-Term Scheduling of Batch and Continuous Processes,” *Industrial & Engineering Chemistry Research*, vol. 43, pp. 105–118, 2004.

[11] P. M. Castro, I. Harjunkoski, and I. E. Grossmann, “New Continuous-Time Scheduling Formulation for Continuous Plants under Variable Electricity Cost,” *Industrial & Engineering Chemistry Research*, vol. 48, pp. 6701–6714, jul 2009.

[12] P. M. Castro, I. Harjunkoski, and I. E. Grossmann, “Optimal scheduling of continuous plants with energy constraints,” *Computers & Chemical Engineering*, vol. 35, no. 2, pp. 372–387, 2011.

[13] C. T. Maravelias and I. E. Grossmann, “New General Continuous-Time StateTask Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants,” *Industrial & Engineering Chemistry Research*, vol. 42, no. 13, pp. 3056–3074, 2003.

[14] M. G. Ierapetritou and C. A. Floudas, “Effective continuous-time formulation for short-term scheduling. Part 2. Continuous and semicontinuous processes,” *Industrial & Engineering Chemistry Research*, vol. 37, no. 11, pp. 4360–4374, 1998.

[15] M. Ierapetritou, T. Hené, and C. Floudas, “Short-term scheduling of batch plants with multiple intermediate due dates,” *Computers & Chemical Engineering*, vol. 23, pp. S515–S518, jun 1999.

[16] S. L. Janak, X. Lin, and C. a. Floudas, “Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Pro-

cesses: Resource Constraints and Mixed Storage Policies," *Industrial and Engineering Chemistry Research*, vol. 43, no. 10, p. 2516, 2004.

[17] M. A. Shaik and C. A. Floudas, "Improved unit-specific event-based continuous-time model for short-term scheduling of continuous processes: Rigorous treatment of storage requirements," *Industrial and Engineering Chemistry Research*, vol. 46, no. 6, pp. 1764–1779, 2007.

[18] M. A. Shaik and C. A. Floudas, "Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework," *Computers & Chemical Engineering*, vol. 32, no. 1, pp. 260–274, 2008.

[19] M. A. Shaik and C. A. Floudas, "Novel Unified Modeling Approach for Short-Term Scheduling," *Industrial & Engineering Chemistry Research*, vol. 48, pp. 2947–2964, mar 2009.

[20] M. A. Shaik, S. L. Janak, and C. A. Floudas, "Continuous-time models for short-term scheduling of multipurpose batch plants: A comparative study," *Industrial and Engineering Chemistry Research*, vol. 45, no. 18, pp. 6190–6209, 2006.

[21] J. Li, N. Susarla, I. A. Karimi, M. A. Shaik, and C. A. Floudas, "An Analysis of Some Unit-Specific Event-Based Models for the Short-Term Scheduling of Noncontinuous Processes," *Industrial & Engineering Chemistry Research*, vol. 49, pp. 633–647, jan 2010.

[22] J. Li and C. A. Floudas, "Optimal Event Point Determination for Short-Term Scheduling of Multipurpose Batch Plants via Unit-Specific Event-Based Continuous-Time Approaches," *Industrial & Engineering Chemistry Research*, vol. 49, pp. 7446–7469, aug 2010.

[23] A. L. Soyster, "Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming," *Operations Research*, vol. 21, pp. 1154–1157, oct 1973.

[24] A. Ben-Tal, L. El-Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton University Press, 2009.

[25] A. Ben-Tal and A. Nemirovski, "Robust solutions of Linear Programming problems contaminated with uncertain data," *Mathematical Programming*, vol. 88, pp. 411–424, sep 2000.

[26] D. Bertsimas and M. Sim, "Tractable Approximations to Robust Conic Optimization Problems," *Mathematical Programming*, vol. 107, pp. 5–36, jun 2006.

[27] Z. Li, R. Ding, and C. A. Floudas, "A Comparative Theoretical and Computational Study on Robust Counterpart Optimization: I. Robust Linear Optimization and Robust Mixed Integer Linear Optimization," *Industrial & Engineering Chemistry Research*, vol. 50, pp. 10567–10603, sep 2011.

[28] P. M. Verderame and C. A. Floudas, "Multisite Planning under Demand and Transportation Time Uncertainty: Robust Optimization and Conditional Value-at-Risk Frameworks," *Industrial & Engineering Chemistry Research*, vol. 50, pp. 4959–4982, may 2011.

[29] D. Bertsimas and M. Sim, "The Price of Robustness," *Operations Research*, vol. 52, pp. 35–53, feb 2004.

[30] Z. Li, Q. Tang, and C. A. Floudas, "A Comparative Theoretical and Computational Study on Robust Counterpart Optimization: II. Probabilistic Guarantees on Constraint Satisfaction," *Industrial & Engineering Chemistry Research*, vol. 51, pp. 6769–6788, may 2012.

[31] Z. Li and C. A. Floudas, “A Comparative Theoretical and Computational Study on Robust Counterpart Optimization: III. Improving the Quality of Robust Solutions,” *Industrial & Engineering Chemistry Research*, vol. 53, pp. 13112–13124, aug 2014.

[32] Y. A. Guzman, L. R. Matthews, and C. A. Floudas, “New a priori and a posteriori probabilistic bounds for robust counterpart optimization: I. Unknown probability distributions,” *Computers & Chemical Engineering*, vol. 84, pp. 568–598, 2016.

[33] Y. A. Guzman, L. R. Matthews, and C. A. Floudas, “New a priori and a posteriori probabilistic bounds for robust counterpart optimization: II. A priori bounds for known symmetric and asymmetric probability distributions,” *Computers & Chemical Engineering*, vol. 101, pp. 279–311, jun 2017.

[34] Y. A. Guzman, L. R. Matthews, and C. A. Floudas, “New a priori and a posteriori probabilistic bounds for robust counterpart optimization: III. Exact and near-exact a posteriori expressions for known probability distributions,” *Computers & Chemical Engineering*, vol. 103, pp. 116–143, aug 2017.

[35] L. Matthews, Y. Guzman, and C. Floudas, “PROTO: Platform for Robust OpTimizatiOn.,” 2018.

[36] Z. Li and M. Ierapetritou, “Process scheduling under uncertainty: Review and challenges,” *Computers & Chemical Engineering*, vol. 32, no. 4, pp. 715–727, 2008.

[37] K. B. Kanakamedala, G. V. Reklaitis, and V. Venkatasubramanian, “Reactive schedule modification in multipurpose batch chemical plants,” *Industrial & Engineering Chemistry Research*, vol. 33, pp. 77–90, jan 1994.

[38] M. Rodrigues, L. Gimeno, C. Passos, and M. Campos, “Reactive scheduling approach for multipurpose chemical batch plants,” *Computers & Chemical Engineering*, vol. 20, pp. S1215–S1220, jan 1996.

[39] G. M. Kopanos, E. Capon-Garcia, A. Espuna, and L. Puigjaner, "Costs for Rescheduling Actions: A Critical Issue for Reducing the Gap between Scheduling Theory and Practice," *Industrial & Engineering Chemistry Research*, vol. 47, pp. 8785–8795, nov 2008.

[40] J. P. Vin and M. G. Ierapetritou, "A New Approach for Efficient Rescheduling of Multiproduct Batch Plants," *Industrial & Engineering Chemistry Research*, vol. 39, pp. 4228–4238, 2000.

[41] S. L. Janak, C. A. Floudas, J. Kallrath, and N. Vormbrock, "Production scheduling of a large-scale industrial batch plant. II. Reactive scheduling," *Industrial and Engineering Chemistry Research*, vol. 45, no. 25, pp. 8253–8269, 2006.

[42] S. L. Janak, X. Lin, and C. A. Floudas, "A new robust optimization approach for scheduling under uncertainty. II. Uncertainty with known probability distribution," *Computers and Chemical Engineering*, vol. 31, no. 3, pp. 171–195, 2007.

[43] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, "Adjustable robust solutions of uncertain linear programs," *Mathematical Programming*, vol. 99, pp. 351–376, mar 2004.

[44] H. Shi and F. You, "A computational framework and solution algorithms for two-stage adaptive robust scheduling of batch manufacturing processes under uncertainty," *AIChE Journal*, vol. 62, pp. 687–703, mar 2016.

[45] N. H. Lappas and C. E. Gounaris, "Multi-stage adjustable robust optimization for process scheduling under uncertainty," *AIChE Journal*, vol. 62, pp. 1646–1667, may 2016.

[46] P. M. Verderame, J. A. Elia, J. Li, and C. A. Floudas, "Planning and Scheduling under Uncertainty: A Review Across Multiple Sectors," *Industrial & Engineering*

Chemistry Research, vol. 49, pp. 3993–4017, may 2010.

[47] L. S. Dias and M. G. Ierapetritou, “Integration of scheduling and control under uncertainties: Review and challenges,” *Chemical Engineering Research and Design*, vol. 116, pp. 98–113, 2016.

[48] I. Gurobi Optimization, “Gurobi Optimizer Reference Manual,” 2016.

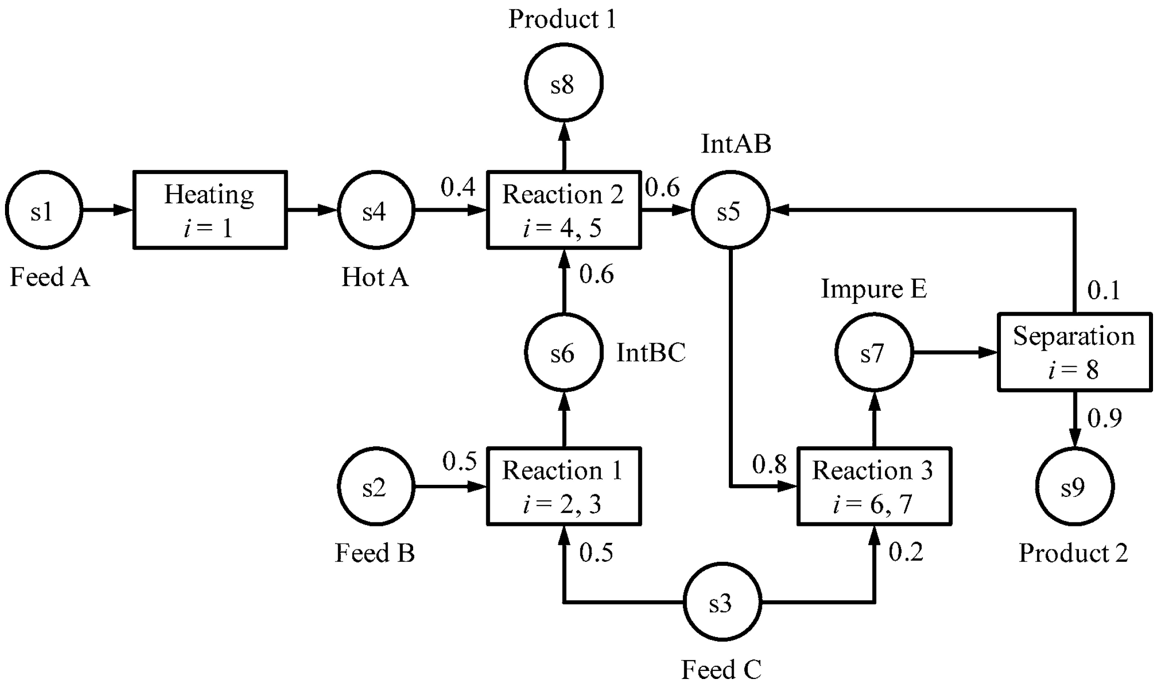
[49] GAMS Development Corporation, “General Algebraic Modelling System (GAMS),” 2013.

APPENDIX A

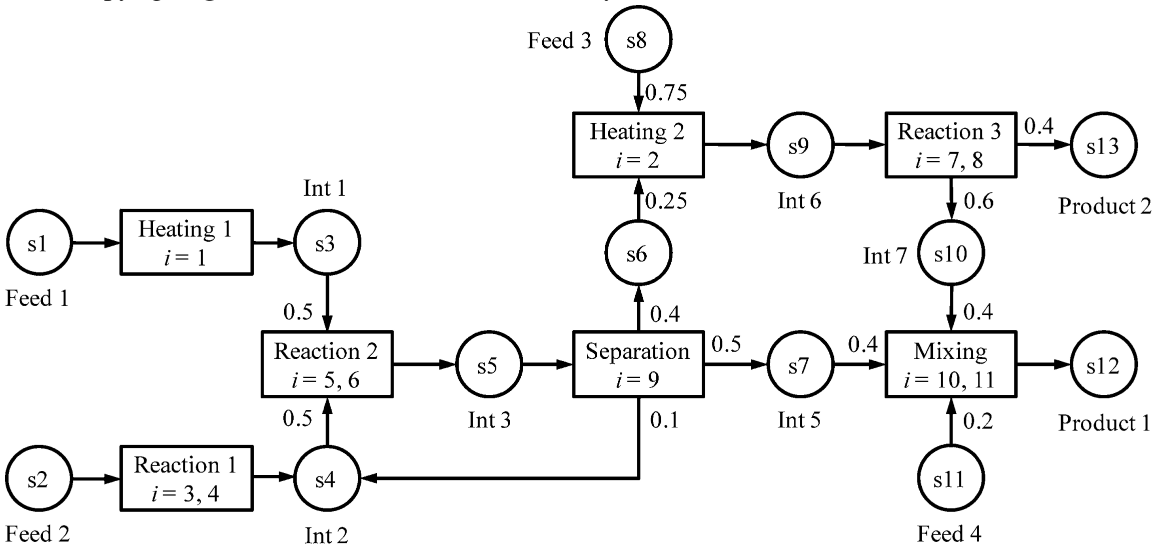
DATA FOR EXAMPLE PROBLEMS

Table A.1: Batch Size Data for motivating example and Examples 2-4

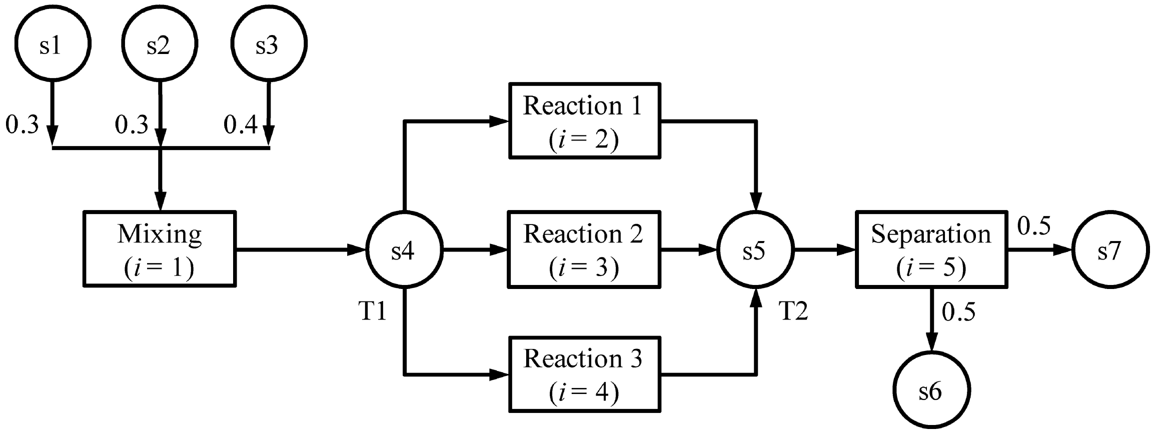
task i		**unit j**	α_{ij}	β_{ij}	$B_{ij}^L - B_{ij}^U$ (μ)
		Motivating Example			
heating	$(i = 1)$	heater	0.667	0.00667	0-100
reaction 1	$(i = 2)$	reactor 1	1.334	0.02664	0-50
	$(i = 3)$	reactor 2	1.334	0.01665	0-80
reaction 2	$(i = 4)$	reactor 1	1.334	0.02664	0-50
	$(i = 5)$	reactor 2	1.334	0.01665	0-80
reaction 3	$(i = 6)$	reactor 1	0.667	0.01332	0-50
	$(i = 7)$	reactor 2	0.667	0.00833	0-80
separation	$(i = 8)$	separator	1.3342	0.00666	0-200
		Example 2			
heating 1	$(i = 1)$	heater	0.667	0.00667	0-100
heating 2	$(i = 2)$	heater	1.000	0.0100	0-100
reaction 1	$(i = 3)$	reactor 1	1.333	0.01333	0-100
	$(i = 4)$	reactor 2	1.333	0.00889	0-150
reaction 2	$(i = 5)$	reactor 1	0.667	0.00667	0-100
	$(i = 6)$	reactor 2	0.667	0.00445	0-150
reaction 3	$(i = 7)$	reactor 1	1.333	0.01333	0-100
	$(i = 8)$	reactor 2	1.333	0.00889	0-150
separation	$(i = 9)$	separator	2.000	0.00667	0-300
mixing	$(i = 10)$	mixer 1	1.333	0.00667	20-200
	$(i = 11)$	mixer 2	1.333	0.00667	20-200
		Example 3			
task 1	$(i = 1)$	mixer	1.5	0	0-150
task 2	$(i = 2)$	reactor A	4.5	0	0-60
task 3	$(i = 3)$	reactor B	1.5	0	0-30
task 4	$(i = 4)$	reactor C	1.5	0	0-30
task 5	$(i = 5)$	separator	3.0	0	0-150
		Example 4			
task 1	$(i = 1)$	unit 1	1.666	0.03335	0-40
task 2	$(i = 2)$	unit 2	2.333	0.08335	0-20
task 3	$(i = 3)$	unit 3	0.667	0.0666	0-5
task 4	$(i = 4)$	unit 4	2.667	0.008325	0-40



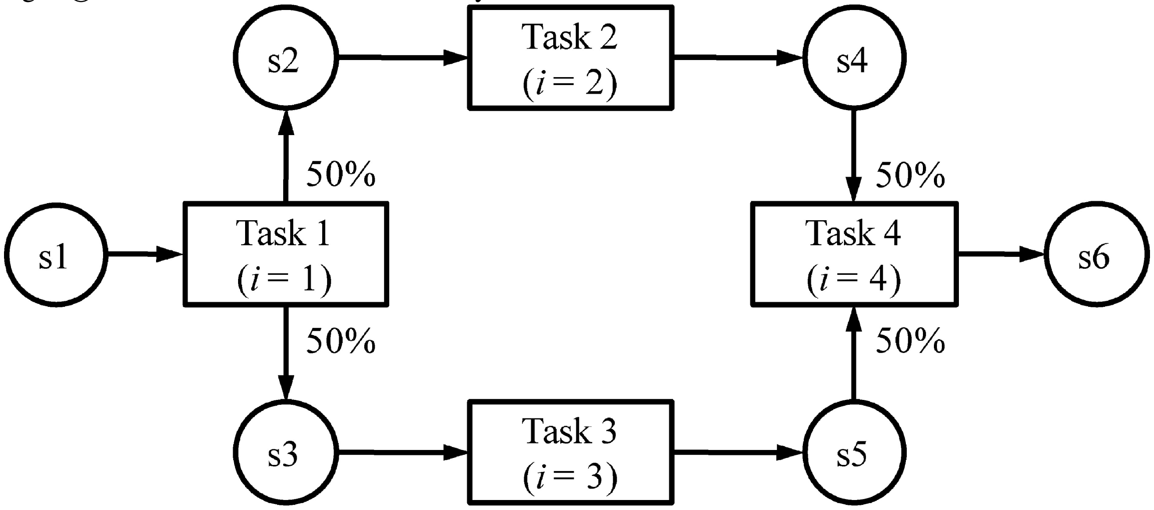
(a) STN of Motivating Example. Prices of s8 and s9 is 10 $/$\mu$. Reprinted with permission from [22]. Copyright ©2010 American Chemical Society.



(b) STN of Example 2. Prices of s12 and s13 is 5 $/$\mu$. Reprinted with permission from [22]. Copyright ©2010 American Chemical Society.



(a) STN of Example 3. Prices of s6 and s7 is 1 \$/μ. Reprinted with permission from [22]. Copyright ©2010 American Chemical Society.



(b) STN of Example 4. Prices of s6 is 10 \$/μ. Reprinted with permission from [22]. Copyright ©2010 American Chemical Society.