Solution Strategies in Short-term Scheduling for Multitasking Multipurpose Plants

by

Do Yeon Lee

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Chemical Engineering

Waterloo, Ontario, Canada, 2018

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Section 2.2, Section 2.4, and Chapter 3 of this thesis consist of a paper that was co-authored by myself, a former MMath student, Mr. Lagzi, and my supervisors, Dr. Ricardez-Sandoval and Dr. Fukasawa. This paper was published by the American Chemical Society in the Industrial & Engineering Chemistry Research journal on July 14, 2017, available online: https://pubs.acs.org/doi/abs/10.1021/acs.iecr.7b01718. S. Lagzi, D. Y. Lee, R. Fukasawa and L. Ricardez-Sandoval, "A Computational Study of Continuous and Discrete Time Formulations for a Class of Short-term Scheduling Problems for Multipurpose Plants". Mr. Lagzi performed part of the literature review of section 2.2, formalized the problem description of section 3.1, and formulated the short-term scheduling models of section 3.2. Mr. Lagzi's contributions are also incorporated in his 2016 MMath thesis, "A Study of Time Representation in a Class of Short Term Scheduling Problems". My main contributions include designing and performing the computational study of section 3.3, including the proposal of new discretization schemes presented in section 3.3.1. These contributions presented in section 3.3 are original and my own, even though though the developments overlap, otherwise. I was also responsible for final preparation and revisions of the published paper.

Section 2.3, Section 2.4, Chapter 4 and the Appendices of this thesis consist of a manuscript that has been submitted for publication. D. Y. Lee, R. Fukasawa and L. Ricardez-Sandoval, "Bi-objective short-term scheduling in a rolling horizon framework: *a priori* approaches with alternative operational objectives". Submitted to the Computers & Operations Research journal published by Elsevier on Feb. 14, 2018, under editor evaluation COR-D-18-00161. This manuscript was co-authored by myself and my supervisors, Dr. Ricardez-Sandoval and Dr. Fukasawa.

The abstracts, introductions and conclusions from both of the aforementioned manuscripts are also incorporated into the abstract, introduction and conclusion of this thesis.

Abstract

This thesis addresses challenges in short-term scheduling of multipurpose facilities using mathematical optimization. Such approach involves the formulation of a predictive model and an objective function, and the development of a solution strategy around such scheduling model formulation in order to obtain an operating schedule that achieves certain objectives, such as maximization of throughput or minimization of makespan. There are many choices that must be made in these aspects of short-term scheduling, and these choices often lead to a trade-off between the solution quality and computational time. This thesis presents two studies analyzing the quality-CPU time trade-off in two major aspects: time representations in model formulation, and the strategy for handling multiple conflicting objectives. The ultimate goal is to develop bi-objective short-term scheduling approaches to tackle industrial-sized problems for multitasking multipurpose plants that are computationally inexpensive, but provide practical schedules with a good balance between throughput and makespan.

The first study addresses the first aspect of interest and compares two different time representation approaches: discrete-time and continuous-time approaches. This comparison is made considering maximization of throughput as the sole objective. We show that, for the modeling framework implemented in this work, the selected discrete-time formulation typically obtained higher quality solutions, and required less time to solve compared to the selected continuous-time formulation, as the continuous-time formulation exhibited detrimental trade-off between computational time and solution quality. We also show that within the scope of this study, non-uniform discretization schemes typically yielded solutions of similar quality compared to a fine uniform discretization scheme, but required only a fraction of the computational time.

The second study builds on the first study and develops a strategy around an efficient non-uniform discretization approach to handle the conflicting objectives of throughput maximization and makespan minimization, focusing on *a priori* multi-objective methods. Two main contributions are presented in this regard. The first contribution is to propose *a priori* bi-objective methods based on the hybridization of compromise programming and the ε-constraint method. The second is to present short-term operational objective functions, that can be used within short-term scheduling to optimize desired long term objectives of maximizing throughput and minimizing makespan. Two numerical case studies, one in a semiconductor processing plant and an analytical services facility, are presented using a rolling horizon framework, which demonstrate the potential for the proposed meth-

ods to improve solution quality over a traditional *a priori* approach.

Acknowledgements

The financial supports provided by Natural Sciences and Engineering Research Council of Canada (NSERC), Ontario Centers for Excellence (OCE) and the industrial partner in analytical services sector are gratefully acknowledged.

I would also like to thank my supervisors Dr. Luis Ricardez-Sandoval of the Department of Chemical Engineering and Dr. Ricardo Fukasawa of the Department of Combinatorics and Optimization for their guidance and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With improving hardware technologies and computation techniques, industries are increasingly looking to implement decision making techniques based on data and advanced optimization-based methods. Short-term scheduling of operations based on mathematical optimization is a field that has been gaining traction in the Process Systems Engineering (PSE) community[21;89;95;108;132]. Such approach involves the use of short-term scheduling models to determine the optimal timings of operations subject to operational objectives and constraints. This approach can effectively streamline operations of complex systems and has found applications in various fields of interest to chemical engineers[101], such as food processing[36;64;84], oil refinery operations[78;109;121], and pharmaceutical manufacturing[83;129].

The main problem of interest in this thesis is short-term scheduling of an actual *multipurpose plant* in the analytical services sector. A *multipurpose plant* is a facility that requires different *processes* (e.g., drying, filtration) to carry out various *tasks* (e.g., manufacturing a product, performing certain analysis for a client), and different tasks may follow different *paths* (sequence of processes, e.g., production path, recipe) through the network of processes[101]. Given their prevalence in various industries, optimal scheduling of multipurpose plants have received significant attention from researchers in the Operations Research and PSE communities[19;31;47;54;74;76;86;87;89;97;101–103;120;131;133]. The analytical services sector is a major sector in which various types of analysis are carried out on a set of samples to determine its properties and chemical composition, which can be used in the decision-making process by end-customers in various industries. For example, composition analysis of meat[52] is used by food processing companies to produce realistic meat analogues[44]. Analytical services facilities may receive samples in the order of thousands to

be processed through a complex system with over a hundred processing units, each processing unit containing multiple machines to carry out processes (e.g., heaters, filtration units), on a daily basis with operational features such as material splitting (a subset of samples from a task may be processed in any given machine) and multitasking (a machine may process samples from several different tasks simultaneously). Furthermore, short-term scheduling models provide operational level day-to-day decisions, and often solutions need to be provided within a short amount of time, using only information available up to a limited time horizon, such as availability of raw materials and resources. One challenge that arises from these situations is that the short amount of time available for a decision maker (DM) to take a decision means that one needs to carefully consider that, whatever approach is considered, it must not take too long to provide a solution. Therefore, there is a need to develop practical approaches that can be effective without taking too much computational time.

Central to solving any process optimization problem is the formulation of the optimization model, which has three key elements[11]:

- An objective function that needs to be maximized, e.g., profit, *throughput* (the amount of material passing through a network of processes over a specific period of time), or minimized, e.g., pollutant emissions, *average makespan* (the average completion time of a task on the last process of its path over a specific period of time).
- A predictive model that describes the system using a set of constraints (equations and inequalities), e.g., material balances and capacity constraints.
- A set of decision variables that appear in the objective function and the predictive model, e.g., the batch size and timing of a unit operation. If a set of values for the decision variables satisfy all constraints, it is considered to be a feasible solution; and if the value of the objective function for a feasible solution is the minimum among all feasible solutions, then it is also the optimal solution.

In short-term scheduling applications, formulations are dominantly based on mixed-integer linear programming (MILP)[11;132] due to their rigorosity and flexibility[75;129]. In an MILP formulation, the objective function and all constraints are linear, and one or more decision variables are integer or binary variables[40]. Such MILP models can be solved efficiently using modern branch-and-bound based solvers such as CPLEX[71]. Dominance of linearity in short-term scheduling formulations is in contrast to other common

applications of mathematical optimization in chemical engineering, such as process design and control[11;80;81;122;135] given that predictive models in chemical engineering often exhibit nonlinear behavior (e.g., blending of oil products[109]). Maintaining linearity of scheduling models as MILP formulations through various methods such as problem simplication and linear approximation/reformulation is of great interest, as this helps to reduce computational complexity for finding a global optimal solution[15;47;75;129;135]. Given that tractability is a key issue in short-term scheduling, this thesis also focuses on MILP formulations, and attention is given to maintaining linearity of the objective function and the predictive model. Fortunately, the problem of the analytical services facility naturally leads to a MILP formulation assuming that all processing times are constant and known in advance.

There are several choices which must be made in the implementation of MILP scheduling models, which may have significant effects on both the quality (the values of the objective function) of the schedules and the time needed to obtain these schedules. This thesis addresses two major challenges: the selection of a time representation approach, and the expression and handling of multiple conflicting objectives in short-term scheduling for an analytical services (multipurpose) facility.

For time representations, the two most common approaches are: discrete-time approaches, which prespecify a finite set of time points at which scheduling decisions may occur, and continuous-time approaches, in which the optimization model determines the positions of time points through the use of continuous decision variables. Furthermore, when using a discrete-time approach, the DM must choose between using a uniform or non-uniform discretization scheme, and decide on the spacing between the time points. While a discretization scheme with finely and uniformly spaced time points can provide solutions of higher quality when compared to using a coarsely and uniformly spaced time points, the fine uniform discretization scheme may require considerably longer time to obtain a solution. On the other hand, a non-uniform discretization scheme aims to provide a good balance between the solution quality and the CPU time by using a fine discretization only when necessary.

Similarly, when using the continuous-time approach, the DM must choose between using a global time scale (analogous to using a uniform discretization scheme) or unit-specific time scales (analogous to using a non-uniform discretization scheme), and decide on the number of time points to use. A unit-specific continuous-time formulation aims to provide a balance between using a global time scale using a high number of time points and using a global time scale using fewer time points. In general, using more time points increases the

quality of the schedule for both discrete and continuous time approaches as timings of operations can be determined with greater details and more precisely; however, this leads to larger MILP formulations and higher computational demands. Furthermore, while the use of continuous variables to represent time may provide higher precision than using prespecified time points, the constraints required to describe such continuous variables may lead to a more complex formulation. Therefore, balancing of tractability and solution quality is a key challenge in selecting an appropriate time representation approach in the model formulation. For this purpose, this thesis performs computational studies comparing the discrete-time and the continuous-time approaches incorporating new developments in modeling methodologies and operational characteristics, such as multitasking, not addressed by other similar studies[110;129;132] available in the literature.

In addition to selecting an appropriate model formulation approach, it is important for the DM to build an appropriate solution strategy around the model formulation. One common challenge is that, for a business, there are multiple conflicting objectives, which must be balanced in order to maximize their profit. In most cases, it may be desirable to solve scheduling problems with profit maximization as the only objective[47;108;132;141]. However, modeling the profit of each operation in the entire system may be challenging and insufficient data may be available to write a single profit function for the entire system. In such cases, the solution strategy must focus on secondary objectives such as maximization of the total throughput (TTP) and minimization of the average makespan (AMS), which are in conflict of each other (i.e., a schedule providing the maximum feasible throughput does not provide the minimum feasible makespan, and vice versa). This issue of conflicting objectives can be addressed by selecting one single objective function to solve for, or by combining objectives into a single function with arbitrarily selected weights. However, it can be advantageous to use an appropriate multi-objective optimization method, which can provide the DM with a greater control over the quality of the trade-off between the conflicting objectives, and guarantee the efficiency of the trade-off[26]. In keeping with the theme of this thesis, the implementation of such multi-objective approach and the associated benefits have to be considered taking into account any additional computational time required.

Moreover, another challenge often found inside organizations is that TTP and AMS are objectives that are realized over a longer period of time (e.g., weeks) than what is considered in short-term scheduling (e.g., days or hours). For instance, a sequence of operations to be scheduled may not have any measurable AMS or TTP after several days, which would in turn make it no better than just scheduling no operations (from the point of view of the optimization model that only considers a limited time horizon, e.g., one day of oper-

ations). Thus, there is a need to consider short-term objective functions that are not the actual TTP or AMS, but instead objective functions designed with the aim of achieving high TTP or low AMS over multiple periods of operation using a rolling horizon framework.

Given the challenges identified above and in the introduction, the objectives of the thesis are as follows:

1. Study the strengths and weaknesses of the different time representation approaches relative to each other, and by doing so, determine the favorable approach to solving industrial-sized short-term scheduling problems for a multipurpose plant where multitasking is a key operational feature.

2. Present multi-objective algorithms that can provide practical solutions to multi-objective problems that emerge in short-term scheduling.

3. Study objective function formulations that can account for TTP and AMS in short-term scheduling to address the issue of the difference in time scales between short-term scheduling (e.g., days or hours) and the long-term realization of TTP and AMS (e.g., weeks).

The remainder of this thesis is structured as follows: In Chapter 2, literature review and background are provided for time representations and multi-objective optimization methods, as well as gaps in literature, the objectives for this thesis. The study on time representations is presented in Chapter 3, which includes the description of the operations of a multipurpose plant and presentation of the multitasking MILP scheduling formulations and proposed time discretization schemes. The study on bi-objective short-term scheduling is presented in Chapter 4, which includes presentation of new bi-objective methods and proposed alternative objective functions to approximate TTP and AMS for the use in a rolling horizon framework. Finally, concluding remarks and potential future research directions are provided in Chapter 5.

Chapter 2

Background and Literature Review

This chapter provides the necessary background and literature reviews for the issues discussed in this thesis. First, a general background on the different types of optimization problems and the techniques for solving these problems is provided in Section 2.1. Literature reviews on the two main issues addressed in this thesis are presented in Sections 2.2 and 2.3, discussing time representations in short-term scheduling formulations and multi-objective optimization approaches. In these sections, and in fact, all throughout this thesis, there exists a recurring theme of trade-off between computational cost and solution quality, which should be kept in mind. In the following literature reviews, specific challenges with regards to short-term scheduling applications are identified, and the gaps in literature with respect to these challenges are identified in section 2.4.

2.1 Problem classifications and optimization techniques

Optimization problems can largely be classified according to two different characteristics: the existence or the absence of nonlinearity in the objective function and the constraints, and whether or not there are binary or integer variables in the model formulation. A problem where the objective function and the constraints are linear and only continuous variables exist (i.e., no binary or integer variables) is called a linear program (LP). A problem with one or more nonlinear constraints and/or a nonlinear objective function is called a nonlinear program (NLP). If one or more of the decision variables are discrete variables (binary or integer), a linear or nonlinear problem is referred to as a mixed integer linear program (MILP) or a mixed integer nonlinear program (MINLP). These classifications are

Table 2.1: Classifications of optimization problems

Formulation linearity	Type of decision variables		
	All continuous	Mixed continuous/discrete (Mixed Integer Programs - MIP)	All discrete (Integer Programs - IP)
Linear	Linear Program (LP)	Mixed Integer Linear Program (MILP)	Integer Linear Program (ILP)
Nonlinear	Nonlinear Program (NLP)	Mixed Integer Nonlinear Program (MINLP)	Integer Nonlinear Program (INLP)

summarized in Table 2.1 [11;40]. Different types of problems require different types of optimization techniques, and these techniques are used by solvers (computer programs) to solve optimization problems with varying degrees of effectiveness.

The techniques for solving LP problems, such as the simplex algorithm and barrier methods are among the most widely used and effective optimization techniques [40]. In fact, branch-and-bound methods, which are the most widely used techniques for solving mixed integer linear programs (MILP), repeatedly call on LP techniques while solving a MILP. In the first step of branch-and-bound for solving a MILP problem, all discrete variables are assumed to be continuous, i.e., the MILP is converted into a LP; this is called *LP relaxation* and this initial LP problem is called the *root node*. This root node relaxation can be solved using a LP technique, and if the optimal LP solution to the root note meets all integrality requirements (i.e., *integer feasible*), then the optimal LP solution is the optimal solution to the original MILP problem. Otherwise, the optimal LP objective value is considered to be the *best bound*, and *branching* occurs at the root node. In the branching step, a variable with a fractional value is chosen, and two subproblems called *nodes* are created imposing that the fractional value is a number less or greater than the nearest integer values. For instance, if a variable x_j has value 2.3, then one of the subproblems would impose that $x_j \leq 2$, while the other imposes that $x_j \geq 3$. If a subproblem is optimal and integer feasible, then the integer optimal solution is compared to the best feasible solution found so far and, if better, it is considered to be the *incumbent*, and its objective value is the *best integer*; this node is then considered to be *fathomed* and no longer branched on. If a subproblem is either infeasible or has a worse objective value than the incumbent, it is eliminated from the search (these nodes are also fathomed); otherwise, this node must be further branched on. At any point during the search, for a minimization problem, the

best bound is the minimum of the LP optimal objective values (the maximum for a maximization problem) of all unfathomed nodes[58]. The *optimality gap* may be calculated as the difference between the current best bound and best integer. The branching process in this algorithm continues, creating a search tree of nodes over time, until the optimality gap falls below a tolerance criteria[40;73]. The incumbent (the integer feasible solution with the best objective value on the search tree) at the time of termination is considered the optimal solution to the original MILP.

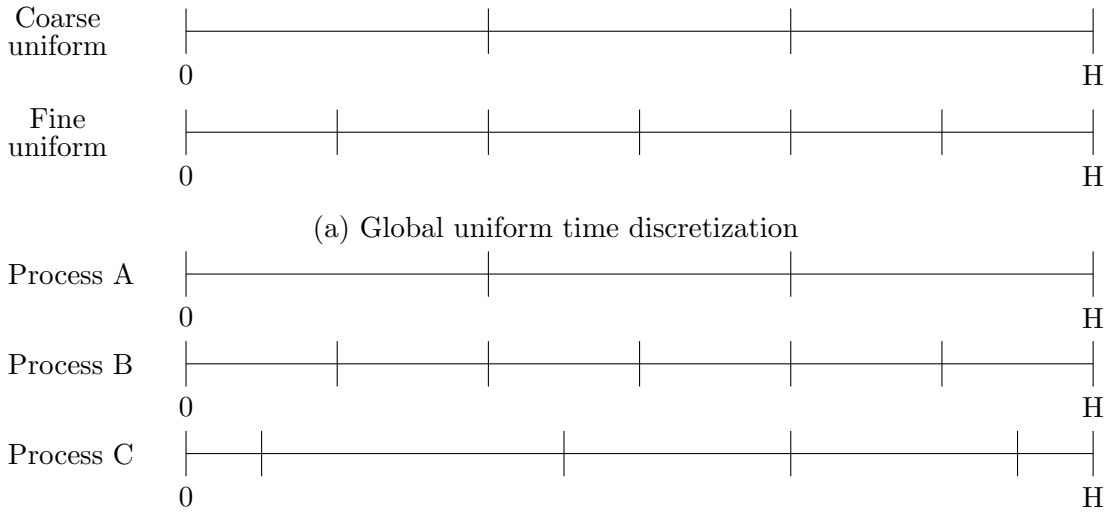
Given the dependence of the branch-and-bound technique for solving a MILP on solving multiple LP subproblems, solvers inherently require more time and memory to solve a MILP than to solve its LP counterpart. Different MILP formulations also require different amounts of time to solve depending on the number of variables, the problem structure, and data[40]. Furthermore, the optimality gap at the root node relaxation may be used as an indicator for how computationally demanding a MILP formulation may be to solve. In general, a formulation leading to a small optimality gap at the root node relaxation (also said to be a *tight* formulation) is computationally less expensive when compared to a formulation leading to a large optimality gap at the root node relaxation. Therefore, modern commercial solvers, such as CPLEX, attempt to tighten the formulation by adding *cuts*, starting with the root node. A *cut* is a constraint added to the model, which rules out certain fractional solutions without eliminating legal integer solutions, and adding cuts helps prevent the search tree from getting too large[73].

Despite these challenges in solving MILP problems, short-term scheduling formulations are still predominantly MILPs[11;132], as opposed to NLPs and MINLPs as the techniques for solving such nonlinear problems, such as general reduced gradient methods, face unique challenges that affect the ability of nonlinear solvers to obtain and recognize a solution. For example, an inaccurate estimation of first derivatives, required by all major NLP algorithms, can lead to a very slow algorithmic progress, or in extreme cases, declare a suboptimal solution to be the optimal solution[40]. Solving MINLPs tend to be even more challenging given that the branch-and-bound methods for solving MINLPs depend on solving NLP subproblems. Therefore, it is generally desirable to try to avoid solving nonlinear problems when possible even if the problems of interest contain inherent nonlinearity[15;47;75;129;135].

2.2 Time representations for operations scheduling models

Over the past few decades, several discrete-time formulations have been proposed, in which scheduling decisions, such as the beginning of sample processing by a machine, can only occur on a finite set of prespecified time points. A general discrete-time MILP formulation for short-term scheduling of batch operations in a multiproduct/multipurpose plant was proposed by Kondili et al.[82], which included batch operations and material/products explicitly as network nodes in a *State-Task Network (STN)*. Computational issues surrounding this approach were discussed by Shah et al.[128]; Pantelides[118] presented the more general *Resource-Task Network (RTN)* representation as a unified framework. The STN representation is still widely used today. Patil et al.[120] presented an Integer Linear Program (ILP), a special case of MILP with only integer decision variables, based on a uniform discrete-time formulation adapted from the STN representation based on flow conservation equations. This ILP formulation by Patil et al.[120] included the multitasking feature, which is a key operational feature for the problems considered in this thesis.

Traditionally, as Velez and Maravelias[140] have noted, discrete-time models only represented time on a single global time grid with uniformly spaced time points where events may occur[8;82;118;120;128] with the time grid being shared between all processing units. These uniform discrete-time formulations could result in MILPs involving large numbers of binary variables, leading to large computational demands[82;128]. To address this issue, Velez and Maravelias[140] proposed a flexible discrete-time formulation with multiple and possibly non-uniform time grids. In a follow up study, Velez and Maravelias[141] generalized their ideas into a general framework for developing such scheduling models, and further explored the algorithms for generating the non-uniform time grids. In this approach, each processing unit is given its own discretized time grid, and events can happen at different times for different processing units, effectively allowing many decisions to be made for some of the units without defining unnecessary time points for the rest of the process network. Furthermore, when using multiple time grids, each time grid does not necessarily need to have uniformly spaced time points. Such an approach has the advantage of obtaining a better balance between solution quality and computational demand, since finer discretizations are only used when needed, for example, for processing units with relatively short processing times. Velez et al.[142] successfully implemented the multi-grid approach to solve three large examples, including a case study based on production operations by the Dow Chemical Company, while extending the approach to consider various features including changeovers and storage policies. More recently, Lagzi[85] extended the multitasking ILP



(a) Global uniform time discretization

(b) Unit specific flexible non-uniform time discretization

Figure 2.1: Different discretization approaches (predefined time points)

formulation of Patil et al.[120] to include the flexible discretization scheme of Velez and Maravelias[140]. This ILP multitasking flexible discrete-time formulation implemented in this thesis using both the uniform discrete (UD) and the non-uniform discrete (NUD) time representations.

Both of these time discretization approaches are illustrated in Figure 2.1. In this figure, the length of each time grid represents the prespecified length of the scheduling horizon H starting at time 0. Each tick mark represents the prespecified location of each time point at which an event (e.g., turning on a machine) may occur. Figure 2.1a shows two different approaches of uniform time discretization where events for all processes occur on either a shared coarsely or finely discretized time grid with uniformly distributed time points. Figure 2.1b, on the other hand, shows the approach by Velez and Maravelias[140] where events for each processing unit occurs on its unit-specific time grid with possibly non-uniform time point distribution; events across all unit-specific time grids are then scheduled simultaneously by the optimization algorithm.

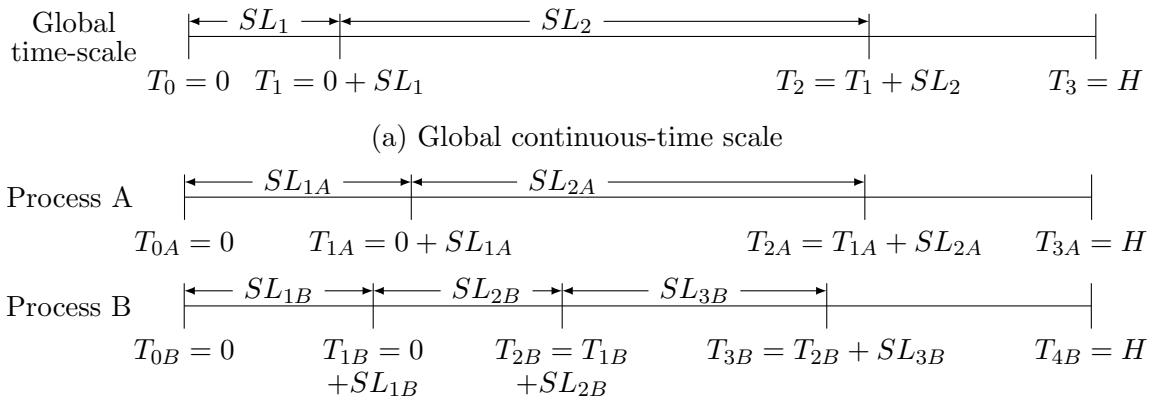
One important aspect of the discrete-time formulations is that a very fine discretization (e.g., every minute) allows for a high degree of flexibility in the solutions, possibly resulting in higher quality solutions compared to a coarser discretization. However, such

a fine discretization requires a larger number of decision variables, and results in larger (and more computationally intensive) optimization problems to solve. Thus, there exists a trade-off between computational time and solution quality.

A different time representation approach only predefines the number of time points, and the optimization algorithm determines the optimal positions of the time points (i.e., the times within the scheduling horizon at which scheduling decisions may occur) as continuous decision variables. The formulations implementing this approach are referred to as continuous-time formulations. Events may either occur on a single global time scale for all units[19;57;87;90;95;102;107;112;126;131;143;154], or on multiple unit-specific time scales[20;51;54;74;76;97;113;133]. Recently, Lagzi et al.[86] adapted the flow conservation equations of Patil et al.[120] and the continuous-time formulation of Sundaramoorthy and Karimi[131] and presented a MILP multitasking global continuous-time formulation. However, a multitasking unit-specific continuous-time formulation has not yet been developed, and not included in the scope of this thesis. Regardless, these two approaches of continuous-time representation are illustrated in Figure 2.2 for a predefined scheduling horizon length of H (i.e., H is an input parameter).

In Figure 2.2, variable T_n represents the position of the n$^{\text{th}}$ time point, variable SL_n represents the distance between the nth and $(n-1)^{\text{th}}$ time points, and the second alphabetical index for SL_{nX} in Figure 2.2b refers to the specific processing unit X. The values of these continuous variables, i.e., the locations of the time points, are determined by the optimization algorithm to provide the optimal objective value. The global continuous-time scale presented in Figure 2.2a is analogous to the uniform discretization approach presented in Figure 2.1a in that events for all processes occur on a shared time scale. On the other hand, the unit-specific continuous-time scales are analogous to the flexible non-uniform time grids presented in Figure 2.1b.

Since the timing of events is not predetermined in continuous-time formulations, more accurate solutions may be obtained without considerably increasing the size of the formulations. Furthermore, the continuous-time formulations are capable of modeling some operational features, such as variable machine processing times, with relative ease, which are challenging for discrete-time formulations. The flexibility provided by continuous-time approaches also make them suitable for developing frameworks for scheduling under uncertainty, as Lappas and Gounaris[88] have demonstrated by successfully developing an adjustable robust optimization (ARO) framework utilizing the global event-based model by Castro et al.[19]; Wang et al.[144] developed a rescheduling framework for crude-oil oper-



(a) Global continuous-time scale

(b) Unit-specific continuous-time scales

Figure 2.2: Different continuous-time approaches (time points are decision variables)

ations under uncertainty utilizing the unit-specific model by Furman et al.[51]. Similar to the discrete-time approach, however, there exists a trade-off between computational time and solution quality as higher number of time points are required to improve solution quality. There also exists an upper limit of the number of time points for a continuous-time formulation for a given problem beyond which no improvement in solution quality can be observed[19;86].

While discrete-time and continuous-time formulations have often been studied separately, few studies have considered a comparison between these two approaches[110;129;132]. Stefansson et al.[129] compared these two approaches with a case study based on a multi-stage, multi-product pharmaceutical production facility. They solved their strongly NP-hard problem by decomposing the problem into two parts, and solving the individual parts using discrete-time and continuous-time formulations. Stefansson et al.[129] found the continuous-time approach to be more suitable in solving their problem as it required less time to solve, and the resulting schedules were more precise. Sundaramoorthy and Maravelias[132] compared these two approaches using a collection of more than 100 problem instances. They found that while continuous-time formulations appeared to solve faster than their discrete-time counterparts for relatively short scheduling horizons, the performance of the continuous-time formulations suffered significantly with increasing length of horizons and increasing number of time points. On the other hand, the performance of discrete-time formulations was found to be relatively consistent. Sundaramoorthy and Maravelias[132] also found the discrete-time formulations to consistently provide better so-

lutions. Merchan et al.[110] compared four discrete-time models against a continuous-time model using a total of 1808 runs, and found that all the proposed discrete-time models outperformed the continuous-time model as the discrete-time models required significantly less time to solve.

2.3 Multi-objective methods

When there exists more than one objective function that must be minimized in an optimization problem, it is said to be a multi-objective optimization problem. The general form of such problem is $\min\{f_i(\vec{x}) : \vec{x} \in \chi \mid \forall i = 1, \ldots, k\}$, where k is the number of objective functions, $\vec{x}$ is a decision vector of n variables, and $\vec{x}$ is the feasible set of constraints. The main concern in a multi-objective optimization problem is that the objectives may be conflicting with each other, i.e., improving the objective value in one objective function may deteriorate the objective values in other objective functions. The first attempt to methodically address the issue of conflicting objectives is usually attributed to Pareto[119], who introduced the concept now referred to as *Pareto optimality*. In short, a solution is said to be efficient or *Pareto optimal* if there exists no solution that is strictly better for at least one objective and at least as good for all other objectives. A set of *Pareto solutions* forms the *trade-off surface* or the *Pareto set*, also called the *Pareto front* or the *Pareto frontier*. A more formal definition of *Pareto optimality* is as follows[111]:

Definition 2.3.1. A decision vector $\vec{x}^* \in \chi$ is *Pareto optimal* if there does not exist another decision vector $\vec{x} \in \chi$ such that $f_i(\vec{x}) \leq f_i(\vec{x}^*) \forall i = 1, \ldots, k$ and $f_j(\vec{x}) < f_j(\vec{x}^*)$ for at least one index j.

One way to handle problems with conflicting objectives is to express the various objectives in a single objective function as a sum of several weighted objectives, i.e. the weighted sum method[149]. The weights may, for example, be chosen arbitrarily based on experience, or by trial and error. While this approach may be straight forward to implement and to solve, the decision maker (DM) may face difficulties in objectively evaluating the quality of the solution with regards to the conflicting objectives. Furthermore, the arbitrary selection of the weights may introduce bias in the decision making process, which deviates from the preferences of the DM. Furthermore, the weighted sum method is not guaranteed to return a Pareto optimal solution for a nonconvex problem[26]. This is undesirable for the problem of interest for this thesis given that MILP problems are inherently nonconvex[40].

Various other multi-objective optimization techniques address the shortcomings of the traditional weighted sum method by providing the DM with a set of efficient solutions, or by providing the DM with greater flexibility and control over the trade-off between the conflicting objectives. The multi-objective techniques typically fall under two main categories: *a priori* and *a posteriori* methods. *A posteriori* methods aim to provide a DM with a set of Pareto optimal solutions. The DM can then afterwards look at the set of Pareto solutions provided and select one that complies with the DM's needs. Currently, the most dominant *a posteriori* methods in the multi-objective optimization research community are approximation methods based on heuristics and metaheuristics. These methods are typically inspired by behaviours and mechanisms in nature, e.g., biology, evolution, and physics [27]. The popularity of such methods are evident in several reviews of multi-objective methods [25;27;53;91;130;146;156], including those focusing on scheduling problems [25;53;91;130;146] (e.g., Yenisey and Yagmahan [146] noted that more than 80% of the articles reviewed in their survey dealt with heuristic and metaheuristic and approaches). These metaheuristics methods include variants of Evolutionary/Genetic Algorithms (EA/GA) [17;23;66;116;127;136], Simulated Annealing (SA) [39;98], Tabu Search (TS) [43;123], Ant Colony Optimization (ACO) [9;67;92], and Particle Swarm Optimization (PSO) [153]. The metaheuristics methods are useful for solving "hard" optimization problems, such as "NP-hard" combinatorial problems, such as integer programs (IP) including MILP problems [26]. Furthermore, EAs are particularly well suited to generating a set of non-dominated solutions due to the possibility of searching for several solutions concurrently [49], then allowing the DM to choose *a posteriori* a solution to implement. However, these methods are stochastic in nature, as they employ stochastic search methods and the quality of solutions depend on the initial population and this stochastic process can negatively impact computational efficiency [26]. Given that the DM often has a limited amount of time to make a decision in short-term scheduling applications, such *a posteriori* methods may not be suitable in such cases, and obtaining a set of alternative solutions may be unnecessary or impractical. Since this thesis aims at providing practical solutions in reasonable computational times, this study will focus on *a priori* methods. Comprehensive reviews on multi-objective methods can be found elsewhere [25–27;41;53;91;105;130;146;156].

In contrast to *a posteriori* methods, the DM's preferences are selected in advance in *a priori* methods; hence, a single trade-off solution is obtained after a unique search [26;41], rather than a set of solutions for the DM to evaluate. Reference point methods, such as compromise programming [147], are very popular *a priori* methods that have been widely used in practice [16;45]. The benefit of compromise programming lies in that it returns the compromise solution that best approximates an ideal unattainable solution [16;99]. The ε-

constraint method, first proposed by Haimes et al.[61] is another method that has enjoyed popularity due to its relative simplicity[24;26]. In this method, a prioritized objective is optimized with other objectives being transformed into bounding constraints to guarantee the minimum/maximum values (ε) of the non-prioritized objectives. In other words, the original multi-objective problem is transformed into a single-objective problem that can be solved using efficient MILP solvers such as CPLEX. While the ε-constraint method is typically classified as *a posteriori* method (although this is not based on heuristics or metaheuristics typical of other popular *a posteriori* methods)[24;35;111], there are studies that have classified it as *a priori* method[27;105], as the DM's preferences can be expressed in the choice of the prioritized objective, and the selection of the ε values. Similar to other *a posteriori* methods, the ε-constraint method as *a posteriori* method suffers the drawback of being computationally demanding due to its iterative nature[26]. Therefore this thesis focuses on the *a priori* aspect of the ε-constraint method in this thesis. This thesis, however, does not utilize the original ε-constraint method by Haimes et al.[61] as this method is not guaranteed to return a non-dominated solution in certain cases involving integer decision variables. To address this issue, Özlen and Azizolu[117] presented a general method for finding all non-dominated solutions within the output efficiency range for a multiobjective integer programming problem with integer objective values by successively solving constrained weighted single objective integer programming problem with progressively changing ε bounds. This thesis utilizes this variation of ε-constraint method by Özlen and Azizolu[117] in order to propose new practical *a priori* methods through its hybridization with the compromise programming method of Yu[147].

2.4 Gaps in literature and thesis contributions

In performing literature reviews as presented in Sections 2.2 and 2.3, several gaps in literature were identified, providing objectives for the studies presented in this thesis.

First, on the issue of time representations, relatively few studies have directly compared the discrete-time and the continuous-time formulations[110;129;132]. Nevertheless, these comparison studies either did not consider the situation where the discrete-time formulation is capable of accommodating flexible time discretization[129;132], or the continuous-time formulation is unable to handle material splitting[110;129] or multitasking[110;129;132]. These gaps in literature are addressed in Chapter 3 through computational studies comparing the flexible discrete-time and the global continuous-time formulations presented by Lagzi[85],

which incorporate these features. Small and industrial-sized instances are considered with discrete batches, material splitting, and multitasking, where batches are not allowed to mix, and the machines are single purpose machines with constant and identical processing times for all compatible tasks. These instances are based on an actual analytical services facility. Chapter 3 also presents heuristics for non-uniform discretization using the flexible discrete-time formulation with the aim of balancing the solution quality and the computational demands.

Second, on the issue of multiple conflicting objectives, researchers in the multi-objective optimization community are predominantly focused on methods based on heuristics and metaheuristics. Despite having enjoyed practical popularity and advances in methodologies[99;117] there have been relatively few studies on scheduling problems that have applied reference point methods[3] or the ε-constraint method[22;59;148] in recent years. Allouche et al.[3] solved a small job shop scheduling problem (5 jobs, 2 machines) using compromise programming to simultaneously consider the makespan, the total flow time, and the total tardiness. Gutierrez-Limon et al.[59] applied the ε-constraint method to solve a problem in simultaneous scheduling and control of a single reactor with 3 to 5 products with potentially simultaneous reactions to manage the trade-off between economic profits and dynamic performance. Both Yue and You[148] and Castro et al.[22] applied the ε-constraint method for problems in batch plant scheduling with 4-11 products and 3-14 stages, respectively. While Yue and You[148] considered the economic and environmental objectives, Castro et al.[22] considered the makespan and the total utility demand. A common factor in those studies is that the size of the problems considered are relatively small. Furthermore, to the authors' knowledge, a multi-objective short-term scheduling study that considers a rolling horizon approach is not available. These issues are addressed in Chapter 4 where *a priori* multi-objective algorithms based on the hybridization of the ε-constraint method and compromise programming are presented with the aim to provide practical solutions to multi-objective problems in short-term scheduling. The performances of these new approaches are compared against the traditional compromise programming approach. Computational studies are performed using a rolling horizon framework, testing the performances of several alternative objective functions that account for TTP and AMS in short-term scheduling to address the issue of the difference in time scales between short-term scheduling (e.g., days or hours) and the long-term realization (e.g., weeks) of these conflicting objectives.

Chapter 3

Study of Time Representations

This chapter presents the study comparing the different time representation approaches (discrete-time and continuous-time) discussed in Section 2.2. The aim of this study is to use existing discrete-time and continuous-time formulations in the literature, which are able to readily handle the operational details as described in the problem definition of Section 3.1 without major modifications. These are the ILP flexible discrete-time formulation and the MILP global continuous-time formulation by Lagzi[85], which were developed by combining existing ideas in the scheduling literature. These formulations were also presented in the published manuscript[87], for which the author of this thesis is a co-author; they are presented again in this thesis for completeness and to reflect some of the minor changes that were made from the original formulations presented by Lagzi[85]. Note that while this thesis presents these scheduling models as ILP and MILP formulations, some of the integer decision variables can be relaxed as continuous decision variables in order to handle a variety of problems using these formulations. For example, tasks do not need to be composed of strictly discrete samples or materials (e.g., a task can be composed of 4.7 kg of a liquid reagent).

While Lagzi[85] focused primarily on the development of model formulations, the study presented in this chapter represents a much more rigorous and extensive computational study comparing the performances of these formulations using a total of 190 small and industrial sized instances, comprised of 1030 runs with 25 processing units and up to 200 tasks based on data from the actual analytical services facility. In contrast, Lagzi[85] presented 12 instances with 3-8 processing units and 2-9 tasks, where the data are not based on the actual analytical services facility. The 25 processing units considered in this study retain the characteristics of the majority of the operations at this facility; however, not

all units at this facility were considered in this chapter due to tractability concerns for the computationally demanding approaches and instances. This chapter also presents and compares heuristics for discretization using the flexible discrete-time formulation, which were not considered by Lagzi[85], with the aim of balancing solution quality and CPU time.

This study focuses on the single-objective optimization approach, and serves as the basis for the study on multi-objective methods presented in Chapter 4. A multi-objective approach is not considered for this study given that such comparison may be too computationally demanding. This study focuses on throughput maximization assuming that for this particular problem, throughput maximization is the sole key priority in the absence of an appropriate multi-objective framework.

The rest of the chapter is organized as follows: First, the problem definition and terminology describing the operations of the multipurpose facility are provided in section 3.1. Then, the flexible discrete-time and the continuous-time formulations for the problem are presented in Sections 3.2.1 and 3.2.2, respectively. Computational results and discussions are presented in Section 3.3, along with proposed discretization schemes for the flexible-discrete time formulation. Finally, a chapter summary in Section 3.4.

3.1 Problem definition

The main problem that will be considered in this thesis is based on an actual industrial analytical services facility; this problem is studied in both Chapters 3 and 4. The facility receives a set of tasks (a task may represent a single request from a client to perform certain analysis on some materials), I, and needs to process them within a scheduling horizon, H, using a set of processing units, P. Each processing unit, $p \in P$, consists of a set of identical machines (e.g., heaters, filtration units), J_p, that perform a specific process (e.g., drying, filtration). The set of all machines in the facility is denoted by J, where J is the disjoint union of J_p's.

For task $i \in I$, a_{ik} number of samples (e.g., in discrete containers such as test tubes) become available at the beginning of the scheduling horizon for process $p_k^i \in P$, which need to visit a specific sequence of processing units, called a path. For example, a sample may be some amount of meat or liquid mixture, for which a client would like to know its

chemical compositions. The path of tasks i, denoted by $\wp_i$, is a sequence of $n(i)$ distinct processing units $\{p^i_1, \ldots, p^i_{n(i)}\}$, where $p^i_k \in P$ for all $i \in I$, $k = 1, \ldots, n(i)$. Within the context of this work, samples for a task may be introduced at any processing unit in the path at the beginning of the scheduling horizon. The incoming samples $a_{ik}, \forall 1 < k \leq n(i)$ simulate samples that had been processed at the preceding processing unit p^i_{k-1} during a previous scheduling horizon. The samples in task i introduced at p^i_k must visit each subsequent processing unit in $\wp_i$ sequentially, that is, p^i_{k+1} in $\wp_i$ can only be visited if p^i_k has already been visited. Samples are considered to have visited processing unit, p, if it has been processed by one of the machines in J_p. It is assumed that at the beginning of the scheduling horizon, the machines in the facility, $j \in J$, will become available to start processing materials at the facility. We also assume that there is no transportation time between different processing units, and that there are no restrictions on intermediate storage.

An illustration of material flow through processing units is presented in Figure 3.1. This illustration contains two tasks, $i = 1, 2$ with paths $\wp_1 = [1, 3, 4]$ and $\wp_2 = [2, 3, 5]$. There are 5 unique processing units, $p = 1, \ldots, 5$ where $p = 3$ is a common process in the paths of both tasks. It is possible for the materials from the two tasks coming from $p = 1$ and $p = 2$ to be processed simultaneously at $p = 3$. Note, however, that this multitasking operation is not a blending or mixing operation (the materials are solids, or separated in individual containers). After having been processed simultaneously at $p = 3$, the materials for the two different tasks go onto two different processing units, $p = 4$ and $p = 5$; this is called material splitting. For an example map of the process network of the analytical services facility considered in this thesis, readers can refer to Figure 3.2.

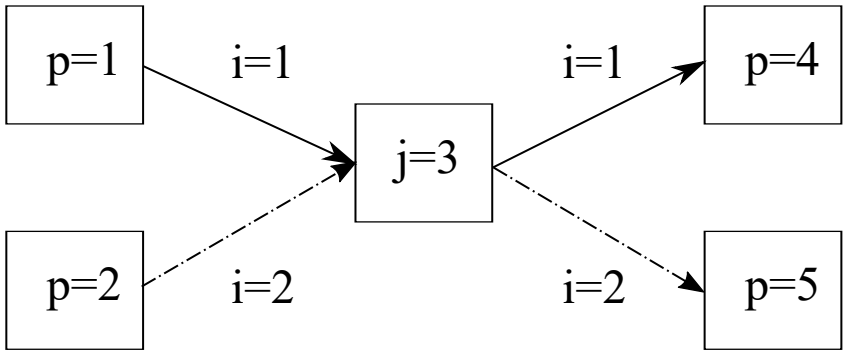


Figure 3.1: Illustration of material flow featuring multitasking and material splitting

Machines in a processing unit p have a specific capacity, denoted as β_p, and an associated processing time, denoted as $\tau(p)$. This means that machine $j \in J_p$ can be loaded with at most β_p number of samples from potentially different tasks. Once a machine has been turned on to process the samples, it will run without interruption for a time $\tau(p)$. After this time, the machine is considered to be available; also, the samples are considered to have visited the corresponding processing unit and they are ready to visit the next processing unit in their path. Since machines are assumed to be identical, the processing time of the machines in a processing unit can be referred to as the processing time of the processing unit. Furthermore, there is no minimum working capacity for any machine, that is, the machines can be turned on with any number of samples between 0 and β_p. It is assumed that the information described above is available and known *a priori*.

3.2 Short-term scheduling formulations

For the facility under consideration, maximization of the number of samples to be processed is a key priority. Ideally, we would like to maximize the true throughput of the facility over an extended period of time, defining the throughput to be the number of samples for which the last process in their respective paths have been completed. However, we can only define the objective function over the scheduling horizon, and defining the objective function to maximize the throughput can only lead to suboptimal resource utilization over an extended period of time in a capacity constrained environment. This suboptimal resource utilization happens when there is not enough time or capacity to finish processing some samples within the scheduling horizon, and a simple throughput maximization objective would have no objective incentive to start processing these samples. With such an objective function, processing units with processing times that are longer than the scheduling horizon will always be idle. Therefore, the objective functions to be presented for the discrete and continuous-time formulations in Sections 3.2.1 and 3.2.2 will give incentive for samples to be processed even if they cannot reach the end of their respective paths, and even if processing cannot be completed within the scheduling horizon. Similarly, flow constraints for these formulations will allow processes to continue processing even after the end of the scheduling horizon, once processing starts for a machine. Processing may also start at the end of the scheduling horizon, but no processes are allowed to start after the end of the scheduling horizon. This particular feature in our formulation is indeed required to reflect the actual operation in analytical service facilities.

3.2.1 Flexible discrete-time formulation

The flexible discrete-time formulation presented in this study extends the multitasking uniform discrete-time ILP formulation presented by Patil et al.[120], by incorporating the multiple non-uniform time grids approach proposed by Velez and Maravelias[140]. To derive such extension, additional notation and assumptions are needed. The revised formulation is presented below. The time domain for each individual processing unit will be discretized into a predetermined series of time points.

Each $p \in P$ will have a time step $\Delta(p)$, which represents the time elapsed between two consecutive time points for processing unit p. Let $\mathcal{E}(p) = (0, \varphi_{p1}, \varphi_{p2}, \ldots, \varphi_{p(|\mathcal{E}(p)|-2)}, H)$ represent the increasing sequence of time points of processing unit p along the axis of time. Let $\varphi_{pt}, \forall t = 0, \ldots, |\mathcal{E}(p)| - 1$ represent the t^{th} element of $\mathcal{E}(p)$, i.e., the time value of the t^{th} time point of a discretized time grid. Given the above descriptions for the flexible discrete-time representation, we define a time step as

$$\Delta(p) := \varphi_{p(t+1)} - \varphi_{pt};\ \forall t = 1, \ldots, |\mathcal{E}(p)| - 2 \tag{3.1}$$

Note that, $\forall\, p \in P$, $\mathcal{E}(p)$ is known *a priori*; also the machines in the processing unit can only be turned on at the beginning of a time point of the processing unit. The present formulation assumes that the time steps are fixed for each processing unit; however, this condition can be relaxed to consider variable time steps. The different discretization schemes to be considered for this work will be discussed in Section 3.3.1.

Since the machines in a processing unit are assumed to be identical in the flexible discrete-time formulation, it is not needed to consider the machines J_p of processing unit p on an individual basis, rather they are considered as resources of the processing units. Accordingly, $R_p := |J_p|$ represents the number of machines in processing unit p, for all $p \in P$.

The first set of decision variables for the flexible discrete-time formulation are described next.
B_{ikt}: Is a nonnegative integer variable representing the number of samples from task i that are set to start being processed at the k^{th} processing unit in the path of task i, $p_k^i \in \wp_i$, at the time point t, $\forall\ i \in I,\ 1 \leq k \leq n(i),\ t = 0, \ldots, |\mathcal{E}(p_k^i)| - 1$.
X_{pt}: Is a nonnegative integer variable representing the number of machines from processing unit p that are being used at time point t, $\forall\ p \in P,\ t = 0, \ldots, |\mathcal{E}(p)| - 1$.

Constraints (3.2)-(3.3) ensure that the machines in the facility are not overloaded with samples.

$$\sum_{i,k:p=p^i_k} B_{ikt} \leq X_{pt}\beta_p; \quad \forall\ p \in P,\ t = 0,\ldots,|\mathcal{E}(p)| - 1 \tag{3.2}$$

$$\sum_{\theta\in\mathcal{E}(p):\varphi_{pt}<\varphi_{p\theta}+\tau(p)\leq\varphi_{pt}+\tau(p)} X_{p\theta} \leq R_p; \quad \forall\ p \in P,\ t = 0,\ldots,|\mathcal{E}(p)| - 1 \tag{3.3}$$

As mentioned above, a sample from task i can visit processing unit p^i_k in $\wp_i$ only if it has already visited the previous processing unit, p^i_{k-1}, in $\wp_i$. Also, a sample from task i is considered to have visited processing unit p^i_k in $\wp_i$, if it has been processed by one of the machines in J_p. To account for these conditions, the following notation is considered:

W_{ikt}: is a nonnegative integer variable representing the number of samples from task i that have visited p^i_{k-1} and are ready to visit processing unit p^i_k at time point t , $\forall\ i \in I,\ k = 2,\ldots,n(i),\ t = 0,\ldots,|\mathcal{E}(p^i_k)| - 1$.

Constraint (3.4) ensures that a subset of materials from task i can visit processing unit p^i_k in $\wp_i$ at time point t, if it has already visited processing unit p^i_{k-1} in $\wp_i$ before time point t. Furthermore, it ensures that the same number of samples from task i that enter a processing unit, leave the processing unit after completing their processing, i.e., constraint (3.4) is a flow conservation constraint, preventing samples from being created or lost. Because of its structure, constraint (3.4) can not be extended to the first processing unit in the path of a task and the first time point of the processing unit; hence, constraint (3.5) is introduced to extend constraint (3.4) to the first process in the paths of the tasks.

$$B_{ikt} + W_{ikt} = W_{ik(t-1)} + \sum_{\substack{\theta=1,\ldots,|\mathcal{E}(p^i_{k-1})|:\\ \varphi_{p^i_k t-1}<\varphi_{p^i_{k-1}\theta}+\tau(p^i_{k-1})\leq\varphi_{p^i_k t}}} B_{i(k-1)\theta}; \tag{3.4}$$

$$\forall\ i \in I,\ k = 2,\ldots,n(i),\ t = 1,\ldots,|\mathcal{E}(p^i_k)| - 1$$

$$B_{i1t} + W_{i1t} = W_{i1(t-1)}; \quad \forall\ i \in I,\ t = 1,\ldots,|\mathcal{E}(p^i_k)| - 1 \tag{3.5}$$

Constraint (3.6) introduces a_{ik} samples for task i at processing unit p^i_k at the beginning of the scheduling horizon as the number of samples waiting to start processing at that unit.

$$W_{ik0} = a_{ik}; \quad \forall\ i \in I,\ k = 1\ldots n(i) \tag{3.6}$$

Given that the samples are introduced as samples waiting to start processing at the first time point, constraint (3.7) prevents samples from being processed at the first time point.

Therefore, for each discretization scheme and for all processing units, discretization was carried out to set the value of the first two time points, φ_{p0} and φ_{p1}, to zero in order to allow events to occur at the beginning of the scheduling horizon. For this reason, eq (3.1) defined the time steps starting at $t = 1$ rather than $t = 0$.

$$B_{ik0} = 0; \quad \forall\ i \in I,\ k = 1 \ldots n(i) \tag{3.7}$$

The following objective function aims to maximize throughput while optimally utilizing available resources.

$$maximize \sum_{i \in I} \sum_{k=1}^{n(i)} \sum_{t=0}^{|\mathcal{E}(p_k^i)|-1} \frac{k}{n(i)} B_{ikt} \tag{3.8}$$

The weight $\frac{k}{n(i)}$ is the relative position of processing unit p_k^i in path $\wp_i$ with $n(i)$ number of processing units. For example, the 4^{th} processing unit (p_4^i) in a path with 7 processing units $(n(i) = 7)$ will have a weight of $\frac{4}{7}$. This weight $\frac{k}{n(i)}$, therefore, gives priority to beginning the last process of a task's path. This approach is utilized over an explicit throughput maximization approach in order to avoid situations where available resources are left idling if a task may not be completed by the end of the scheduling horizon as discussed in the opening of this chapter. This objective function represents one of the several possible approaches for approximating the total throughput (TTP) for a short-term scheduling horizon as discussed further later in this thesis in Section 4.2.2. This particular objective function is used for this study given its relative simplicity of expression, while we deemed it to be a better approximate expression of the TTP when compared to using, for example, a constant weight of 1.

3.2.2 Continuous-time formulation

Lagzi et al.[86] introduced a global event-based MILP formulation that has been modified to model the problem considered in this work. For the purpose of completeness, we will describe the formulation here as used in this work. The formulation was modified to remove features that were not considered for this work, namely, minimum processing load, and the feature allowing samples to be introduced after the beginning of the scheduling horizon. An *idle* task will be assigned to a machine whenever the machine is being idle. The idle task is denoted as task number 0 whereas $I = 1, ..., |I|$ is the set of the actual tasks. Note that, for every $p \in P$ and $j \in J_p$, we denote by I_j, the subset of tasks $i \in I$ where $p \in \wp_i$.

The time domain will be partitioned into $N+1$ predetermined number of time points, where the distance between two consecutive time points is called a time slot. The time points are shared by all the tasks and machines and their locations along the time axis is determined through the optimization model. Note that the first and last time points, 0 and N, have their locations fixed at the beginning and at the end of the scheduling horizon, i.e. times 0 and H, respectively. Accordingly, $T_n \in [0, H],\ \forall\ n = 0, \ldots, N$, is the decision variable representing the location of time point n and $SL_n \in [0, H]$ is the decision variable representing the length of time slot n, where time slot n is the time between T_n and T_{n-1}, for all $n \in 1, \ldots, N$. Notice that the number of time points is assumed to be given as an input. The problem of determining the appropriate value of N will be discussed in Section 3.3.1. The rest of the decision variables used in the continuous-time formulation are as follows:

$Z_{jn} \in \{0, 1\}$: 1 if machine j is turned on at time point n, and 0 otherwise; $\forall\ j \in J,\ \ n = 0, \ldots, N$.

$Y_{ijn} \in \{0, 1\}$: 1 if samples from task i start being processed at machine j at time point n, and 0 otherwise; $\forall\ j \in J,\ i \in I \cup \{0\},\ n = 0, \ldots, N$.

$YE_{ijn} \in \{0, 1\}$: 1 if machine j is set to finish processing samples from task i at time point n, and 0 otherwise; $\forall\ j \in J,\ i \in I \cup \{0\},\ n = 0, \ldots, N$.

$YR_{ijn} \in \{0, 1\}$: 1 if machine j is set to continue processing samples from task i at time point n, and 0 otherwise; $\forall\ j \in J,\ i \in I \cup \{0\},\ n = 0, \ldots, N$.

TR_{ijn}: A nonnegative continuous variable, representing the amount of time remaining to complete processing samples from task i that are set to continue being processed at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

B_{ijn}: A nonnegative integer variable representing the number of samples from task i that are set to begin processing at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

BE_{ijn}: A nonnegative integer variable representing the number of samples from task i that are set to finish being processed at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

BR_{ijn}: A nonnegative integer variable representing the number of samples from task i that are set to continue being processed at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

W_{ikn}: A nonnegative integer variable representing the number of samples from task i that have visited p^i_{k-1} and are ready to visit processing unit p^i_k at time point n; $\forall\ i \in I \cup \{0\},\ k = 2, \ldots, n(i),\ n = 0, \ldots, N$.

Constraints (3.9)-(3.11) model the relationship between T_n and SL_n.

$$\sum_{n=1}^{N} SL_n = H \tag{3.9}$$

$$T_n - T_{n-1} = SL_n; \quad \forall\, n = 1, \ldots, N \tag{3.10}$$

$$T_0 = 0 \tag{3.11}$$

Constraints (3.12)-(3.18) ensure that turning machine j on at time point n is coordinated with machine j starting, continuing, or finishing processing samples from a set of tasks. They also assure that a machine cannot continue processing samples and start processing new samples, at any time point.

$$Z_{jn} \geq Y_{ijn}; \quad \forall\, j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N \tag{3.12}$$

$$Z_{jn} \leq \sum_{i \in I_j \cup \{0\}} Y_{ijn}; \quad \forall\, j \in J,\ n = 0, \ldots, N \tag{3.13}$$

$$Z_{jn} \geq YE_{ijn}; \quad \forall\, j \in J,\ i \in I_j \cup \{0\},\ n = 1 \ldots, N \tag{3.14}$$

$$\sum_{i \in J} \sum_{i \in I_j \cup \{0\}} YE_{ij0} = 0 \tag{3.15}$$

$$Z_{jn} \leq 1 - YR_{ijn}; \quad \forall\, j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N \tag{3.16}$$

$$(1 - \sum_{i \in I_j \cup \{0\}} YR_{ijn}) \leq Z_{jn}; \quad \forall\, j \in J,\ n = 1, \ldots, N \tag{3.17}$$

$$\sum_{i \in I_j \cup \{0\}} YR_{ij0} = 0; \quad \forall\, j \in J \tag{3.18}$$

Constraint (3.19) ensures that, if samples from task i started or continued to be processed at machine j at time point n, then, at the next time point the samples will either complete their processing or continue being processed in machine j.

$$YR_{ijn} = YR_{ij(n-1)} + Y_{ij(n-1)} - YE_{ijn}; \quad \forall\, j \in J,\ i \in I_j \cup \{0\},\ n = 1, \ldots, N \tag{3.19}$$

Constraint (3.20) ensures that a machine cannot process samples while the machine is being idle.

$$Y_{ijn} \leq 1 - Y_{0jn}; \quad \forall\, j \in J,\ i \in I_j, n = 0, \ldots, N \tag{3.20}$$

Constraints (3.21)-(3.28) ensure that the number of samples in a machine is at most equal to the capacity machine j.

$$B_{ijn} \leq \beta_p Y_{ijn}; \quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N \tag{3.21}$$

$$Y_{ijn} \leq B_{ijn}; \quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 1, \ldots, N \tag{3.22}$$

$$\sum_{i \in I_j \cup \{0\}} B_{ijn} \leq \beta_p Z_{jn}; \quad \forall\, p \in P,\ j \in J_p,\ n = 0, \ldots, N \tag{3.23}$$

$$BR_{ijn} \leq \beta_p YR_{ijn}; \quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N \tag{3.24}$$

$$\sum_{i \in I_j \cup \{0\}} BR_{ijn} \leq \beta_p (1 - Z_{jn}); \quad \forall\, p \in P,\ j \in J_p,\ n = 0, \ldots, N \tag{3.25}$$

$$BE_{ijn} \leq \beta_p YE_{ijn}; \quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N \tag{3.26}$$

$$YE_{ijn} \leq BE_{ijn}; \quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 1, \ldots, N \tag{3.27}$$

$$\sum_{i \in I_j \cup \{0\}} BE_{ijn} \leq \beta_p Z_{jn}; \quad \forall\, p \in P,\ j \in J_p,\ n = 0, \ldots, N \tag{3.28}$$

Constraint (3.29) introduces a_{ik} samples for task i at processing unit p_k^i at the beginning of the scheduling horizon as the number of samples waiting to start processing at that unit.

$$W_{ik0} = a_{ik}; \quad \forall i \in I,\ k = 1, \ldots, n(i) \tag{3.29}$$

Given that the samples are introduced as samples waiting to start processing at the beginning of the scheduling horizon, all machines are set to idle for the first time point according to constraint(3.30). Note that the optimal schedule can still have machines starting processing at the beginning of the scheduling horizon if $T_1 = 0$.

$$Y_{0j0} = 1; \quad \forall j \in J \tag{3.30}$$

Constraint (3.31) ensures that the number of samples that finishes processing or continues to be processed at machine j at time point n is equal to the number of samples that started or continued to be processed in machine j at the previous time points, i.e., samples are not created or lost.

$$BR_{ijn} + BE_{ijn} = BR_{ij(n-1)} + B_{ij(n-1)}; \quad \forall\, j \in J,\ i \in I_j,\ n = 1, \ldots, N \tag{3.31}$$

Constraint (3.32) ensures that a sample from task i can visit processing unit p_k^i in $\wp_i$ at time point n, if it has already visited processing unit p_{k-1}^i in $\wp_i$ before time point n.

$$\sum_{j \in J_p:\ p = p_k^i} B_{ijn} + W_{ikn} = W_{ik(n-1)} + \sum_{j \in J_p:\ p = p_{k-1}^i} BE_{ijn}; \quad \forall\, i \in I,\ \ k = 2, \ldots, n(i),\ n = 1, \ldots, N \tag{3.32}$$

Constraint (3.33) is a flow constraint similar to constraint (3.32), but for the first processing unit of path $\wp_i$.

$$\sum_{j \in J_p:\ p=p_1^i} B_{ijn} + W_{i1n} = W_{i1(n-1)};\quad \forall\ i \in I,\ n = 1, \ldots, N \tag{3.33}$$

Constraints (3.34)-(3.35) regulate the amount of time remaining to finish processing samples from task i at machine j at time point n if machine j is set to continue processing samples from task i at time point n.

$$TR_{ijn} \leq \tau(p) Y R_{ijn};\quad \forall\ p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N \tag{3.34}$$

$$TR_{ij(n+1)} \geq TR_{ijn} + \tau(p) Y_{ijn} - SL_{n+1};\quad \forall\ p \in P,\ j \in J_p,\ i \in I_j,\ n = 0, \ldots, N-1 \tag{3.35}$$

Note that constraint (3.34) is not considered for the idle task, since a machine can stop being idle at any time if a set of samples are assigned to start being processed at the machine at that time.

The objective function for the continuous-time formulation is expressed so that it can be directly comparable to the objective function for the flexible discrete-time formulation presented in eq (3.8).

$$maximize \sum_{n=0}^{N} \sum_{i=1}^{|I|} \sum_{k=1}^{n(i)} \sum_{j \in J_p : p=p_k^i} \frac{k}{n(i)} B_{ijn} \tag{3.36}$$

3.3 Computational study

In this section, we present the case study, composed of a total of 190 instances and 1,030 runs, based on an actual analytical services facility. In Section 3.3.1, we define the network of processing units defined by various paths, we present model parameters used for this study, we present the metrics by which comparisons were made, and we present the discretization schemes that were compared. In Sections 3.3.1 – 3.3.4, we present and discuss the results of these comparisons.

Table 3.1: Paths of analytical processes defining the process network. Default path precursor A-B-C-D considered in the analysis but omitted in the table for brevity.

Path ID	Sequence of processing units				
P1	X	F	M	P	K
P2	X	F	E	K	
P3	X	G	M	T	L
P4	X	F	R	K	
P5	X	F	M	T	L
P6	Y	F	M	Q	J
P7	H	M	T	L	
P8	X	W	S	K	
P9	X	W	N	U	L
P10	X	W	O	I	V
P11	X	W	E	K	

3.3.1 Experimental data and design of instances

The network of the processing units being considered in this work is presented in Figure 3.2. This network resembles the actual analytical services facility that was studied. The identity of the industrial partner and the names of involved processes are not disclosed as per our non-disclosure agreement with the industrial partner. However, the proposed computational studies retain the characteristics of the majority of the operations at this facility. This network of 25 processing units is defined by the 11 paths presented in Table 3.1. Each task entering the facility goes through processes A, B, C, and D prior to starting processes defined by the paths in Table 3.1. For example, a task with Path ID, P1, will have a path of A-B-C-D-X-F-M-P-K; the first part of the path, A-B-C-D, is the default precursor path, and X-F-M-P-K is the unique path defined in Table 3.1.

In this work, the resources in each process unit p have the same processing capacity β_p and processing time $\tau(p)$, where a resource may represent a machine, a worker, or a machine-worker combination. Each processing unit represents a group of such resources performing a specific process or unit operation (e.g., drying, filtration). The number of resources, the capacity, and the processing time for each processing unit considered for this study are reported in Table 3.2. Instances created with the process information values reported in Table 3.2 will be referred to as baseline instances.

Table 3.2: Process capacity, resources, and processing time information

Process	Capacity	Resources	Processing Time (min)
A	500	2	15
B	60	3	60
C	2250	2	1440
D	50	4	375
E	42	2	40
F	216	2	300
G	21	2	150
H	48	1	615
I	7	1	1440
J	150	1	240
K	480	3	180
L	440	2	240
M	216	4	120
N	440	4	220
O	1	1	10
P	180	1	390
Q	240	1	1440
R	720	10	735
S	480	10	471
T	112	8	1256
U	135	8	1141
V	22	1	60
W	440	4	1620
X	10	6	10
Y	10	1	10

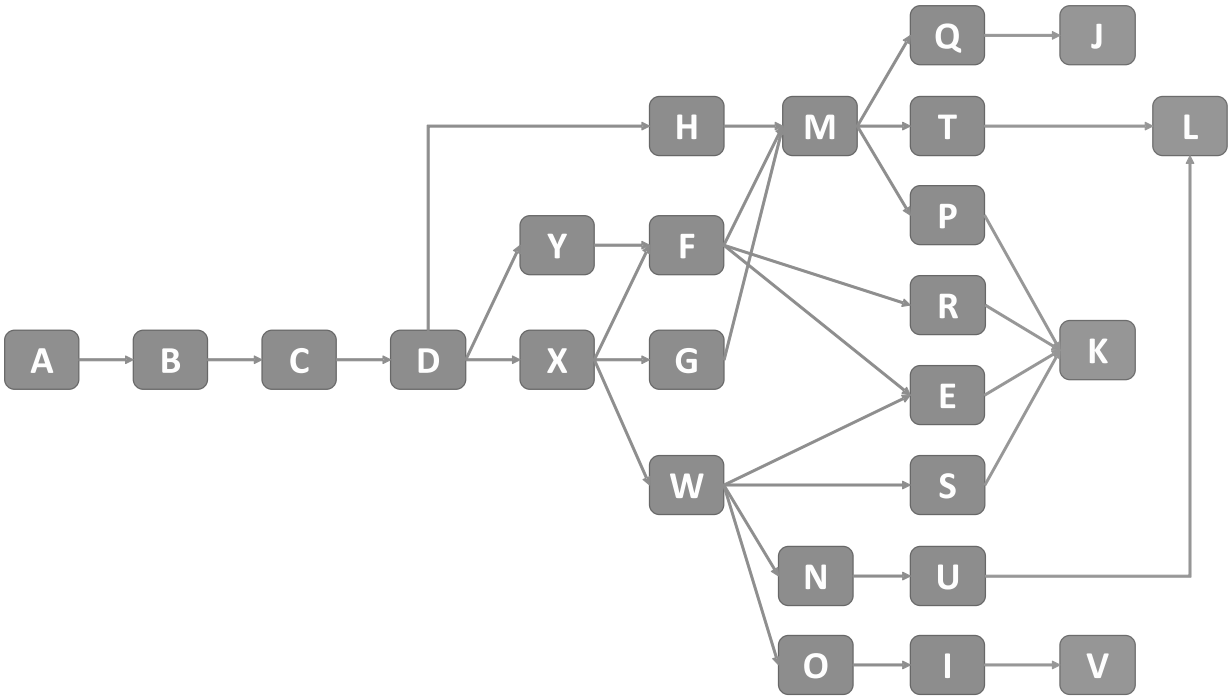


Figure 3.2: Map of the process network

In all computational studies, experiments are designed with the following experimental design parameters: number of tasks ($|I|$), length of the scheduling horizon (H), number of time points for the continuous-time formulation (N), and processing time variability (Γ), which has been defined in this work as follows::

$$\Gamma = \frac{\max\limits_{p \in P} \tau(p) - \min\limits_{p \in P} \tau(p)}{\min\limits_{p \in P} \tau(p)} \tag{3.37}$$

Note that in the context of these computational studies, processing time variability as defined by eq (3.37) does not refer to variability of processing time with respect to a dependent variable, but it is a relative measure of the discrepancy between the longest and

the shortest processing times considered in these instances. The baseline instances have processing time variability of 161.

In the present work, a set of instances contains ten instances with the same experimental design parameters. However, for each of the ten instances within a set, each task considers:
a) a randomly generated number of samples between 10 and 500, which is the typically observed range of task size at the analytical services facility under study,
b) a randomly assigned Path ID and the path associated with it, and
c) a randomly generated starting point within the path.

In what follows, *UD* will refer to implementation of the discrete-time formulation with uniform time discretization, and *NUD* will refer to implementation of the discrete-time formulation with non-uniform time discretization. When UD or NUD is followed by a number, the number will indicate the discretization scheme. For UD, the number will indicate the size of the uniform time steps. For example, each $p \in P$ in UD10 will have a time step $\Delta(p) = 10$. For NUD, the number will indicate the maximum discretization value. For example, for NUD30, each $p' \in P$ with $\tau(p') < 30$ will have a time step $\Delta(p') = \tau(p')$ and each $p'' \in P$ with $\tau(p'') \geq 30$ will have a time step $\Delta(p'') = 30$. For either UD or NUD, given a time step $\Delta(p)$ for processing unit p, the unit-specific time grid is discretized according to eq (3.38).

$$\mathcal{E}(p) = (0, 0, \Delta(p), 2\Delta(p), \ldots, (\lceil \frac{H}{\Delta(p)} \rceil - 2)\Delta(p), (\lceil \frac{H}{\Delta(p)} \rceil - 1)\Delta(p), H) \tag{3.38}$$

$\mathcal{E}(p)$ has $(\lceil \frac{H}{\Delta(p)} \rceil - 1) + 3$ time points index from $t = 0$ to $t = |\mathcal{E}(p)| - 1$. Note that if $\Delta(p)$ is not a factor of H, then $H - \varphi_{p(|\mathcal{E}(p)|-2)} \neq \Delta(p)$. For this reason, eq (3.1) defined the time steps up to $t = |\mathcal{E}(p)| - 2$ rather than $t = |\mathcal{E}(p)| - 1$.

Based on the above, UD10, UD30, UD60, NUD30, and NUD60 discretization schemes are studied in the following computational studies. In practice, for short-term scheduling, we would like to be able to make decisions at least on an hourly basis, but it can also becomes impractical to define schedules down to the minute. We consider UD10 to be the finest discretization where the uniform $\Delta(p)$ equals to the minimum allowable accuracy, whereas UD60 represents the *coarse* discretization where the uniform $\Delta(p)$ equals to the maximum allowable accuracy. For all sets of instances, each instance was solved for all five of these discretization schemes, and the continuous-time formulation was solved in addition

to these five discretization schemes when the continuous-time formulation was compared against the flexible discrete-time formulation.

Regarding the continuous-time formulation, each of the instances was solved for a selected number of time points, and instances with different numbers of time points were considered to analyze the effect of increasing number of time points on the computational time. Increasing the number of time points, starting with a small number of time points, will increase both the solution quality and computational demand, until a point when adding time points will not improve the quality of the solution. The procedure of starting with a small number of time points and increasing it until no improvement can is observed was used by Ierapetritou and Floudas[74]. However, before the maximum number of useful time points is reached, the problem may become intractable, or fail to be solved to optimality within a given CPU time limit. In this work, we initialized the continuous-time formulation with 6 time points, and aimed to increase the number of time points until no improvement in the solutions are detected. A CPU time limit of 8 hours was considered for these simulations. This CPU time limit was imposed based on the expected time needed to arrive to optimal solutions for industrial-case problems, especially in a short-term scheduling context for analytical service facilities (i.e., a new schedule may be required every day).

For the remainder of this study, the computation of a particular discretization scheme or the continuous-time formulation has been referred to as a *run*. To summarize, a set of instances has ten instances with the same experimental design parameters, and each instance has five or six instances depending on whether or not the continuous-time formulation is considered for that instance.

In principle, UD60, as the *coarse* discretization scheme, may result in the worst objective value and the lowest computational time among those five discretization schemes, and the continuous-time formulation should also yield better solutions than UD60 perhaps at the cost of additional computational time, provided enough number of time points are specified in the continuous-time formulation. Note that comparing absolute values of the objective function and the computational time between different instances may not provide insightful or meaningful conclusions due to different experimental design parameters. Therefore, to evaluate each approach fairly, we compute, for each run within an instance, the 'Relative Objective Benefit (ROB)' and the 'Relative CPU time Disadvantage (RCD)' compared to the objective value and CPU time yielded by UD60 for that instance. ROB and RCD are defined as follows:

$$ROB(\chi) = \frac{ObjVal(\chi) - ObjVal(UD60)}{ObjVal(UD60)} \tag{3.39}$$

$$RCD(\chi) = \frac{CPUtime(\chi) - CPUtime(UD60)}{CPUtime(UD60)} \tag{3.40}$$

where χ is one of the discretization schemes or the continuous-time formulation under consideration.

When the mean of a value over a set of instances is reported, the associated standard error of the mean (SEM) is calculated and reported as follows:

$$SEM = \frac{STDEV.S}{\sqrt{NIST}} \tag{3.41}$$

where STDEV.S is the sample standard deviation, and NIST is the number of instances.

As stated in the Introduction, the following computational experiments are expected to evaluate the strengths and weaknesses of the different time representation approaches relative to each other, and by doing so, determine the favorable approach to solving industrial-sized short-term scheduling problems for a multipurpose plant. All the computational experiments were performed on a Linux server with 250 GB of RAM and 4 CPUs, each with 12 cores and a processing speed of 2.4 GHz using IBM ILOG CPLEX Optimizer 12.6.0[71]. CPLEX was run with the default settings, except for the CPU time limit of 8 hours. Particularly, this means that the setting for parallelism allowed utilization of all 48 available logical cores.

3.3.2 Comparing the discrete-time and the continuous-time formulations

In order to compare the different discretization schemes along with the continuous-time formulation, 80 relatively small sized instances (i.e., eight sets of instances, each set with ten randomly generated instances with the same experimental design parameters) were considered with the number of tasks ranging from 5 to 10, the number of time points for the continuous-time formulation ranging from 6 to 8, with a set scheduling horizon of 8

hours. We aimed to start with a low number of time points for the continuous-time formulation, and increase the number of time points until the computational time for the continuous-time formulation becomes prohibitive. For each instance, six runs were made for the five discretization schemes and the continuous-time formulation.

Results from these 480 runs are shown in Figure 3.3, where each point on the plot represents a run, and demonstrates the trade-off between the solution quality and the computational time depending on its position within the plot. There are 400 points shown on Figure 3.3 since 80 of the 480 runs are for UD60, whereas $ROB(UD60) = 0$, $RCD(UD60) = 0$ as per eq (3.40) and (3.41) respectively. Thus, these values are not shown for brevity. In this figure, the RCD values are plotted on the horizontal axis, while the ROB values are plotted on the vertical axis. As shown in Figure 3.3, a perfectly vertical trend would represent improved relative solution quality at absolutely no deterioration in relative computational cost. On the other hand, a perfectly horizontal trend would represent deterioration in relative computational cost with absolutely no relative gain in solution quality. Therefore, a time representation scheme with a steep trend resembling a vertical line close to the vertical axis would be generally more desirable over a scheme with a flat trend resembling a horizontal line far from the vertical axis. The horizontal axis for the RCDs is given in logarithmic scale for readability. Note that with the logarithmic scale, a trend that is closer to the vertical axis has a steeper slope and a narrower spread compared to a trend that appears to have similar spread and slope, but further from the vertical axis.

In addition to the slope of the trend, the positions of the scattered points are important performance indicators of a particular scheme. Having scattered points closer to the vertical axis indicates the ability to provide solutions quickly, while having scattered points far from the horizontal axis indicates the ability to provide high quality solutions. Comparing the maximum ROB values (the heights of the trends) of the different schemes is a simple way to compare the potentials of the different schemes to provide high quality solutions.

The mean ROB and RCD values for each set of ten instances and their associated standard errors are reported in Tables 3.4–3.7. All runs reached optimality for the flexible discrete-time formulation, whereas 72 of 80 runs for the continuous-time formulation reached optimality, and 8 runs failed to reach optimality within a CPU time limit of 8 hours. The CPU time limit was set to 8 hours in order to be able to complete these runs within a reasonable time frame. More detailed information including the optimality gaps on these 8 runs are reported in Table 3.3.

Table 3.3: Runs that failed to reach optimality within eight hours

Run Type	Number of Tasks	Number of Samples	Horizon (hr)	Number of Time Points	Optimality Gap (%)
Continuous	6	1624	8	6	1.99
Continuous	7	1275	8	6	5.36
Continuous	8	2975	8	6	16.01
Continuous	9	2163	8	6	5.41
Continuous	9	2845	8	6	10.96
Continuous	9	2555	8	6	13.37
Continuous	10	1803	8	6	16.39
Continuous	10	2367	8	6	13.55
Continuous	10	2742	8	6	9.92
Continuous	5	1418	8	7	0.44
Continuous	5	1440	8	8	3.9
Continuous	5	1364	8	8	2.13

These runs that failed to reach optimality within 8 hours still appear in Figure 3.3 and are included in mean value calculations. The ROB and RCD for these failed runs were computed using the objective value at the point of forced termination of the optimization run using 8 hours as the CPU time. Here, optimality gap is calculated as:

$$\frac{UB - OBJ}{UB} \tag{3.42}$$

where UB denotes the smallest upper bound on the objective function value obtained from the branch-and-bound tree search, and OBJ is the objective function value of the best feasible solution found within the CPU time limit of eight hours.

In Figure 3.3, we can observe the discrete-time formulation forming steep vertical trends and the continuous-time formulation forming a relatively flat trend. For the multi-tasking modeling framework presented in Section 3.2 for this case study, the discrete-time formulation was also generally able to provide higher quality solutions than the continuous-time formulation. In particular, NUD30, NUD60, UD10, and UD30 returned significantly higher maximum ROB values compared to the continuous-time formulation. Figure 3.3 suggests that the modeling framework developed in this work may not be suitable for solving large instances while using the continuous-time formulation as computational cost deteriorates rapidly for a small gain in solution quality.

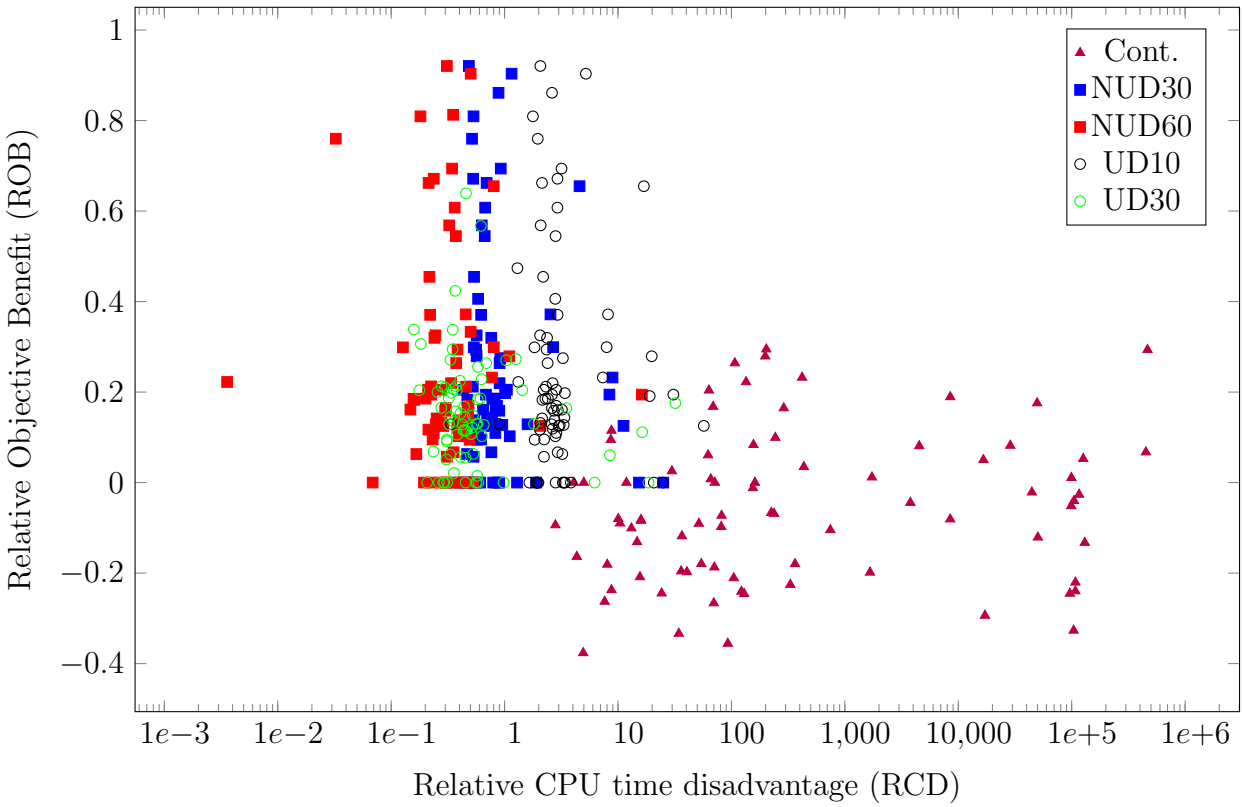


Figure 3.3: Comparison of different time representations

As reported in Table 3.4, the continuous-time formulation generally provided better quality solutions than UD60 with 5 tasks when implemented with more than 6 time points. However, as reported in Table 3.6, the computational time increased rapidly for the continuous-time formulation with increasing number of time points, as well as increasing number of tasks. The continuous-time formulation required, on average, 395 seconds for a run with 5 tasks and 6 time points, but a run with 5 tasks and 8 time points required 4,939 seconds, and a run with 10 tasks and 6 time points required 9,153 seconds. On the other hand, NUD30, NUD60, UD10, and UD30 all produced lower mean RCDs with 10 tasks than they did with 5 tasks, generally requiring less than 8 seconds to compute.

These results are in line with the observations of optimality gap at the root node relaxation, the number of variables, and the number of constraints as reported in Tables 3.8, 3.9, and 3.10. While the continuous-time formulation returned lower optimality gap at the root node relaxation than UD10 and UD60 for certain instances, it returned, on average, much higher optimality gap compared to its NUD and UD counterparts. This indicates that, within the scope of this work, the flexible discrete-time formulation generally led to tighter problems than the continuous-time formulation presented in this work (recall that a tighter formulation is generally less computationally demanding as discussed in Section 2.1). In addition, the average value of this gap for the continuous-time formulation increased 3 fold when the number of time points was increased from 6 to 8, which indicates a significant trade-off between computational time and solution quality. The continuous-time formulation also produced much higher average numbers of variables and constraints compared to its NUD and UD counterparts, indicating much higher computational demand.

Furthermore, while Standard Error of Mean (SEM) values of ROB were lower for the continuous-time formulation compared to the discrete-time formulation as reported in Table 3.5, SEM values of RCD were much higher for the continuous-time formulation as reported in Table 3.7, and this gap in SEM values of RCD increased with increasing number of tasks and increasing number of time points. This indicates that, for the application studied in this work, while the continuous-time formulation may provide solutions with more consistent quality, it may be computationally less consistent, and the CPU time consistency deteriorates with increasing problem size.

These results suggest that the discrete-time representation is more suitable for the present multi-tasking modeling framework since it is able to handle industrial-sized short-term scheduling problems of this nature with hundreds of tasks. However, the continuous-time formulation demonstrated that it can obtain higher quality solutions than the coarse

Table 3.4: Mean Relative Objective Benefit (ROB) over UD60

Number of Tasks	Number of Time Points (C)	NUD30	NUD60	UD10	UD30	C	UD60 ObjVal
5	6	0.24	0.23	0.24	0.16	0.00	747
5	7	0.24	0.23	0.24	0.16	0.08	747
5	8	0.24	0.23	0.24	0.16	0.06	747
6	6	0.35	0.35	0.35	0.19	−0.05	898
7	6	0.25	0.25	0.25	0.13	−0.15	1,198
8	6	0.22	0.22	0.22	0.14	−0.11	1,178
9	6	0.22	0.22	0.22	0.14	−0.14	1,205
10	6	0.19	0.18	0.19	0.13	−0.17	1,245

Table 3.5: Standard Error of Mean (SEM) of ROB over UD60

Number of Tasks	Number of Time Points (C)	NUD30	NUD60	UD10	UD30	C
5	6	0.04	0.04	0.04	0.03	0.00
5	7	0.04	0.04	0.04	0.03	0.00
5	8	0.04	0.04	0.04	0.03	0.00
6	6	0.11	0.11	0.11	0.05	0.00
7	6	0.05	0.05	0.05	0.02	0.00
8	6	0.07	0.07	0.07	0.04	0.00
9	6	0.07	0.07	0.07	0.03	0.00
10	6	0.06	0.06	0.06	0.02	0.00

Table 3.6: Mean Relative CPU time disadvantage (RCD) over UD60

Number of Tasks	Number of Time Points (C)	NUD30	NUD 60	UD10	UD30	C	UD60 CPU time (s)
5	6	2.83	0.68	8.01	3.10	506	0.78
5	7	2.83	0.68	8.01	3.10	1,444	0.78
5	8	2.83	0.68	8.01	3.10	6,331	0.78
6	6	0.61	0.26	2.18	0.37	6,912	0.24
7	6	0.73	0.28	2.55	0.37	13,172	0.26
8	6	0.69	0.29	2.63	0.38	17,790	0.27
9	6	0.77	0.34	2.91	0.47	26,828	0.27
10	6	0.79	0.33	3.17	0.49	33,898	0.27

Table 3.7: Standard Error of Mean (SEM) of RCD over UD60

Number of Tasks	Number of Time Points (C)	NUD30	NUD60	UD10	UD30	C
5	6	1.04	0.55	2.31	1.31	1.40
5	7	1.04	0.55	2.31	1.31	3.03
5	8	1.04	0.55	2.31	1.31	17.13
6	6	0.05	0.04	0.09	0.04	15.53
7	6	0.05	0.03	0.10	0.02	40.26
8	6	0.03	0.03	0.17	0.04	42.02
9	6	0.09	0.04	0.12	0.05	51.88
10	6	0.05	0.03	0.07	0.04	54.28

Table 3.8: Mean optimality gap (%) at the root node relaxation

Number of Tasks	Number of Time Points (C)	NUD30	NUD60	UD10	UD30	UD60	C
5	6	1.16	0.76	6.38	3.25	0.49	93.33
5	7	1.16	0.76	6.38	3.25	0.49	73.65
5	8	1.16	0.76	6.38	3.25	0.49	288.43
6	6	0.00	0.55	7.46	5.25	0.81	54.69
7	6	0.97	0.26	2.68	1.75	0.16	56.75
8	6	1.51	0.00	6.17	0.00	16.67	52.99
9	6	5.11	0.00	6.30	1.33	0.00	76.58
10	6	0.87	0.00	8.35	2.11	0.00	148.98

Table 3.9: Mean number of variables for the discrete and continuous time formulations

Number of Tasks	Number of Time Points (C)	NUD30	NUD60	UD10	UD30	UD60	C
5	6	2,669	1,968	5,671	2,042	1,135	23,569
5	7	2,669	1,968	5,671	2,042	1,135	27,608
5	8	2,669	1,968	5,671	2,042	1,135	31,705
6	6	3,078	2,270	6,541	2,355	1,309	27,276
7	6	3,477	2,568	7,391	2,661	1,479	31,059
8	6	3,930	2,894	8,351	3,007	1,671	34,780
9	6	4,331	3,206	9,171	3,302	1,835	38,598
10	6	4,687	3,455	9,991	3,597	1,999	42,345

Table 3.10: Mean number of constraints for the discrete and continuous time formulations

Number of Tasks	Number of Time Points (C)	NUD30	NUD60	UD10	UD30	UD60	C
5	6	2,280	1,661	4,903	1,793	1,015	21,034
5	7	2,280	1,661	4,903	1,793	1,015	24,759
5	8	2,280	1,661	4,903	1,793	1,015	28,968
6	6	2,493	1,821	5,347	1,958	1,111	23,636
7	6	2,702	1,978	5,780	2,120	1,204	26,851
8	6	2,937	2,151	6,270	2,302	1,310	29,483
9	6	3,146	2,315	6,688	2,458	1,400	32,998
10	6	3,332	2,448	7,106	2,614	1,490	35,923

uniform discretization scheme when dealing with small numbers of tasks. One advantage of the continuous-time formulation over the coarse uniform discretization scheme is that while the coarse uniform discretization scheme can only provide schedules with events occurring at coarsely predefined time points (e.g., every hour) events can occur anywhere along the scheduling horizon for a schedule provided by the continuous-time formulation. For some scheduling problems involving small number of tasks, for example, scheduling for an industrial chemical plant producing small number of products but in large volumes with long processing times should be able to successfully implement a continuous-time formulation, especially if large CPU time limits can be accepted. For these problems, having precise time points can be important, and defining very fine time points over a long scheduling horizon can make the problems intractable. As reported in Table 3.6, UD10 required more time to compute by several factors compared to the coarser discretization schemes even when handling a small number of tasks. In addition, the continuous-time approaches must be considered if variable machine processing times is a significant feature of the scheduling problem since discrete-time approaches cannot readily handle this feature without resulting in large, computationally demanding formulations[86;132].

Comparing among the different discretization schemes, NUD30, NUD60, and UD10 provided solutions of very similar quality as shown in Figure 3.3 and Table 3.4. UD30 provided significantly worse solutions with little benefit to the computational time, if at all. While UD10 required more time to compute than the other discretization schemes, it still only required seconds to compute. Therefore, when considering short-term scheduling problems with such small number of tasks, implementing a fine uniform discretization scheme may seem to be suitable over coarser discretization schemes as the fine uniform

discretization scheme can provide schedules with more precise event points.

In the following subsections, much larger instances with hundreds of tasks will be considered as the objective of this work is to study industrial-sized problems. However, these large instances are intractable for the continuous-time formulation considered in this study as made evident above with just 6 to 10 tasks, therefore the following subsections will not consider the continuous-time formulation.

3.3.3 Comparing uniform and non-uniform discretization: varying $|I|$ and H

In order to compare the performances of the different discretization schemes, 90 instances with the number of tasks ranging from 100 to 200 and the length of the scheduling horizon ranging from 24 to 40 hours were considered. The instances with 100 tasks consisted of 25,600 samples on average. Given that the facility under consideration typically receives around 1000 – 4000 samples for every 8 hours of operation, the sizes of these instances reflect operations of an actual analytical services facility with particularly heavy client demands. All 450 runs for these instances reached optimality.

In Figure 3.4, the different discretization schemes can be distinguished easily. While NUD30, NUD60, and UD10 all provided significantly better solutions than UD60, the computational time deteriorated relatively quickly for UD10 compared to the non-uniform discretization schemes. While the trend for NUD60 is much steeper compared to the trend for UD10 (recall that the horizontal axis is in logarithmic scale), the maximum ROB value of NUD60 on this plot is almost as high as that of UD10. That is, NUD60 provided solutions that were virtually as good as those provided by UD10 at a fraction of the computational cost compared to UD10 (99% less RCD, on average, as reported on Table 3.13).

Superior tractability of NUD60 compared to UD10 is clearly demonstrated in Tables 3.15–3.17, which show that NUD60 required, on average, 67% less variables, 66% less constraints, and produced 89% lower optimality gap at the root node relaxation compared to UD10. Similarly, NUD30 also produced, on average, 84% less RCD, required 54% less variables and constraints, and produced, on average, 53% lower optimality gap at the root node relaxation. However, NUD30 provided the same quality of solutions as UD10. While UD30 required around the same amount of time to solve as NUD30, and NUD30 required 34% less variables and 33% less constraints, UD30 produced, on average, 51% and 48%

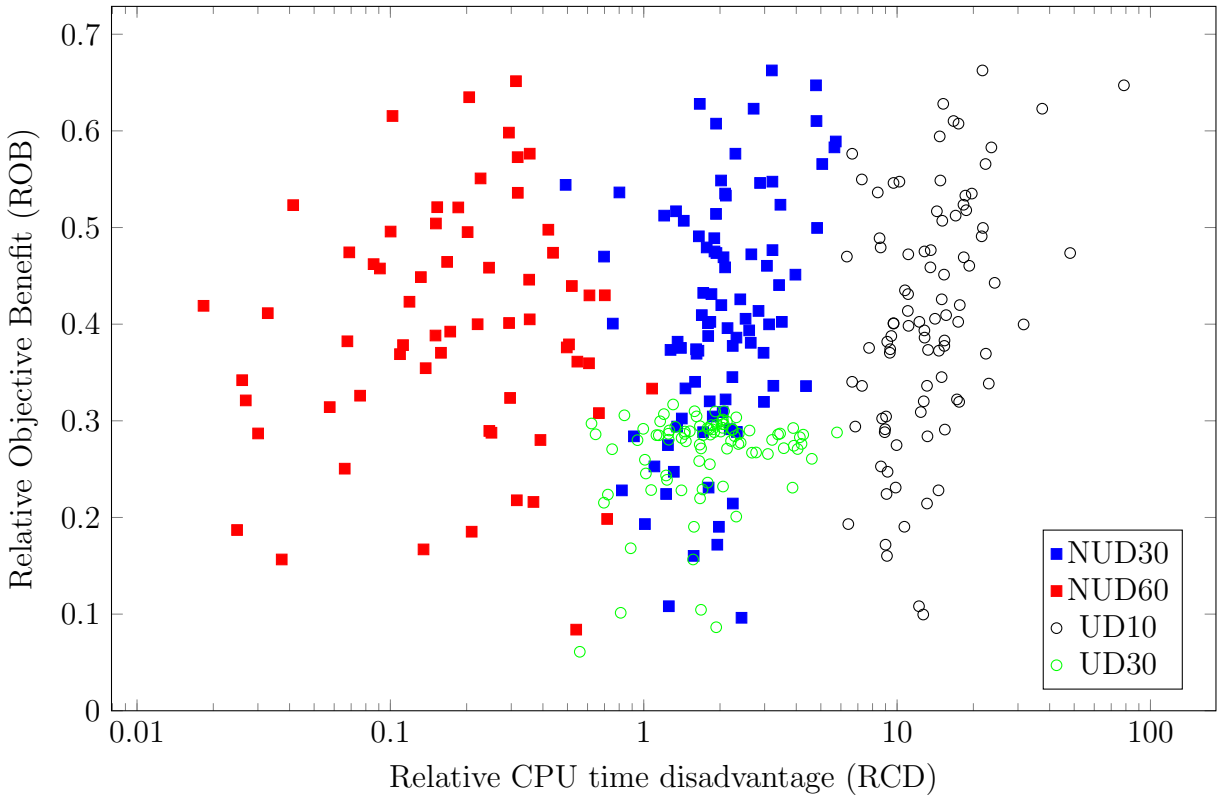


Figure 3.4: Comparison of different discretization schemes

Table 3.11: Mean Relative Objective Benefit (ROB) over UD60

Number of Tasks	Horizon (h)	NUD30	NUD60	UD10	UD30	UD60 ObjVal
100	24	0.28	0.28	0.28	0.22	10,021
100	32	0.29	0.28	0.29	0.24	10,189
100	40	0.29	0.29	0.29	0.25	10,160
150	24	0.40	0.39	0.40	0.28	10,705
150	32	0.46	0.45	0.46	0.29	10,539
150	40	0.45	0.44	0.45	0.29	10,660
200	24	0.47	0.47	0.47	0.27	11,080
200	32	0.55	0.54	0.55	0.28	10,957
200	40	0.45	0.44	0.45	0.27	11,122

Table 3.12: Standard Error of Mean (SEM) of ROB over UD60

Number of Tasks	Horizon (h)	NUD30	NUD60	UD10	UD30
100	24	0.03	0.03	0.03	0.03
100	32	0.04	0.04	0.04	0.02
100	40	0.03	0.03	0.03	0.02
150	24	0.03	0.03	0.03	0.01
150	32	0.03	0.03	0.03	0.01
150	40	0.02	0.02	0.02	0.00
200	24	0.03	0.03	0.03	0.01
200	32	0.03	0.02	0.03	0.00
200	40	0.03	0.03	0.03	0.01

lower ROB compared to NUD30 and NUD60, respectively, as reported on Table 3.11. Furthermore, while NUD60 required merely 3% less variables and constraints compared to UD30, NUD60 produced, on average, 92% lower RCD, since NUD60 led to much tighter problems with 85% lower optimality gap at the root node relaxation. These results clearly demonstrate the superiority of non-uniform discretization schemes over uniform discretization schemes when handling large problems of this nature.

By inspecting Table 3.11, we can also observe the relative advantage of finer discretization schemes (NUD30, NUD60, and UD10) becoming more significant compared to the coarser discretization schemes (UD30 and UD60) as higher numbers of tasks are consid-

Table 3.13: Mean Relative CPU time disadvantage (RCD) over UD60

Number of Tasks	Horizon (h)	NUD30	NUD60	UD10	UD30	UD60 CPU time (s)
100	24	1.90	0.19	10.30	1.77	7
100	32	2.22	0.40	12.41	2.21	6
100	40	1.47	0.12	9.84	1.85	8
150	24	2.26	0.06	14.06	2.04	11
150	32	1.97	0.15	13.04	1.59	12
150	40	2.06	0.04	14.20	2.14	12
200	24	3.18	0.16	28.63	2.47	14
200	32	2.91	0.18	16.71	1.97	15
200	40	2.17	0.08	14.73	1.85	17

ered. However, by inspecting Table 3.13, we can observe mean RCD values increasing noticeably with increasing number of tasks for UD10, while the increases in mean RCD values are not significant for NUD30, and the mean RCD values appear to decrease for NUD60 with increasing number of tasks. In addition, while Standard Error of Mean (SEM) values of ROB between NUD30, NUD60, and UD10 were very close as reported in 3.12, SEM values of RCD were much higher for UD10, as well as for UD30 to a certain degree, compared to NUD30 and NUD60 as reported in 3.14, and this gap in SEM values of RCD increased with increasing number of tasks. These results indicate that the gap between non-uniform discretization schemes and uniform discretization schemes only enlarges with increasing problem size, and that the non-uniform discretization schemes perform more consistently with regards to the CPU time.

Table 3.14: Standard Error of Mean (SEM) of RCD over UD60

Number of Tasks	Horizon (h)	NUD30	NUD60	UD10	UD30
100	24	0.20	0.06	0.77	0.36
100	32	0.31	0.12	1.57	0.40
100	40	0.12	0.06	0.69	0.26
150	24	0.26	0.06	0.79	0.22
150	32	0.28	0.08	1.05	0.24
150	40	0.36	0.05	1.95	0.42
200	24	0.49	0.08	6.56	0.34
200	32	0.43	0.09	2.68	0.30
200	40	0.28	0.07	1.15	0.35

Table 3.15: Mean optimality gap (%) at the root node relaxation

Number of Tasks	Horizon (h)	NUD30	NUD60	UD10	UD30	UD60
100	24	1.39	0.43	4.24	1.35	3.13
100	32	1.61	0.34	2.99	2.82	2.36
100	40	3.45	0.73	5.74	7.20	4.92
150	24	2.30	0.03	3.44	3.70	2.78
150	32	0.63	0.42	6.64	1.80	1.48
150	40	1.86	0.13	4.35	4.29	1.49
200	24	0.10	0.46	1.26	1.24	0.85
200	32	1.61	0.20	1.46	3.31	1.65
200	40	2.28	0.12	6.06	1.25	1.40

Table 3.16: Mean number of variables for the NUD and UD discretization schemes

Number of Tasks	Horizon (h)	NUD30	NUD60	UD10	UD30	UD60
100	24	116,866	84,933	255,501	87,501	45,501
100	32	118,271	86,267	257,165	88,071	45,797
100	40	117,108	84,977	256,377	87,801	45,657
150	24	174,885	127,150	382,112	130,861	68,048
150	32	175,747	127,959	382,930	131,141	68,194
150	40	175,564	127,694	383,134	131,211	68,230
200	24	232,277	168,636	508,256	174,061	90,512
200	32	232,670	169,290	507,818	173,911	90,434
200	40	233,297	169,570	509,366	174,441	90,710

Table 3.17: Mean number of constraints for the NUD and UD discretization schemes

Number of Tasks	Horizon (h)	NUD30	NUD60	UD10	UD30	UD60
100	24	61,829	45,054	134,525	46,637	24,665
100	32	62,537	45,727	135,362	46,927	24,818
100	40	61,953	45,079	134,966	46,790	24,746
150	24	91,272	66,596	198,264	68,750	36,372
150	32	91,706	67,004	198,675	68,893	36,447
150	40	91,615	66,872	198,778	68,929	36,466
200	24	120,400	87,771	261,768	90,782	48,036
200	32	120,595	88,097	261,547	90,706	47,995
200	40	120,914	88,242	262,326	90,976	48,138

3.3.4 Comparing uniform and non-uniform discretization: effects of processing time variability

The preceding computational studies clearly demonstrated the superiority of non-uniform discretization schemes compared to uniform discretization schemes in solving large instances of the nature of problems considered in this study. However, those instances only considered instances with the baseline processing time variability Γ of 161. Note that at Γ of 1, NUD30 and NUD60 would be equivalent to UD10, where 10 minutes is the smallest given processing time. Therefore, as Γ approaches 1 from the baseline value of 161, the relative performance superiority of NUD30 and NUD60 over UD10 as observed previously should gradually diminish. To observe whether or not the conclusions drawn previously for the baseline instances still holds at relatively lower values of Γ still exceeding 1, the set of instances with 100 tasks and 24 hour scheduling horizon as used in Section 3.3.3 were re-run at Γ values of 40 and 100. To set the Γ value to 40, for example, minimum processing time was kept at 10 minutes, and any processing times exceeding 400 minutes were set to 400 minutes. At $\Gamma = 40$, we can still observe superior efficiency of the non-uniform discrete schemes compared to the uniform discrete schemes, as shown in Figure 3.5.

Since we are interested in observing how the relative performance superiority of NUD30 and NUD60 over UD10 changes with changing Γ, the mean ROB values at their corresponding Γ values, as well as their SEM values given as error bars, are presented in Figure 3.6 instead. The mean ROB values at each Γ value were spaced out horizontally for readability. Note that in Figure 3.6, the mean ROB values for NUD30, NUD60, and UD10 appear close to each other for $\Gamma = 161$. For $\Gamma = 40$, there is clearly a gap between NUD60 and the finer discretizations, NUD30 and UD10. This indicates that the superiority of non-uniform discretization schemes becomes less pronounced with lower processing time variability. When large CPU time limits can be accepted, some of the finer uniform discretization schemes may be more appealing than some of the coarser non-uniform discretization schemes, especially for instances with relatively lower numbers of tasks given the observations made in Section 3.3.3. Similarly, for problems with low processing time variability, finer non-uniform discretization schemes may be more desirable than coarser non-uniform discretization schemes.

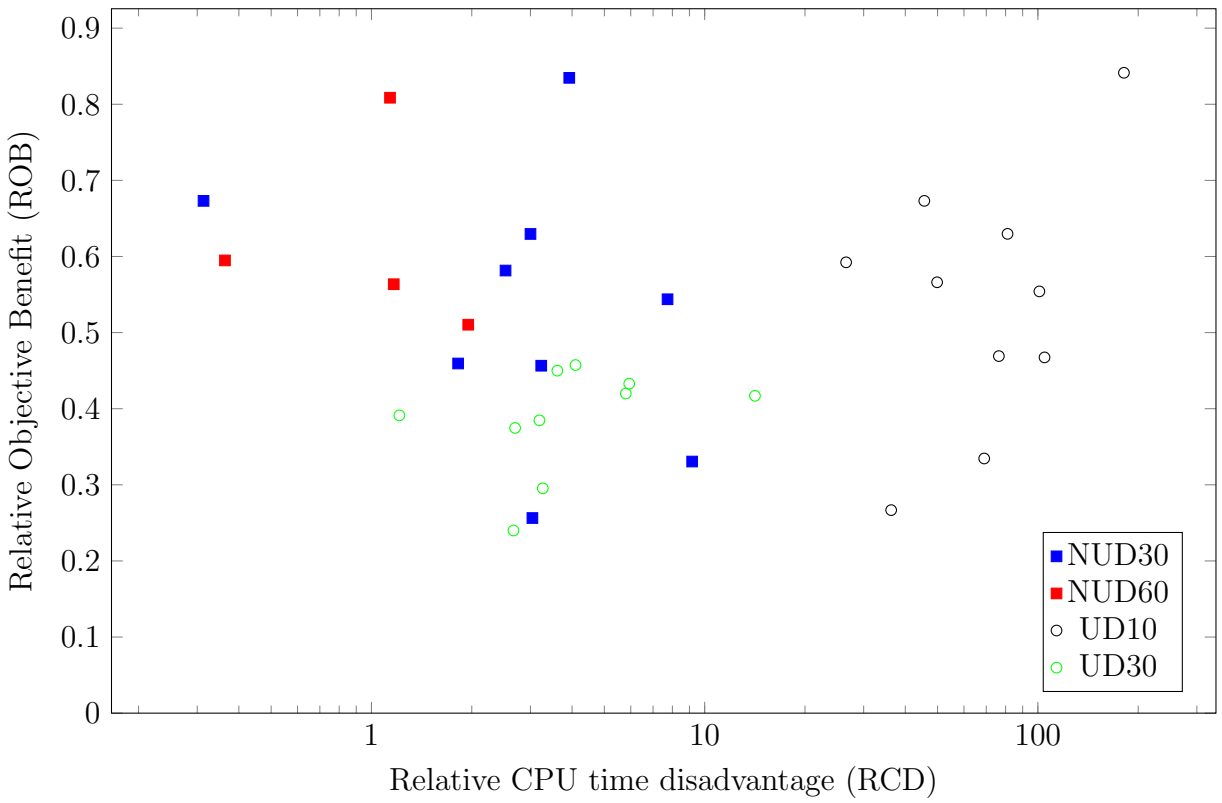


Figure 3.5: NUD vs UD at processing time variability Γ =40 ($|I|$ = 100 and H=24 h)

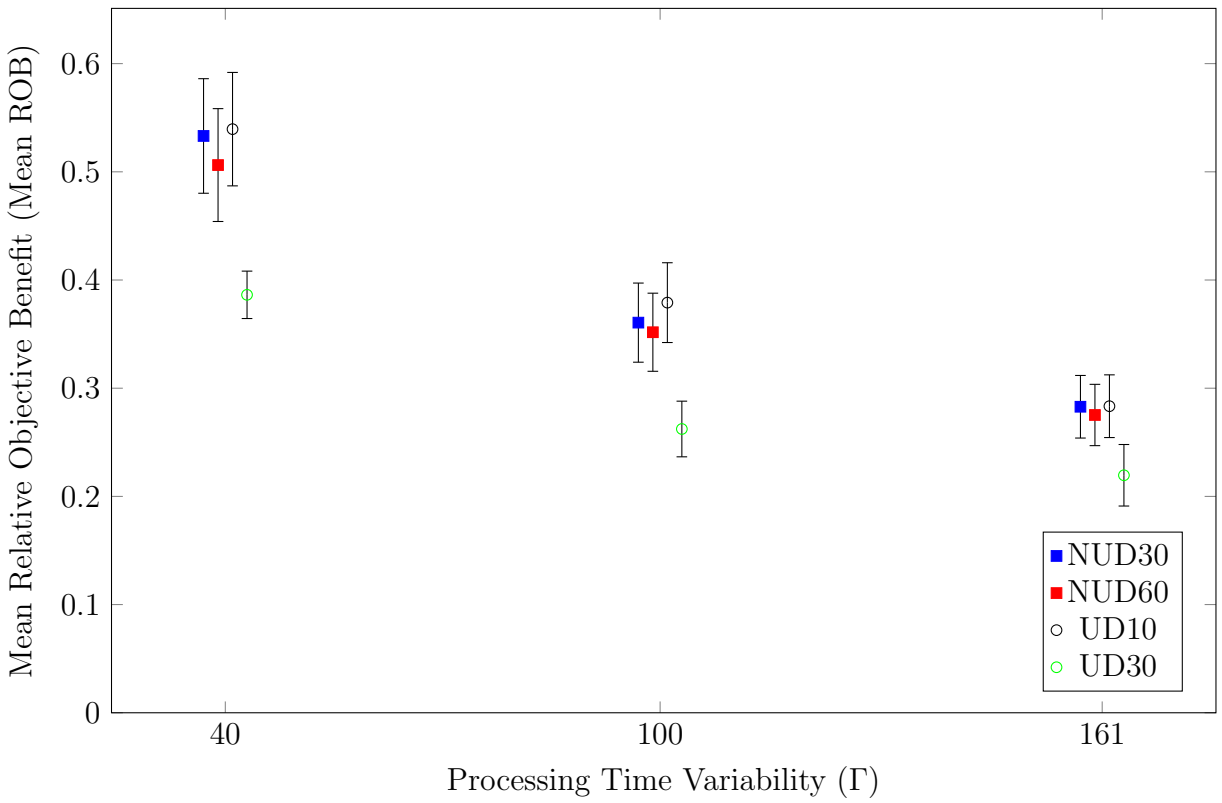


Figure 3.6: Trend of change in Mean ROB with changing processing time variability Γ

3.4 Chapter summary

This chapter presented comparisons between existing multitasking discrete-time and continuous-time formulations, as well as comparison between different discretization schemes for the flexible-discrete time formulation. The continuous-time formulation considered in this study appeared not to be suitable for handling industrial-sized problems for the problem of interest for this thesis due to a detrimental trade-off between solution quality and CPU time. On the other hand, the non-uniform discretization schemes exhibited good balance between solution quality and CPU time, especially NUD60.

Chapter 4

Study of Conflicting Objective Functions

This chapter presents the study of short-term scheduling problems with conflicting objectives, focusing on *a priori* approaches as discussed in Section 2.3. The aim of this study is to develop and explore practical approaches for handling conflicting objectives in short-term scheduling using a rolling horizon framework. In addition, this study will also explore different objective function formulations that can account for TTP and AMS in short-term scheduling. These two issues will be studied within the context of two application settings: a semiconductor manufacturing plant and the actual analytical services facility also studied in Chapter 3.

In this chapter, we utilize the non-uniform discretization approach as the results of the study presented in the previous chapter showed that this modeling approach is effective in solving large-scale integer linear programming (ILP) short-term scheduling problems where multitasking is a key operational feature. In contrast to the last chapter, however, the study in this chapter models the entire network of the facility containing over 100 unique processing units, where as the previous chapter only considered a portion of it due to tractability concerns for the computationally demanding approaches. The problem size is increased in this chapter given that this thesis ultimately aims to present methods for solving real-life industrial-sized problems.

The organization of this chapter is as follows. Section 4.1 presents two new methods considered in this work to address multi-objective problems in short-term scheduling. The

specific objective functions to address TTP and AMS in short-term scheduling are presented in Section 4.2. Extensions to the flexible discrete-time formulation presented in Section 3.2.1 required to incorporate the rolling horizon framework and operational characteristics of the semiconductor problem are presented in Section 4.3. The two case studies featuring multi-objective problems for short-term scheduling are presented in Section 4.4. Finally, a summary of the chapter is provided in Section 4.5.

4.1 Multi-objective a priori scalarization methods

In this section, we first present a reference point method of relevance to the present study, i.e., minimizing the 1-norm distance to the utopia point (1NM method). We then present two new methods that are based on the hybridization of the 1NM method and the ε-constraint method.

4.1.1 1-norm minimization (1NM) method

Assume χ is the set of constraints, $\vec{x}$ is the vector of decision variables, and f_1, f_2 are functions that we wish to maximize/minimize, respectively. In this study, we choose as the reference point the utopia point at the coordinate $(UB(f_1), LB(f_2))$ as shown on Figure 4.1. The coordinates $(UB(f_1), UB(f_2))$ and $(LB(f_1), LB(f_2))$ represent the upper and lower bounds of the output efficiency range, which are computed by solving the single-objective problems $\max\{f_1(\vec{x}) : \vec{x} \in \chi\}$ and $\min\{f_2(\vec{x}) : \vec{x} \in \chi\}$, respectively. A compromise solution on the trade-off surface is obtained by minimizing a p-norm distance (δ_p) from the utopia point[147]. This approach is also referred to as compromise programming, and it is considered to be a special case of the reference point methods[41;99].

In the *a priori* compromise programming approach, the DM's preference is expressed in the selection of the p-norm distance to minimize. In the 1NM method, we minimize the 1-norm distance (δ_1) to the utopia point, and we refer to the obtained compromise solution as the 1NM Point. For the following bi-objective problem:

$$\begin{aligned} &\max\ f_1(\vec{x}) \\ &\min\ f_2(\vec{x}) \\ &s.t.\ \vec{x} \in \chi \end{aligned} \tag{4.1}$$

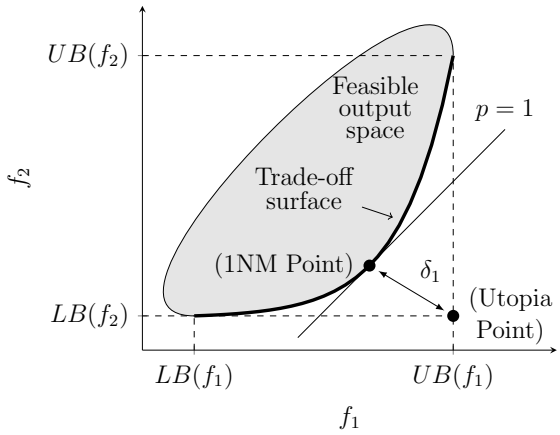


Figure 4.1: 1-norm minimum trade-off solution on trade-off surface

the 1NM Point is computed by solving the following aggregated and scalarized problem[55]:

$$\begin{aligned} &\min\ (1 - \hat{f}_1(\vec{x})) + \hat{f}_2(\vec{x}) \\ &s.t.\ \vec{x} \in \chi \end{aligned} \tag{4.2}$$

where $\hat{f}_1(\vec{x})$ and $\hat{f}_2(\vec{x})$ are normalizations of $f_1(\vec{x})$ and $f_2(\vec{x})$:

$$\hat{f}_1(\vec{x}) = \frac{f_1(\vec{x}) - LB(f_1)}{UB(f_1) - LB(f_1)} \qquad \hat{f}_2(\vec{x}) = \frac{f_2(\vec{x}) - LB(f_2)}{UB(f_1) - LB(f_2)} \tag{4.3}$$

The 1NM method thus has the advantage of simplifying decision making by providing the DM with a single ideal compromise solution, which best approximates the unobtainable utopia point. The 1NM method is also guaranteed to return a non-dominated solution even if the output space is nonconvex[147]; also, this 1-norm based approach leads to a linear formulation given linear objective functions and constraints as shown in problem (4.2). However, one downside of the 1NM method is that the DM preferences are inherently assumed, i.e., $p = 1$, and the DM has no freedom to express other preferences. In general, the articulation of the DM preferences can be challenging given that $1 < p < \infty$ norms lead to nonlinear aggregating functions, and a ∞-norm solution may be a dominated solution for a nonconvex output space[147], which are more common in pure integer problems[55]. Furthermore, in practice, the DM may want to express their preferences in terms of specific objectives, rather than in terms of the p-norm. Therefore, we present in this work two

methods to address these shortcomings, which are presented next.

4.1.2 Modified ε-constraint method (Mod-ε)

Another option of *a priori* method is to compute the Pareto frontier (a set of non-dominated solutions), and choose one trade-off solution to implement according to a rule. In fact, the 1NM method can be considered to be an application of such a rule to select the 1NM Point within the Pareto frontier. In this work, it is desired to consider different solutions that may be preferred by the DM than the 1NM Point. One alternative way to devise an *a priori* approach is to compute the Pareto frontier (e.g., using the ε-constraint method) and having a pre-determined rule on how to choose a point from it. However, such approach is typically computationally expensive. In this Section, we propose a non-iterative hybrid method between the 1NM method described in Section 4.1.1 and the ε-constraint method, referred henceforth as the Mod-ε method. We use a variant of *a posteiori* ε-constraint method presented by Özlen and Azizolu[117] as the basis given that it is adapted for integer programming problems as we would like to be able to solve in this work. We propose two options for searching of the trade-off solution: search for a trade-off solution in the direction of increasing objective values (i.e., from the 1NM Point towards (UB(f_1), UB(f_2)) in Figure 4.1, or in the decreasing search direction (i.e., from the 1NM Point towards (LB(f_1), LB(f_2)) in Figure 4.1). We present the Mod-ε method below for the bi-objective problem (4.1):

1. Compute the bounds of the output efficiency range: LB(f_1), LB(f_2), UB(f_1), UB(f_1).
2. Compute the 1NM Point, $\vec{x}^*$ (i.e. solve problem (4.2)).
3. Set weight $w_1 = \frac{1}{UB(f_1)-LB(f_1)+1}$ for the f_1 objective (increasing search direction), or set $w_2 = \frac{1}{UB(f_2)-LB(f_2)+1}$ for the f_2 objective (decreasing search direction).
4. Given the *a priori* DM preference value, $0 < \Upsilon < 1$ ($\Upsilon = 0$ and $\Upsilon = 1$ correspond to the 1NM Point and either the lower or upper bound of the output efficiency range, respectively), and the 1NM Point, $\vec{x}^*$, set bound $\varepsilon_1 = f_1(\vec{x}^*) + \Upsilon(UB(f_1) - f_1(\vec{x}^*))$ for the f_1 objective (increasing search direction), or set bound $\varepsilon_2 = f_2(\vec{x}^*) - \Upsilon(f_2(\vec{x}^*) - LB(f_2))$ for the f_2 objective (decreasing search direction).
5. Solve the constrained weighted problem (4.4) (increasing search direction), or problem (4.5) (decreasing search direction). This returns the final DM preferred solution

in the desired search direction.

$$\begin{aligned} \min\ & f_2(\vec{x}) - w_1 f_1(\vec{x}) \\ s.t.\ & f_1(\vec{x}) \geq \varepsilon_1 \\ & \vec{x} \in \chi \end{aligned} \qquad (4.4) \qquad\qquad \begin{aligned} \max\ & f_1(\vec{x}) - w_2 f_2(\vec{x}) \\ s.t.\ & f_2(\vec{x}) \leq \varepsilon_2 \\ & \vec{x} \in \chi \end{aligned} \qquad (4.5)$$

Note that the weights w_1 and w_2 shown above are expressed such that the solution of problem (4.4) or (4.5) are bi-objective efficient for an integer programming problem (see Theorems 2.1-2.3 of Özlen and Azizolu[117]). Therefore, in this work, we limit the discussion and implementation of the Mod-ε method to integer programming problems.

The motivation for the proposed approach is that the DM preferred solution is typically close to the 1NM Point; hence, we express the DM preferences relative to the 1NM Point and the bounds of the output efficiency range. Expression of DM preferences and the non-iterative structure are the major differences between the Mod-ε method and the iterative *a posteriori* approach of Özlen and Azizolu[117], which lacks any expression of DM preferences; also, we aim to reduce the time required to obtain a good practical solution using the Mod-ε method through these differences. The Mod-ε method also addresses the concerns regarding compromise programming raised in Section 4.1.1, as the constrained weighted problems (4.4) and (4.5) maintains linearity for linear constraint $\vec{x} \in \chi$.

Furthermore, expressing the DM preferences in Υ with respect to the 1NM Point and the bounds of the output efficiency range can potentially give the DM greater control over the degree of trade-off between the conflicting objectives compared to some arbitrary selection of p-norm in compromise programming or the value of ε in the ε-constraint method. Note that if we consider the 1NM Point to be the ideal attainable compromise between the conflicting objectives, then moving further away from the 1NM Point along the Pareto frontier (i.e., higher Υ) would typically mean greater deviation from ideal trade-off. In fact, the Mod-ε method can be considered as an extension of the 1NM method, which allows the DM to express *a priori* the degree of deviation from ideal objective trade-off the DM is willing to tolerate in order to prioritize one objective over the other, without having to compute the entire Pareto frontier.

4.1.3 1NM ε-constraint method (1NM-ε)

In this section, we propose the 1NM ε-constraint method (1NM-ε), which has a similar structure to the Mod-ε method described in the previous section. While the motivation for this method is the same, i.e. that the DM's preference would be close to the 1NM Point, the major difference is that rather than solving the constrained weighted problem (4.4) or (4.5), we search for the compromise solution with the lowest 1-norm distance on the ε-constrained Pareto frontier that excludes at least the 1NM Point. We present the 1NM-ε method below for the bi-objective problem (4.1):

1. Compute the bounds of the output efficiency range: LB(f_1), LB(f_2), UB(f_1), UB(f_1).
2. Compute the 1NM Point, $\vec{x}^*$ (solve problem (4.2)).
3. Given the *a priori* DM preference value, $0 < \Upsilon < 1$, and the 1NM Point, $\vec{x}^*$, set bound $\varepsilon_1^+ = f_1(\vec{x}^*) + \Upsilon(UB(f_1) - f_1(\vec{x}^*))$ for the f_1 objective and set bound $\varepsilon_2^+ = f_2(\vec{x}^*)$ for the f_2 objective (increasing search direction); alternatively, set bound $\varepsilon_2^- = f_2(\vec{x}^*) - \Upsilon(f_2(\vec{x}^*) - LB(f_2))$ for the f_2 objective and set bound $\varepsilon_1^- = f_1(\vec{x}^*)$ for the f_1 objective (decreasing search direction).
4. Solve the 1-norm minimization problem (4.6) (increasing search direction), or problem (4.7) (decreasing search direction). This returns the final DM preferred solution in the desired search direction.

$$\begin{aligned} \min\ & (1 - \hat{f}_1(\vec{x})) + \hat{f}_2(\vec{x}) \\ s.t.\ & f_1(\vec{x}) \geq \varepsilon_1^+ \\ & f_2(\vec{x}) \geq \varepsilon_2^+ \\ & \vec{x} \in \chi \end{aligned} \tag{4.6}$$

$$\begin{aligned} \min\ & (1 - \hat{f}_1(\vec{x})) + \hat{f}_2(\vec{x}) \\ s.t.\ & f_1(\vec{x}) \leq \varepsilon_1^- \\ & f_2(\vec{x}) \leq \varepsilon_2^- \\ & \vec{x} \in \chi \end{aligned} \tag{4.7}$$

By solving a 1-norm minimization problem in its last step rather than solving a constrained weighted problem like the Mod-ε method, the 1NM-ε method has a larger emphasis on obtaining a good trade-off between the conflicting objectives compared to the Mod-ε method. This behavior arises from the fact that for the Pareto frontier of the 1NM-ε method, moving from the 1NM Point towards either the upper or lower bound of the output efficiency range, we would inherently observe, by design, a continuous increase in the normalized 1-norm distance ($\hat{\delta}_1$). This is not necessarily the case for the Mod-ε method, and it is possible to observe localized decrease in $\hat{\delta}_1$ (i.e., localized decrease in trade-off quality). Therefore, the Pareto frontier of the 1NM-ε method is either equivalent

to the Pareto frontier of the Mod-ε method, or a subset of the Mod-ε Pareto frontier that excludes localized decrease in $\hat{\delta}_1$. However, this does not necessarily mean that the 1NM-ε method would always provide results that are more preferable to the DM. For example, given the same problem and using the same value of Υ, the Mod-ε method cannot return higher values for f_1 and f_2 than the 1NM-ε method in the direction of increasing f_1 and f_2 from the 1NM Point if the 1NM-ε Pareto frontier is a smaller subset of the Mod-ε Pareto frontier; however, the DM might prefer the Mod-ε solution with lower values of f_1 and f_2, even if the corresponding 1NM-ε solution has lower 1-norm distance to the utopia point. Furthermore, if the two methods have the same Pareto frontier, then their solutions would also be the same using the same value of Υ.

4.2 Short-term objective functions

As discussed in the introduction, maximizing the total throughput (TTP) and minimizing the average makespan (AMS) can be challenging due to the difference in time scale between tactical level decision making (TTP/AMS) and short-term scheduling. In order to address this issue of approximating TTP and AMS for short-term scheduling, we present here a rolling horizon approach for short-term scheduling applications. While the present approaches have been tested on two particular applications, i.e., a semiconductor plant and the analytical services facility introduced in Chapter 3, a wide variety of industrial applications require this approach to schedule operations in an optimal fashion, e.g., steelmaking[94] and industrial gases supply chain[150]. In order to provide the necessary rolling horizon context for the objective functions proposed in this work, we first define our rolling horizon framework in Section 4.2.1 before presenting the alternative objective functions in Section 4.2.2. In section 4.2.1, we also introduce the concept of operational modes in order to incorporate the characteristics of the semiconductor plant. The notation used in the remainder of this section is consistent with the notation introduced in Sections 3.1 and 3.2.1.

4.2.1 Rolling horizon framework and operational modes

For the problems considered in this study, a facility receives a set of tasks I that need to be processed within a rolling scheduling horizon (RH). A predetermined number of constituent horizons (CH), each with a scheduling horizon of length H, form the overall RH.

Within the context of this study, a CH represents a single day of operations (e.g., there can be 30 CHs (days) within the RH (month), with $H = 8$ hours or 24 hours). Each task i is introduced at the beginning of a CH, but not necessarily at the beginning of the RH, and at the beginning of a path $\wp_i$.

For the semiconductor case study of Section 4.4.1, we introduce the concept of operational modes in order to allow for potentially different processing times for each processing unit. For each task i and its path $\wp_i$, there is a specific sequence of operational modes $\{m^i_1, \ldots, m^i_{n(i)}\}$, denoted by M_i, and mode m^i_k dictates the length of processing time $\tau_{p^i_k m^i_k}$. All machines in a processing unit p have $|\Psi_p| \geq 1$ possible predetermined values of processing times τ_{pm}, where Ψ_p denotes the set of one or more possible operational modes for p, $\Psi_p = \{1, \ldots, |\Psi_p|\}$.

Based on this description of the operational modes, we also present the following modification to the NUD60 discretization scheme: $\mathcal{E}(p) = (ST, \varphi_{p1}, \varphi_{p2}, \ldots, \varphi_{p(|\mathcal{E}(p)|-3)}, \varphi_{p(|\mathcal{E}(p)|-2)}, ST + H)$ represents the unit-specific sequence of time points of length $|\mathcal{E}(p)|$ for processing unit p along the axis of time for a CH starting at time ST and ending at time $ST + H$ within the RH, where φ_{pt} represents the actual time value of the t^{th} time point for processing unit p. In this work, we set the difference in time values of two consecutive to be the minimum value between the greatest common divisor of $(\tau_{p1}, \ldots, \tau_{p|\Psi_|})$ and 60. This modified NUD60 scheme is used in the semiconductor case study presented in Section 4.4.1.

For the problems being considered in this study, we allow a machine $j \in J_{p^i_k}$ to continue to process materials for task i during the gap between the CHs, if any, as long as the machine was turned on for mode m^i_k during or at the end of the CH. If the gap is long enough for the machine to complete processing before the beginning of the next CH, then machine j will become available to process more materials at the beginning of the following CH, and the materials that completed processing during the gap will become available to be processed at $(k+1)^{th}$ process in its path if $k < n(i)$. Otherwise, the machine will be occupied, and the materials unavailable for further processing until processing has been completed at that machine. At the beginning of the RH, machines in J may be occupied, or not, depending on the specific operating conditions of the facility.

4.2.2 Objective functions

Objective functions for throughput maximization (f_1)

Within the context of this study, we define the throughput of a task i (TP_i) as the cumulative sum of the amount of materials for task i, which have either started or completed being processed at the last processing unit in its path, $p^i_{n(i)}$ for the RH. We also define the total throughput as $TTP := \sum TP_i,\ \forall i \in I$, and a key objective is to maximize TTP for the RH. However, simply maximizing the amount of materials starting from the last process $\sum_{i\in I}\sum_{t=1}^{|\mathcal{E}(p^i_{n(i)})|} B_{i,n(i),t}$ for each CH does not necessarily maximize TTP for the RH. For instance, if none of the materials can complete the last process in their respective paths during the CH under consideration, which may happen, for example, at the beginning of a production campaign, the value of such objective function would be 0 for all feasible solutions. In such instance, a schedule that does not process any materials during the CH would be considered to be just as good as any other feasible schedule. Therefore, we propose operational throughput maximization objective functions (i.e. f_1 in problem (4.1)) of the following form:

$$\max \sum_{i\in I}\sum_{k=1}^{n(i)}\sum_{t=0}^{|\mathcal{E}(p^i_k)|-1} (weight) B_{ikt} \tag{4.8}$$

The above objective function uses the weights presented in Table 4.1, each with an identifying f_1ID, which attempt to produce schedules with close to optimal TTP for the RH. In this work, we consider the optimal TTP for the RH to be the TTP that could be achieved by explicitly maximizing the TTP by solving a single large problem with a scheduling horizon equivalent to the length of the RH. Note that objective functions of the form shown in equation (4.8) can easily account for other problem specific details, such as rush jobs, by adding another weight factor in front of B_{ikt} to prioritize tasks with higher priority.

Table 4.1: Proposed weights for the throughput maximization objective functions

f_1ID	$weight$	Description/Rationale
0	1	Maximizes the sum of all materials starting to be processed at all processing units for all tasks during the CH.
1	$\frac{k}{n(i)}$	Provides higher priority to materials starting to be processed at processing units later in the path $\wp_i$, relative to the units earlier in $\wp_i$ (Closer approximation to TTP for the CH than using the weight of 1)
2	$\left(\frac{k}{n(i)}\right)^2$	Shifts the priority towards the end of $\wp_i$ compared to $\frac{k}{n(i)}$.
3	$\frac{k}{\sum_{j=1}^{n(i)} j}$	If there are no materials for task i, which started being processed during the CH, but did not arrive at the last processing unit by the end of the CH, then $\sum_{k=1}^{n(i)} \sum_{t=1}^{\lvert\mathcal{E}(p_k^i)\rvert} \frac{k}{\sum_{j=1}^{n(i)} j} B_{ikt} = TP_i$ for the CH (Closer approximation to TPP for the CH than other weights where $\sum_{k=1}^{n(i)} \sum_{t=1}^{\lvert\mathcal{E}(p_k^i)\rvert} (weight) B_{ikt} > TP_i$ for the CH).
4	$\frac{\sum_{j=1}^{k} \tau_{p_k^i m_k^i}}{\sum_{j=1}^{n(i)} \tau_{p_k^i m_k^i}}$	Similar to the ratio of the length of $\wp_i$ up to and including p_k^i to the total length of $\wp_i$ in terms of processing times. Prioritizes processing units with longer processing times compared to other weights.

Objective functions for makespan minimization

Within the context of this study, we consider task i to be completed when there are no materials waiting to start being processed at p_k^i, $1 \leq k \leq n(i)$, and we define the makespan of task i (MS_i) as the time elapsed since the task was made available (AT_i) to be processed at p_1^i until the expected time of completion of $p_{n(i)}^i$ for the last remaining materials in task i. For example, if a batch of material unit size 5 became available to be processed at the first processing unit on day 1 at 9 hours, and all 5 units of material reached the last processing unit requiring 1 hour of processing time on day 3 at 15 hours (i.e. 3 p.m.), then the makespan of this task is {24 hours/day $\times$2 days +(15 − 9) hours +1 processing hour = 55 hours}. Given this definition for MS_i, we define the average makespan (AMS) for a given set of tasks I to be the arithmetic mean of MS_i for tasks $i \in I$ that are completed. For short-term scheduling problems where the system is sufficiently constrained, or one or more tasks have paths that are longer than the length of the scheduling horizon, explicitly minimizing AMS as the objective function may be undesirable. For instance, if there are no tasks that can be completed within the CH, then the value of such objective function would be 0 for all feasible solutions. Given the above considerations, we propose the following operational makespan minimization objective functions (i.e. f_2 in problem (4.1)) of the following form:

$$\min \sum_{i \in I} \sum_{k=1}^{n(i)} (ST - AT_i + \sum_{j=k}^{n(i)} \tau_{p_j^i m_j^i})(variable) \tag{4.9}$$

The above objective function uses the variables presented in Table 4.2, each with an identifying f_2ID, which attempt to produce schedules with close to optimal AMS for the RH. In eq (4.9), ST represents the start time of the CH. Note that in contrast to the f_1 objective function, the f_2 objective functions are differentiated using different decision variables, rather than different weights.

The constraints (4.10) and (4.11) shown below define the binary decision variable S_{ik} for f_2ID = 0 and f_2ID = 1, respectively. When implementing the objective functions, f_2ID = 0 and f_2ID = 1, S_{ik} must be introduced as a decision variable, in addition to the other decision variables, and the respective constraint (4.10) or (4.11) must be added to the scheduling model according to the f_2ID.

$$\frac{W_{ik|\mathcal{E}(p_k^i)|}}{1 + \sum_{j=1}^{n(i)} \sum_{t=1}^{|\mathcal{E}(p_k^i)|} a_{ijt}} \leq S_{ik} \leq W_{ik|\mathcal{E}(p_k^i)|} \quad \forall i \in I, k = 1, \ldots, n(i) \tag{4.10}$$

Table 4.2: Proposed weights for the makespan minimization objective functions (f_2)

f_2ID	*variable*	Description/rationale
0	S_{ik}, eq (4.10)	$S_{ik} = 1$ if processing unit p_k^i has materials waiting to be processed at the end of the CH; otherwise, $S_{ik} = 0$. Promotes materials to arrive at the last processing unit together, since a task is not completed until all materials get to the end.
1	S_{ik}, eq (4.11)	$S_{ik} = 1$ if processing unit p_k^i has materials waiting to be processed at the end of the CH, or materials that started being processed at p_k^i and expected to continue being processed at p_k^i after the end of the CH; otherwise, $S_{ik} = 0$. Penalizes in-process materials for being at earlier processing units rather than later, in addition to waiting materials.
2	$W_{ik\|\mathcal{E}(p_k^i)\|}$	Minimizes the amount of materials waiting to be processed at each processing unit at the end of the CH. Higher priorities are given to tasks with more waiting samples.

$$\frac{W_{ik|\mathcal{E}(p_k^i)|} + \sum_{t'=1,\ldots,|\mathcal{E}(p_k^i)|:\varphi_{p_k^i t'}+\tau_{p_k^i m_k^i}>\varphi_{p_k^i|\mathcal{E}(p_k^i)|}} B_{ikt'}}{1 + \sum_{j=1}^{n(i)} \sum_{t=1}^{|\mathcal{E}(p_k^i)|} a_{ijt}} \leq S_{ik} \leq W_{ik|\mathcal{E}(p_k^i)|} + \sum_{t'=1,\ldots,|\mathcal{E}(p_k^i)|:\varphi_{p_k^i t'}+\tau_{p_k^i m_k^i}>\varphi_{p_k^i|\mathcal{E}(p_k^i)|}} B_{ikt'} \quad \forall i \in I, k = 1,\ldots,n(i) \tag{4.11}$$

For each f_2, the general approach is to consider as weight, for each task $i \in I : AT_i \leq ST$ and process unit p_k^i, $1 \leq k \leq n(i)$, the sum of elapsed time since AT_i to ST and the total processing time remaining in path $\wp_i$, then provide incentive for task completion by promoting materials to progress through $\wp_i$ during the CH. This weight prioritizes processing of materials at process units that are earlier in $\wp_i$ compared to the units that are later in $\wp_i$, and completion of tasks that are introduced earlier and tasks with longer paths.

4.3 Modifications to the flexible discrete-time formulation

The modifications made to the scheduling model presented in Section 3.2.1 incorporating operational modes and the rolling horizon framework described in Section 4.2.1 are presented below.

While B_{ikt} and W_{ikt} did not change from the original formulation presented in Section 3.2.1, the index, m was added to X_{pmt} (the number of machines being used) to account for the different operational modes of processing unit p and redefined below.

X_{pmt}: the number of machines from processing unit p that are operating in mode m and being used at time point t, $\forall\ p \in P,\ m \in \Psi_p,\ t = 0, \ldots, |\mathcal{E}(p)| - 1$

Thus, capacity and equipment resource constraints, i.e. equations (3.2) and (3.3) from Section 3.2.1, are reformulated as follows:

$$\sum_{i,k:p=p_k^i} B_{ikt} \leq\ X_{pmt}\beta_p;\quad \forall\ p \in P,\ m \in \Psi_p,\ t = 0, \ldots, |\mathcal{E}(p)| - 1 \tag{4.12}$$

$$z_{pt} + \sum_{m \in \Psi_p} \sum_{\theta} X_{pm\theta} \leq R_p;\quad \forall\ p \in P,\ t = 0, \ldots, |\mathcal{E}(p)| - 1,$$
$$\theta \in \mathcal{E}(p) : \varphi_{pt} < \varphi_{p\theta} + \tau_{pm} \leq \varphi_{pt} + \tau_{pm} \tag{4.13}$$

β_p is the capacity of a machine in processing unit p that has R_p number of machines as defined in Section 3.1. In constraint (4.13), z_{pt} is an input parameter representing the number of machines for processing unit p that are occupied up until time point t. For a particular CH within the RH, the values of z_{pt} are calculated in pre-processing based on the solutions from preceding CHs (the number of machines turned on, the time at which a machine was turned on, and processing time of the machine for the given operational mode). For the very first CH in the RH, the values of z_{pt} are based on the status of the facility being considered. If all the machines in the facility are available at the beginning of the RH, then $z_{pt} = 0\ \forall\ p \in P,\ t = 0, \ldots, |\mathcal{E}(p)| - 1$. Otherwise, z_{pt} has a non-zero positive value for some or all $p \in P,\ t = 0, \ldots, |\mathcal{E}(p)| - 1$.

The flow conservation eq (3.4) from Section 3.2.1 has been reformulated as follows:

$$B_{ikt} + W_{ikt} = W_{ik(t-1)} + \sum_{\substack{\theta=1,\ldots,|\mathcal{E}(p^i_{k-1})|: \\ \varphi_{p^i_k t-1} < \varphi_{p^i_{k-1}\theta} + \tau_{p^i_{k-1} m^i_k} \leq \varphi_{p^i_k t}}} B_{i(k-1)\theta} + a_{ikt} \qquad (4.14)$$
$$\forall\ i \in I,\ k = 2, \ldots, n(i),\ t = 1, \ldots, |\mathcal{E}(p^i_k)| - 1$$

In eq (4.14), a_{ikt} is an input parameter representing the amount of materials from task i becoming available to be processed at processing unit p^i_k at time point t. For a particular CH and $a_{ikt} > 0,\ \forall\ i \in I, k > 1, t = 0, \ldots, |\mathcal{E}(p^i_k)| - 1$, a_{ikt} materials must have started being processing at p^i_{k-1} during one of the preceding CHs, and have completed being processed by time point t. Note that a_{ikt}, as an input parameter, does not account for materials that started being processed at p^i_{k-1} during the current CH, which are accounted for by the decision variable W_{ikt}.

4.4 Computational studies

We test the performance of our proposed multi-objective formulations using two different case studies. The first case study is based on the semiconductor processing case study presented by Senties et al.[127], whereas the second is a full industrial-sized case study of the analytical services scheduling problem previously reported in the literature[86;87;120]. For the different methods (single-objective, 1NM, Mod-ε and 1NM-ε), we compare the CPU time along with the total throughput (TTP) and the average makespan (AMS) as defined in Sections 4.2.2 and 4.2.2, respectively. For all rolling horizon (RH) results presented in Sections 4.4.1 and 4.4.2, we present mean CPU times and mean optimality gaps as mean across the entire RH for one constituent horizon (CH).

For all *a priori* Mod-ε and 1NM-ε implementations, we chose the DM preference value of $\Upsilon = 0.4$ to calculate the values of ε and $\varepsilon^{+/-}$ as defined in Sections 4.1.2 and 4.1.3. This value was chosen to obtain a trade-off solution about half way between the 1NM Point and the bounds of the output efficiency range, in order to produce results that were distinguishable from the single-objective implementations and the 1NM method. However, we did not want to potentially exclude the 'half way point' solution of $\Upsilon = 0.5$.

In order to simplify reference to specific objectives, we use the notation $f_1^{f_1 ID}$ and $f_2^{f_2 ID}$ when referring to f_1 and f_2 objectives with specific f_1ID weight and f_2ID variable as presented in Tables 4.1 and 4.2, respectively. For example, f_1^1 refers to f_1 objective function with $f_1\text{ID} = 1$ (weight shown in Table 4.1). For both case studies, we do not implement the Mod-ε method with f_1^1 and f_1^2 as these f_1 objectives lead to non-integer values, and the Mod-ε method is intended to be used for problems with integer-valued objectives as discussed in Section 4.1.2.

All reported CPU times include operations leading up to the final optimization run of each algorithm (i.e. model generation, solver pre-processing, obtaining bounds of the output efficiency range, and computation of 1NM Point). All implementations were performed using Julia programming language (0.6.0)[10] and JuMP modeling language (0.18.0)[37]. Optimization runs were performed using IBM ILOG CPLEX Optimizer (12.6.0)[71] on a shared Linux server with 250 GB of RAM and 4 CPUs, each with 12 cores and a processing speed of 2.4 GHz. For all optimization runs, we used a CPLEX solver time limit of 1 hour (unless stated otherwise) and 8 GB size limit on the MIP branch-and-cut tree[72].

4.4.1 Semiconductor case study

We solved two instances of the semiconductor case study adapted from the case study presented by Senties et al.[127], which contains 5 different product recipes (paths) and 12 processing units with a total of 14 equipment resources, where two of those processing units had two identical equipment resources in parallel. The network of the processing units defined by the product recipes are presented in Figure 4.2. The product recipes and the data on each processing unit are presented in Tables 4.1 and 4.2.

For this case study, we define a task as the group of wafer lots (the processed materials) with the same arrival time belonging to the same product. For the rolling horizon (RH) approach, we consider a constituent horizon (CH) to be a single day, and the plant is assumed to operate 24 hours/day (i.e. $H = 24$ hours) for each day in the RH without any interruption. The CH is discretized according to the modified NUD60 discretization scheme described in Section 4.2.1. The first instance is a problem containing 50 tasks consisted of 90 wafer lots. The second instance is a larger problem with 63 tasks consisted of 900 wafer lots. Full instance data are presented in Tables 4.3 and 4.4.

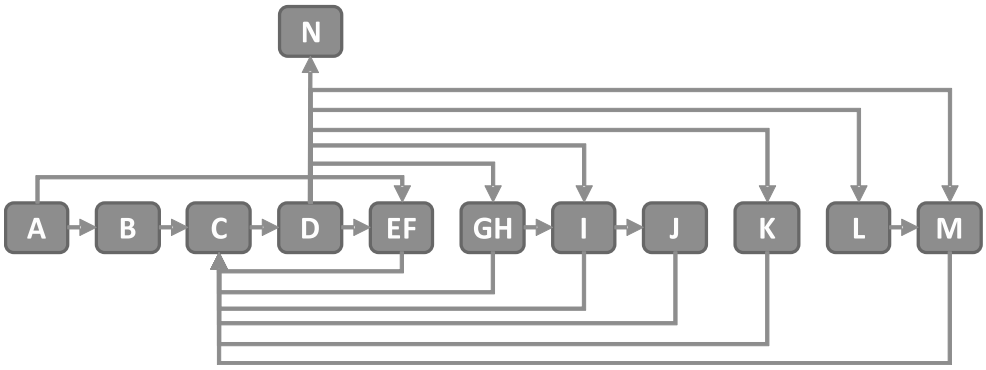


Figure 4.2: Map of the process network for the semiconductor case study

Table 4.1: Product recipes for the semiconductor case study

Product 1		Product 2		Product 3		Product 4		Product 5	
Proc.	Mode	Proc.	Mode	Proc.	Mode	Proc.	Mode	Proc.	Mode
A	1	A	1	A	1	A	1	A	1
EF	4	EF	1	B	3	B	1	B	2
C	1	C	1	C	1	C	1	C	1
D	1	D	1	D	1	D	1	D	1
GH	4	GH	3	EF	4	EF	3	EF	2
C	1	C	1	C	1	C	1	C	1
D	1	D	1	D	1	D	1	D	1
I	1	I	1	GH	2	GH	4	GH	1
J	1	J	2	C	1	I	1	I	1
C	2	C	2	D	1	C	2	C	2
D	2	D	2	I	1	D	2	D	2
L	1	L	2	J	3	K	1	K	2
M	1	M	1	C	2	C	2	C	2
C	1	C	1	D	2	D	2	D	2
D	2	D	2	L	3	M	1	M	1
N	1	N	1	M	1	C	1	C	1

Table 4.2: Process capacity, resources, and processing time information for the semiconductor case study (both instances)

Process	Number of Resources	Zone	Capacity [lots]	Processing times by lot [min]			
				Mode 1	Mode 2	Mode 3	Mode 4
A	1 (5)	Diffusion	1 (2)	120			
B	1 (5)	Diffusion	4 (8)	700	850	1000	
C	1 (5)	Photo	1 (2)	20	30		
D	1 (15)	Engraving	1 (2)	15	20		
EF	2 (10)	Diffusion	2 (4)	200	300	400	500
GH	2 (10)	Diffusion	2 (4)	400	500	600	700
I	1 (5)	Test	1 (2)	1			
J	1 (5)	Diffusion	2 (4)	350	400	500	
K	1 (5)	Diffusion	2 (4)	400	500		
L	1 (5)	Engraving	1 (2)	140	180	200	
M	1 (5)	Metal	1 (2)	120			
N	1 (5)	Metal	1 (2)	20			

Values for the number of resources and capacity for each resource for the second instance are given inside ( )

Table 4.3: Daily product demand for each task for the first semiconductor instance

Arrival Day	Daily product demands [lots]				
	Product 1	Product 2	Product 3	Product 4	Product 5
1	6	6	6	6	6
3	1	1		2	1
4	1	2		1	1
5	1	1	2	1	
6		1		3	1
7	1		2	1	1
8		1	2	1	1
9	1	2	1		1
10	2	1	2		
11		1	1	2	1
12		1	1		3
13	1		2	1	1
14	1	1	2		1

Table 4.4: Daily product demand for each task for the second semiconductor instance

Arrival Day	Daily product demands [lots]				
	Product 1	Product 2	Product 3	Product 4	Product 5
1	60	60	60	60	60
3	7	13	5	13	12
4	7	5	15	5	18
5	15	12	11	1	11
6	8	3	25	11	3
7	25		1		24
8	1	9	3	16	21
9	5	8	14	10	13
10	11	13	15	6	5
11	5	14	1	16	14
12	18	4	7	10	11
13	33	1	3	12	1
14	24	1	7	7	11

Rolling horizon results: semiconductor

For this case study, we compare TTP and AMS at the end of a 15-day RH given that for both instances, more than 80% of tasks were completed by the end of day 15 for all approaches. While more comprehensive results are presented in Appendix A, including the number of days required to complete all tasks for each approach and the AMS at 100% task completion, we present the key results and comparative analysis in this section.

First, for each instance, we identified the f_1ID and f_2ID for the single-objective max f_1 and min f_2 RH runs, which produced the highest TTP (TTP*) and lowest AMS (AMS*) at the end of the 15-day RH, as presented in Table 4.5. In order to perform easier comparisons of TTP and AMS for different approaches, we compute the relative differences in TTP (d_r(TTP)) and AMS (d_r(AMS)) for a particular approach to the best case TTP* and AMS* as follows:

$$d_r(\text{TTP}) = \frac{TTP - TTP^*}{TTP^*} \qquad (4.15) \qquad d_r(\text{AMS}) = \frac{AMS - AMS^*}{AMS^*} \qquad (4.16)$$

Table 4.5: Rolling horizon results for single-objective implementations for the semiconductor case study

	f_1 ID	f_2 ID	TTP	d_r(TTP) [%]	AMS	d_r(AMS) [%]	Mean CPU time [sec]	Mean Gap [%]
	0	-	81	−0.01	4685	0.04	42	0.00
	1	-	81	−0.01	4740	0.05	11	0.01
	2	-	81	−0.01	5067	0.12	65	0.00
1st	3*	-	82[a]	0.00	4951	0.10	21	0.00
inst.	4	-	82	0.00	4892	0.08	21	0.00
	-	0	74	−0.10	6506	0.44	133	0.00
	-	1*	38	−0.54	4515[b]	0.00	84	0.00
	-	2	74	−0.10	6478	0.43	101	0.00
	0*	-	876[a]	0.00	4800	0.02	47	0.00
	1	-	871	−0.01	4742	0.01	38	0.00
	2	-	875	−0.00	4863	0.04	148	0.00
2nd	3	-	865	−0.01	4844	0.03	62	0.00
inst.	4	-	858	−0.02	4694	0.00	85	0.00
	-	0	736	−0.16	8174	0.74	49	0.00
	-	1*	605	−0.31	4692[b]	0.00	286	0.00
	-	2	739	−0.16	7726	0.65	73	0.00

Values with superscripts a and b represent TTP* and AMS*, respectively. The f_1ID and f_2ID corresponding to TTP* and AMS* are identified with *.

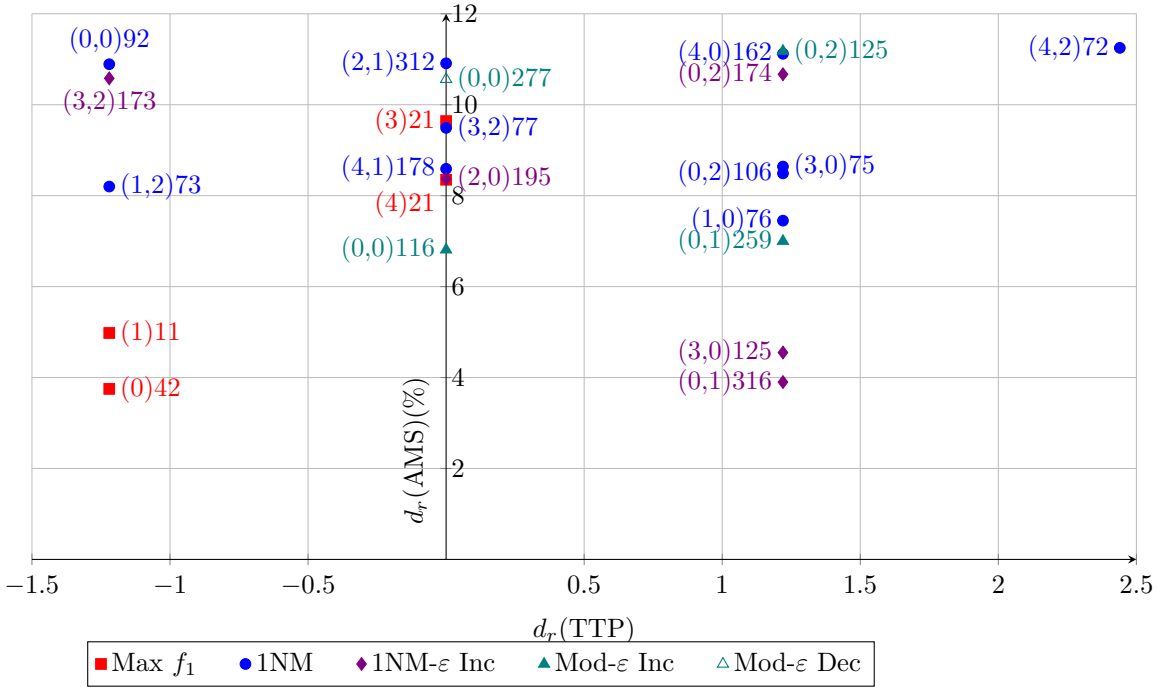


Figure 4.3: d_r(TTP) vs d_r(AMS) for the first instance of the semiconductor case study

In Figures 4.3 and 4.4, we compare the different approaches by plotting d_r(TTP) on the horizontal axis, and d_r(AMS) on the vertical axis for each instance. For each data point, the corresponding method is identified by the marker, the (f_1ID, f_2ID) combination is identified inside the parenthesis, and mean CPU time is provided next to the parenthesis. For the 1NM-ε and Mod-ε, we identify the search direction as the suffixes 'Inc' and 'Dec'. In comparing the d_r(TTP) and d_r(AMS) of different approaches, we can describe a dominance relationship similar to the notion of Pareto efficiency: approach A dominates over approach B if $\{d_r(\text{TTP})^A > d_r(\text{TTP})^B$ && $d_r(\text{AMS})^A \leq d_r(\text{AMS})^B\}$ or $\{d_r(\text{TTP})^A \geq d_r(\text{TTP})^B$ && $d_r(\text{AMS})^A < d_r(\text{AMS})^B\}$. Based on this dominance relationship, we present results that are closer to the bottom right quadrant, and exclude most of the dominated results.

In Figure 4.3, three non-dominated approaches are observed for the first instance: Max f_1^0, 1NM-ε Inc (f_1^0, f_2^1), and 1NM (f_1^4, f_2^2). In Figure 4.4, one non-dominated approach is observed: 1NM-ε Inc (f_1^4, f_2^1). These results demonstrate that the *a priori* multi-objective scalarization methods can be used to improve TTP and AMS over a rolling horizon compared to a single-objective approach. Furthermore, the *a priori* 1NM-ε and Mod-ε ap-

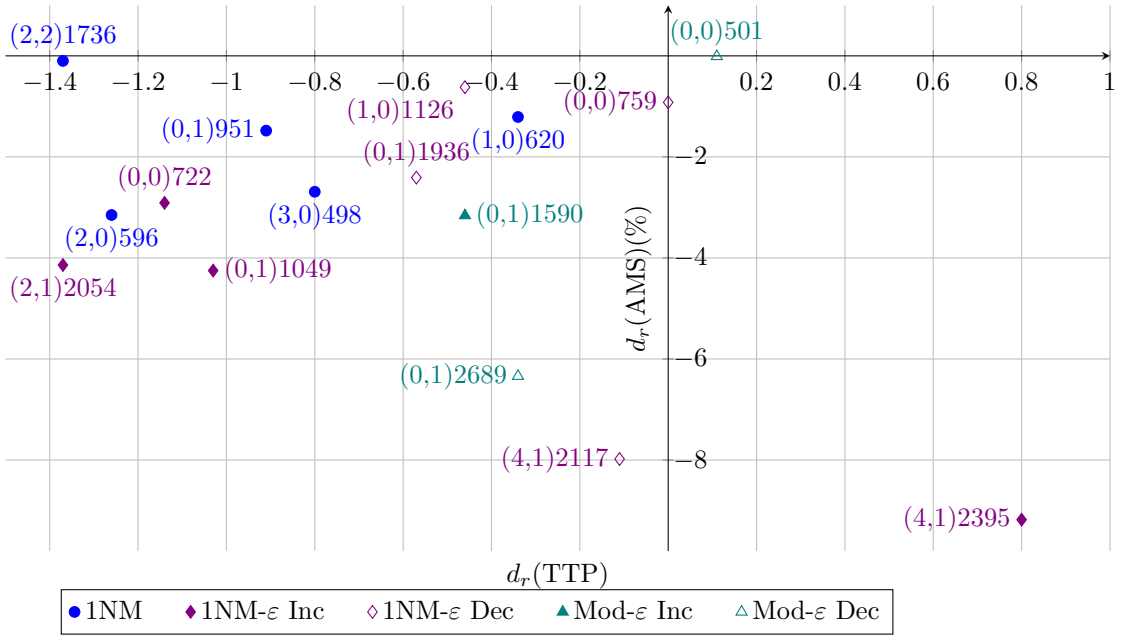


Figure 4.4: d_r(TTP)vs d_r(AMS)for the second instance of the semiconductor case study

proaches proposed in this work were able to improve TTP or AMS, if not both, compared to the 1NM method. For example, for the second instance, Mod-ε Inc (f_1^0, f_2^1) produced both higher TTP and lower AMS (d_r(TTP) $= -0.41$, d_r(AMS) $= -3.16$) compared to 1NM (f_1^0, f_2^1) (d_r(TTP) $= -0.91$, d_r(AMS) $= -1.48$) as shown in Figure 4.4. However, the improvements in TTP and AMS were generally achieved at the expense of additional CPU time.

Based on the results of Figures 4.3 and 4.4, we compare how d_r(TTP) and d_r(AMS) change throughout the RH for the two instances in Figures 4.5 and 4.6. Given that only one non-dominated approach is observed for the second instance, we identified three additional dominant approaches excluding 1NM-ε Inc (f_1^4, f_2^1), and included them in Figure 4.6. Furthermore, we compare these results to the single-objective Max f_1 run producing TTP* for each instance as reported in Table 4.5. We did not, however, compare the results to the single-objective Min f_2^1 run given that it produced significantly lower d_r(TTP) at all points throughout the RH compared to the other methods for each instance (-51 to -100% and -28 to -100% for the first and second instances, respectively).

In Figures 4.5 and 4.6, we can observe that the TTPs of the dominant approaches

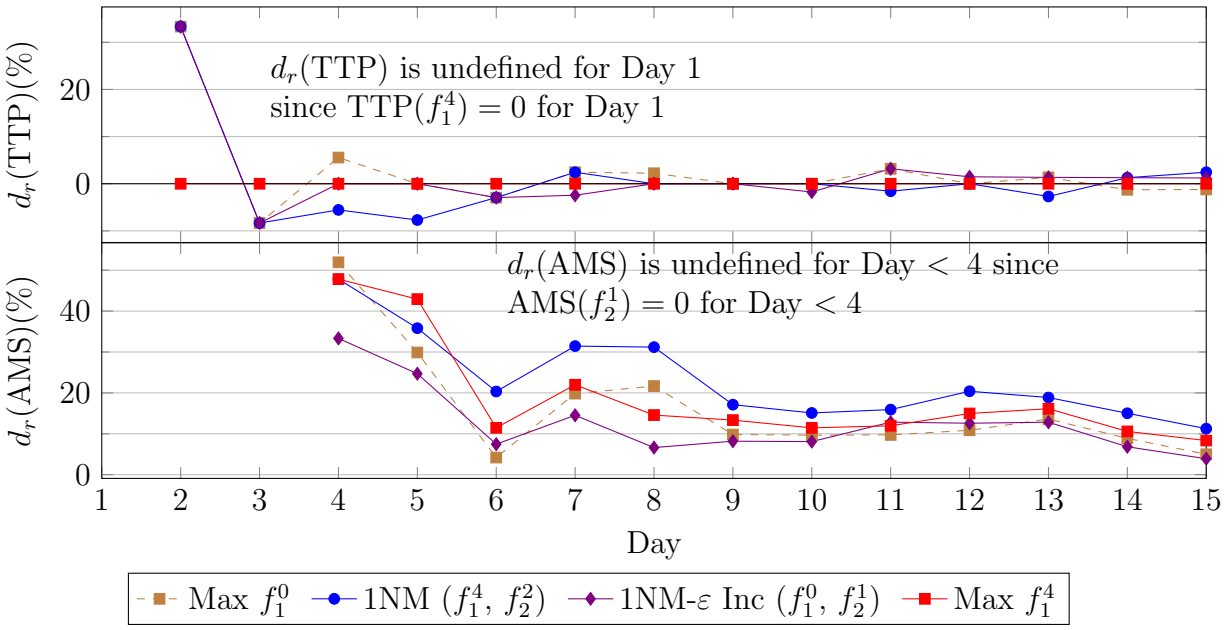


Figure 4.5: Change in d_r(TTP) and d_r(AMS) throughout the RH for the first instance

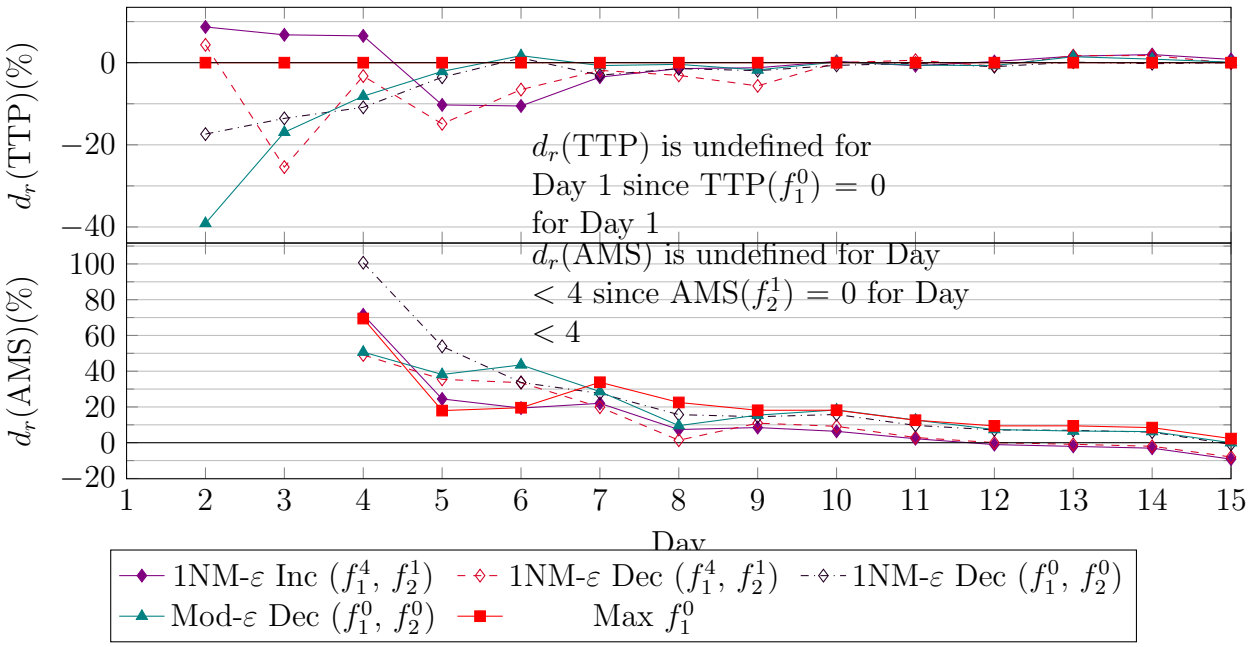


Figure 4.6: Change in d_r(TTP) and d_r(AMS) throughout the RH for the second instance

tend to converge around day 8 of the 15-day RH for both instances. Given this behavior of converging TTP, the main benefit of applying a multi-objective scalarization method over a single-objective f_1 maximization approach is to reduce AMS without a substantial trade-off in TTP. For both instances, the 1NM-ε Inc approach provided the lowest AMS for most of the RH among the approaches compared in Figures 4.5 and 4.6, while its d_r(TTP) did not fall below -10.5% at any point during the RH for 1NM-ε Inc.

Overall, it was beneficial to obtain a trade-off solution within the output efficiency range in the direction of increasing f_1 from the 1NM Point using the 1NM-ε method for this particular case study. f_1^0 and f_1^4 were identified as the dominant throughput maximization objectives, while f_2^1 was the dominant makespan minimization objective. However, depending on the problem being considered, different objectives may be dominant, and the Mod-ε method may be more appropriate (e.g., if the Mod-ε method has better or similar solution quality, with similar or lower CPU times).

4.4.2 Analytical services case study

While the case study presented in Section 3.3 considered a network of 25 processing units defined by 11 paths, in this study, we consider a more comprehensive network of 118 processing units and 117 unique paths; a part of this network formed by 39 processing units and 19 paths is presented in Figure 4.7. The 19 paths defining this partial network are presented in Tables 4.6, and capacities, resources, and processing times are presented in Table 4.7. The identity of the industrial partner and the complete data for this case study are not disclosed as per our nondisclosure agreement with the industrial partner.

For this analytical services case, a task consists of a group discrete samples received from a client that must follow a specific path of processing units to be analyzed. We consider a constituent horizon (CH) to be a single day, and the plant is assumed to operate 8 hours/day (i.e. $H = 8$ hours) for each day in the RH. The CH is discretized according to the NUD60 discretization scheme presented in Section 3.3.1. The instance studied in this work was generated by first taking a snapshot of the analytical services facility to capture the tasks that are currently in the system, then introducing new tasks for each day within the rolling horizon. A task for each day is generated through uniform random assignment of task size between 1 and 100 samples, and tasks are generated in this manner until the total number of samples arriving that day exceeds 4,000, the approximate nominal daily

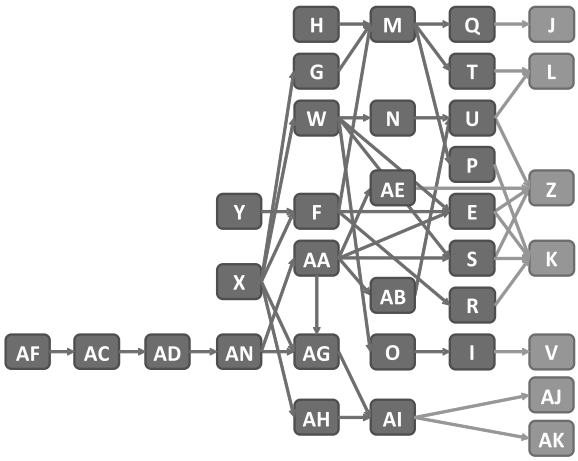


Figure 4.7: Partial map of the process network for the analytical services case study

Table 4.6: Paths of processes defining the partial analytical services network of Figure 4.7

Path ID	Sequence of processing units							
P1	X	F	M	P	K			
P2	X	F	E	K				
P3	X	G	M	T	L			
P4	X	F	R	K				
P5	X	F	M	T	L			
P6	Y	F	M	Q	J			
P7	H	M	T	L				
P8	X	W	S	K				
P9	X	W	N	U	L			
P10	X	W	O	I	V			
P11	X	W	E	K				
P12	AF	AC	AD	AN	AA	E	Z	
P13	AF	AC	AD	AN	AA	AB	U	Z
P14	AF	AC	AD	AN	AA	AE	Z	
P15	AF	AC	AD	AN	AA	S	Z	
P16	AF	AC	AD	AN	AG	AI	AJ	
P17	AF	AC	AD	AN	AA	AE	Z	
P18	X	AG	AI	AK				
P19	X	AH	AI	AK				

Table 4.7: Process capacity, resources, and processing time information for analytical services case study

Process	Normalized Capacity	Number of Resources	Scaled Processing Time
E	0.06	2	4
F	0.30	2	30
G	0.03	2	15
H	0.07	1	62
I	0.01	1	144
J	0.21	1	24
K	0.67	3	18
L	0.61	2	24
M	0.30	4	12
N	0.61	4	22
O	0.00	1	1
P	0.25	1	39
Q	0.33	1	144
R	1	10	74
S	0.67	10	47
T	0.16	8	126
U	0.19	8	114
V	0.03	1	6
W	0.61	4	162
X	0.01	6	1
Y	0.01	1	1
Z	0.06	2	1
AA	0.12	1	6
AB	0.01	2	33
AC	0.06	2	4
AD	0.06	2	2
AE	0.06	2	8
AF	0.06	1	4
AG	0.50	1	1,008
AH	0.50	1	1,008
AI	0.48	6	48
AJ	0.20	1	1
AK	0.02	1	2
AN	0.06	2	7

Reported capacities are normalized to a maximum value of 1, and processing times are scaled to a minimum value of 1. Actual plant numbers cannot be disclosed for confidentiality.

processing capacity of this network of units considered in this study. Paths were assigned to the tasks to generate a distribution of paths reflecting the facility's actual operating distribution of paths.

Rolling horizon results: analytical services

In this section, we compare TTP and AMS at the end of a 30-day RH, at which point 67-78% of tasks are completed for the multi-objective methods. While more comprehensive results tables are presented in Appendix B, we present key results and comparative analysis in this section.

For this case study, Max f_1^1 and Min f_2^1 single-objective RH runs produced the highest TTP (TTP*) and lowest AMS (AMS*) at the end of the 30-day RH, as presented in Table 4.8. Therefore, relative differences d_r(TTP) and d_r(AMS) are presented in this section with respect to Max f_1^1 and Min f_2^1 in this section as defined in eq (4.15)-(4.16).

In Figure 4.8, d_r(TTP) and d_r(AMS) of the different approaches are compared. In this case study, bi-objective implementations were not performed with f_1^2 as each single-objective Max f_1^2 iteration required 4 hours, on average, to compute due to computationally inefficient model generation. Furthermore, based on the observation that f_2^2 consistently provided inferior trade-off results due to producing significantly high d_r(AMS) (35-37%), the Mod-ε and 1NM-ε methods were not implemented with f_2^2 objective (see Tables B.1 and B.2 in Appendix B). Results for the Mod-ε and 1NM-ε methods for the search direction of decreasing f_1 and f_2 could not be obtained as no feasible solution could be found in this direction on Day 1 of the RH for any (f_1, f_2). During the 1NM method implementations, we also noticed that the final 1-norm minimization run often reached the solver time limit of 1 hour. To address this issue, we also performed bi-objective implementations using a 6 minute solver time limit, based on our observation that most single objective runs compute the bounds of the output efficiency range in less than 6 minutes, and the optimality gap for the 1-norm minimization runs typically reached below 1% within the first 6 minutes.

In Figure 4.8, three non-dominated approaches are observed: Mod-ε Inc (f_1^0, f_2^1), 1NM (f_1^1, f_2^1) (6 minute time limit), and 1NM-ε Inc (f_1^1, f_2^1). These results demonstrate that the proposed 1NM-ε and Mod-ε methods, as well as the 1NM method can be used to produce results that are non-inferior, if not dominant, when compared to a single-objective

Table 4.8: Rolling horizon results for single-objective runs for the analytical services case study

f_1 ID	f_2 ID	TTP	d_r(TTP) [%]	AMS	d_r(AMS) [%]	Mean CPU time [sec]	Mean Gap [%]
0	-	61,321	−1.08	21,132	18.3	88	0.00
1*	-	61990[a]	0	20,447	14.5	88	0.00
2	-	61,246	−1.20	20,540	15.0	14,343	0.00
3	-	61,205	−1.27	20,331	13.8	84	0.00
4	-	59,229	−4.45	20,729	16.1	141	0.00
-	0	58,343	−5.88	20,664	15.7	1,214	0.02
-	1*	32,025	−48.34	17859[b]	0	251	0.01
-	2	60,049	−3.13	25,206	41.1	98	0.00

Values with superscripts a and b represent TTP* and AMS*, respectively. The f_1ID and f_2ID corresponding to TTP* and AMS* are identified with *.

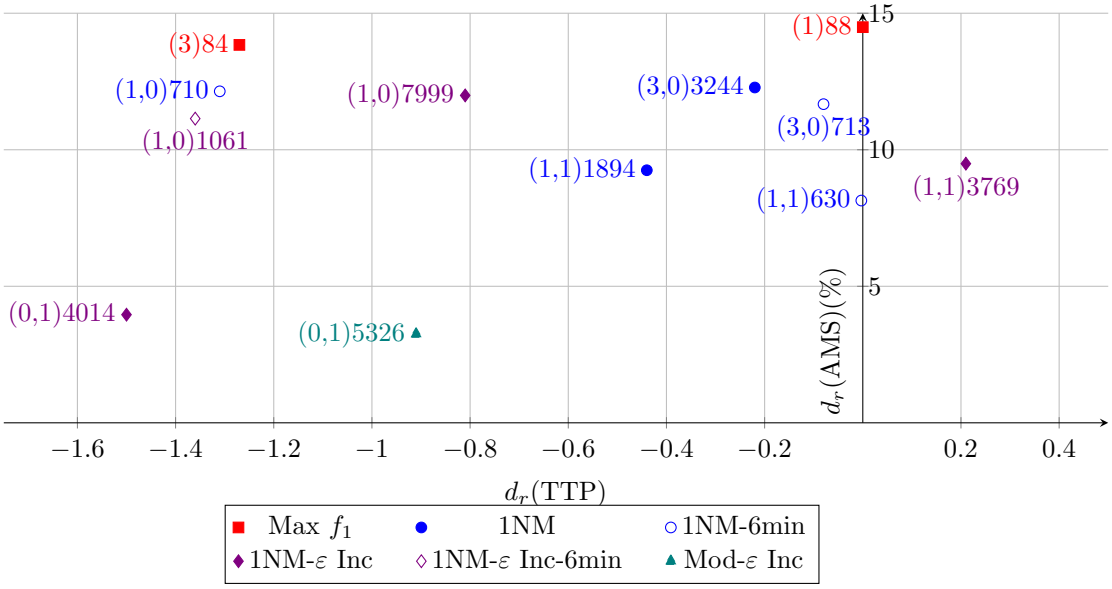


Figure 4.8: d_r(TTP) vs d_r(AMS) for the analytical services case study

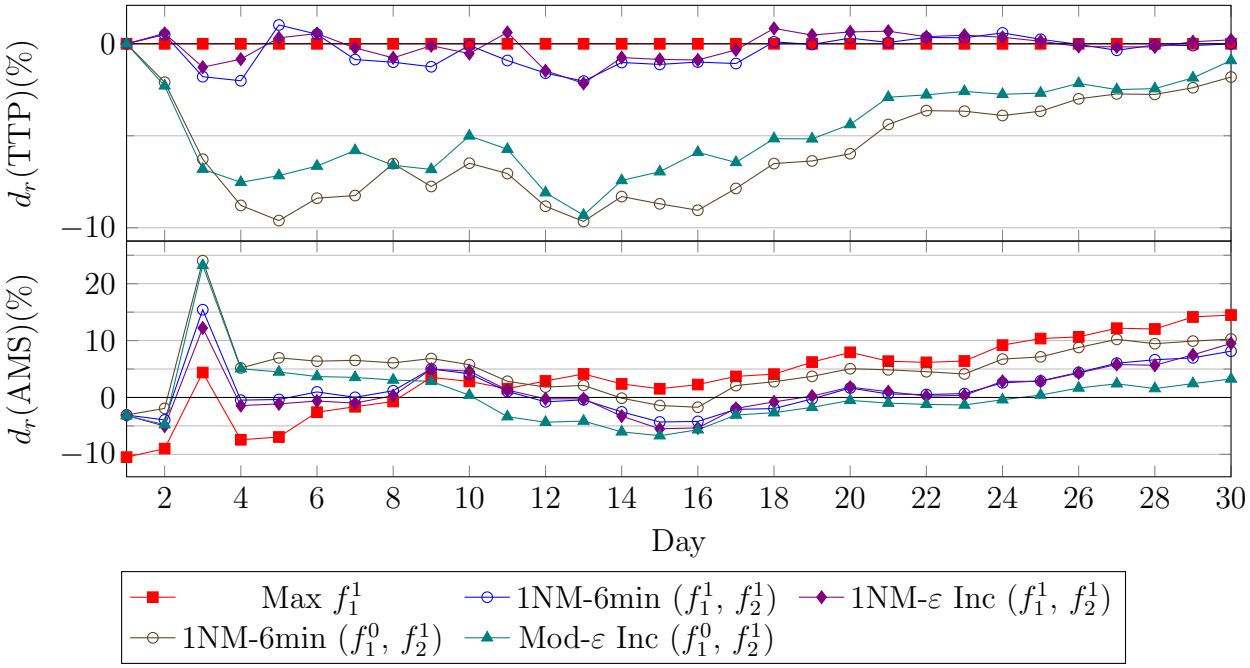


Figure 4.9: Change in d_r(TTP) and d_r(AMS) throughout the rolling horizon

approach for an industrial sized problem using a rolling horizon approach. Furthermore, for the 1NM method, the overall CPU time could be reduced by 67% and 78% for (f_1^1, f_2^1) and (f_1^3, f_2^0), respectively, by setting the solver time limit to 6 minutes instead of 1 hour, while d_r(TTP) and d_r(AMS) improved by 0.43 and 1.12 percentage points, respectively, for (f_1^1, f_2^1), and by 0.14 and 0.61 percentage points, respectively, for (f_1^3, f_2^0). These results show that, since the operational f_1 and f_2 objectives do not explicitly maximize TTP and minimize AMS, but rather approximate expressions with the goals to maximize TTP and minimize AMS, it is not necessary to solve problems to optimality when computing the 1NM Point to obtain a good trade-off solutions in terms of TTP and AMS. However, the same success could not be obtained for the 1NM-ε and Mod-ε methods using a 6 minute time limit, as 5 of 8 (f_1, f_2) combinations failed to produce a feasible solution in the increasing search direction within 6 minutes at some point during the RH.

In Figure 4.9, the changes in d_r(TTP) and d_r(AMS) of the three non-dominated approaches throughout the rolling horizon are presented, along with Max f_1^1, as the reference for computing d_r(TTP), as well as 1NM (f_1^0, f_2^1) to provide a comparison parallel to 1NM-ε Inc (f_1^1, f_2^1) vs 1NM (f_1^1, f_2^1). These results demonstrate that the choice of the f_1 objec-

tive in a bi-objective implementation produces a significant difference in TTP, as single objective max f_1^1, 1NM (f_1^1, f_2^1) and 1NM-εInc (f_1^1, f_2^1) produced average d_r(TTP) of -0.28% over the RH, while 1NM (f_1^0, f_2^1) and Mod-εInc (f_1^0, f_2^1) produced average d_r(TTP) of -5.29%. Furthermore, the effectiveness of taking an alternative trade-off solution after finding the 1NM solution appeared to depend on the choice of f_1. On average, throughout the RH, the 1NM-ε method only increased d_r(TTP)by 0.27% point and reduced d_r(AMS) by 0.29% point over 1NM method for (f_1^1, f_2^1), while the Mod-ε method increased d_r(TTP) by 1.05% point and reduced d_r(AMS) by -4.94% point for (f_1^0, f_2^1).

These results demonstrated potential benefits to using *a priori* scalarization methods for industrial sized problems in a rolling horizon applications. For this particular case study, the traditional 1NM approach provided good trade-off solutions within 6 minutes, and the proposed 1NM-ε and Mod-ε methods can be used to further increase the TTP or reduce the AMS using the dominant objective functions (f_1^0 and f_1^1 for throughput maximization, and f_2^1 for makespan minimization).

4.5 Chapter summary

In this chapter, two new *a priori* approaches for handling problems with conflicting objectives of maximizing the TTP and minimizing the AMS, which were compared against the traditional compromise programming approach. In addition, alternative objective function formulations for approximating the TTP and the AMS for the use in a rolling horizon framework were presented. The two case studies in this chapter showed that some of the proposed objective function formulations were dominant over others. Furthermore, the multi-objective methods were shown to produce favourable trade-off solutions when compared to single-objective implementations, and the new *a priori* approaches proposed in this work demonstrated the potential to improve the quality of the solutions when compared to the traditional compromise programming approach.

Chapter 5

Conclusions

The overall goal of this thesis was to present approaches to handle industrial-sized problems in short-term scheduling with conflicting objectives of throughput maximization, and makespan minimization. The problem was tackled in two major aspects: time representation in the model formulation, and the approaches for handling the conflicting objectives.

Chapter 3 presented a comparison between a flexible discrete-time formulation, with both uniform and non-uniform time discretizations, and a continuous-time formulation in the context of short-term scheduling of operations in an analytical services facility. Through a series of computational experiments designed based on an actual analytical services facility, the strengths and weaknesses of the different time representation approaches relative to each other were studied in order to determine the favorable approach to solving industrial-sized short-term scheduling problems for a multipurpose plant.

Based on the results of the computational study presented in Section 3.3.2, the multitasking continuous-time formulation based on conservation of flow proposed by Lagzi et al.[87] may not be suitable for addressing large-scale problems. For the instances considered in this work, the flexible discrete-time formulation led to tighter and smaller problems compared to the continuous-time formulation. Computational cost for the continuous-time formulation became prohibitive, taking hundreds to tens of thousands of times longer to compute compared to the flexible discrete-time formulation, while almost all of the instances resulted in poorer solution quality for the continuous-time formulation. The performance of the continuous-time formulation presented in this work deteriorates even more with increasing problem size. These results are in contrast to some of the previous results

reported in the literature on comparing the performance of discrete (without flexible time discretization capability) and continuous-time formulations[129]. This contrast highlights the importance of further investigations to be conducted on discrete-time formulations, since the conclusions may vary from application to application.

For the flexible discrete-time formulation, non-uniform discretization schemes solved large instances more efficiently compared to the uniform discretization schemes within the scope of this study. While the quality of solutions for the non-uniform discretization schemes was virtually as good as the quality of solutions for the finest uniform discretization scheme, the computational times for the non-uniform discretization schemes were merely fractions of the computational times for the finest uniform discretization scheme, as the non-uniform discretization schemes generally led to tighter problems with smaller numbers of variables and constraints. The superior relative performance of the non-uniform discretization schemes become even more apparent with increasing problem size, as the computational cost deteriorates much more rapidly for the uniform discretization schemes compared to the non-uniform discretization schemes. Superior relative performance of the non-uniform discretization schemes also become more pronounced with increasing processing time variability. For problems with low processing time variability, however, a fine uniform discretization scheme might be more desirable compared to a relatively coarse non-uniform discretization scheme if solution quality is of utmost importance while long CPU time limits can be tolerated.

In Chapter 4 effective alternatives to address some of the issues regarding short-term scheduling of multipurpose plants were proposed, where there exists two conflicting tactical objectives in maximizing the total throughput of the plant, and minimizing the average makespan of tasks being completed. The contributions in addressing these issues are *a priori* multi-objective optimization methods, and approximation of the tactical objectives as alternative operational objective functions. The effectiveness and limitations of these different approaches are studied using a semiconductor processing plant case study, and a larger, industrial-sized analytical services sector case study. In these case studies, a rolling horizon approach was used to compare the performances of the different multi-objective approaches and the operational objective functions in terms of the total throughput of the plant, and the average makespan of completed tasks.

The multi-objective methods implemented in this work were focused around the reference point based compromise programming approach, an *a priori* scalarization method, which minimizes the 1-norm distance of a trade-off solution to the utopia point (1NM

method). In this work, two new algorithms, i.e., the Mod-ε and 1NM-ε methods, are proposed based on hybridization of the ε-constraint method and the 1-norm distance based compromise programming. For both case studies, with appropriately selected operational objective functions, the *a priori* methods provided lower average makespan than single-objective throughput maximization approach, without a significant reduction in the total throughput, if any. In particular, the *a priori* 1NM-ε approach, searching in the direction of increasing operational objective values, was particularly effective in this regard. However, the effectiveness and required CPU time varied between the different instances and the different combinations of the operational objective functions. Therefore, it is important to understand these problem-specific behaviors for their implementation in practice. Furthermore, when dealing with a large-scale problem, consideration should be made to impose a relatively small solver time limit for the 1-norm minimization run, and it was shown that this approach can significantly reduce the CPU time for an industrial-sized problem without a significant degradation in solution quality, if any.

5.1 Potential future research directions

There are multiple potential future research directions to take in time representations. For discrete-time approaches, one direction is to explore algorithms for discretizing the multiple and non-uniform time grids for the flexible discrete-time approaches, which do not necessarily produce time grids with equally spaced time points. These advanced discretization schemes may provide better balance between precision of event points and the computational cost.

For the continuous-time approaches, one direction is to explore alternatives that would improve the solution time for the continuous-time approaches by developing formulations with smaller number binary variables, using new model representations (e.g., state-task network, STN), or by tightening the LP relaxations of the existing formulations. Unit-specific continuous-time formulations may also be extended to include more operational characteristics, such as multitasking. Researchers have been successful in reducing formulation size and computational demands by implementing unit-specific approaches[54;74;76;97;113;133].

When it comes to methods for handling multiple conflicting objectives, this thesis focused on relatively simple *a priori* approaches with demonstrated practical applicability,

but have not been given prominent attention from researchers in the multi-objective community. Therefore, possible future research directions include developing and applying state of the art *a Priori* multi-objective methods based on heuristics and metaheuristics as methods in this category are continually improved by the majority of the researchers in this field. Furthermore, generalization of the Mod-ε and 1NM-ε methods proposed in this work to handle more than two objectives is another possible future research direction.

References

[1] S. C. Aggarwal. A focussed review of scheduling in services. *Eur. J. Oper. Res.*, 9 (2):114–121, 1982.

[2] M. S. Alam, M. A. Hossain, S. Algoul, M. A. A. Majumader, M. A. Al-mamun, G. Sexton, and R. Phillips. Multi-objective multi-drug scheduling schemes for cell cycle specific cancer treatment. *Comput. Chem. Eng.*, 58:14–32, 2013.

[3] M. A. Allouche, B. Aouni, J. M. Martel, T. Loukil, and A. Rebaï. Solving multi-criteria scheduling flow shop problem through compromise programming and satisfaction functions. *Eur. J. Oper. Res.*, 192(2):460–467, 2009.

[4] E. Álvarez-miranda, V. Cacchiani, T. Dorneth, M. Jünger, F. Liers, A. Lodi, T. Parriani, and D. R. Schmidt. Models and Algorithms for Robust Network Design with Several Traffic Scenarios. In *Comb. Optim. Second Int. Symp. ISCO 2012*, pages 261–272, 2012. URL http://link.springer.com/10.1007/3-540-29297-7.

[5] M. J. Alves and J. Clímaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *Eur. J. Oper. Res.*, 180(1):99–115, 2007.

[6] S. Bartolini, I. Casini, and P. Detti. Solving Graph Partitioning Problems Arising in Tagless Cache Management. In *Comb. Optim. Third Int. Symp. ISCO 2014*, pages 50–61, Cham, 2014. Springer International Publishing Switzerland.

[7] M. H. Bassett, J. F. Pekny, and G. V. Reklaitis. Decomposition Techniques for the Solution of Large-Scale Scheduling Problems. *AIChE J.*, 42(12):3373–3387, 1996.

[8] M. H. Bassett, J. Penky, and G. V. Reklaitis. Using Detailed Scheduling To Obtain Realistic Operating Policies for a Batch Processing Facility. *Ind. Eng. Chem. Res.*, 36:1717–1726, 1997.

[9] A. Berrichi, F. Yalaoui, L. Amodeo, and M. Mezghiche. Bi-Objective Ant Colony Optimization approach to optimize production and maintenance scheduling. *Comput. Oper. Res.*, 37(9):1584–1596, 2010.

[10] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[11] L. T. Biegler. Introduction to Process Optimization. In *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*, chapter 1, pages 1–16. SIAM, Philadelphia, 2010.

[12] B. Bilgen and K. Dogan. Multistage Production Planning in the Dairy Industry: A Mixed-Integer Programming Approach. *Ind. Eng. Chem. Res.*, page acs.iecr.5b02247, 2015.

[13] M. Blom, A. R. Pearce, and P. J. Stuckey. Short-term scheduling of an open-pit mine with multiple objectives. *Eng. Optim.*, 49(5):777–795, 2017.

[14] M. L. Blom, A. R. Pearce, and P. J. Stuckey. A Decomposition-Based Algorithm for the Scheduling of Open-Pit Networks Over Multiple Time Periods. *Manage. Sci.*, 62 (10), 2016.

[15] S. K. Bose and S. Bhattacharya. A State Task Network model for scheduling operations in cascaded continuous processing units. *Comput. Chem. Eng.*, 33(1):287–295, 2009.

[16] C. Büsing, K. S. Goetzmann, J. Matuschke, and S. Stiller. Reference points and approximation algorithms in multicriteria discrete optimization. *Eur. J. Oper. Res.*, 260(3):829–840, 2017.

[17] T. Cai, S. Wang, and Q. Xu. Scheduling of multiple chemical plant start-ups to minimize regional air quality impacts. *Comput. Chem. Eng.*, 54:68–78, 2013.

[18] E. Capón-garcía, S. Papadokonstantakis, and K. Hungerbühler. Multi-objective optimization of industrial waste management in chemical sites coupled with heat integration issues. *Comput. Chem. Eng.*, 62:21–36, 2014.

[19] P. Castro, A. P. F. D. Barbosa-Póvoa, and H. Matos. An Improved RTN Continuous-Time Formulation for the Short-term Scheduling of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.*, 40(9):2059–2068, 2001.

[20] P. M. Castro, I. E. Grossmann, and A. Q. Novais. Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers. *Ind. Eng. Chem. Res.*, 45(18):6210–6226, 2006.

[21] P. M. Castro, I. Harjunkoski, and I. E. Grossmann. Optimal short-term scheduling of large-scale multistage batch plants. *Ind. Eng. Chem. Res.*, 48(24):11002–11016, 2009.

[22] P. M. Castro, B. Custódio, and H. A. Matos. Optimal scheduling of single stage batch plants with direct heat integration. *Comput. Chem. Eng.*, 82:172–185, 2015.

[23] Y.-t. Chang. Multiobjective Permutation Flow Shop Scheduling using MOEA / D with Local Search. In *2016 Conf. Technol. Appl. Artif. Intell. (TAAI)*, number 5, pages 262–269, 2016.

[24] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio. *Comparison of multi-objective optimization methodologies for engineering applications*, volume 63. Elsevier Ltd, 2012.

[25] T. C. Chiang. Enhancing rule-based scheduling in wafer fabrication facilities by evolutionary algorithms: Review and opportunity. *Comput. Ind. Eng.*, 64(1):524–535, 2013.

[26] Y. Collette and P. Siarry. *Multiobjective Optimization: Principles and Case Studies*. Springer, Berlin, New York, 2003.

[27] Y. Cui, Z. Geng, Q. Zhu, and Y. Han. Review: Multi-objective optimization methods and application in energy saving. *Energy*, 125:681–704, 2017.

[28] I. Das and J. E. Dennis. Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *Siam J. Optim.*, 8(3):631–657, 1998.

[29] K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, New York, 2001.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6(2):182–197, 2002.

[31] I. T. Dedopoulos and N. Shah. Optimal Short-Term Scheduling of Maintenance and Production for Multipurpose Plants. *Ind. Eng. Chem. Res.*, 34(1):192–201, 1995.

[32] A. Dietz, L. Pibouleau, and S. Domenech. A Framework for Multiproduct Batch Plant Design with Environmental Consideration : Application To Protein Production. *Ind. Eng. Chem. Res.*, 44:2191–2206, 2005.

[33] A. Dietz, C. Azzaro-Pantel, L. Pibouleau, and S. Domenech. Multiobjective optimization for multiproduct batch plant design under economic and environmental considerations. *Comput. Chem. Eng.*, 30(4):599–613, 2006.

[34] T. L. Dinh. *Multi-Objective Linear Programming Problem*. Springer, Cham, 2016.

[35] U. Diwekar. *Introduction to Applied Optimization*, volume 22. Springer, Boston, 2008.

[36] P. Doganis and H. Sarimveis. Optimal production scheduling for the dairy industry. *Ann. Oper. Res.*, 159(1):315–331, 2008.

[37] I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[38] M. Durantoksari. Taylor series approach to fuzzy multiobjective linear fractional programming. *Inf. Sci. (Ny).*, 178(4):1189–1204, 2008.

[39] M. Ebrahimi, S. M. Fatemi Ghomi, and B. Karimi. Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates. *Appl. Math. Model.*, 38(9-10):2490–2504, 2014.

[40] T. F. Edgar, D. M. Himmelblau, and L. S. Lasdon. *Optimization of Chemical Processes*. McGraw-Hill, New York, 2nd edition, 2001.

[41] M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin, Heidelberg, second edi edition, 2005.

[42] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectr.*, 22(4):425–460, 2000.

[43] P. Fattahi, M. Saidi Mehrabad, and F. Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *J. Intell. Manuf.*, 18(3):331–342, 2007.

[44] M. Fellet. A fresh take on fake meat. *ACS Central Science*, 1(7):347–349, 2015. PMID: 27162992.

[45] J. Figueira, S. Greco, and M. Ehrgott. *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer, Boston, 2005.

[46] A. Flores-Tlacuahuac, P. Morales, and M. Rivera-Toledo. Multiobjective Nonlinear Model Predictive Control of a Class of Chemical Reactors. *Ind. Eng. Chem. Res.*, 51(17):5891–5899, 2011.

[47] C. A. Floudas and X. Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Comput. Chem. Eng.*, 28(11):2109–2129, 2004.

[48] C. Fonseca and P. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussions and Generalization. In *Proc. Fifth Int. Conf. Genet. Algorithms*, pages 416–423, 1993.

[49] C. M. Fonseca and P. J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evol. Comput.*, 3(1):1–16, 1995.

[50] M. Freimer and P. L. Yu. Some New Results on Compromise Solutions for Group Decision Problems. *Manage. Sci.*, 22(6):688–693, 1976.

[51] K. C. Furman, Z. Jia, and M. G. Ierapetritou. A robust event-based continuous time formulation for tank transfer scheduling. *Ind. Eng. Chem. Res.*, 46(26):9126–9136, 2007.

[52] P. Geladi, D. MacDougall, and H. Martens. Correction for Near-Infrared Reflectance Spectra of Meat. *Appl. Spectrosc.*, 39(3):491–500, 1985.

[53] M. Gen and L. Lin. Multiobjective evolutionary algorithm for manufacturing scheduling problems : state-of-the-art survey. *J. Intell. Manuf.*, 25:849–866, 2014.

[54] N. F. Giannelos and M. C. Georgiadis. A Simple New Continuous-Time Formulation for Short-Term Scheduling of Multipurpose Batch Processes. *Ind. Eng. Chem. Res.*, 41:2178–2184, 2002.

[55] I. E. Grossmann, R. Drabbant, and R. K. Jain. Incorporating Toxicology in the Synthesis of Industrial Chemical Complexes. *Chem. Eng. Commun.*, 17(1-6):151–170, sep 1982.

[56] G. Guillén-Gosálbez, F. D. Mele, and I. E. Grossmann. A bi-criterion optimization approach for the design and planning of hydrogen supply chains for vehicle use. *AIChE J.*, 56(3):650–667, 2010.

[57] S. Gupta and I. A. Karimi. An Improved MILP Formulation for Scheduling Multiproduct, Multistage Batch Plants. *Ind. Eng. Chem. Res.*, 42(11):2365–2380, 2003.

[58] Gurobi. Mixed-Integer Programming (MIP) - A Primer on the Basics. URL http://www.gurobi.com/resources/getting-started/mip-basics.

[59] M. A. Gutierrez-Limon, A. Flores-Tlacuahuac, and I. E. Grossmann. A Multiobjective Optimization Approach for the Simultaneous Single Line Scheduling and Control of CSTRs. *Ind. Eng. Chem. Res.*, 51:5881–5890, 2011.

[60] Y. Y. Haimes. Integrated system identification and optimization. *Cont. Dyn. Sys.*, 10:435–453, 1973.

[61] Y. Y. Haimes, L. S. Lasdon, and D. A. Wismer. On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization. *IEEE Trans. Syst. Man, Cybern. Syst.*, 1(3):296–297, 1971.

[62] Z. Han, T. Cheng, Y. Zhou, and P. Zhang. Multi-Objective Optimal Scheduling for Hydro-Thermal-Wind Power System. In *TENCON 2015 - 2015 IEEE Reg. 10 Conf.*, 2015.

[63] S. Hanoun, S. Nahavandi, D. Creighton, and H. Kull. Solving a multiobjective job shop scheduling problem using Pareto Archived Cuckoo Search. *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, 2012.

[64] M. J. Hazaras, C. L. E. Swartz, and T. E. Marlin. Industrial Application of a Continuous-Time Scheduling Framework for Process Analysis and Improvement. *Ind. Eng. Chem. Res.*, 53(1):259–273, 2014.

[65] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Michigan, 1975.

[66] Y. Hou, N. Q. Wu, M. C. Zhou, and Z. W. Li. Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm. *IEEE Trans. Syst. Man, Cybern. Syst.*, 47(3):517–530, 2017.

[67] K. Huang and C. Liao. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Comput. Oper. Res.*, 35(4):1030–1046, 2008.

[68] C. W. Hui, A. Gupta, and H. A. J. Van Der Meulen. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. *Comput. Chem. Eng.*, 24(12):2705–2717, 2000.

[69] G. L. Hwang and A. S. M. Masud. *Multiple Objective Decision Making - Methods and Applications*. Springer, Berlin, 1979.

[70] IBM ILOG. CPLEX tolerance usage, 2011. URL http://www-01.ibm.com/support/docview.wss?uid=swg21567945.

[71] IBM ILOG. CPLEX Optimization Studio, 2013. URL http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud.

[72] IBM ILOG. Tree memory limit, 2016. URL https://www.ibm.com/support/knowledgecenter/en/SSSA5P{_}12.7.0/ilog.odms.cplex.help/CPLEX/Parameters/topics/TreLim.html.

[73] IBM ILOG. Branch and cut in CPLEX, 2017. URL https://www.ibm.com/support/knowledgecenter/SSSA5P{_}12.8.0/ilog.odms.cplex.help/refcppcplex/html/branch.html.

[74] M. G. Ierapetritou and C. A. Floudas. Effective Continuous-Time Formulation for Short-Term Scheduling. 1. Multipurpose Batch Processes. *Ind. Eng. Chem. Res.*, 37 (11):4341–4359, 1998.

[75] S. L. Janak and C. A. Floudas. Improving unit-specific event based continuous-time approaches for batch processes: Integrality gap and task splitting. *Comput. Chem. Eng.*, 32(4-5):913–955, 2008.

[76] S. L. Janak, X. Lin, and C. A. Floudas. Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes: Resource Constraints and Mixed Storage Policies. *Ind. Eng. Chem. Res.*, 43 (10):2516, 2004.

[77] Z. Jia and M. G. Ierapetritou. Generate Pareto optimal solutions of scheduling problems using normal boundary intersection technique. *Comput. Chem. Eng.*, 31 (4):268–280, 2007.

[78] R. Karuppiah, K. C. Furman, and I. E. Grossmann. Global optimization for scheduling refinery crude oil operations. *Comput. Chem. Eng.*, 32(11):2745–2766, 2008.

[79] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science (80-.).*, 220(4598):671–680, 1983.

[80] R. W. Koller and L. A. Ricardez-Sandoval. A Dynamic Optimization Framework for Integration of Design, Control and Scheduling of Multi-product Chemical Processes under Disturbance and Uncertainty. *Comput. Chem. Eng.*, 106(2):147–159, 2017.

[81] R. W. Koller, L. A. Ricardez-Sandoval, and L. T. Biegler. Stochastic back-off algorithm for simultaneous design, control, and scheduling of multiproduct systems under uncertainty. *AIChE J.*, 2018.

[82] E. Kondili, C. C. Pantelides, and R. W. H. Sargent. A General Algorithm for Short-Term Scheduling of Batch-Operations . I. Milp Formulation. *Comput. Chem. Eng.*, 17(2):211–227, 1993.

[83] G. M. Kopanos, C. A. Méndez, and L. Puigjaner. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. *Eur. J. Oper. Res.*, 207(2):644–655, 2010.

[84] G. M. Kopanos, L. Puigjaner, and M. C. Georgiadis. Optimal production scheduling and lot-sizing in dairy plants: The yogurt production line. *Ind. Eng. Chem. Res.*, 49 (2):701–718, 2010.

[85] S. Lagzi. A study of time representation in a class of short term scheduling problems. Master's thesis, 2016. URL http://hdl.handle.net/10012/10642.

[86] S. Lagzi, R. Fukasawa, and L. Ricardez-Sandoval. A multitasking continuous time formulation for short-term scheduling of operations in multipurpose plants. *Comput. Chem. Eng.*, 97:135–146, 2017.

[87] S. Lagzi, D. Y. Lee, R. Fukasawa, and L. Ricardez-Sandoval. A Computational Study of Continuous and Discrete Time Formulations for a Class of Short-Term Scheduling Problems for Multipurpose Plants. *Ind. Eng. Chem. Res.*, 56(31):8940–8953, 2017.

[88] N. H. Lappas and C. E. Gounaris. Multi-stage adjustable robust optimization for process scheduling under uncertainty. *AIChE J.*, 62(5):1646–1667, 2016.

[89] H. Lee and C. T. Maravelias. Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments. *Comput. Chem. Eng.*, 107: 171–183, 2017.

[90] K. H. Lee, H. I. Park, and I. B. Lee. A novel nonuniform discrete time formulation for short-term scheduling of batch and continuous processes. *Ind. Eng. Chem. Res.*, 40(22):4902, 2001.

[91] D. Lei. Multi-objective production scheduling: A survey. *Int. J. Adv. Manuf. Technol.*, 43(9-10):925–938, 2009.

[92] C. W. Leung, T. N. Wong, K. L. Mak, and R. Y. K. Fung. Integrated process planning and scheduling by an agent-based ant colony optimization. *Comput. Ind. Eng.*, 59(1):166–180, 2010.

[93] H. Li and Q. Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Trans. Evol. Comput.*, 13(2):284–302, 2009.

[94] J. Li, X. Xiao, Q. Tang, and C. A. Floudas. Production scheduling of a large-scale steelmaking continuous casting process via unit-specific event-based continuous-time models: Short-term and medium-term scheduling. *Ind. Eng. Chem. Res.*, 51(21): 7300–7319, 2012.

[95] J. Li, X. Xiao, Q. Tang, and C. A. Floudas. Production scheduling of a large-scale steelmaking continuous casting process via unit-specific event-based continuous-time models: Short-term and medium-term scheduling. *Ind. Eng. Chem. Res.*, 51(21): 7300–7319, 2012.

[96] X. Li. Multiobjective Discrete Artificial Bee Colony Algorithm for Multiobjective Permutation. *IEEE Trans. Eng. Manag.*, 64(2):1–17, 2017.

[97] X. Lin and C. A. Floudas. Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. *Comput. Chem. Eng.*, 25(4-6): 665–674, 2001.

[98] T. Loukil, J. Teghem, and P. Fortemps. A multi-objective production scheduling case study solved by simulated annealing. *Eur. J. Oper. Res.*, 179(3):709–722, 2007.

[99] M. Luque, A. B. Ruiz, R. Saborido, and Ó. D. Marcenaro-Gutiérrez. On the use of the Lp distance in reference point-based approaches for multiobjective optimization. *Ann. Oper. Res.*, 235(1):559–579, 2015.

[100] C. Lyra and L. R. M. Ferreira. A Multiobjective Approach to the Short-term scheduling of a Hydroelectric Power System. *IEEE Trans. Power Syst.*, 10(4), 1995.

[101] R. S. H. Mah. Scheduling of Batch Plants. In *Chem. Process Struct. Inf. Flows*, chapter 6, pages 241–302. Butterworths, Boston, 1989.

[102] C. T. Maravelias and I. E. Grossmann. New General Continuous-Time StateTask Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.*, 42(13):3056–3074, 2003.

[103] C. T. Maravelias and I. E. Grossmann. A hybrid MILP/CP decomposition approach for the short term scheduling of multipurpose batch plants. *Comput. Chem. Eng.*, 28:1921–1949, 2004.

[104] Ó. D. Marcenaro-gutiérrez. On the use of the L p distance in reference point-based approaches for multiobjective optimization. *Ann. Oper. Res.*, 235(1):559–579, 2015.

[105] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Struct. Multidiscip. Optim.*, 26(6):369–395, 2004.

[106] G. Mavrotas and K. Florios. An improved version of the augmented s-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. *Appl. Math. Comput.*, 219(18):9652–9669, 2013.

[107] C. M. Mcdonald and I. A. Karimi. Planning and Scheduling of Parallel Semicontinuous Processes . 1 . Production Planning. *Society*, 36(1978):2691–2700, 1997.

[108] 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. *Comput. Chem. Eng.*, 30(6-7):913–946, 2006.

[109] C. A. Méndez, I. E. Grossmann, I. Harjunkoski, and P. Kaboré. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. *Comput. Chem. Eng.*, 30(4):614–634, 2006.

[110] A. F. Merchan, H. Lee, and C. T. Maravelias. Discrete-time mixed-integer programming models and solution methods for production scheduling in multistage facilities. *Comput. Chem. Eng.*, 94:387–410, 2016.

[111] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12. Springer US, New York, 1998.

[112] L. Mockus and G. V. Reklaitis. Continuous time representation approach to batch and continuous process scheduling. 2. Computational issues. *Ind. Eng. Chem. Res.*, 38(1):204–210, 1999.

[113] S. Mouret, I. E. Grossmann, and P. Pestiaux. A novel priority-slot based continuous-time formulation for crude-oil scheduling problems. *Ind. Eng. Chem. Res.*, 48(18): 8515–8528, 2009.

[114] T. Murata, M. Gen, and H. Ishibuchi. Multi-Objective Scheduling with Fuzzy Due-Date Completion. *Comput. Ind. Eng.*, 35(3-4):439–442, 1998.

[115] T. Murata, H. Ishibuchi, and M. Gen. Multi-Objective Fuzzy Scheduling with the OWA Operator for Handling Different Scheduling Criteria and Different Job Importance. *IEEE Int. Fuzzy Syst. Conf. Proc.*, pages 773–778, 1999.

[116] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Trans. Evol. Comput.*, 18(2):193–208, 2014.

[117] M. Özlen and M. Azizolu. Multi-objective integer programming: A general approach for generating all non-dominated solutions. *Eur. J. Oper. Res.*, 199(1):25–35, 2009.

[118] C. C. Pantelides. Unified frameworks for optimal process planning and scheduling. In *Proceedings on the second conference on foundations of computer aided operations*, pages 253—-274. Cache Publications New York, 1994.

[119] V. Pareto. *Manuale di economia politica*. Società Editrice Libraria, 1906.

[120] B. P. Patil, R. Fukasawa, and L. A. Ricardez-Sandoval. Scheduling of operations in a large-scale scientific services facility via multicommodity flow and an optimization-based algorithm. *Ind. Eng. Chem. Res.*, 54(5):1628–1639, 2015.

[121] J. M. Pinto, M. Joly, and L. F. Moro. Planning and scheduling models for refinery operations. *Comput. Chem. Eng.*, 24(9-10):2259–2276, 2000.

[122] M. Rafiei-Shishavan, S. Mehta, and L. A. Ricardez-Sandoval. Simultaneous design and control under uncertainty: A back-off approach using power series expansions. *Comput. Chem. Eng.*, 99:66–81, 2017.

[123] M. Saidi-Mehrabad and P. Fattahi. Flexible job shop scheduling with tabu search algorithms. *Int. J. Adv. Manuf. Technol.*, 32(5-6):563–570, 2007.

[124] M. Sakawa, H. Yano, and I. Nishizaki. *Linear and Multiobjective Programming with Fuzzy Stochastic Extensions*, volume 203. Springer, New York, 2013.

[125] S. Schaible. On Sensitivity in Linear Multiobjective. *Optimization*, 107(3):615–626, 2000.

[126] G. Schilling and C. Pantelides. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Comput. Chem. Eng.*, 20:S1221–S1226, 1996.

[127] O. B. Senties, C. Azzaro-Pantel, L. Pibouleau, and S. Domenech. Multiobjective scheduling for semiconductor manufacturing plants. *Comput. Chem. Eng.*, 34(4): 555–566, 2010.

[128] N. Shah, C. C. Pantelides, and R. W. H. Sargent. A General Algorithm for Short-Term Scheduling of Batch-Operations. II. Computational Issues. *Comput. Chem. Eng.*, 17(2):211–227, 1993.

[129] H. Stefansson, S. Sigmarsdottir, P. Jensson, and N. Shah. Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry. *Eur. J. Oper. Res.*, 215(2): 383–392, 2011.

[130] Y. Sun, C. Zhang, L. Gao, and X. Wang. Multi-objective optimization algorithms for flow shop scheduling problem: A review and prospects. *Int. J. Adv. Manuf. Technol.*, 55(5-8):723–739, 2011.

[131] A. Sundaramoorthy and I. A. Karimi. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chem. Eng. Sci.*, 60(10):2679–2702, 2005.

[132] A. Sundaramoorthy and C. T. Maravelias. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Ind. Eng. Chem. Res.*, 50(9):5023–5040, 2011.

[133] N. Susarla, J. Li, and I. A. Karimi. A novel approach to scheduling multipurpose batch plants using unit-slots. *AIChE J.*, 56(7):1859–1879, 2010.

[134] J. Sylva and A. Crema. A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *Eur. J. Oper. Res.*, 180 (3):1011–1027, 2007.

[135] F. Trespalacios and I. E. Grossmann. Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie-Ingenieur-Technik*, 86(7):991–1012, 2014.

[136] A. Trivedi, N. M. Pindoriya, and D. Srinivasan. Modified NSGA-II for day-ahead multi-objective thermal generation scheduling. *2010 9th Int. Power Energy Conf. IPEC 2010*, pages 752–757, 2010.

[137] E. Ulungu, J. Teghem, P. Fortemps, and D. Tuyttens. MOSA method: A tool for solving MOCO problems. *J. Multi-Criteria Decis. Anal.*, 8:221–236, 1999.

[138] G. Upton and I. Cook. *Understanding Statistics*. Oxford University Press, Oxford, 1996.

[139] M. A. H. Van Elzakker, E. Zondervan, N. B. Raikar, I. E. Grossmann, and P. M. M. Bongers. Scheduling in the FMCG industry: An industrial case study. *Ind. Eng. Chem. Res.*, 51(22):7800–7815, 2012.

[140] S. Velez and C. T. Maravelias. Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling. *Comput. Chem. Eng.*, 53:70–85, 2013.

[141] S. Velez and C. T. Maravelias. Theoretical framework for formulating MIP scheduling models with multiple and non-uniform discrete-time grids. *Comput. Chem. Eng.*, 72: 233–254, 2015.

[142] S. Velez, A. F. Merchan, and C. T. Maravelias. On the solution of large-scale mixed integer programming scheduling models. *Chem. Eng. Sci.*, 136:139–157, 2015.

[143] S. Wang and M. Guignard. Redefining event variables for efficient modeling of continuous-time batch processing. *Ann. Oper. Res.*, 116(1-4):113–126, 2002.

[144] Z. Wang, Z. Li, Y. Feng, and G. Rong. Crude-Oil Operations under Uncertainty: A Continuous-Time Rescheduling Framework and a Simulation Environment for Validation. *Ind. Eng. Chem. Res.*, 55(43):11383–11401, 2016.

[145] X. Y. Wu, C. T. Cheng, J. J. Shen, B. Luo, S. L. Liao, and G. Li. A multi-objective short term hydropower scheduling model for peak shaving. *Int. J. Electr. Power Energy Syst.*, 68:278–293, 2015.

[146] M. M. Yenisey and B. Yagmahan. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, 45:119–135, 2014.

[147] P. L. Yu. A Class of Solutions for Group Decision Problems. *Manage. Sci.*, 19(8): 936, 1973.

[148] D. Yue and F. You. Sustainable scheduling of batch processes under economic and environmental criteria with MINLP models and algorithms. *Comput. Chem. Eng.*, 54:44–59, 2013.

[149] L. Zadeh. Optimality and non-scalar-valued performance criteria. *Autom. Control. IEEE Trans.*, 8(1):59, 1963.

[150] M. Zamarripa, P. A. Marchetti, I. E. Grossmann, T. Singh, I. Lotero, A. Gopalakrishnan, B. Besancon, and J. André. Rolling Horizon Approach for Production-Distribution Coordination of Industrial Gases Supply Chains. *Ind. Eng. Chem. Res.*, 55(9):2646–2660, 2016.

[151] V. M. Zavala and A. Flores-Tlacuahuac. Stability of multiobjective predictive control: A utopia-tracking approach. *Automatica*, 48(10):2627–2632, 2012.

[152] M. Zelany. A concept of compromise solutions and the method of the displaced ideal. *Comput. Oper. Res.*, 1(3):479–496, 1974.

[153] G. Zhang, X. Shao, P. Li, and L. Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Comput. Ind. Eng.*, 56(4):1309–1318, 2009.

[154] X. Zhang and R. W. H. Sargent. The optimal operation of mixed production facilitiesa general formulation and some approaches for the solution. *Comput. Chem. Eng.*, 20(6-7):897–904, 1996.

[155] H. Zhao, J. Shen, Y. Li, and J. Bentsman. Preference adjustable multi-objective NMPC: An unreachable prioritized point tracking method. *ISA Trans.*, 66:134–142, 2017.

[156] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm Evol. Comput.*, 1 (1):32–49, 2011.

APPENDICES

Appendix A

Full rolling horizon results for the semiconductor case study (Section 4.4.1)

In the following results tables for the semiconductor case study, 'Max Iter.' refers to the number of days required to complete 100% of the tasks, which failed to complete all tasks by the end of day 30, which was the maximum length of the rolling horizon we were considering.

Table A.1: Rolling horizon results for the 1NM-ε and Mod-ε methods for the 1st semiconductor instance

Method	f_1 ID	f_2 ID	Max Iter.	at max iter. AMS	at day 15 TTP	at day 15 AMS	Mean CPU time [sec]	Mean gap [%]
1NMEps Inc	0	0	17	5,273	55	5,184	77	0.00
1NMEps Inc	0	1	17	4,825	56	4,691	282	0.00
1NMEps Inc	0	2	17	5,064	60	4,997	154	0.00
1NMEps Inc	1	0	18	5,436	52	5,420	325	0.18
1NMEps Inc	1	1	17	5,186	55	5,169	221	0.00
1NMEps Inc	1	2	18	5,221	57	5,140	70	0.00
1NMEps Inc	2	0	18	5,117	57	4,893	165	0.00
1NMEps Inc	2	1	17	5,181	52	5,157	338	0.00
1NMEps Inc	2	2	17	5,229	56	5,193	228	0.09
1NMEps Inc	3	0	17	4,816	58	4,721	111	0.00
1NMEps Inc	3	1	17	5,181	55	5,126	222	0.00
1NMEps Inc	3	2	18	5,074	58	4,993	145	0.00
1NMEps Inc	4	0	18	5,253	57	5,100	134	0.00
1NMEps Inc	4	1	18	5,265	57	5,237	217	0.00
1NMEps Inc	4	2	18	5,441	56	5,261	183	0.00
1NMEps Dec	0	0	18	5,416	57	5,301	198	0.00
1NMEps Dec	0	2	18	5,264	56	5,230	158	0.00
1NMEps Dec	1	0	18	5,278	58	5,206	193	0.00
1NMEps Dec	1	1	18	5,121	58	5,086	236	0.00
1NMEps Dec	1	2	18	5,376	56	5,300	126	0.00
1NMEps Dec	2	0	18	5,444	55	5,215	181	0.00
1NMEps Dec	2	1	17	5,689	50	5,762	508	0.00
1NMEps Dec	2	2	18	5,337	57	5,246	320	0.00
1NMEps Dec	3	0	18	5,374	57	5,248	110	0.00
1NMEps Dec	3	1	17	5,394	49	5,436	584	0.00
1NMEps Dec	3	2	17	5,301	56	5,181	140	0.00
1NMEps Dec	4	0	18	5,404	56	5,268	130	0.00
1NMEps Dec	4	1	18	5,276	58	5,138	418	0.00
1NMEps Dec	4	2	18	5,352	56	5,276	286	0.00
ModEps Inc	0	0	18	5,034	58	4,823	98	0.00
ModEps Inc	0	1	17	4,914	56	4,832	230	0.00
ModEps Inc	0	2	17	5,092	57	5,020	110	0.00
ModEps Dec	0	0	17	5,046	55	4,992	245	0.00
ModEps Dec	0	1	17	5,581	54	5,640	623	0.00
ModEps Dec	0	2	18	5,153	56	5,040	150	0.00

Note: 1NMEps Dec for (f_1^0, f_2^1) is not included due to not being able to find a feasible solution for the ε bounded problem on day 5 of the rolling horizon.

Table A.2: Rolling horizon results for the 1NM-ε and Mod-ε methods for the 2nd semiconductor instance

Method	f_1 ID	f_2 ID	Max Iter.	at max iter. AMS	at day 15 TTP	at day 15 AMS	Mean CPU time [sec]	Mean gap [%]
1NMEps Inc	0	0	17	4,579	612	4,556	638	0.11
1NMEps Inc	0	1	16	4,477	614	4,493	984	0.12
1NMEps Inc	0	2	17	4,752	598	4,750	704	0.75
1NMEps Inc	1	0	17	4,757	616	4,751	1,223	0.01
1NMEps Inc	1	1	16	4,660	585	4,686	1,918	0.55
1NMEps Inc	1	2	16	4,768	609	4,800	1,207	0.46
1NMEps Inc	2	0	17	4,739	608	4,761	1,116	0.15
1NMEps Inc	2	1	17	4,502	611	4,498	1,813	0.12
1NMEps Inc	2	2	17	4,857	604	4,875	1,117	1.25
1NMEps Inc	3	0	17	4,763	614	4,759	776	5.43
1NMEps Inc	3	1	17	4,812	578	4,815	1,333	0.36
1NMEps Inc	3	2	17	4,723	603	4,726	1,906	0.39
1NMEps Inc	4	0	17	4,800	610	4,807	792	0.00
1NMEps Inc	4	1	17	4,285	610	4,261	2,115	0.44
1NMEps Inc	4	2	17	4,722	606	4,728	1,549	0.23
1NMEps Dec	0	0	17	4,669	604	4,649	670	0.00
1NMEps Dec	0	1	16	4,552	605	4,579	1,816	0.07
1NMEps Dec	0	2	17	4,745	610	4,751	1,227	0.32
1NMEps Dec	1	0	17	4,681	598	4,663	994	0.01
1NMEps Dec	1	1	16	4,476	593	4,507	1,255	0.13
1NMEps Dec	1	2	16	4,683	604	4,713	1,163	0.39
1NMEps Dec	2	0	17	4,712	606	4,699	957	0.00
1NMEps Dec	2	1	16	4,708	524	4,732	1,609	0.00
1NMEps Dec	2	2	17	4,818	605	4,814	1,882	0.09
1NMEps Dec	3	0	16	4,683	607	4,691	804	0.00
1NMEps Dec	3	1	16	4,630	587	4,637	3,043	0.11
1NMEps Dec	3	2	17	4,851	592	4,854	1,602	0.03
1NMEps Dec	4	0	16	4,613	604	4,631	817	0.19
1NMEps Dec	4	1	16	4,305	608	4,318	1,986	0.13
1NMEps Dec	4	2	16	4,787	593	4,830	1,287	0.00
ModEps Inc	0	0	17	4,749	596	4,754	345	0.25
ModEps Inc	0	1	16	4,548	605	4,544	1,491	0.61
ModEps Inc	0	2	16	4,736	603	4,775	510	0.00
ModEps Dec	0	0	17	4,690	609	4,692	443	0.00
ModEps Dec	0	1	16	4,366	596	4,395	2,522	0.31
ModEps Dec	0	2	16	4,650	609	4,662	1,513	0.01

Appendix B

Full rolling horizon results for the analytical services case study (Section 4.4.2)

Table B.1: Rolling horizon results for the 1NM method for the analytical services case study

f_1 ID	f_2 ID	TTP	d_r(TTP) [%]	AMS	d_r(AMS) [%]	CPU t. [sec]	Gap [%]
0	0	60,418	−2.54	20,413	14.3	3,681	0.11
0	1	60,706	−2.07	18,657	4.5	1,887	0.03
0	2	60,228	−2.84	24,546	37.4	206	0.00
1	0	60,915	−1.73	20,053	12.3	2,158	0.06
1	1	61,719	−0.44	19,512	9.3	1,894	0.03
1	2	60,399	−2.57	24,318	36.2	268	0.00
3	0	61,854	−0.22	20,052	12.3	3,244	0.06
3	1	60,422	−2.53	19,566	9.6	2,039	0.03
3	2	61,136	−1.38	24,142	35.2	165	0.00
4	0	59,739	−3.63	19,703	10.3	2,773	0.06
4	1	58,735	−5.25	19,850	11.1	2,276	0.03
4	2	60,419	−2.53	24,205	35.5	355	0.00

Table B.2: Rolling horizon results for the 1NM method with 6 minute solver time limit

f_1 ID	f_2 ID	TTP	d_r(TTP) [%]	AMS	d_r(AMS) [%]	Mean CPU time [sec]	Gap [%]
0	0	60,335	−2.67	20,266	13.5	738	0.13
0	1	60,870	−1.81	19,694	10.3	649	0.05
1	0	61,178	−1.31	20,028	12.1	710	0.07
1	1	61,988	− 0.003	19,312	8.1	630	0.04
3	0	61,943	−0.08	19,943	11.7	713	0.09
3	1	60,885	−1.78	19,613	9.8	575	0.03
4	0	59,373	−4.22	19,709	10.4	819	0.20
4	1	58,540	−5.57	19,838	11.1	670	0.05

Table B.3: Rolling horizon results for the 1NM-ε and Mod-ε methods

Method	f_1 ID	f_2 ID	TTP	d_r(TTP) [%]	AMS	d_r(AMS) [%]	Mean CPU time [sec]	Gap [%]
1NM-ε Inc	0	0	60,329	−2.68	20,310	13.7	6,314	0.22
	0	1	61,059	−1.50	18,567	4.0	4,014	0.09
	1	0	61,490	−0.81	20,001	12.0	7,999	0.18
	1	1	62,123	0.21	19,555	9.5	3,769	0.10
	3	1	60,906	−1.75	19,324	8.2	5,595	0.14
	4	0	59,307	−4.33	19,588	9.7	7,933	0.12
	4	1	58,812	−5.13	19,763	10.7	6,175	0.11
1NM-ε Inc (6 min)	0	0	60,214	−2.86	20,357	14.0	1,086	1.70
	0	1	60,859	−1.82	19,359	8.4	1,091	0.42
	1	0	61,150	−1.36	19,849	11.1	1,061	0.35
Mod-ε Inc	0	0	60,352	−2.64	20,268	13.5	4,446	0.03
	0	1	61,427	−0.91	18,441	3.3	5,326	0.05
Mod-ε Inc (6 min)	0	0	60,336	−2.67	20,308	13.7	1,113	0.09

Results for the search direction of decreasing f_1 and f_2 are not shown since no feasible solution was found in this direction for any of the (f_1, f_2) combination on day 1 of the RH. (f_1^3, f_2^0) is omitted for the 1NM-ε method, since no feasible solution was found in the search direction of increasing f_1 and f_2.