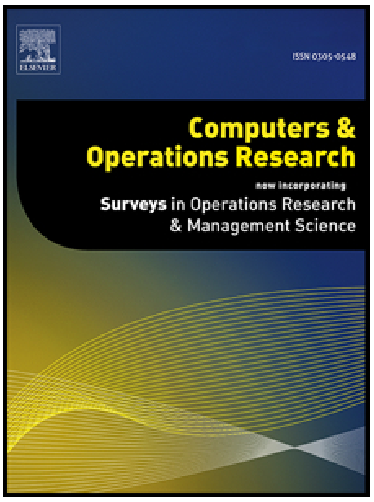

Bi-objective short-term scheduling in a rolling horizon framework: a priori approaches with alternative operational objectives

Do Yeon Lee, Ricardo Fukasawa, Luis Ricardez-Sandoval

PII:	S0305-0548(19)30166-2
DOI:	https://doi.org/10.1016/j.cor.2019.06.006
Reference:	CAOR 4732
To appear in:	*Computers and Operations Research*
Received date:	14 February 2018
Revised date:	25 April 2019
Accepted date:	17 June 2019

Highlights

- Addressed all referees comments
- Explain with more background material how our work is different from previous works
- Gave more insightful analysis of computational results.

Bi-objective short-term scheduling in a rolling horizon framework: *a priori* approaches with alternative operational objectives

Do Yeon Lee[a], Ricardo Fukasawa[b,*], Luis Ricardez-Sandoval[a]

[a] *Department of Chemical Engineering, University of Waterloo, Waterloo, Ontario, Canada*
[b] *Department of Combinatorics and Optimization, University of Waterloo, 200 University Ave W, Waterloo, Ontario, Canada N2L 3G1*

Abstract

This study addresses short-term scheduling problems with throughput and makespan as conflicting objectives, focusing on *a priori* multi-objective methods. Two contributions are presented. The first contribution is to propose *a priori* methods based on the hybridization of compromise programming and the ε-constraint method that have computational benefits over the original methods. The second is to present short-term operational objective functions, that can be used within short-term scheduling to optimize desired long term objectives. Two numerical case studies, one in a semiconductor processing plant and a scientific services facility, are presented using a rolling horizon framework, which demonstrate the potential for the proposed methods to improve solution quality over a traditional *a priori* approach.

Keywords: multi-objective short-term scheduling rolling horizon compromise programming a priori method ε-constraint

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 [35, 6, 42, 29, 27]. Such approach involves the use of short-term scheduling models to determine the optimal timings of operations subject to operational objectives and constraints. One common challenge is that, inside an organization, there are multiple conflicting objectives. These issues can be addressed using one single objective function, or combining objectives into a single function with arbitrarily selected weights; however, it can be advantageous

*Corresponding author
Email address: `rfukasawa@uwaterloo.ca` (Ricardo Fukasawa)

to use an appropriate multi-objective optimization method. For instance, two conflicting objectives often considered in operations scheduling are maximization of the total throughput (TTP) and minimization of the average makespan (AMS).

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. Due to this situation, some multi-objective approaches may be too time consuming for the purposes of short-term scheduling, e.g. *a posteriori* methods, which consumes more CPU time than *a priori* methods [34]. While a priori methods such as compromise programming and ε-constraint method are quite popular [2, 34], they heavily rely on the DMs expertise to select appropriate weights that may return non-dominated solutions that comply with the DMs expectations. Therefore, there is a need to develop practical computationally attractive multi-objective approaches that rely on intuitive weights that can be followed by DMs.

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 operations). 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.

One of the goals of this work is to present *a priori* multi-objective algorithms that can provide practical solutions to multi-objective problems that emerge in short-term scheduling. The key idea is to combine the features of compromise programming and ε-constraint methods to generate two hybrid optimization algorithms that take into account the DMs preferences using intuitive weights based on reference point methods, but that are guaranteed to provide alternative a-priori solutions to such methods. The proposed hybrid algorithms are expected to be computationally efficient since they do not require iterations, i.e. they will search for acceptable solutions in a reduced space, which is mostly determined by the specific objectives (i.e. weights) that the DM would like to consider in the analysis. This is particularly relevant in an industrial setting where large-scale (intensive) multi-objective optimization formulations may need to be formulated and solved in short computational times. To our knowledge, the solution of industrial-scale multi-objective problems using a priori optimization methods still remains an open challenge. In addition, this work will also study 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 an actual scientific services facility.

The organization of this paper is as follows. The rest of this section reviews the prominent contributions on multi-objective approaches. Section 2 presents two new methods considered in this work to address multi-objective problems in short-term scheduling, which are hybridizations of compromise programming and the ε-constraint method. The specific objective functions to address TTP and AMS in short-term scheduling are presented in Section 3. The two case studies featuring multi-objective problems for short-term scheduling are presented in Section 4. Finally, concluding remarks are presented in Section 5.

1.1. Background and literature review on multi-objective approaches

A general multi-objective optimization problem can be formulated as follows:

$$\begin{aligned} \min \ & \mathbf{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_\pi(\vec{x})]^T \\ \text{s.t.} \ & \vec{x} \in \chi \end{aligned} \tag{1}$$

where $\chi \in \mathbb{R}^n$ denotes the set of feasible solutions. $\mathbf{f}(\vec{x}) \in \mathbb{R}^\pi$ is a vector of objective functions of length π; each element in $\mathbf{f}(\vec{x})$ is called an individual objective, cost or performance function.

There are two main types of methods to dealing with multi-objective optimization problems: *a priori* and *a posteriori* methods. *A posteriori* methods aim to provide a DM with efficient trade-off solutions called *Pareto* optimal/*Pareto* efficient/non-dominated solutions [10, 15]. The DM can then afterwards look at the set of Pareto solutions provided and select one that complies with the DM's needs. However, the DM often has a limited amount of time to make a decision, particularly for short-term scheduling applications. Thus, obtaining a set of alternative solutions may be unnecessary or impractical in such cases. Given that the present work 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 [10, 34, 15, 28, 41, 9, 45, 18, 49, 11].

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 [10, 15], rather than a set of solutions for the DM to evaluate. Reference point methods, such as compromise programming [46], are very popular *a priori* methods that have been widely used in practice [16, 4].

These methods aim to minimize the difference between the potential optimal point and a utopia (ideal) point $\mathbf{f}^*$ defined as follows:

$$\mathbf{f}^* := [f_1^*, f_2^*, \ldots, f_\pi^*]^T$$

where

$$f_l^* = \min\{f_l(\vec{x}) : \vec{x} \in \chi\}, \forall l = 1, \ldots, \pi$$

For most of the applications, the utopia point is often unattainable. Hence, different methods have been developed to identify a (compromise) solution point that is as close as possible to the utopia point. Most of the compromising programming methods differ in the way the methods specify the distance between the utopia point and the optimal point. One of the most popular compromise programming methods is based on the finding the point in χ that minimizes the distance to the utopia point, i.e.

$$\begin{aligned} \min \quad & \left(\sum_{l=1}^{\pi} (f_l(\vec{x}) - f_l^*)^p \right)^{\frac{1}{p}} \\ \text{s.t.} \quad & \vec{x} \in \chi \end{aligned} \tag{2}$$

where p ($1 \leq p \leq \infty$) is a parameter selected by the DM that denotes the L_p norm used to measure the distance between the optimal (compromise) point and the utopia point. Setting $p = 1$ or $p = \infty$ leads to a natural way to reformulate the objective function using only linear expressions. Setting $p = 1$ will always return a non-dominated solution even if the feasible solution space is non-convex whereas $p = \infty$ may not necessarily return a non-dominated solution for a non-convex feasible solutions space [17]. Note that the implementation of compromise programming methods often requires the application of transformation methods (e.g. fractional deviation or normalization) to specify dimensionless (normalized) optimization problems. More details about this method are presented in the next section.

In general, the benefit of compromise programming lies in that it returns the compromise solution that best approximates an ideal unattainable solution [32, 4]. The ε-constraint method [21] is another method that has enjoyed popularity due to its relative simplicity [10, 8]. 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.

For a bi-objective problem (i.e. $\pi = 2$), the ε-constraint formulation can be posed as follows. Consider the following two optimization problems:

$$\begin{aligned} \min \quad & f_1(\vec{x}) \\ \text{s.t.} \quad & \vec{x} \in \chi \\ & f_2(\vec{x}) \leq \varepsilon_2 \end{aligned} \tag{P1}$$

and

$$\begin{aligned} \min \quad & f_2(\vec{x}) \\ \text{s.t.} \quad & \vec{x} \in \chi \\ & f_1(\vec{x}) \leq \varepsilon_1 \end{aligned} \tag{P2}$$

where ε_1 and ε_2 represent the limits imposed on each objective function, respectively. The ε-constraint method consists in first obtaining f_1^* - equivalently solving (P1) for a high value of ε_2 - and then iteratively solving (P1) for smaller and smaller values of ε_2. Alternatively, a similar procedure can be done iteratively by solving (P2) and changing the values of ε_1. Details about this method can be found elsewhere [36, 7]. Such a method is an a-posteriori method, in that it computes several candidate points that the DM can choose a-posteriori. However, one concern is that, at any iteration, the point obtained by solving (P1) or (P2) may not be strong Pareto efficient. Özlen and Azizolu [37] propose an alternative to this method that changes the objective function of (P1) and (P2) so that all computed solutions are strong Pareto efficient. In fact, their method can compute all possible strong Pareto efficient solutions, with a careful choice of ε_1 and ε_2.

While the ε-constraint method is typically classified as *a posteriori* method [36, 13, 8], there are studies that have classified it as *a priori* method [34, 11], as the DM's preferences can be expressed in the choice of the prioritized objective, and the selection of the ε values. As mentioned above, we focus on the *a priori* aspect of the ε-constraint method in this work, and propose two methods that use a hybridization of compromise programming and the ε-constraint method in order to obtain computationally cheaper algorithms.

There have been relatively few studies on scheduling problems that have applied reference point methods [1] or the ε-constraint method [20, 47, 5]. Allouche et al. [1] 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. [20] 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 [47] and Castro et al. [5] applied the ε-constraint method for problems in batch plant scheduling with 4-11 products and 3-14 stages, respectively. While Yue and You [47] considered the economic and environmental objectives, Castro et al. [5] 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; hence the motivation to develop new *a priori* multi-objective approaches to address these issues.

2. 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

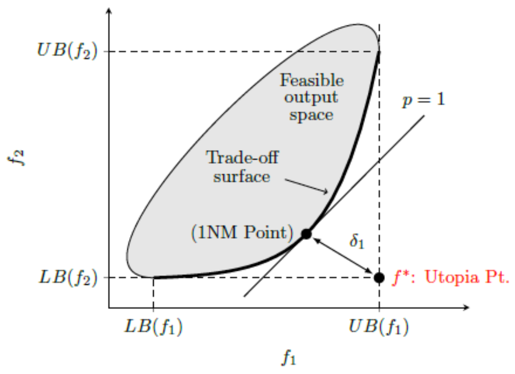

Figure 1: 1-norm minimum trade-off solution on trade-off surface

of the 1NM method and the ε-constraint method.

Throughout this section, and from this point forward in the study, we will switch slightly from the generic notation of Section 1 to the more particular case that we are interested in solving, namely, the following bi-objective problem:

$$\begin{aligned} &\max \ f_1(\vec{x}) \\ &\min \ f_2(\vec{x}) \\ &\text{s.t.} \quad \vec{x} \in \chi \end{aligned} \tag{3}$$

2.1. 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 $f^* := (UB(f_1), LB(f_2))$ as shown on Figure 1. The coordinates $UB(f_1)$ and $LB(f_2)$ 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. The coordinates $LB(f_1)$ and $UB(f_2)$ in Figure 1 can be obtained by solving $\max\{f_1(\vec{x}) : \vec{x} \in \chi, f_2(\vec{x}) = LB(f_2)\}$ and $\min\{f_2(\vec{x}) : \vec{x} \in \chi, f_1(\vec{x}) = UB(f_1)\}$, respectively. A compromise solution on the trade-off surface is obtained by minimizing a p-norm distance (δ_p) from the utopia point [46]. This approach is also referred to as compromise programming, and it is considered to be a special case of more general reference point methods, which may use any reference point, not just the utopia point [44, 15, 39, 32].

In the *a priori* compromise programming approach, the DM's preference can be expressed in the selection of the p-norm distance to minimize, or through a strategy of selecting weights [46, 32]. 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 (3), the 1NM Point is computed by solving the following aggregated and scalarized problem [19]:

$$\begin{aligned} &\min\ (1 - \hat{f}_1(\vec{x})) + \hat{f}_2(\vec{x}) \\ &\text{s.t.}\quad \vec{x} \in \chi \end{aligned} \tag{4}$$

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

$$\hat{f}_1(\vec{x}) = \frac{f_1(\vec{x}) - LB(f_1)}{UB(f_1) - LB(f_1)} \qquad \hat{f}_2(\vec{x}) = \frac{f_2(\vec{x}) - LB(f_2)}{UB(f_2) - LB(f_2)} \tag{5}$$

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 [46]; also, this 1-norm based approach leads to a linear formulation given linear objective functions and constraints as shown in problem (4). While the 1NM method is popular, one downside is that the DM preferences are inherently assumed, i.e., $p = 1$, and the DM has no freedom to express other preferences. Indeed, one of the main motivations to our approach is to be able to provide to the DM a-priori solutions that are different than the 1NM point. For instance, if the DM considers that the 1NM point consistently gives a point with too low value of $f_1(\vec{x})$, we would like to provide a tool for the DM to specify a different choice of point.

Other options that have been proposed to obtain different *a-priori* efficient solutions include obtaining the closest point to the utopia point using a different p-norm. This, however, 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 [46], which are more common in pure integer problems [19]. Furthermore, in practice, the DM may want to express their preferences in terms of specific objectives, rather than in terms of the p-norm which can be rather non-intuitive.

Another alternative is to use one of the weighed compromise programming methods available, e.g. weighted sum methods [43, 2], augmented weighted term methods [36, 32]. However, one of the downsides of such methods is not being able to generate all Pareto optimal points. In addition, putting different weights in the objective functions may not be very intuitive for a DM. For example, setting a weight of 2 for $\hat{f}_2(\vec{x})$ says that objective function $\hat{f}_2$ is twice as important as $\hat{f}_1$. Such a quantification of importance of the normalized objectives seems to be a bit artificial. In addition, it is quite possible that, for a particular weight choice, the returned point will still be the 1NM point, thus effectively not providing the DM with any alternative to the 1NM point.

A final option that should be mentioned that can provide the DM with alternatives to the 1NM point is to compute all (strong) Pareto efficient points using an a-posteriori method and then picking the one that has the desired property. However such approach is bound to be computationally expensive since computing the Pareto frontier is an iterative process that may require many iterations to finish. Hence, their application to industrial-scale systems is computationally intractable.

Therefore, we present in this work two practical approaches to address these shortcomings. The idea is to base our methods on a parameter Υ that, with a value of $\Upsilon = 0$ is equivalent to returning the 1NM point, with a value equal to 1 is equivalent to returning a point with $f_1(\vec{x}) = UB(f_1)$ or $f_2(\vec{x}) = LB(f_2)$. Any value in between measures how far from the 1NM point does the DM want to go either *increasing* f_1 towards the maximum value $UB(f_1)$ or *decreasing* f_2 towards the minimum value $LB(f_2)$. In particular, any value $\Upsilon \neq 0$ is guaranteed to find a Pareto solution that is different from the 1NM point.

2.2. Modified ε-constraint method (Mod-ε)

As previously mentioned, 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 2.1 and the ε-constraint method, referred henceforth as the Mod-ε method. We use a variant of *a posteriori* ε-constraint method presented by Özlen and Azizolu [37] 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 1, or in the decreasing search direction (i.e., from the 1NM Point towards (LB(f_1), LB(f_2)) in Figure 1). We present the Mod-ε method below for the bi-objective problem (3):

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)).
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

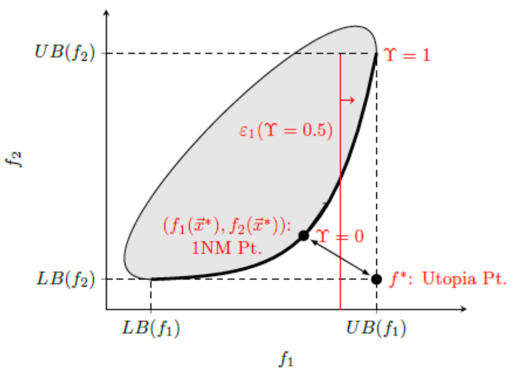

Figure 2: Using the DM preference value in the increasing search direction for Mod-ε

objective (decreasing search direction). This step is visualized in Figure 2 for the increasing search direction.

5. Solve the constrained weighted problem (6) (increasing search direction), or problem (7) (decreasing search direction). This returns the final DM preferred solution in the desired search direction.

$$\begin{aligned} \min \quad & f_2(\vec{x}) - w_1 f_1(\vec{x}) \\ \text{s.t.} \quad & f_1(\vec{x}) \geq \varepsilon_1 \\ & \vec{x} \in \chi \end{aligned} \tag{6}$$

$$\begin{aligned} \max \quad & f_1(\vec{x}) - w_2 f_2(\vec{x}) \\ \text{s.t.} \quad & f_2(\vec{x}) \leq \varepsilon_2 \\ & \vec{x} \in \chi \end{aligned} \tag{7}$$

Note that the weights w_1 and w_2 shown above are expressed such that the solution of problem (6) or (7) are bi-objective efficient for an integer programming problem (see Theorems 2.1-2.3 of Özlen and Azizolu [37]). Therefore, in this work, we limit the discussion and implementation of the Mod-ε method to integer programming problems. Furthermore, as mentioned in Özlen and Azizolu [37], in this approach, f_1 and f_2 are assumed to have only integral values, which can be attained through scaling any rational objective function, therefore the results of Özlen and Azizolu [37] can directly be applied to this method, thus guaranteeing that any solution obtained will be strong Pareto optimal.

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. Note that a different way to compute the exact same a-priori point is to run the algorithm of Özlen and Azizolu [37] until the desired point is found. Indeed, this can be seen as just one iteration of the algorithm of Özlen and Azizolu [37] with a particular starting value of ε_1 or ε_2 being generated. The advantage of the proposed approach is that, by starting with the 1NM point, we are ensured to find the desired point in one iteration, while Özlen and Azizolu [37] may need to compute nearly the entire Pareto frontier to reach the desired point. In addition, since the approach is based on Özlen and Azizolu [37], it overcomes the concerns regarding compromise programming discussed in section 2.1.

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. 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. Therefore, such expression of Υ can be valuable if the DM decides to stay relatively close to the 1NM Point rather than explicitly selecting a bound within the objective range without any insight with respect to the ideal trade-off.

2.3. 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 (6) or (7), 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 (3):

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)).
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 (8) (increasing search direction), or problem (9) (decreasing search direction). This returns the final DM preferred solution in the desired search direction.

$$\begin{aligned} \min\ & (1-\hat{f}_1(\vec{x}))+\hat{f}_2(\vec{x}) \\ \text{s.t.}\ & f_1(\vec{x}) \geq \varepsilon_1^+ \\ & f_2(\vec{x}) \geq \varepsilon_2^+ \\ & \vec{x} \in \chi \end{aligned} \tag{8}$$

$$\begin{aligned} \min\ & (1-\hat{f}_1(\vec{x}))+\hat{f}_2(\vec{x}) \\ \text{s.t.}\ & f_1(\vec{x}) \leq \varepsilon_1^- \\ & f_2(\vec{x}) \leq \varepsilon_2^- \\ & \vec{x} \in \chi \end{aligned} \tag{9}$$

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 $\hat{\delta_1}$. This difference is demonstrated in Figure 3, which simultaneously shows trade-off solutions and their corresponding normalized 1-norm distances for all relevant choices of Υ in both increasing and decreasing directions. Note that if Mod-ε is used for all possible Υ values, it is equivalent to using the a-posteriori method of Özlen and Azizolu [37], thus the label on the picture specifies it as Mod-ε/O&A. We stress that the proposed methods are a-priori and, thus for a fixed Υ would only return one solution illustrated in Figure 3. The purpose of Figure 3 is to illustrate how solutions may differ depending on using Mod-ε or 1NM-ε.

However, this does not necessarily mean that the 1NM-ε method would always provide results that are more preferable to the DM. For the example given in Figure 3, searching in the decreasing search direction (from the 1NM Point towards LB) using the bound $f_2(\vec{x}) \leq 4$, the Mod-ε method returns the solution at $(f_1, f_2) = (28, 3.8)$ with $\hat{\delta_1} = 0.5113$, while the 1NM-ε method returns the solution at $(25, 2.6)$ with $\hat{\delta_1} = 0.5111$. In this case, while the 1NM-ε solution is slightly closer to the Utopia Point than the Mod-ε solution, the DM might still prefer the Mod-ε solution with higher f_1 value. Furthermore, if the two methods have equivalent Pareto frontier, then their solutions would also be the same using the same value of Υ.

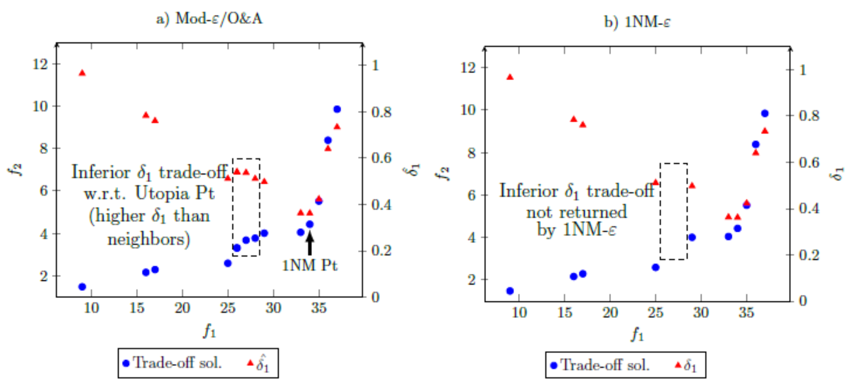

Figure 3: Comparison of Pareto sets from Mod-ε/O&A and 1NM-ε. The dashed rectangle represents solutions that cannot be generated by the 1NM-ε method.

3. Short-term objective functions

As discussed in the introduction, maximizing the total throughput (TTP) and minimizing the average makespan (AMS) can often be computationally intractable 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 rolling horizon approaches to scheduling are common (e.g. [12, 33, 31]), the typical assumption is that there is an associated given cost for each activity and the total cost is to be minimized. Even if one looks simply at short term scheduling, this assumption on the objective function also seems to hold (e.g. [35, 24]). Our approach differs from those in that the objectives that are given are not in terms of immediate costs of finishing/performing any actual task, but the long term objectives that will unlikely be realized within a singe CH. While the present approaches have been tested on two particular applications, i.e., a semiconductor plant and an actual scientific services facility, a wide variety of industrial applications require this approach to schedule operations in an optimal fashion, e.g., steelmaking [30] and industrial gases supply chain [48]. In order to provide the necessary context and notation for the objective functions proposed in this work, we first provide the problem definition in section 3.1 before presenting the alternative objective functions. A number of alternative operational objective functions are proposed in section 3.2.

3.1. Problem Definition

The following problem definition applies to both of the problems considered in this study (see Section 4). 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 $i \in I$ is composed of a discrete number of materials. These materials are processed using a set of processing units P and each processing unit $p \in P$ consists of a set J_p of identical machines that perform a specific process. 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 specific sequence of processing units, called a path. The path of a task i, denoted by $\wp_i$, is a sequence of $n(i)$ distinct processing units $(p_1^i, \dots, p_{n(i)}^i)$, where $p_k^i \in P$ for all $i \in I$, $k = 1, \dots, n(i)$. The materials in task i must visit each subsequent processing unit in $\wp_i$ sequentially, that is, p_{k+1}^i in $\wp_i$ can only be visited if p_k^i has already been visited. Materials are considered to have visited processing unit p if it has been processed by one of the machines in J_p. We assume that there is no transportation time between different processing units, and that there are no restrictions on intermediate storage.

All machines in a processing unit p have an associated integer number $nm_p \geq 1$ of operational modes. Each $m \in \Psi_p := \{1, \dots, nm_p\}$ represents a predetermined value of processing time τ_{pm} associated to a machine in p used in mode m. For each task $i \in I$ and its path $\wp_i$, there is a specific sequence of operational modes $(m_1^i, \dots, m_{n(i)}^i)$ that determines that the k-th processing unit in $\wp_i$ must be used in mode $m_k^i \in \Psi_{p_k^i}$, for all $k = 1, \dots, n(i)$.

Machines in a processing unit p have a specific capacity, denoted as β_p, and a machine $j \in J_p$ may be loaded with at most β_p amount of materials from potentially different tasks. Once a machine $j \in J_p$ has been turned on to process the materials for operational mode m, it will run without interruption for a time τ_{pm}. After this time, the machine is considered to be available; also, the materials are considered to have visited the corresponding processing unit, and they are ready to visit the next processing unit in their path. 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*.

For the problems being considered in this study, we allow a machine $j \in J_{p_k^i}$ 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_k^i 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.

Given the above problem definition, we use the integer linear programming (ILP) scheduling model based on the multitasking flexible discrete-time formulation presented by Lagzi et al. [26]. That study has shown that this modeling approach is effective in solving large-scale ILP short-term scheduling problems where multitasking is a key operational feature. Modifications were made for this study in order to allow for potentially different processing times for each processing unit, and to fit with the rolling horizon approach. These modifications are presented in Appendix B. The following are the nonnegative integer decision variables from that formulation that are relevant to the objective functions being proposed in this work:

- B_{ikt}: the amount of materials from task i that are set to start being processed at the k^{th} processing unit in the path of task i, $p^i_k \in \wp_i$, at the time point t, $\forall\ i \in I,\ 1 \leq k \leq n(i),\ t = 1, \ldots, |\mathcal{E}(p^i_k)|$
- W_{ikt} the amount of materials waiting to start being processed at processing unit p^i_k at time point t , $\forall\ i \in I,\ k = 1, \ldots, n(i),\ t = 1, \ldots, |\mathcal{E}(p^i_k)|$ where $|\mathcal{E}(p)|$ represents the length of unit-specific sequence of time points.

3.2. Objective functions

The ultimate goal of this work is to either maximize throughput or minimize makespan, or obtain a Pareto optimal solution to both these objectives over the entire RH. However, as mentioned before, performing any of these tasks over the entire RH is computationally intractable, even for smaller sized instances. Thus, the need to do a RH approach arises. However, most tasks cannot be finished over a single CH, thus any attempt to directly maximize throughput or minimize makespan over that single CH will have zero throughput or makespan undefined. For this reason, we needed to design objective functions for a single CH that would make progress towards having good throughput/low makespan. These are presented in the following subsections.

3.2.1. 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

Table 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	$\frac{k}{\sum\limits_{j=1}^{n(i)} j}$	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	$\frac{\sum\limits_{j=1}^{k} \tau_{p_k^i m_k^i}}{\sum\limits_{j=1}^{n(i)} \tau_{p_k^i m_k^i}}$	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.

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 (3)) of the following form:

$$\max \sum_{i\in I} \sum_{k=1}^{n(i)} \sum_{t=1}^{\lvert\mathcal{E}(p_k^i)\rvert} (weight) B_{ikt} \tag{10}$$

The above objective function uses the weights presented in Table 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 (10) 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.

3.2.2. 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^i_k, $1 \leq k \leq n(i)$ (i.e., $\sum_{i \in I} \sum_{k=1}^{n(i)} \sum_{t=1}^{|\mathcal{E}(p^i_k)|} W_{ikt} = 0$ for the entire RH), 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^i_1 until the expected time of completion of $p^i_{n(i)}$ 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 (3)) of the following form:

$$\min \sum_{i \in I} \sum_{k=1}^{n(i)} (ST - AT_i + \sum_{j=k}^{n(i)} \tau_{p^i_j m^i_j})(variable) \tag{11}$$

The above objective function uses the variables presented in Table 2, each with an identifying f_2ID, which attempt to produce schedules with close to optimal AMS for the RH. In eq (11), 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 (12) and (13) shown below define the binary decision variable S_{ik} for $f_2\text{ID} = 0$ and $f_2\text{ID} = 1$, respectively. When implementing the objective functions, $f_2\text{ID} = 0$ and $f_2\text{ID} = 1$, S_{ik} must be introduced as a decision variable, in addition to the other decision variables, and the respective constraint (12) or (13) must be added to the scheduling model presented in Appendix B according to the f_2ID.

$$\frac{W_{ik|\mathcal{E}(p^i_k)|}}{1 + \sum_{j=1}^{n(i)} \sum_{t=1}^{|\mathcal{E}(p^i_k)|} a_{ijt}} \leq S_{ik} \leq W_{ik|\mathcal{E}(p^i_k)|} \quad \forall i \in I, k = 1, \ldots, n(i) \tag{12}$$

Table 2: Proposed weights for the makespan minimization objective functions (f_2)

f_2ID	*variable*	Description/rationale
0	S_{ik}, eq (12)	$S_{ik} = 1$ 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 (13)	$S_{ik} = 1$ 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.

$$\frac{W_{ik|\mathcal{E}(p_k^i)|} + \sum_{t'=1,\ldots,|\mathcal{E}(p_k^i)|:\varphi_{p_k^i t'}+\tau_{p_k^i m_k^i}>\varphi_{p_k^i |\mathcal{E}(p_k^i)|}} B_{ikt'}}{1 + \sum_{j=1}^{n(i)} \sum_{t=1}^{|\mathcal{E}(p_k^i)|} a_{ijt}} \leq S_{ik} \leq W_{ik|\mathcal{E}(p_k^i)|}$$
$$+ \sum_{t'=1,\ldots,|\mathcal{E}(p_k^i)|:\varphi_{p_k^i t'}+\tau_{p_k^i m_k^i}>\varphi_{p_k^i |\mathcal{E}(p_k^i)|}} B_{ikt'} \quad \forall i \in I, k = 1, \ldots, n(i) \tag{13}$$

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.

We end this section by observing that both TTP and AMS objectives ultimately aim at finishing tasks as early as possible, but ultimately these objectives do conflict in which tasks to prioritize. Nonetheless, since the objective functions proposed are merely heuristic short-term suggestions to simulate TTP and AMS, it is also possible that an objective function designed purely with TTP in mind performs well relative to the AMS objective (and vice-versa).

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. [40], whereas the second is a full industrial-sized case study of the scientific services scheduling problem previously reported in the literature [38, 25, 26]. 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 3.2.1 and 3.2.2, respectively. For all rolling horizon (RH) results presented in sections 4.1.1 and 4.2.1, 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 2.2 and 2.3. 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 this way, we could compare a different range of choices of the DM: the 1NM, the extreme choices of individual objectives, and a Pareto solution in between.

In order to simplify reference to specific objectives, we use the notation $f_1^{f_1ID}$ and $f_2^{f_2ID}$ when referring to f_1 and f_2 objectives with specific f_1ID weight and f_2ID variable as presented in Tables 1 and 2, respectively. For example, f_1^1 refers to f_1 objective function with $f_1\text{ID} = 1$ (weight shown in Table 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 2.2.

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) [3] and JuMP modeling language (0.18.0) [14]. Optimization runs were performed using IBM ILOG CPLEX Optimizer (12.6.0) [22] 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 [23].

4.1. Semiconductor case study

We solved two instances of the semiconductor case study adapted from the case study presented by Senties et al. [40], which contains 5 different product

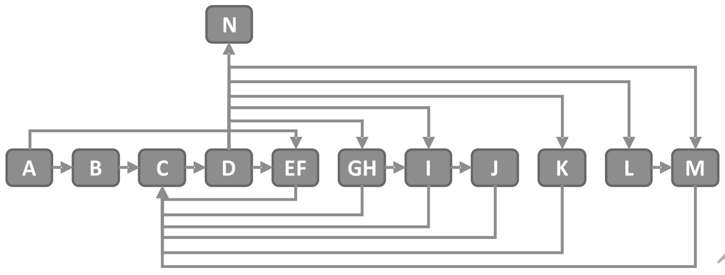

Figure 4: Process map for the semiconductor case study

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. The product recipes and the data on each processing unit are presented in Tables B1 and B2 in Appendix C.

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 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 B3 and B4 in Appendix C.

4.1.1. Rolling horizon results: semiconductor

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 except single-objective f_2 minimization. While more comprehensive results are presented in Appendix D, 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 3. 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:

$$d_r(\text{TTP}) = \frac{TTP - TTP^*}{TTP^*} \qquad (14) \qquad d_r(\text{AMS}) = \frac{AMS - AMS^*}{AMS^*} \qquad (15)$$

While we use such benchmarks against the best single-objective short-term

Table 3: Best case TTP and AMS for single-objective runs 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 [%]
1st	4	-	82^a	0	4892	8.35	21	0.00
inst.	-	1	38	-53.66	4515^b	0	84	0.00
2nd	0	-	876^a	0	4800	2.29	47	0.00
inst.	-	1	605	-30.94	4692^b	0	286	0.00

Values with superscripts a and b represent TTP* and AMS*, respectively

approximations for ease of comparison, TTP* and AMS* should not be considered to be optimal long-term values, i.e., d_r(TTP) can be positive and d_r(AMS) can be negative. Note that each single-objective optimization run produces a solution for a short-term constituent horizon that is optimal with respect to its short-term approximating objective function, and does not guarantee long term optimality in TTP or AMS, which we deem to be unobtainable, due to computationally prohibitive runtimes. In particular, a positive d_r(TTP) value would suggest that particular combination of f_1 and f_2 short-term objectives leads to better results for TTP over time than the best available naive single-objective implementation.

In Figures 5 and 6, 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 $\{d_r(\text{TTP})^A > d_r(\text{TTP})^B$ and $d_r(\text{AMS})^A \leq d_r(\text{AMS})^B\}$ or $\{d_r(\text{TTP})^A \geq d_r(\text{TTP})^B$ and $d_r(\text{AMS})^A < d_r(\text{AMS})^B\}$. Based on this dominance relationship, we present only the results that are closer to the bottom right quadrant, and exclude most of the dominated results for readability.

In Figure 5, 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 6, 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 approaches 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 ($d_r(\text{TTP}) = -0.41$, $d_r(\text{AMS}) = -3.16$) com-

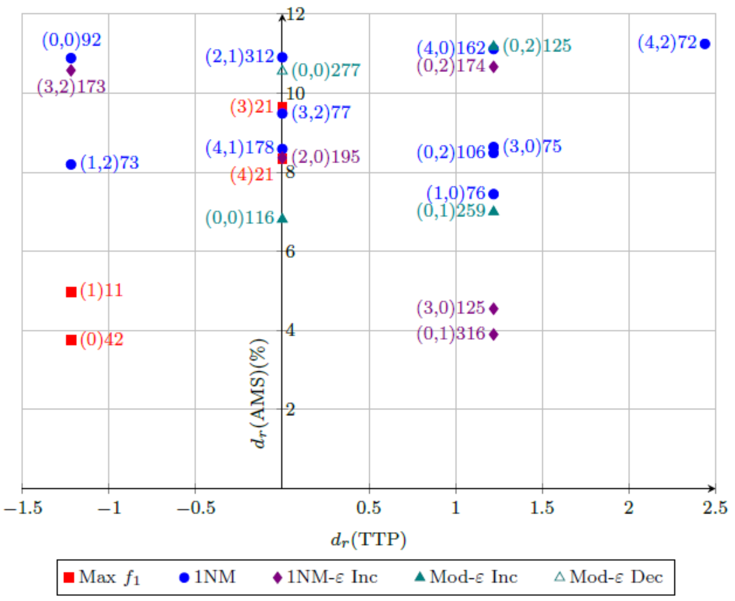

Figure 5: d_r(TTP) vs d_r(AMS) for the first instance of the semiconductor case study

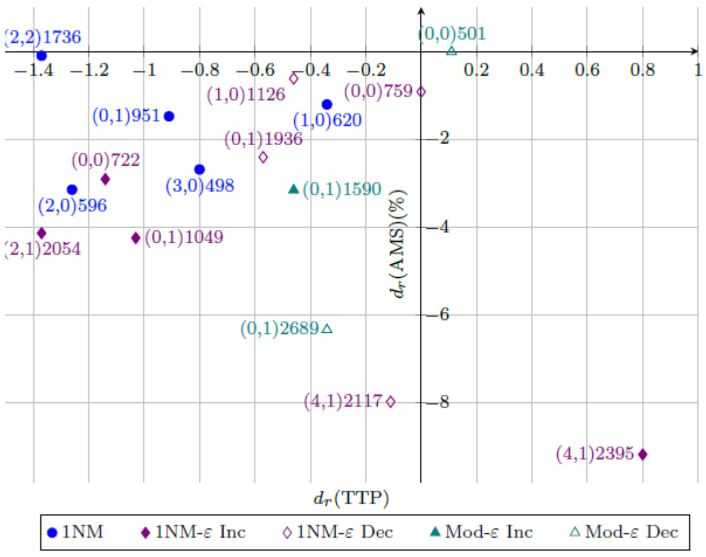

Figure 6: d_r(TTP)vs d_r(AMS)for the second instance of the semiconductor case study

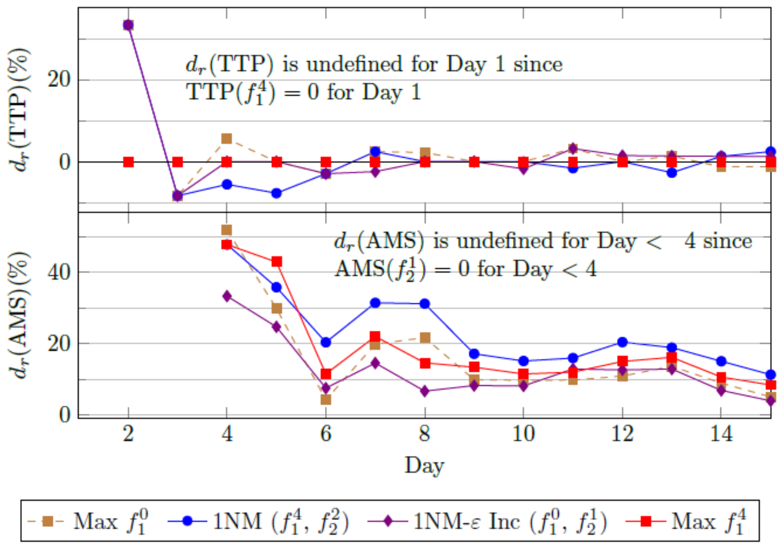

Figure 7: Change in d_r(TTP) and d_r(AMS) throughout the RH for the first instance

pared to 1NM (f_1^0, f_2^1) (d_r(TTP) = −0.91, d_r(AMS) = −1.48) as shown in Figure 6. However, the improvements in TTP and AMS were generally achieved at the expense of additional CPU time.

Based on the results of Figures 5 and 6, we compare how d_r(TTP) and d_r(AMS) change throughout the RH for the two instances in Figures 7 and 8. 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 8. Furthermore, we compare these results to the single-objective Max f_1 run producing TTP* for each instance as reported in Table 3. 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 7 and 8, we can observe that the TTPs of the dominant approaches 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 7 and 8, while its d_r(TTP) did not fall

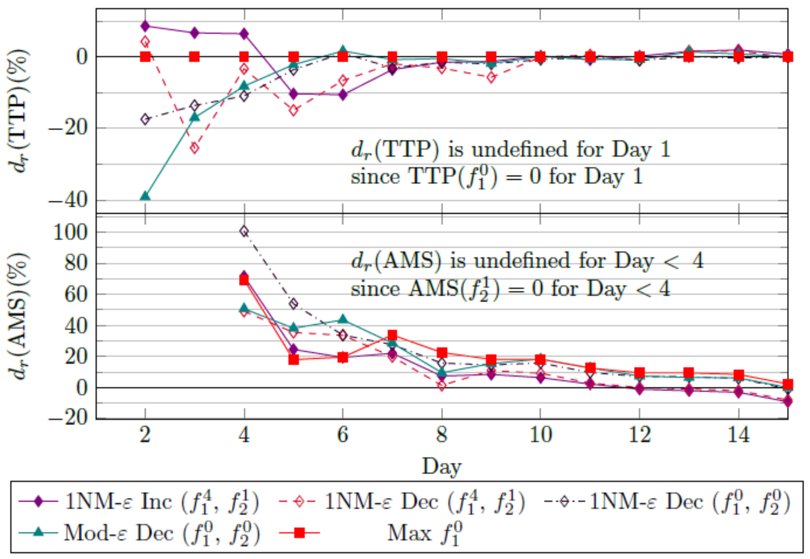

Figure 8: Change in d_r(TTP) and d_r(AMS) throughout the RH for the second instance

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).

4.2. Scientific services case study

The scientific services facility model considered in this work was that presented by Lagzi et al. [26] and consists of 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 9. In Appendix E, the 19 paths defining this partial network are presented in Tables D1, and capacities, resources, and processing times are presented in Table D2. Unlike the semiconductor case, the scientific services facility in consideration does not utilize different operational modes, i.e., each processing unit has only one possible processing time. The identity of the industrial partner and the complete

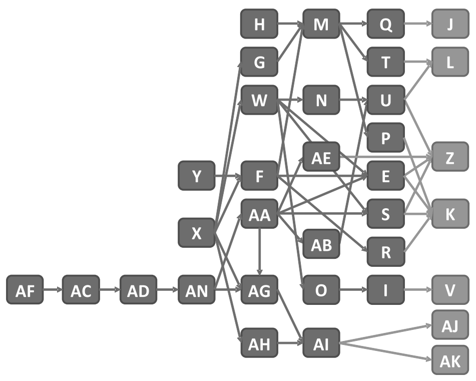

Figure 9: Partial process map for the scientific services case study

data are not disclosed as per our nondisclosure agreement with the industrial partner. Refer to section 3.1 of Lagzi et al. [26] for more details, including part of the data set used in this study.

For this scientific 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 instance studied in this work was generated by first taking a snapshot of the scientific 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 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.

4.2.1. Rolling horizon results: scientific 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 F, 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. 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 (14)-(15).

Table 4: Best case TTP and AMS for single-objective runs for the scientific services case study

f_1 ID	f_2 ID	TTP	d_r(TTP) [%]	AMS	d_r(AMS) [%]	Mean CPU time [sec]	Mean Gap [%]
1	-	61990[a]	0	20,447	14.5	88	0.00
-	1	32,025	-48.34	17859[b]	0	251	0.01

Values with superscripts a and b represent TTP* and AMS*, respectively

In Figure 10, 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 (see Table E1 in Appendix F). 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 E2 and E3 in Appendix F). 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 10, 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 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

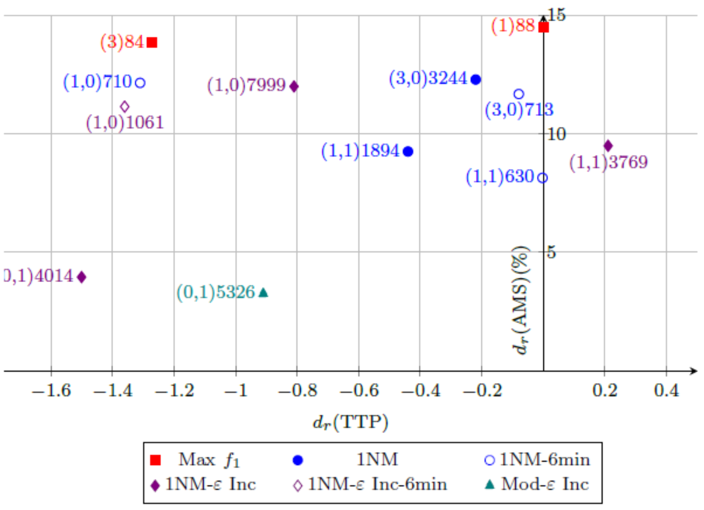

Figure 10: d_r(TTP) vs d_r(AMS) for the scientific services case study

in the increasing search direction within 6 minutes at some point during the RH.

In Figure 11, 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 objective 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). It is worth mentioning that the results obtained can change sig-

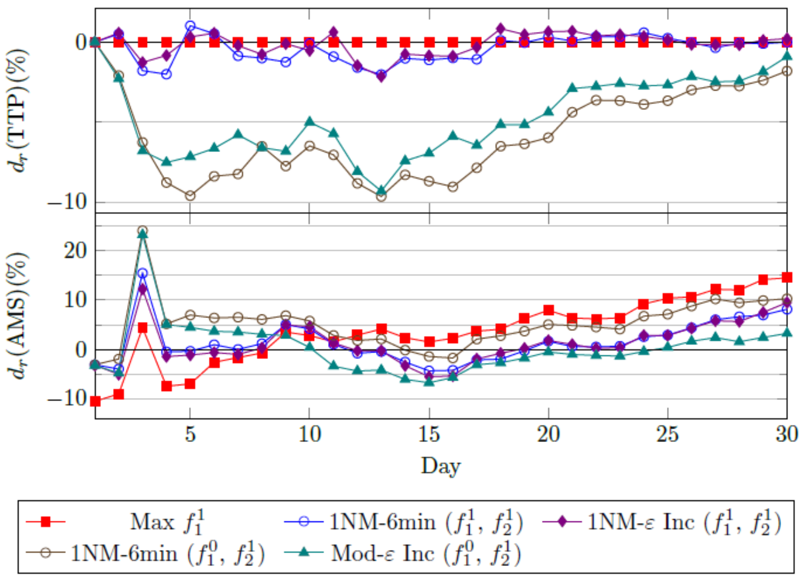

Figure 11: Change in d_r(TTP) and d_r(AMS) throughout the rolling horizon

nificantly if the computational time allowed varies. This is because some more computationally intensive choices may be more promising if given more time. Therefore, one needs to carefully consider this and other factors when considering these results in different contexts.

4.3. Conclusions from experiments

While there was not a single choice of approach that consistently outperformed all others in our experiments (two semiconductor cases and one scientific services), there are certain common conclusions that apply to both and will likely be helpful in similar situations. We outline such conclusions here.

The first observation is that the proposed multi-objective approaches indeed perform better than the single-objective ones. Note that in all case studies, most non-dominated approaches were obtained from multi-objective approaches. In fact, only in the first semiconductor case there was a non-dominated approach that arose from single-objective Max f_1^0 and, in such case, its (TTP, AMS) were $(81, 4684.7)$ which are very close to another non-dominated approach, the 1NM-ε Inc (f_1^0, f_2^1), with $(TTP, AMS) = (83, 4691.4)$. Thus, the choice of using a multi-objective approach seems to consistently outperform single-objective ones.

Secondly, regarding the choice of objective function to use, one relevant observation is that the choice of f_2^1 seems to be suitable and attractive. Indeed, for all case studies, if we limit ourselves to using only approaches that consider f_2^1, we will still keep at least one non-dominated solution. While the choice for f_1 is less clear, one additional observation is that f_1^2 and f_1^3 did not lead to non-dominated solutions and thus discarding them would still leave us with all the non-dominated solutions found. In addition, regarding the multiobjective method, 1NM-ε Inc performs well in our experiments. Indeed, in all cases, there is at least one non-dominated solution that uses 1NM-ε Inc. Narrowing the choices further than what is discussed above will come with a loss of non-dominated solutions.

These results show the value of our proposed methods and study of different objective functions.

5. Conclusion

This study has proposed effective alternatives to address some of the issues regarding short-term scheduling of multipurpose plants, where there exists two conflicting tactical objectives in maximizing the total throughput of the plant, and minimizing the average makespan of tasks being completed. Our 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 scientific services sector case study. In these case studies, we used a rolling horizon approach 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. The proposed algorithms provide an alternative way for the DM to specify preferences without the need to specify how much better is one objective function relative to the other (weighted compromise programming) or specifying a particular p-norm to be minimized. 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 we showed that this approach can significantly reduce the CPU time for an industrial-sized problem without a significant degradation in solution quality, if any.

Several challenges had to be addressed in this work and improving on any of these areas is an interesting area of future research. For instance, due to the long-term nature of the true objective functions of interest no true optimal solutions are available to benchmark our work against. Computing such optimal solutions, or at least better bounds on their values by more advanced methods would be very valuable. Related to such issues, the large scale nature of the problem leads to tractability issues on any of the proposed approaches. Improving computational times would also be an interesting avenue of future research.

In addition, the multi-objective nature of the problem makes it difficult to identify a "best" approach. New ways of evaluating the approaches or narrowing down even further the possible choice of approaches would also be an interesting follow-up study. Finally, this work was an initial and promising proof of concept, but additional tests should be performed in a follow-up study to make sure that its conclusions hold in a broader set of cases.

Acknowledgment

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

References

References

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

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

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

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

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

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

[7] Chankong, V., Haimes, Y., 1983. Multiobjective Decision Making: Theory and Methodology. North-Holland.

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

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

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

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

[12] Dimitriadis, A., Shah, N., Pantelides, C., 1997. Rtn-based rolling horizon algorithms for medium term scheduling of multipurpose plants. Computers and Chem. Eng 21, S1061–S1066.

[13] Diwekar, U., 2008. Introduction to Applied Optimization. Vol. 22. Springer, Boston.

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

[15] Ehrgott, M., 2005. Multicriteria Optimization, 2nd Edition. Springer, Berlin, Heidelberg.

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

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

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

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

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

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

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

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

[24] Ierapetritou, M. G., Floudas, C. A., 1998. Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes. Ind. Eng. Chem. Res. 37 (11), 4341–4359.

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

[26] Lagzi, S., Lee, D. Y., Fukasawa, R., Ricardez-Sandoval, L., 2017. 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.

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

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

[29] Li, J., Xiao, X., Tang, Q., Floudas, C. A., 2012. 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.

[30] Li, J., Xiao, X., Tang, Q., Floudas, C. A., 2012. 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.

[31] Li, Z., Ierapetritou, M. G., 2010. Rolling horizon based planning and scheduling integration with production capacity consideration. Chem. Eng. Science 65 (22), 5887–5900.

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

[33] Maravelias, C. T., Sung, C., 2009. Integration of production planning and scheduling: Overview, challenges and opportunities. Comput. Chem. Eng. 33 (2), 1919–1930.

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

[35] Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Comput. Chem. Eng. 30 (6-7), 913–946.

[36] Miettinen, K., 1998. Nonlinear Multiobjective Optimization. Vol. 12. Springer US, New York.

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

[38] Patil, B. P., Fukasawa, R., Ricardez-Sandoval, L. A., 2015. 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.

[39] Ruiz, F., Luque, M., Cabello, J. M., 2009. A classification of the weighting schemes in refernece point procedures for multiobjective programming. J. Oper. Res. Soc. 60 (4), 544–553.

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

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

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

[43] Voogd, H., 1983. Multicriteria evaluation for urban and regional planning. Pion.

[44] Wierzbicki, A. P., 1980. The Use of Reference Objectives in Multiobjective Optimization. In: Mult. Criteria Decis. Mak. Theory Appl. No. August. pp. 468–486.

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

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

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

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

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

Appendix A Nomenclature

Acronyms/short-forms
AMS = Average Makespan
CH = Constituent Horizon
d_r = relative difference DM = Decision Maker
ILP = Integer Linear Programming
LB = Lower Bound (of the output efficiency range)
Mod-ε = Modified ε-constraint method (section 2.2)
RH = Rolling Horizon
TTP = Total Throughput
UB = Upper Bound (of the output efficiency range)
1NM = 1-norm minimization/minimum
1NM-ε = 1NM ε-constraint method (section 2.3)
δ_p = p-norm distance (from the utopia point)

Indices
i = client job
j = machine
k = order of processing unit
m = operational mode
p = processing unit
m_k^i = k^{th} operational mode for task i
p_k^i = k^{th} processing unit for task i
t = time point

Sets
I = set of tasks
J_p = set of machines for processing unit p
P = set of processing units
χ = set of constraints
Ψ_p = set of operational modes for processing unit p

Parameters
AT$_i$ = arrival time of task i
H = length of constituent horizon
M_i = sequence of operational modes for task i ST = start time of the constituent horizon
w_1 = weight for the f_1 objective
w_2 = weight for the f_2 objective
ε_1 = minimum value for the f_1 objective (Mod-ε)
ε_2 = maximum value for the f_2 objective (Mod-ε)
ε_1^+ = minimum value for the f_1 objective (1NM-ε, increasing search direction)
ε_2^+ = minimum value for the f_2 objective (1NM-ε, increasing search direction)
ε_1^- = maximum value for the f_1 objective (1NM-ε, decreasing search direction)
ε_2^- = maximum value for the f_2 objective (1NM-ε, decreasing search direction)
Υ = DM preference value
$\wp_i$ = path of processing units for task i
n(i) = size of path $\wp_i$ for task i
β_p = capacity for processing unit p τ_{pm} = processing time for processing unit p, mode m

Decision variables
$\vec{x}$ = vector of decision variables
B_{ikt} = the amount of materials starting to be processed
S_{ik} = binary variable used in the f_2 objectives
W_{ikt} = the amount of materials waiting to start being processed
X_{pmt} = the number of resources starting to be used

Objective functions
f_1 = objective to maximize
f_2 = objective to minimize

Appendix B Flexible discrete-time formulation

The modifications made to the scheduling model presented by Lagzi et al. [26] is presented below. Unchanged parts of the formulation are also presented for completeness.
In addition to the decision variables B_{ikt} and W_{ikt}, one more variable, X_{pmt}, is used for this scheduling formulation.

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

While B_{ikt} and W_{ikt} did not change from the original formulation presented in Lagzi et al. [26], 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. Thus, capacity and equipment resource constraints, i.e. equations (2) and (3) from Lagzi et al. [26], are reformulated as follows:

$$\sum_{i,k:p=p_k^i} B_{ikt} \leq\ X_{pmt}\beta_p;\ \ \forall\ p \in P,\ m \in \Psi_p,\ t = 1, \ldots, |\mathcal{E}(p)| \qquad \text{(A1)}$$

$$z_{pt} + \sum_{m\in\Psi_p} \sum_{\theta} X_{pm\theta} \leq |J_p|;\ \ \forall\ p \in P,\ t = 1, \ldots, |\mathcal{E}(p)|,$$
$$\theta \in \mathcal{E}(p) : \varphi_{pt} < \varphi_{p\theta} + \tau_{pm} \leq \varphi_{pt} + \tau_{pm} \qquad \text{(A2)}$$

β_p is the capacity of a machine in processing unit p that has a set J_p of machines as defined in section 3.1. In constraint (A2), 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 $z_{pt} = 0\ \forall\ p \in P,\ t = 1, \ldots, |\mathcal{E}(p)|$. Otherwise, z_{pt} has a non-zero positive value for some or all $p \in P,\ t = 1, \ldots, |\mathcal{E}(p)|$.

The flow conservation eq (4) from Lagzi et al. [26] has been reformulated as follows:

$$B_{ikt} + W_{ikt} = W_{ik(t-1)} + \sum_{\substack{\theta=1,\ldots,|\mathcal{E}(p_{k-1}^i)|: \\ \varphi_{p_k^i t-1} < \varphi_{p_{k-1}^i \theta} + \tau_{p_{k-1}^i m_k^i} \leq \varphi_{p_k^i t}}} B_{i(k-1)\theta} + a_{ikt} \qquad \text{(A3)}$$
$$\forall\ i \in I,\ k = 2, \ldots, n(i),\ t = 2, \ldots, |\mathcal{E}(p_k^i)|$$

In eq (A3), a_{ikt} is an input parameter representing the amount of materials from task i becoming available to be processed at processing unit p_k^i at time point t. For a particular CH and $a_{ikt} > 0,\ \forall\ i \in I, k > 1, t = 1, \ldots, |\mathcal{E}(p_k^i)|$, a_{ikt} materials must have started being processing at p_{k-1}^i 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}.

The remaining eq (5) and eq (6) of the original formulation are unchanged as presented below:

$$W_{ik1} = a_{ik1}; \quad \forall\, i \in I,\ k = 1 \ldots n(i) \tag{A4}$$

$$B_{ik1} = 0; \quad \forall\, i \in I,\ k = 1 \ldots n(i) \tag{A5}$$

In this work, we use the following time discretization scheme: $\mathcal{E}(p) = (ST, \varphi_{p2}, \varphi_{p3}, \ldots, \varphi_{p(|\mathcal{E}(p)|-2)}, \varphi_{p(|\mathcal{E}(p)|-1)}, ST + H)$ 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.

Appendix C Data for the semiconductor case study

Table B1: 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 B2: 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 B3: 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 B4: 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

Appendix D Full rolling horizon results for the semiconductor case study

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, except for single-objective f_2^1 in Table C1, 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 C1: Rolling horizon results for single-objective implementations for the semiconductor case study

	f_1	f_2	Max	at max iter.		at day 15		Mean CPU
Inst.	ID	ID	Iter.	TTP	AMS	TTP	AMS	time [sec]
1st inst.	0	-	17	90	4,826	81	4,685	42
	1	-	17	90	4,952	81	4,740	11
	2	-	17	90	5,234	81	5,067	65
	3	-	17	90	5,062	82	4,951	21
	4	-	18	90	5,022	82	4,892	21
	-	0	24	90	7,320	74	6,506	133
	-	1	30	84	9,718	38	4,515	84
	-	2	24	90	7,176	74	6,478	101
2nd inst.	0	-	17	900	4,805	876	4,800	47
	1	-	17	900	4,750	871	4,742	38
	2	-	17	900	4,847	875	4,863	148
	3	-	17	900	4,841	865	4,844	62
	4	-	16	900	4,677	858	4,694	85
	-	0	26	900	9,362	736	8,174	49
	-	1	30	795	5,965	605	4,692	286
	-	2	25	900	9,216	739	7,726	73

Table C2: Rolling horizon results for the 1NM method for the 1st semiconductor instance

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 [%]
0	0	18	5,078	81	5,007	77	0.00
0	1	18	5,500	82	5,534	198	0.00
0	2	17	4,969	83	4,899	94	0.00
1	0	17	4,932	83	4,852	67	0.62
1	1	17	5,102	83	5,066	231	0.00
1	2	18	4,994	81	4,886	61	0.00
2	0	18	5,148	81	5,089	140	0.00
2	1	18	5,258	82	5,008	264	0.00
2	2	17	5,172	80	5,131	184	0.00
3	0	18	4,986	83	4,905	63	0.00
3	1	18	5,303	81	5,188	152	0.00
3	2	18	4,974	82	4,944	65	0.00
4	0	17	5,086	83	5,017	144	0.00
4	1	18	5,140	82	4,903	151	0.00
4	2	18	5,086	84	5,024	61	0.00

Table C3: Rolling horizon results for the 1NM method for the 2nd semiconductor instance

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 [%]
0	0	17	4,719	875	4,717	129	0.00
0	1	16	4,610	868	4,623	892	0.23
0	2	17	4,845	871	4,841	545	1.64
1	0	16	4,610	873	4,635	581	0.01
1	1	16	4,596	856	4,598	309	0.00
1	2	16	4,724	869	4,752	599	0.10
2	0	17	4,554	865	4,544	526	0.01
2	1	16	4,723	866	4,740	1,630	0.14
2	2	17	4,698	864	4,688	1,533	0.10
3	0	17	4,580	869	4,566	440	0.01
3	1	16	4,661	867	4,713	1,005	0.02
3	2	17	4,875	866	4,879	1,083	0.03
4	0	17	4,718	870	4,717	856	0.01
4	1	16	4,625	848	4,634	1,181	0.00
4	2	16	4,841	868	4,864	1,416	0.11

Table C4: 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 C5: 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 E Partial data for scientific services case study

Table D1: Paths of processes defining the partial scientific services network of Figure 9

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 D2: Process capacity, resources, and processing time information for scientific 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
AN	0.06	2	7
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
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
AI	0.48	6	48
AJ	0.20	1	1
AK	0.02	1	2
AG	0.50	1	1,008
AH	0.50	1	1,008
V	0.03	1	6
W	0.61	4	162
X	0.01	6	1
Y	0.01	1	1

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.

Appendix F Full rolling horizon results for the scientific services case study

Table E1: Rolling horizon results for single-objective runs for the scientific 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	-	61,990	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	17,859	0	251	0.01
-	2	60,049	-3.13	25,206	41.1	98	0.00

Table E2: Rolling horizon results for the 1NM method for the scientific 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 E3: 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 E4: 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.