Real-Time Dynamic Hoist Scheduling for Multistage Material Handling Process Under Uncertainties

Chuanyu Zhao, Jie Fu, and Qiang Xu
Dan F. Smith Dept. of Chemical Engineering, Lamar University, Beaumont, TX 77710

DOI 10.1002/aic.13852
Published online June 22, 2012 in Wiley Online Library (wileyonlinelibrary.com).

Multistage material handling processes are broadly used for manufacturing various products/jobs, where hoists are commonly used to transport inline products according to their processing recipes. When multiple types of jobs with different recipes are simultaneously and continuously handled in a production line, the hoist movement scheduling should be thoroughly investigated to ensure the operational feasibility of every job inline and in the meantime to maximize the productivity if possible. The hoist scheduling will be more complicated, if uncertainties of new coming jobs are considered, that is, the arrival time, type, recipe, and number of new jobs are totally unknown and unpredictable before they join the production line. To process the multiple jobs already inline and the newly added jobs, the hoist movements must be swiftly rescheduled and precisely implemented whenever new job(s) come. Because a reschedule has to be obtained online without violating processing time constraints for each job, the solution identification time for rescheduling must be taken into account by the new schedule itself. All these stringent requisites motivate the development of real-time dynamic hoist scheduling (RDHS) targeting online generation of reschedules for productivity maximization under uncertainties. Hitherto, no systematic and rigorous methodologies have been reported for this study. In this article, a novel RDHS methodology has been developed, which takes into account uncertainties of new coming jobs and targets real-time scheduling optimality and applicability. It generally includes a reinitialization algorithm to accomplish the seamless connection between the previous scheduling and rescheduling operations, and a mixed-integer linear programming model to obtain the optimal hoist reschedule. The RDHS methodology addresses all the major scheduling issues of multistage material handling processes, such as multiple recipes, multiple jobs, multicapacity processing units, diverse processing time requirements, and even optimal processing queue for new coming jobs. The efficacy of the developed methodology is demonstrated through various case studies. © 2012 American Institute of Chemical Engineers *AIChE J,* 59: 465–482, 2013
Keywords: hoist scheduling, real-time optimization, MILP, multistage material handling

Introduction

Multistage material handling processes are widely used in chemical, automotive, electronic, steel, and many other industries for manufacturing various products/jobs. Surface coating operations, such as electroplating and polymeric coating, are typical examples, where job handling is managed by hoist(s). A hoist is a controlled robot to lift, release, or move jobs from one processing unit to another along a production line by following a preset movement schedule based on the processing recipe of each job. Because there are multiple jobs simultaneously processed in the same production line, hoist movements must be well scheduled so as to satisfy the multistage processing request from each job; and in the meantime to maximize the productivity of the entire production line if possible. Obviously, hoist scheduling is one of the most relevant factors to improve the productivity. It is reported that as high as 20% reduction in mean job waiting time and 50% improvement in standard deviation of cycle time can be achieved by hoist scheduling.[1]

Historically, hoist schedules were developed based on process experience. The first reported effort for computerized scheduling was made by Phillips and Unger,[2] who investigated the scheduling problem in a simplified electroplating line, which processed a single type of product by one hoist. In their case, the hoist was scheduled to repeat a fixed movement sequence. This type of complete movement sequence is referred to a ''cycle'',[3] and the corresponding type of scheduling is called ''cyclic hoist scheduling'' (CHS), which has been proven to be an non-deterministic polynomial-time-hard problem.[4] Since then, a number of other new methods, especially mathematical programming based methods, have been introduced.[5–8] Applications can also be coupled with process design and operation issues to bring multiperspective benefits, such as freshwater/wastewater minimization and material and energy savings. Xu and Huang[9] pioneered in the environmental conscious hoist scheduling to integrate fresh water minimization with productivity maximization based on a graph-assisted method. Kuntay et al.[10] developed a two-step optimization algorithm to accomplish the same targets. In addition to operational optimization, Liu et al.[11] considered

Correspondence concerning this article should be addressed to Q. Xu at qiang.xu@lamar.edu.

simultaneous productivity and water-reuse efficiency maximization by coupling CHS and water reuse network design. Very recently, Liu et al.[12] simultaneously addressed a triple-objective optimization problem in terms of productivity maximization, energy saving, and wastewater minimization.

In practice, scheduling of an actual hoist-employed production line may be more complicated than what the CHS assumes. A general method of hoist scheduling should be able to handle such situation that new jobs come to the system randomly. This requires hoist scheduling to work in a dynamic way, that is, dynamic hoist scheduling (DHS). It means when a new job enters a production line for processing, a new hoist schedule needs to be developed to replace the current one. Yin and Yih[13] and Yin[14] studied the hoist scheduling problem in such a way that the hoist schedule for the jobs already in the production line does not change; only the hoist schedule for the new job is to be added. This approach can help reduce the complexity of schedule development but cannot guarantee the optimality of the scheduling solution. Lamothe et al.[15,16] introduced a heuristic method by resorting to a classical backtrack algorithm. Nevertheless, that method was proved to be feasible only when all jobs are of the same type. Goujon and Lacomme[17] developed a dispatching-rule-based heuristic method, which can give a priority to the most urgent job for processing. Although it is simple and reasonable in most cases, it may be incapable of handling various practical situations. Riera and Yorke-Smith[18] provided a comprehensive survey on the history and classification of hoist scheduling problems, where a hybrid algorithm combined with constraint logic programming and mixed integer programming was presented. Zhou and Li[19] also suggested a heuristic method with the assistance of combining sequence search and a linear programming model. It shows success in scheduling a four-tank example, which is much simpler than a real production line. Xu and Huang[20] proposed a heuristic-rule-based solving strategy to tackle the hoist scheduling of a complex electroplating problem, but it cannot theoretically guarantee the robustness of the solution identification and its general applicability. To even enhance the operability or productivity of a production line, research on multihoists scheduling have been studied as well.[21–31]

Probably the most critical type of hoist scheduling is the real-time dynamic hoist scheduling (RDHS). It deals with the dynamic scheduling of multiple types of jobs with different recipes that continuously and randomly arrive to a production line, requesting a new schedule to take care of all the jobs' processing; meanwhile, to ensure the online applicability, the new schedule must be quickly identified with optimality and be seamlessly implemented and connected with the current hoist schedule and processing conditions of the production line. Note that there are inherently more stringent requirements for RDHS than DHS: RDHS requires that the solution identification time for rescheduling must be taken into account by the new schedule itself. The major challenges of an RDHS problem are listed below:

(a) *Multiple types of jobs with different job processing recipes*: In most of previous research, a production line only processes one type of job. RDHS in this article addresses different types of jobs with different processing recipes in one production line. As only one hoist is available in the production line, the hoist-service competition among all jobs will be very intensive.

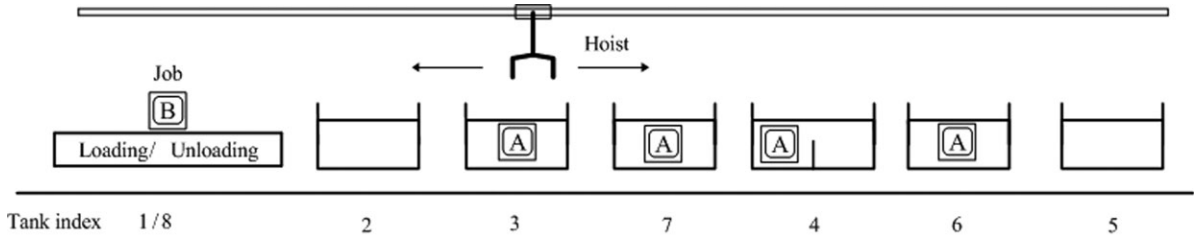
(b) *The random arrival of new jobs*: When new job(s) arrive for processing, the current hoist schedule should be updated. A new schedule has to be generated for serving all the jobs in line, including the new joined jobs. Note that a new schedule should also be seamlessly connected to the current schedule, that is, the initial condition (hoist position and job processing conditions) of the new schedule will be the same as the terminating conditions of the current schedule.

(c) *Units with multijob capacities*: It is common that a unit may simultaneously process multiple jobs at a time to improve the productivity. However, it causes the number of jobs in a production line to increase. This inevitably brings more complexity to an RDHS problem.

(d) *Fast and robust rescheduling with real-time applicability*: This is because slow rescheduling could miss the optimization opportunities and cause a lower production rate, or even product quality problems due to job processing-time violation; meanwhile, the optimal solution for rescheduling must be quickly identified and seamlessly applied under all the process constraints, so that the real-time applicability can be guaranteed.

The challenges listed above make the modeling and solving of RDHS problems extremely difficult. Obviously, the major challenges for RDHS are how to effectively deal with uncertainties, such as unpredictable job arrival time, type, recipe, and number, as well as how to efficiently obtain and adopt an optimal solution for real-time application. Lots of references have addressed uncertainties, especially in the studies of short-time scheduling. In general, the methods dealing with uncertainties can be classified into three categories: (1) Predicting uncertainty behaviors that provide additional information on uncertainty, based on which an optimal scheduling strategy is developed and usually followed by some post modification methods once the uncertainties have been realized. The commonly used approaches include stochastic scheduling approaches,[32–39] chance constrained programming approaches,[40–44] and fuzzy programming methods.[45–47] (2) Parametric programming methods that conduct partition of optimal solutions within uncertainty space based on characterized scheduling model structure prior to uncertainty occurrence. A few studies have been reported in this area.[48–53] (3) Fast responding methods that take the advantage of the quick adaptive capability of a system for post uncertainty processing. Reactive scheduling belongs to this category, which can be further classified into mathematical programming based approaches[54–58] and hybrid approaches that combine heuristic rules with mathematical programming.[20,59–63] The extensive reviews related to the above-mentioned methods for handling uncertainties can be found in publications of Floudas and Lin,[64] Li and Ierapetritou,[65] and Verderame et al.[66]

The way of handling uncertainties from the RDHS study in this article belongs to the third category. A fast response is critically required to generate a reschedule whenever an uncertainty occurs, which means that the solution identification time for rescheduling must be taken into account by the new schedule itself, so that it fully shifts the problem focus from the uncontrollable uncertainty domain to the controllable system domain. This philosophy is actually widely used by industrial manufacturing, business, and people's daily life for handling uncertainties. According to the literature survey made in this article, systematic and rigorous methodologies addressing RDHS are still lack of study. In this article, a



Recipes:

A: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

B: 1 → 2 → 3 → 4 → 5 → 8

C: 1 → 2 → 3 → 6 → 7 → 8

Figure 1. An example of electroplating production line.

novel and rigorous RDHS methodology has been developed, which takes into account uncertainties of new coming jobs and targets real-time scheduling optimality and applicability. It generally includes a reinitialization algorithm to accomplish the seamless connection between the previous scheduling and rescheduling operations, and a mixed-integer linear programming (MILP) model to obtain the optimal hoist reschedule. The RDHS methodology has addressed all the major scheduling issues of a multistage material handling process, such as multiple recipes, multiple jobs, multicapacity processing units, diverse processing time requirements, and even optimal processing queue for new coming jobs. The efficacy of the developed methodology is demonstrated through various case studies.

Problem Statement

Multistage material handling process, for example, electroplating, is performed in a production line that consists of a number of processing units with different processing purposes. As shown in Figure 1, different jobs are picked up from the loading zone initially, then dragged into those processing units by following their processing recipes, and finally the jobs are placed on the unloading zone to complete their processing. To help understand the RDHS problem, the following terminologies are introduced in advance.

Job processing recipe

A job processing recipe gives processing requirements for a job, which includes the processing sequence and resident-time window for each processing step. The recipes for different types of jobs are different. Figure 1 gives the different processing sequences according to the recipes for three types of jobs (A–C), respectively.

Unit job processing capacity

It means the maximum number of jobs that can be simultaneously processed in a unit. In a general production line, each processing unit has a certain job processing capacity. Most units can handle one job at a time, whereas some units have multijob processing capacity. For an example, in Figure 1, Tank 4 is a multijob capacity unit, which consists of two slots for simultaneously processing two jobs if needed.

Free move, loaded move, and idle waiting

When a hoist travels without a job, it is called a free move; otherwise, it is called a loaded move if a hoist travels with a job. The idle waiting means the hoist just staying above some unit without any activity and holding nothing.

Based on the above understandings, an RDHS problem can be summarized as follows.

Given information:

1. One production line with fixed processing units and one hoist;
2. Job capacity of each processing unit;
3. The traveling speed of hoist free move is fixed, which means the hoist traveling time of a free move is totally determined by the spatial distance that the hoist travels;
4. The traveling time of a hoist loaded move is determined by the job type the hoist holds and the picking locations; generally, a loaded move is slower than a free move for the same distance;
5. Initial status of hoist position and inline job processing conditions.

Uncertainties:

1. The arrival time, recipe, and number of new jobs that join the production line. Note that once the new job(s) come, their arrival time, recipe, and number are known immediately.

Information to be determined:

1. The starting time point that the reschedule shall be implemented when a rescheduling is triggered by new coming job(s);
2. Detailed reschedule of hoist free, loaded move, and idle waiting time;
3. Job processing time in each processing unit;
4. The total time span of the reschedule to complete the processing of all the jobs inline.

Assumptions:

1. New job joins the production line via a loading zone and leaves the production line via an unloading zone. The loading and unloading zones share the same deck (see Figure 1).
2. No job capacity limit in the loading/unloading zone;
3. Job processing time limits (or time windows) in each processing unit under every recipe cannot be violated at any time.

General RDHS Methodology

In this article, a novel RDHS methodology has been developed, which takes into account uncertainties of new coming jobs and targets real-time scheduling optimality and applicability. The scheduling activities are implemented in a reactive way. Any occurrence of uncertainties will help initiate a new schedule. This makes an RDHS problem implemented as a series of sequentially connected rescheduling problems. For each rescheduling problem, a MILP model is developed to obtain the optimal hoist schedule to complete processing tasks of all the inline jobs as quick as possible, so that the productivity will be maximized. The global optimal solution can be guaranteed for each rescheduling problem. Meanwhile, to accomplish the seamless connection between the current scheduling and rescheduling operations, a reinitialization algorithm is also introduced. In this section, the general methodology framework is first introduced, and then the RDHS modeling details including the dynamic set designation, rescheduling model equations, and RDHS reinitialization algorithm are presented.

Methodology framework

Figure 2 shows the developed methodology framework of RDHS. First, a general MILP-based rescheduling model needs to be developed offline to minimize the processing completion time of all the jobs in the production line, which is equivalent to maximizing the production rate. With the given initial conditions of the hoist and all the existing jobs in the production line (including the new jobs in the loading zone), the MILP model will be solved to obtain the optimal solution for the current rescheduling problem. Note that during the solution identification, the hoist is actually continuing executing the current scheduling results. From what time point (namely, the switch time point, t^s) the hoist begins to switch its movement from the previous scheduling to the rescheduling results depends on how fast the solution identification can be done for the rescheduling problem.

The rescheduling solution may be available, whereas a hoist is executing a free move or a loaded move. However, the rescheduling can only be started when the hoist holds nothing. This suggests that if the hoist is executing a loaded move when the rescheduling solution is obtained, the hoist has to complete the loaded move in hand first before it has chance to implement the rescheduling results; on the other hand, if the hoist is executing a free move when the rescheduling solution is obtained, the rescheduling can be started at the beginning of the free move, during the free move, or at the end of the free move. Anyway, the switch time point (t^s) that the rescheduling results begin to be implemented has to be projected in advance. Note that when new jobs join the production line, the rescheduling calculation can be immediately conducted at this new job addition time (t^a). The projected t^s should supposedly be behind t^a by enough time to ensure that the time interval of $t^s - t^a$ would be greater than the solution identification time (Δt^c) used for the rescheduling problem (as illustrated in Figure 3), so that the rescheduling results could be implemented in time. Therefore, $t^s - t^a$ should be larger than the upper time limit for solving the MILP rescheduling model. How to project the switch time point and identify the initial conditions of the hoist and all the existing jobs in the production line will be given in details in the later section of RDHS initialization strategy.

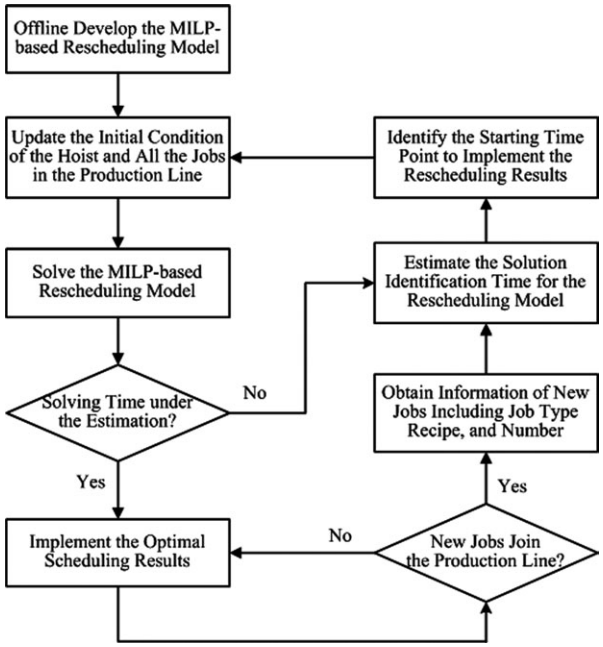


Figure 2. Methodology framework.

As shown in Figure 2, when the rescheduling problem is solved, the actual Δt^c will be examined to see if it is under estimation or not. Two scenarios will be considered.

1. If it is under the estimation (i.e., $t^s - t^a \geq \Delta t^c$), the rescheduling results can be timely implemented (as shown in Figure 3). The hoist will be checked for its availability at the switch time point and start to implement the rescheduling movements after the switch time point; in the meantime, RDHS algorithm will stand by until a new batch of jobs join the production line. Once new jobs arrive, their job types, recipes, and numbers are recognized. Then, the system will estimate the solution identification time for the next round rescheduling to project the new switch time point again and the initial conditions of the production line, including initial hoist position (HI), leftover processing recipes (SI_n, $n = 1,\ldots,N$ with N number of jobs inline) and their elapsed processing time ($IT_{i,n}$, $n = 1,\ldots,N$ and i the unit, in which job n is currently processed) in the current occupied units (BI_n).

2. If the actual solving time is out of the estimation (i.e., $t^s - t^a < \Delta t^c$), the rescheduling results cannot be timely implemented. Then, the rescheduling switch time point has to be projected again to reinitialize a new solution identification; meanwhile, the hoist will continue performing the existing schedule. Iteratively, the rescheduling activities are trigged in a reactive way that any occurrence of uncertainties will initiate a new schedule.

RDHS dynamic sets

To accurately and precisely model the dynamic multistage material handling process, the dynamic sets are used for some particular variables. $recipe_n$ is the processing recipe of job n, which includes the unit processing sequence characterized by a directed array of unit indexes and the processing time specification in each processing unit. Let i represents the unit index; then the processing time specification in unit i for job n is generally bounded by a lower and an upper bound as $RT^{lo}_{i,n}$ and $RT^{up}_{i,n}$, respectively. SI_n is the leftover recipe containing the uncompleted processing units of job n

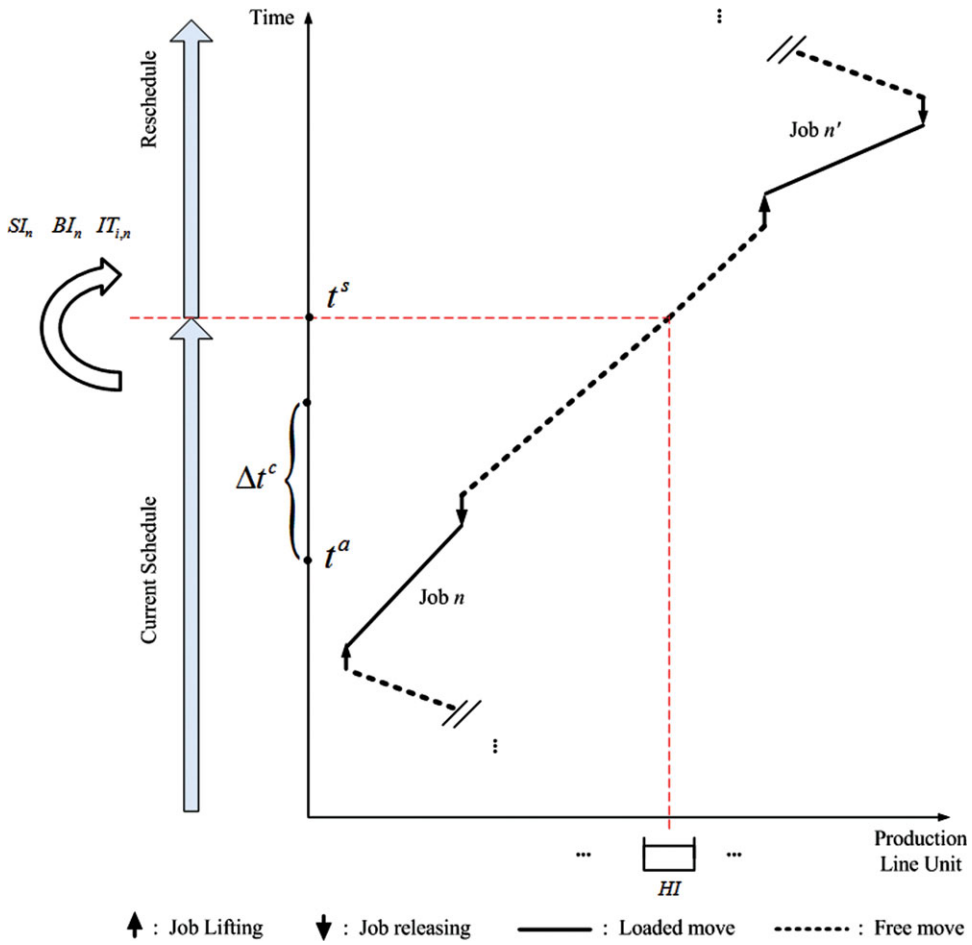


Figure 3. Rescheduling strategy illustration.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

at the beginning of a reschedule (i.e., at the switch time point), which is presented by Eq. 1.

$$\mathrm{SI}_n = \{i_n | \forall i_n, (i_n \in \mathrm{recipe}_n) \wedge (\mathrm{ord}(i_n) \geq \mathrm{ord}(\mathrm{BI}_n))\} \quad (1)$$

where i_n is unit i during the processing of job n; BI_n is the location of job n at the beginning of a reschedule; $\mathrm{ord}(i_n)$ and $\mathrm{ord}(\mathrm{BI}_n)$ are the processing step index of i_n and BI_n, respectively. Thus, $\mathrm{ord}(i_n) \geq \mathrm{ord}(\mathrm{BI}_n)$ suggests those uncompleted processing steps for job n. Equation 2 defines a set $\mathrm{SI}_n^{\mathrm{A}}$ containing all the processing units that the hoist can lift job n from. Because when a job is sent to the unloading deck, its production is already completed, the job should not be picked up by the hoist from the unloading zone (EI). Thus, EI should be excluded from $\mathrm{SI}_n^{\mathrm{A}}$.

$$\mathrm{SI}_n^{\mathrm{A}} = \{i_n | (\forall i_n \in \mathrm{SI}_n) \wedge (i_n \neq \mathrm{EI})\} \quad (2)$$

To clarify the concepts of recipe_n, SI_n, and $\mathrm{SI}_n^{\mathrm{A}}$, suppose a production line consists of eight units including the loading/unloading zone, which are indexed as 1–8. Loading zone and unloading zone are units 1 and 8, respectively. Also suppose a job B is inline and its job index n is 3; its recipe suggests its processing sequence as {1, 2, 3, 4, 5, 6, 7, 8}. If at the beginning of the reschedule this job B is in unit 4, then SI_3 is {4, 5, 6, 7, 8}; and $\mathrm{SI}_3^{\mathrm{A}}$ is {4, 5, 6, 7}. Note that the dynamic sets will change under different initial conditions of each rescheduling problem.

RDHS rescheduling model

Based on the defined dynamic sets, the general MILP rescheduling model is introduced in this section. For illustration, some logic equations and auxiliary figures are used to explain the rescheduling model. All the variables and parameters are explained both in the context and in the notation.

Objective Function. The goal in the rescheduling model is to maximize the job production rate, which is equivalent to minimizing the time span (T) used for processing all the jobs inline as shown in Eq. 3.

$$\mathrm{Obj} = \min T \quad (3)$$

$$T \geq \mathrm{TE}_{\mathrm{EI},n}, \quad \forall n \in \mathrm{SN} \quad (4)$$

where EI is the index of the unloading zone; SN is the set of all the jobs in the production line; $\mathrm{TE}_{\mathrm{EI},n}$ is the ending time that the hoist drops job n into the unloading zone (end of the last loaded move), which is also the production due time of job n. Equation 4 indicates that T should be larger than the production due time of every job in the production line.

Hoist movement Constraints. Hoist movement constraints are used to define and characterize two time variables of $\mathrm{TS}_{i,n}$ and $\mathrm{TE}_{i,n}$, where $\mathrm{TS}_{i,n}$ and $\mathrm{TE}_{i,n}$ are, respectively, the lift time and release time for the hoist to pick up job n from unit i or to drop job n in unit i. As aforementioned, hoist movements include free and loaded moves. Equation 5 describes the relationship of

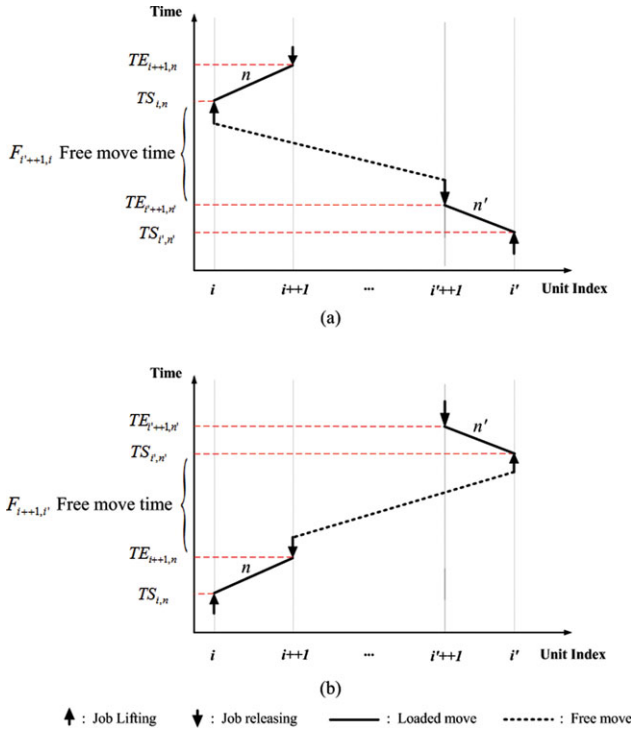


Figure 4. Illustrative examples for binary variable of $w_{i,n,i',n'}$: (a) $w_{i,n,i',n'} = 1$ (the loaded move of job n' from unit i' is ahead of that of job n from unit i); (b) $w_{i,n,i',n'} = 0$ (the loaded move of job n' from unit i' is behind that of job n from unit i).

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

a hoist loaded move, where $L_{i,n}$ is the traveling time of a hoist loaded move for carrying job n from unit i to the next unit.

$$\mathrm{TE}_{i+1,n} = \mathrm{TS}_{i,n} + L_{i,n}, \qquad \forall i \in \mathrm{SI}_n^{\mathrm{A}}; \quad \forall n \in \mathrm{SN} \tag{5}$$

where $\mathrm{TE}_{i+1,n}$ is the release time of carrying job n to the next processing unit according to the processing recipe.

Equation 6 suggests that $\mathrm{TS}_{i,n}$ should be at least larger than a time limit ($F_{\mathrm{HI},i}$) even if the hoist moves directly from the initial position to unit i to pickup job n.

$$\mathrm{TS}_{i,n} \geq F_{\mathrm{HI},i}, \qquad \forall i \in \mathrm{SI}_n^{\mathrm{A}}; \quad \forall n \in \mathrm{SN} \tag{6}$$

where HI is the initial hoist position at the beginning of the rescheduling. Equations 7 and 8 ensure that loaded moves cannot timely conflict, that is, a hoist cannot carry two jobs at a time. Here, a binary variable $w_{i,n,i',n'}$ is used. If the loaded move that picks up job n' from unit i' occurs ahead of the loaded move that picks up job n from unit i, $w_{i,n,i',n'}$ is 1; otherwise, the loaded move that picks up job n' from unit i' occurs after the loaded move that picks up job n from unit i, $w_{i,n,i',n'}$ is 0 (as illustrated in Figure 4).

$$M\left(w_{i,n,i',n'} - 1\right) \leq \mathrm{TE}_{i+1,n} - \mathrm{TS}_{i',n'} \leq M w_{i,n,i',n'},$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}}; \forall i' \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; i \neq i' \text{ or } n \neq n' \tag{7}$$

$$-w_{i,n,i',n'} M \leq \mathrm{TE}_{i'+1,n'} - \mathrm{TS}_{i,n} \leq M\left(1 - w_{i,n,i',n'}\right),$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}}; \forall i' \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; i \neq i' \text{ or } n \neq n' \tag{8}$$

Based on $w_{i,n,i',n'}$, $\mathrm{TS}_{i,n}$ can be restricted by Eq. 9, indicating that if $\mathrm{TS}_{i,n} > \mathrm{TS}_{i',n'}$, then $\mathrm{TS}_{i,n} \geq \mathrm{TE}_{i'+1,n'} + F_{i'+1,i}$; or vice versa (see Figure 4a, b), where $F_{i'+1,i}$ is the traveling time of a hoist free move from unit $i'+1$ to unit i. Note that after a hoist completes a free move, it may experience an idle waiting right above the unit where it will pick up a job, if that job is not ready to leave yet (due to job processing time restriction). The nonlinear constraint of Eq. 9 is linearly presented by Eqs. 10–12, where another positive variable $\mathrm{Sl}_{i,n,i',n'}$ is introduced to substitute $(\mathrm{TE}_{i'+1,n'} + F_{i'+1,i}) w_{i,n,i',n'}$. L and M are the lower and upper bounds of $Sl_{i,n,i',n'}$, respectively.

$$\mathrm{TS}_{i,n} \geq \left(\mathrm{TE}_{i'+1,n'} + F_{i'+1,i}\right) w_{i,n,i',n'},$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}}; \forall i' \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; i \neq i' \text{ or } n \neq n' \tag{9}$$

$$L\left(1 - w_{i,n,i',n'}\right) \leq \left(\mathrm{TE}_{i'+1,n'} + F_{i'+1,i}\right) - \mathrm{Sl}_{i,n,i',n'}$$
$$\leq M\left(1 - w_{i,n,i',n'}\right), \forall i \in \mathrm{SI}_n^{\mathrm{A}}; \forall i' \in \mathrm{SI}_{n'}^{\mathrm{A}};$$
$$\forall n, n' \in \mathrm{SN}; i \neq i' \text{ or } n \neq n' \tag{10}$$

$$L w_{i,n,i',n'} \leq \mathrm{Sl}_{i,n,i',n'} \leq M w_{i,n,i',n'},$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}}; \forall i' \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; i \neq i' \text{ or } n \neq n' \tag{11}$$

$$\mathrm{TS}_{i,n} \geq \mathrm{Sl}_{i,n,i',n'},$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}}; \forall i' \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; i \neq i' \text{ or } n \neq n' \tag{12}$$

Unit Processing Capacity Constraints. The units involved in the production line may have diverse job processing capacities. Some of them can only process one job at a time, so that to release a new job into such a single-capacity unit, the job already staying in the unit must be moved out first. For the multicapacity units, the maximum number of jobs processed in the unit at any time has a limit. Equations 13 and 14 provide the way to count the number of jobs staying in a unit. A binary variable $x_{i,n,n'}$ is defined in the logic constraint of Eq. 13. As illustrated in Figure 5, $x_{i,n,n'}$ is 1 if job n' once a time stays with job n in unit i; otherwise, it is 0. In Eq. 14, Ca_i is job processing capacity of unit i. $\sum_{n' \in \mathrm{SN}} x_{i,n,n'}$ $(n \neq n')$ accounts for the total number of jobs once staying with job n in unit i at a time. Thus, the job capacity of unit i is at least $\sum_{n' \in \mathrm{SN}} x_{i,n,n'} + 1$.

$$\mathrm{TE}_{i,n} \leq \mathrm{TE}_{i,n'} \leq \mathrm{TS}_{i,n} \Leftrightarrow x_{i,n,n'} = 1,$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}} \text{ and } \forall i \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; \quad n \neq n' \tag{13}$$

$$\sum_{n' \in \mathrm{SN}} x_{i,n,n'} + 1 \leq \mathrm{Ca}_i,$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}} \text{ and } \forall i \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; \quad n \neq n' \tag{14}$$

To represent the logic constraint of Eq. 13 with linear algebraic constraints, Eqs. 15–17 are used.

$$-u_{i,n,n'} M < \mathrm{TE}_{i,n'} - \mathrm{TS}_{i,n} \leq M\left(1 - u_{i,n,n'}\right),$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}} \text{ and } \forall i \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; \quad n \neq n' \tag{15}$$

$$-v_{i,n,n'} M < \mathrm{TE}_{i,n} - \mathrm{TE}_{i,n'} \leq M\left(1 - v_{i,n,n'}\right),$$
$$\forall i \in \mathrm{SI}_n^{\mathrm{A}} \text{ and } \forall i \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; \quad n \neq n' \tag{16}$$

$$v_{i,n,n'} + u_{i,n,n'} - x_{i,n,n'} \leq 1, \quad \forall i \in \mathrm{SI}_n^{\mathrm{A}} \text{ and}$$
$$\forall i \in \mathrm{SI}_{n'}^{\mathrm{A}}; \forall n, n' \in \mathrm{SN}; \quad n \neq n' \tag{17}$$

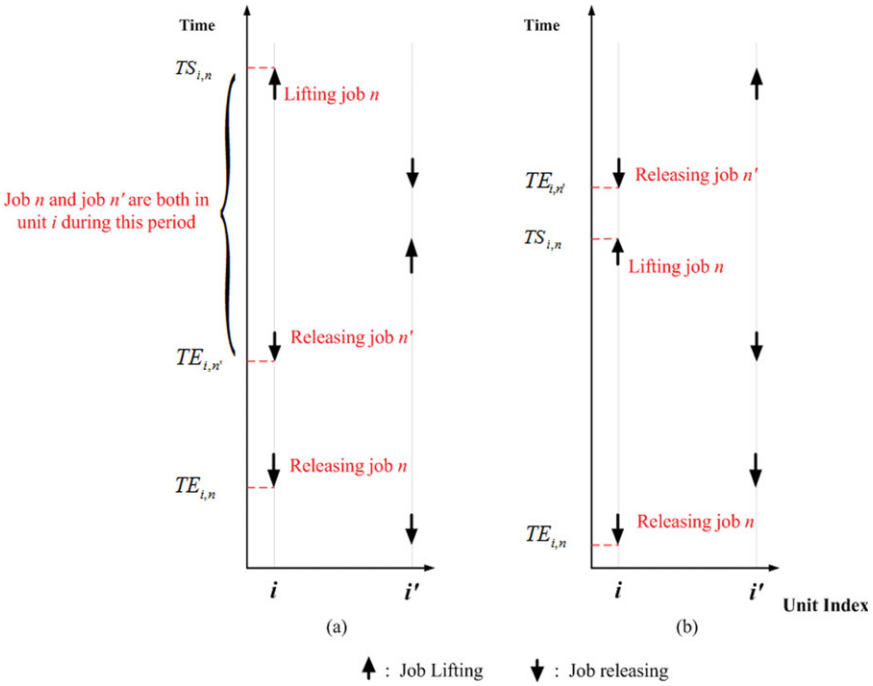


Figure 5. Illustrative examples for binary variables of $x_{i,n,n'}$: (a) $x_{i,n,n'} = 1$ (the job n' is staying with job n in the unit i at one time); (b) $x_{i,n,n'} = 0$ (the job n' is never staying with job n in the unit i at any time).

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

where another two binary variables are introduced to represent the relations between $TS_{i,n}$, $TE_{i,n'}$, and $TE_{i,n}$. As shown in Eqs. 15 and 16, $u_{i,n,n'}$ is 1 if $TE_{i,n'} \leq TS_{i,n}$; otherwise, it is 0. $v_{i,n,n'}$ is 1 if $TE_{i,n} \leq TE_{i,n'}$; otherwise, it is 0. Based on the linearization, all unit processing capacity constraints become linear.

Processing Time Constraints. Generally, a job staying in a processing unit should respect a processing time window specified by its recipe. Equation 18 accounts for the processing time of job n staying in unit i, where $IT_{i,n}$ is the elapsed processing time duration of job n staying in unit i when the reschedule is started. As shown in Eq. 19, $RT^{lo}_{i,n}$ and $RT^{up}_{i,n}$ are the specified lower and upper bounds of the processing time window, respectively.

$$p_{i,n} = TS_{i,n} - TE_{i,n} + IT_{i,n}, \quad \forall i \in SI^{A}_{n};\ \forall n \in SN \tag{18}$$

$$RT^{lo}_{i,n} \leq p_{i,n} \leq RT^{up}_{i,n}, \quad \forall i \in SI^{A}_{n};\ \forall n \in SN \tag{19}$$

Variable Bounds. All the continuous variables have a lower bound of zero and an upper bound of M. As described, u, v, w, and x are defined as binary variables.

$$g_n, p_{i,n}, SI_{i,n,i',n'}, T, TS_{i,n}, TE_{i,n} \geq 0, \quad \forall i, i' \in SI; \forall n, n' \in SN \tag{20}$$

$$g_n, p_{i,n}, SI_{i,n,i',n'}, T, TS_{i,n}, TE_{i,n} \leq M, \quad \forall i, i' \in SI; \forall n, n' \in SN \tag{21}$$

$$u_{i,n,n'}, v_{i,n,n'}, w_{i,n,i',n'}, x_{i,n,n'} \in \{0, 1\}, \quad \forall i, i' \in SI; \forall n, n' \in SN \tag{22}$$

In summary, the developed RDHS rescheduling model consists of the objective function of Eq. 3, and constraints of Eqs. 4–8, 10–12, and 14–22. It is an MILP model and can be efficiently solved with commercial solvers.

RDHS Initialization Strategy

To solve the MILP rescheduling model, its initial conditions need to be determined through a systematic and quantitative way. As aforementioned, the RDHS system needs to trigger a rescheduling activity corresponding to any addition of new jobs. Before the rescheduling result is implemented, the hoist will execute the current schedule. Moreover, the hoist has to first complete the ongoing loaded move if it is holding a job at that moment before it has chance to implement the rescheduling results. Thus, the switch time point (t^s) for rescheduling can be setup at the beginning of a free move, during a free move, or at the end of a free move, as long as the hoist holds nothing at that time. Equation 23 gives the method to identify the switch time point (t^s) for implementing the rescheduling.

$$t^s = \left\{ t_{i,k} \middle| \min_{i,k} \left(t_{i,k} - t^a - \Delta t^c \right) \geq 0 \right\} \tag{23}$$

where t^a represents the time point at which new job(s) just join the production line; Δt^c represents the estimated time consumption for solving the rescheduling model; and $t_{i,k}$ represents the time point at which the hoist moves to unit i for the kth time according to the current schedule. However, the situation that the hoist passing by the unit i during a loaded move is excluded, for the reason that at this time point the hoist is occupied, which is not feasible to begin a reschedule. Equation 23 shows that the switch time point should be larger than t^a with the minimum time difference, and such a time difference should also be larger than Δt^c, so that the rescheduling can take place as soon as possible under any new addition of job(s); and in the meantime, the rescheduling results can be implemented in time. Note that although Eq. 23 suggests an optimization problem, the solution identification for t^s is very fast, because t^s can be obtained through hoist movement search after $t^a + \Delta t^c$ according to the current schedule (Figure 6).

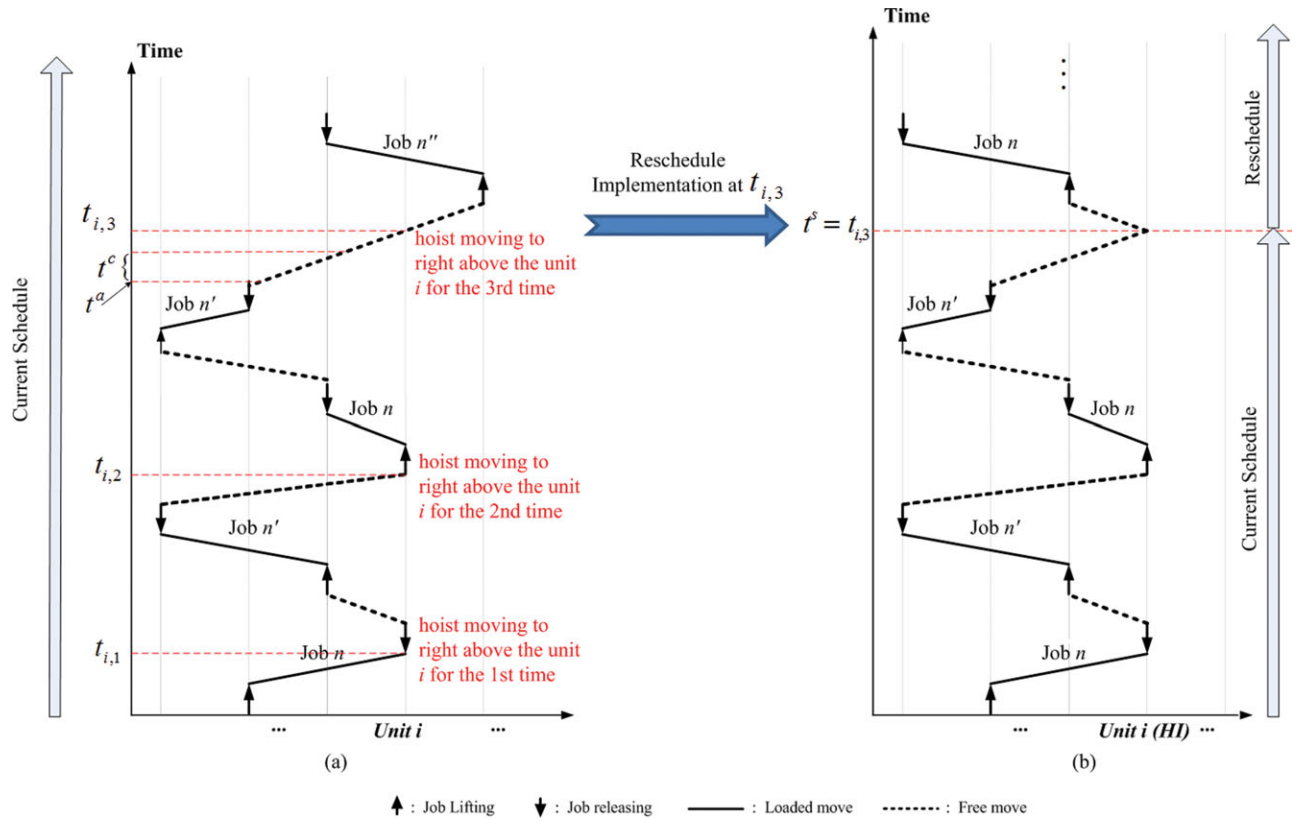


Figure 6. A schematic plot for t^s identification: (a) An indexed hoist moving route for current scheduling; (b) Initial hoist position and starting time point for rescheduling.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

The initial hoist position (HI) of the rescheduling problem can be obtained through a method represented by Eq. 24. At the time of t^s, unit i, where the hoist is hanging above, is indicated as HI. Note that the initial hoist position of a reschedule is characterized by the unit index.

$$\mathrm{HI} = \left\{ i \middle| t_{i,k} = t^s \right\} \tag{24}$$

Besides the hoist initial condition, the processing condition of all the jobs in the production line should also be identified. Equation 25 accounts for the elapsed time duration of job n staying in unit i ($\mathrm{IT}_{i,n}$) when a reschedule is just started. In the equation, the superscript p means the current schedule, so that $\mathrm{TE}^p_{i,n}$, $\mathrm{TS}^p_{i,n}$, $\mathrm{SI}^{p,\mathrm{A}}_n$, and SN^p have similar definitions as those used in the rescheduling model but are, respectively, distinguished. Note that $\mathrm{IT}_{i,n}$ will be used by Eq. 18. To ensure the generality of Eq. 18, if $\mathrm{IT}_{i,n}$ at t^s does not influence the accounting of $p_{i,n}$ during the rescheduling, $\mathrm{IT}_{i,n}$ is given by 0.

$$\mathrm{IT}_{i,n} = \begin{cases} t^s - \mathrm{TE}^p_{i,n}, & \text{if } \mathrm{TS}^p_{i,n} \geq t^s \geq \mathrm{TE}^p_{i,n} \\ 0, & \text{otherwise} \end{cases}, \quad \forall i \in \mathrm{SI}^{p,\mathrm{A}}_n;\ \forall n \in \mathrm{SN}^p \tag{25}$$

Similarly, the units whose processing times are influenced by $\mathrm{IT}_{i,n}$ are the occupied units at the beginning of the rescheduling; meanwhile, for those new coming jobs, their beginning units are loading zone (BI). As shown in Eq. 26, the initial job positions used for rescheduling are provided.

$$\mathrm{BI}_n = \begin{cases} i, & \text{if } \mathrm{TS}^p_{i,n} \geq t^s \geq \mathrm{TE}^p_{i,n} \\ \mathrm{BI}, & \text{otherwise} \end{cases}, \quad \forall i \in \mathrm{SI}^{p,\mathrm{A}}_n;\ \forall n \in \mathrm{SN}^p \tag{26}$$

Finally, when all the initial hoist and job conditions are identified, the rescheduling will be calculated and implemented based on the new time origin, which means the rescheduling switch time point will be setup as 0 (Eq. 27) in the reschedule. Correspondingly, as the elapsed time duration of job n staying in unit i ($\mathrm{IT}_{i,n}$) has been accounted, its release time point $\mathrm{TE}_{\mathrm{BI}_n,n}$ in the reschedule should be set as 0 (Eq. 28).

$$t^s = 0 \tag{27}$$

$$\mathrm{TE}_{\mathrm{BI}_n,n} = 0 \tag{28}$$

Case Studies

The developed RDHS methodology has been examined through a hoist scheduling problem for a multistage material handling process. In this production line (see Figure 1), eight units with various processing purposes (including the loading/unloading zone) are labeled as 1–8; loading zone and unloading zone are indexed as units 1 and 8, respectively. Different types of jobs, for example, A–C with different processing recipes are processed in this production line. The job processing capacity and the allowed processing time window of each unit, as well as the initial conditions of the production line are given in Table 1. The first row means the unit index in the production line; whereas the first column

Table 1. General Information and Initial Condition for the Production Line

Unit	1/8	2	3	7	4	6	5
Ca_i	∞	1	1	1	2	1	1
$RT^{lo}_{i,n}$ (s)	–	60	30	30	75	55	30
$RT^{up}_{i,n}$ (s)	–	300	300	300	300	300	300
Initial jobs	B1	None	A1	A2	A3	A4	None
$IT_{i,n}$ (s)	2	–	4	33	68	22	–

means the job processing capacity, the processing-time lower bound of each unit, the processing-time higher bound of each unit, the initial job in each unit, and the elapsed processing-time duration at the beginning of a schedule. Unit 4 has job processing capacity of two; units 1 and 8 are the loading/unloading zone without job capacity restriction; and the other processing units have single-job processing capacity.

At the beginning of the current scheduling problem, there are four jobs under processing in different units; meanwhile, one job is just added onto the loading zone (see Figure 1). Hoist traveling times for free and loaded moves are given parameters as shown in Tables 2 and 3. For free moves shown in Table 2, the first row and column represent unit indexes of the production line, and it takes 2 s for the hoist traveling between two adjacent processing units. Sometimes, after the hoist drops a job in a certain unit, it will lift another job from the same unit; under this situation, we consider that the hoist free move consumes zero second. For Table 3, the first row is different job type in the system, and the first column is the unit index of the production line. It gives the loaded move time when the hoist is holding a certain job and departing from a certain unit. Note that the developed RDHS methodology is capable of scheduling many types of jobs as long as their recipes are given at the time the jobs join the production line.

All the cases are modeled and solved in GAMS[67] version 23.3 with the solver CPLEX,[68] and the solving time with an 8-Core Xeon 3.2GHz Dell server for all the case studies is within 1 s, which is acceptable for rapid switch to reschedules due to uncertainties. Certainly, more jobs in the production line may increase the solution identification time. Through our pretesting, to reschedule at least three new jobs simultaneously added into the production line, 2 s can be safely adopted as the computational time (i.e., $\Delta t^c = 2$). As shown in the methodology framework, the MILP model will be reactively resolved whenever any uncertainties occur, for example, new jobs come randomly with different recipes.

Case 1: current hoist schedule

In this schedule, five jobs will be processed in the production line of eight units. The initial jobs in the production line include 4 job A in units 3, 4, 6, and 7; and one job B in unit 1. The developed MILP model for the current schedule has 368 binary variables, 697 continuous variables, and 2696 constraints, and the solving time for this case is 0.22 s. Figure 7 shows the global optimal scheduling results for the current schedule based on the given initial condition. The total scheduling time span is 283 s including 175 s of loaded move and 108 s of free move and idle waiting. During the schedule, the hoist starts from lifting job B1 from loading zone; after releases it into unit 2, the hoist free moves to unit 7 to pick up job A2, then the hoist performs a series of loaded moves and free moves according to the schedule, and finally completes all the job processing tasks and returns to the loading/unloading zone. Every starting and ending time point of the loaded moves, as well as the job index carried by the hoist for each loaded move are marked in the figure. For those single-capacity units, the job already in the unit has to be moved out before another job is released; for multicapacity unit as unit 4, a job A3 is under processing at the beginning of the schedule, with another job A1 released at the time 34 s, unit 4 does handle two jobs at that time. Also, after a free move, the hoist may be idle waiting above a certain unit to rest according to the schedule. For example in Figure 7, the hoist free moves from unit 7 at 57 s to unit 2 to lift job B1. The free move time is 4 s; whereas, the hoist lifts job B1 at 64 s, which suggests that there is an idle waiting of 3 s before the hoist lifts job B1.

Table 2. Given Parameters of $F_{i',i}$ (s)

	Unit i							
Unit i'	1	2	3	4	5	6	7	8
1	0	2	4	8	12	10	6	0
2	2	0	2	6	10	8	4	2
3	4	2	0	4	8	6	2	4
4	8	6	4	0	4	2	2	8
5	12	10	8	4	0	2	6	12
6	10	8	6	2	2	0	4	10
7	6	4	2	2	6	4	0	6
8	0	2	4	8	12	10	6	0

Table 3. Given Parameters of $L_{i,n}$ (s)

	Job Type n		
Unit i	A	B	C
1/8	4	4	4
2	10	10	10
3	8	8	14
4	11	11	0
5	5	20	0
6	10	0	10
7	13	0	13

To demonstrate the efficacy of the developed RDHS methodology, another three rescheduling case studies are conducted from different aspects of uncertain job arrival time, recipe, and job numbers.

Case 2: rescheduling when a new job is added during a hoist free move

Suppose that at the time of 145 s, a job C (indexed as C1) comes to the loading zone to be processed. To adopt the processing of this new job C as quick as possible, the hoist movement is going to be rescheduled. At that time, the hoist is experiencing a free move based on the initialization of the rescheduling, t^a is 145 s, Δt^c is 2 s; according to the hoist current movement schedule by considering the computational time, the closest position that the hoist can restart a new reschedule is unit 7, which will be set as the hoist initial position in the reschedule. The time point when the hoist travels right above unit 7 will be set as the switch time for rescheduling (i.e., $t^s = 148$ s from the current schedule, but t^s is set as 0 in the reschedule). The initial condition for this rescheduling case is obtained and listed in Table 4. Other rescheduling initial conditions based on the same current schedule are also listed in Table 4. In Table 4, the first row is the unit index of the production line; the first column is the case index, whereas the second column is the initial job(s) and elapsed processing-time duration at the beginning of a reschedule. In this schedule, four jobs will be processed

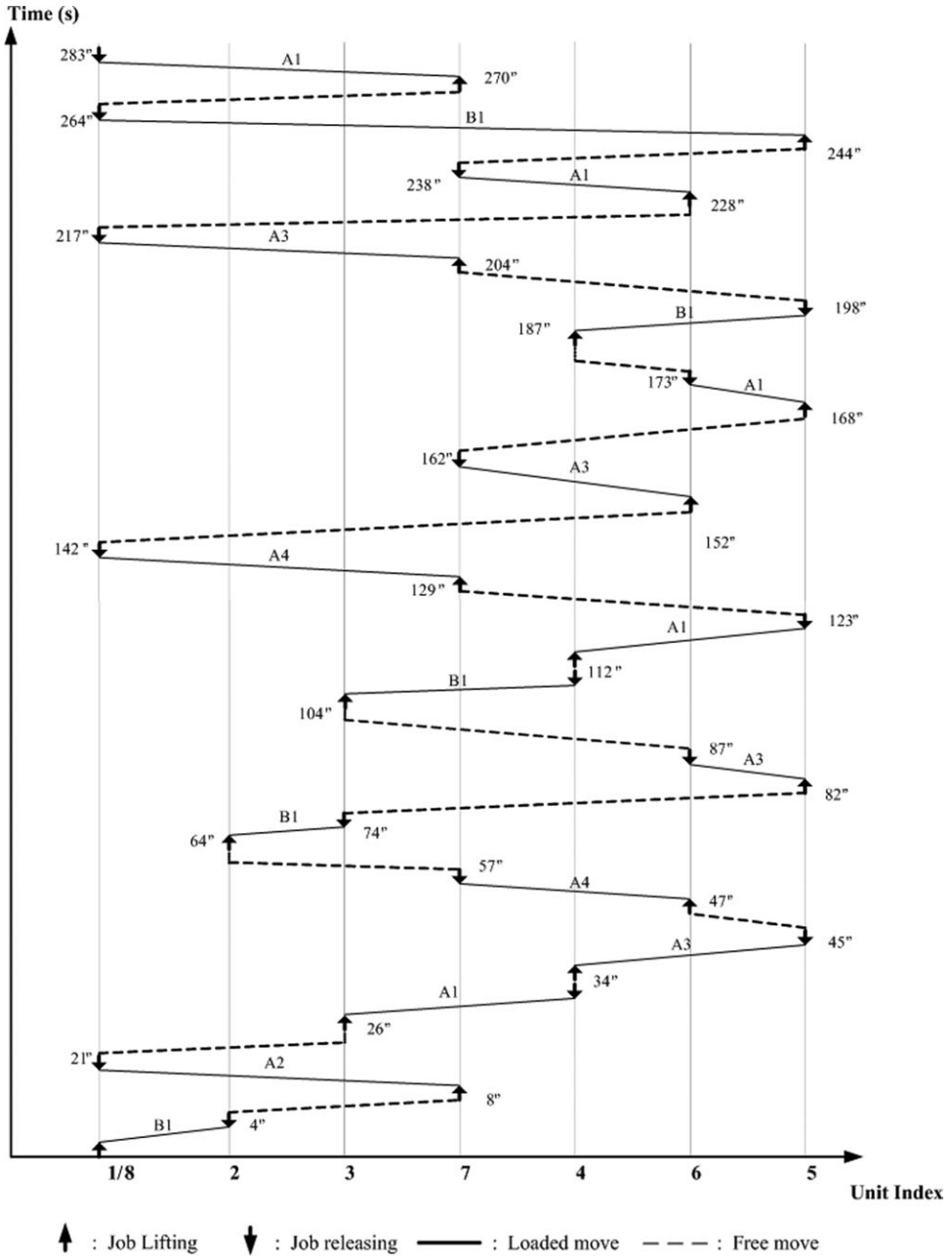


Figure 7. Optimal scheduling results for Case 1.

in the production line of eight units. The initial jobs in the production line include 2 job A in units 5 and 6; one job B in unit 4; and one job C in unit 1. The MILP model for the rescheduling has 174 binary variables, 347 continuous variables, and 1318 constraints, and the solving time for Case 2 is 0.24 s. The rescheduling time span for the global optimal solution is 235 s, including 133 s of loaded move and 102 s of free move and idle waiting. Figure 8 shows the global optimal rescheduling results and how the system is switched from the current schedule to the newly obtained reschedule. In the reschedule, the hoist starts a free move from unit 7 to the loading zone; and then picks up the new coming job C1 at 6 s, and then it is released into its next processing unit 2 at 10 s. By following the reschedule shown in Figure 8, all the jobs in the production line will be processed.

Case 3: rescheduling when a new job is added during a hoist loaded move

If the job C1 comes earlier to the system at time 117 s, then t^a is 117 s and Δt^c is 2 s, the reschedule should be

Table 4. Initial Conditions for the Rescheduling Cases

		Unit						
		1/8	2	3	7	4	6	5
Case 2	Initial jobs	C1	None	None	None	B1	A3	A1
	$IT_{i,n}$ (s)	3	–	–	–	36	61	25
Case 3	Initial jobs	C1	None	None	A4	B1	A3	A1
	$IT_{i,n}$ (s)	6	–	–	66	11	36	0
Case 4	Initial jobs	A5, B2, C1	None	None	None	B1	A3	A1
	$IT_{i,n}$ (s.)	3	–	–	–	36	61	25

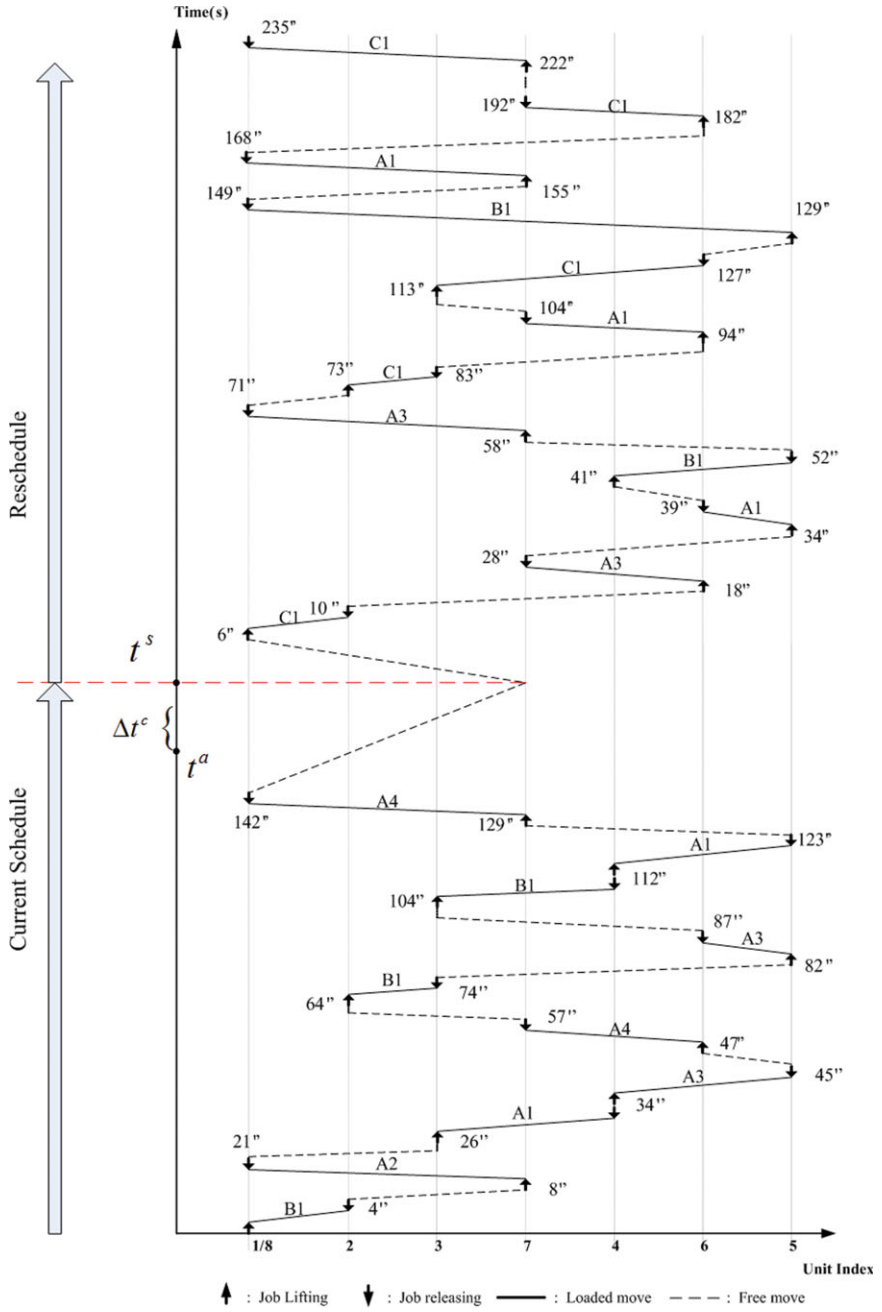


Figure 8. Optimal rescheduling results for Case 2.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

applied after 119 s. However, at that time the hoist is carrying job A1, a reschedule cannot be started with a loaded hoist. Thus, unit 5 where job A1 is released will be set as the hoist initial position for the rescheduling. The time point when the hoist is right above the unit 5 will be set as the switch time for rescheduling (i.e., $t^S = 123$ second from the current schedule, but t^S is set as 0 in the reschedule). The initial condition for this rescheduling case is obtained and also listed in Table 4. In this schedule, five jobs will be processed in the production line of eight units. The initial jobs in the production line include 3 job A in units 5–7; one job B in unit 4; and one job C in unit 1. The MILP model for this case has 216 binary variables, 417 continuous variables, and 1570 constraints, and the solving time for Case 3 is 0.24 s. The time span for the global optimal solution is 248 s including 146 s of loaded move and 102 s of free move and idle waiting. Figure 9 shows the global optimal rescheduling result and how the system is switched from the current schedule to the reschedule. In this reschedule, the hoist first spends a free move and a loaded move before starts to lift the new job C1 from the loading zone at time 19 s.

Case 4: rescheduling when three different jobs simultaneously arrive

Suppose at the time 145 s, three jobs indexed as A5, B2, and C1 simultaneously join the loading zone (instead of one new job in Cases 2 and 3). As shown in Case 2, the switch time for rescheduling will be set at 148 s, and unit 7 will be set as the hoist initial position. The initial condition for this rescheduling case is listed in Table 4. In this schedule, six

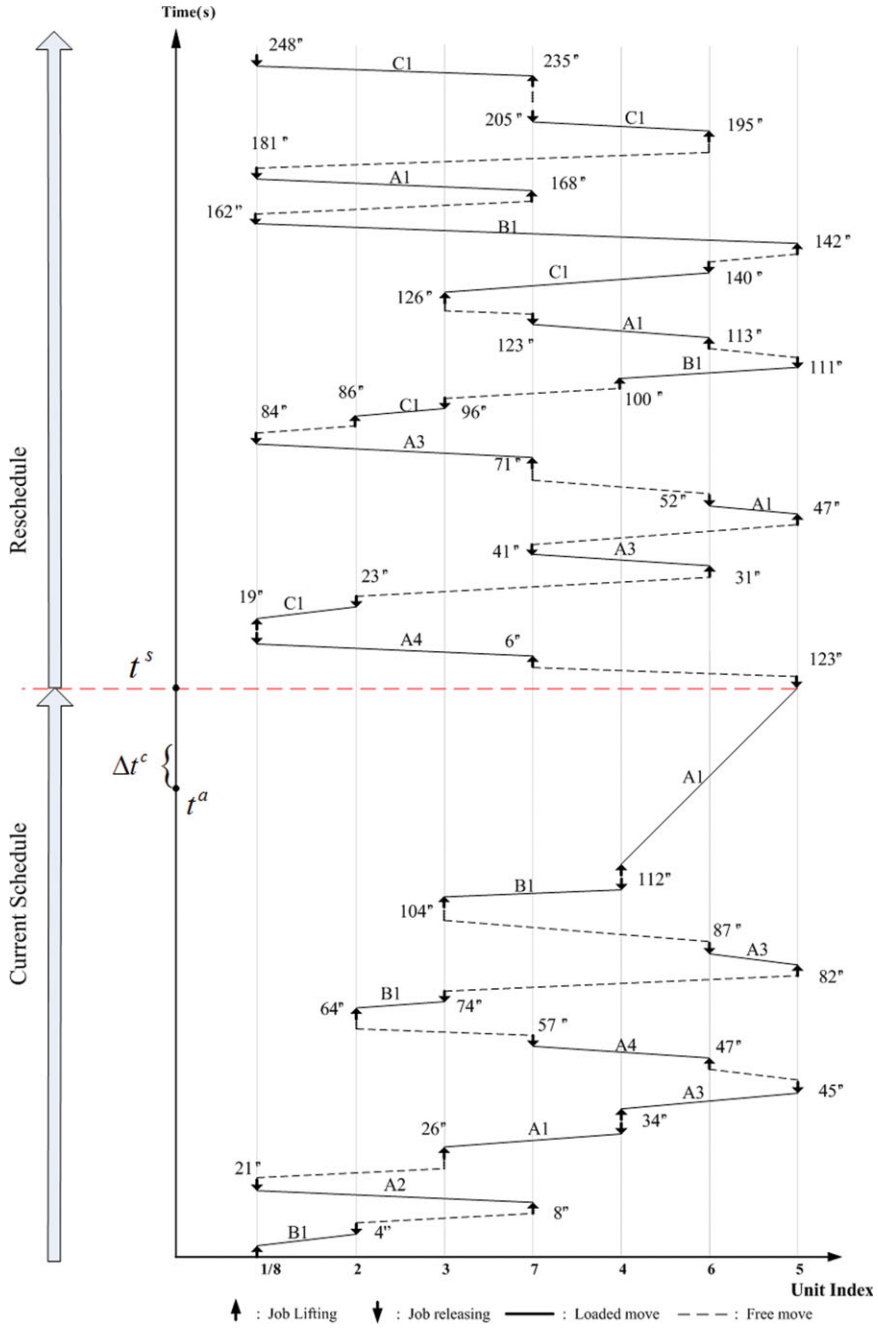


Figure 9. Optimal rescheduling results for Case 3.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

jobs will be processed in the production line of eight units: 3 job A in units 1, 5, and 6; 2 job B in units 1 and 4; and one job C in unit 1. The MILP model for this case has 732 binary variables, 1363 continuous variables and 5394 constraints, and the solving time for Case 4 is 0.73 s. The rescheduling time span for the global optimal solution is 426 s, including 247 s of loaded move and 179 s of free move and idle waiting. Figure 10 shows the global optimal rescheduling results and how the system is switched from the current schedule to the reschedule. In the reschedule, the new job A5 is picked up from the loading zone at 6 s, the new job C1 is picked up at 84 s, and the new job B2 is picked up at 162 second. Therefore, the processing sequence of the new joined jobs is A → C → B, which suggests that the optimal processing queue can actually be obtained through the developed RDHS methodology.

To fully demonstrate the optimal queue capability of the developed RDHS methodology, more scenarios based on Case 4 are investigated. As there are three jobs A–C simultaneously joining the production line, the full combinations of their processing sequence generate six scenarios, including the obtained optimal one (A → C → B). Thus, we generate the other five queue scenarios and enforce the rescheduling to handle the new job processing according to these queue scenarios (see Table 5). Note that for fixed queue scenarios, RDHS is still conducting optimal rescheduling, but must follow the designated job processing queue. All the queue scenarios are solved within 1 s. Their rescheduling time span is

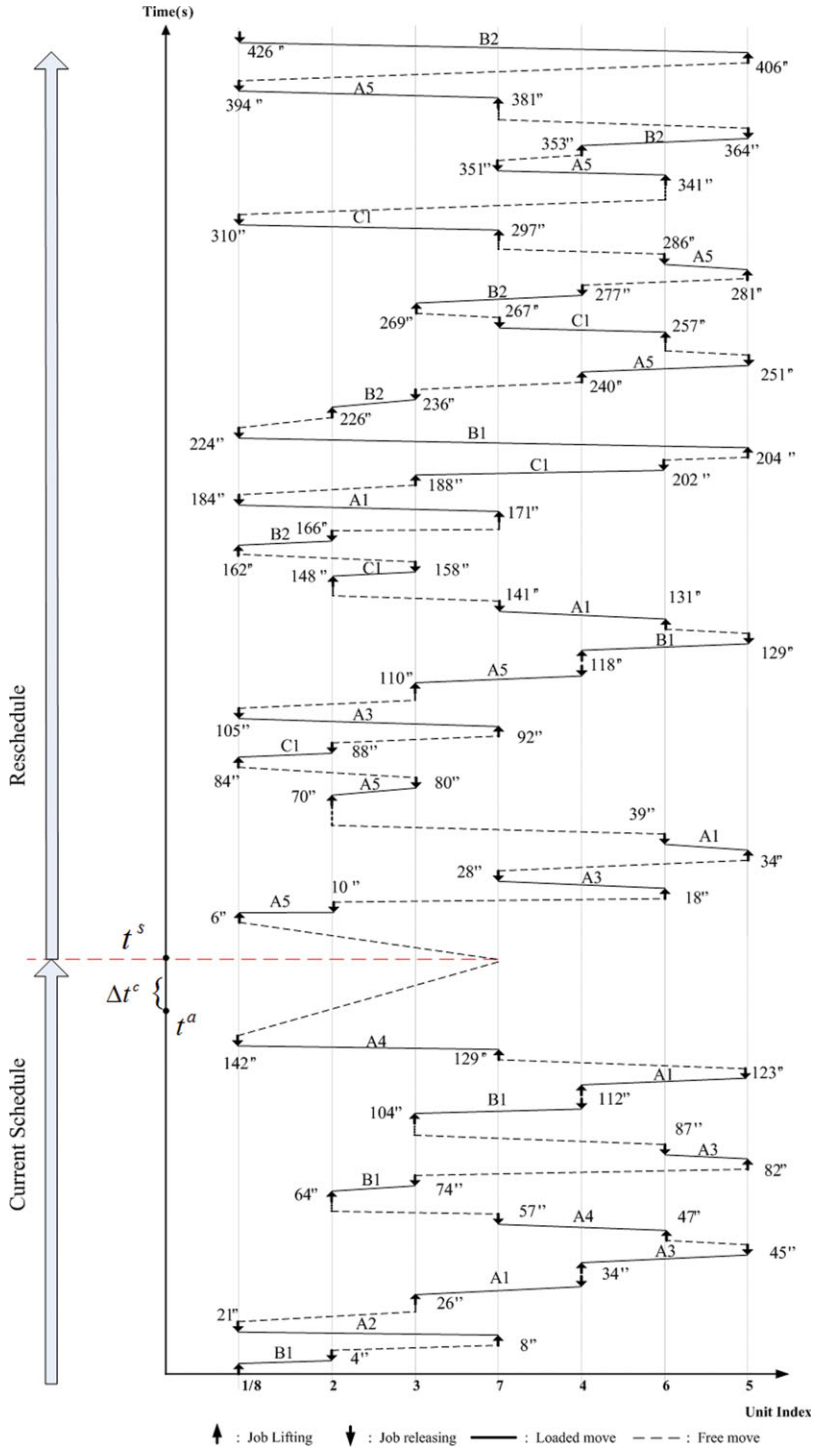


Figure 10. Optimal rescheduling results for Case 4.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

shown in Table 5. It shows that the job processing queue does influence the rescheduling time span. The difference between best one (426 s under the queue of A → C → B) compared with the worst one (504 s under the queue of C → B → A) is 78 s, which is significant. The results on the one hand demonstrate the exceptional merit of the developed RDHS methodology for its optimal queue capability. On the other hand, it also suggests the new processing queue should be considered if possible, because there are significant benefits implicated. Certainly, when more jobs join the

Table 5. Optimal Rescheduling Results under Different Job Queues for Case 4

Scenario No.	Job Queue	Rescheduling Time Span (T) (s)	Solving Time (s)
1	A → B → C	430	0.41
2*	A → C → B	426	0.73
3	B → A → C	464	0.38
4	B → C → A	503	0.27
5	C → A → B	431	0.42
6	C → B → A	504	0.29

*Optimal rescheduling result of Case Study 4.

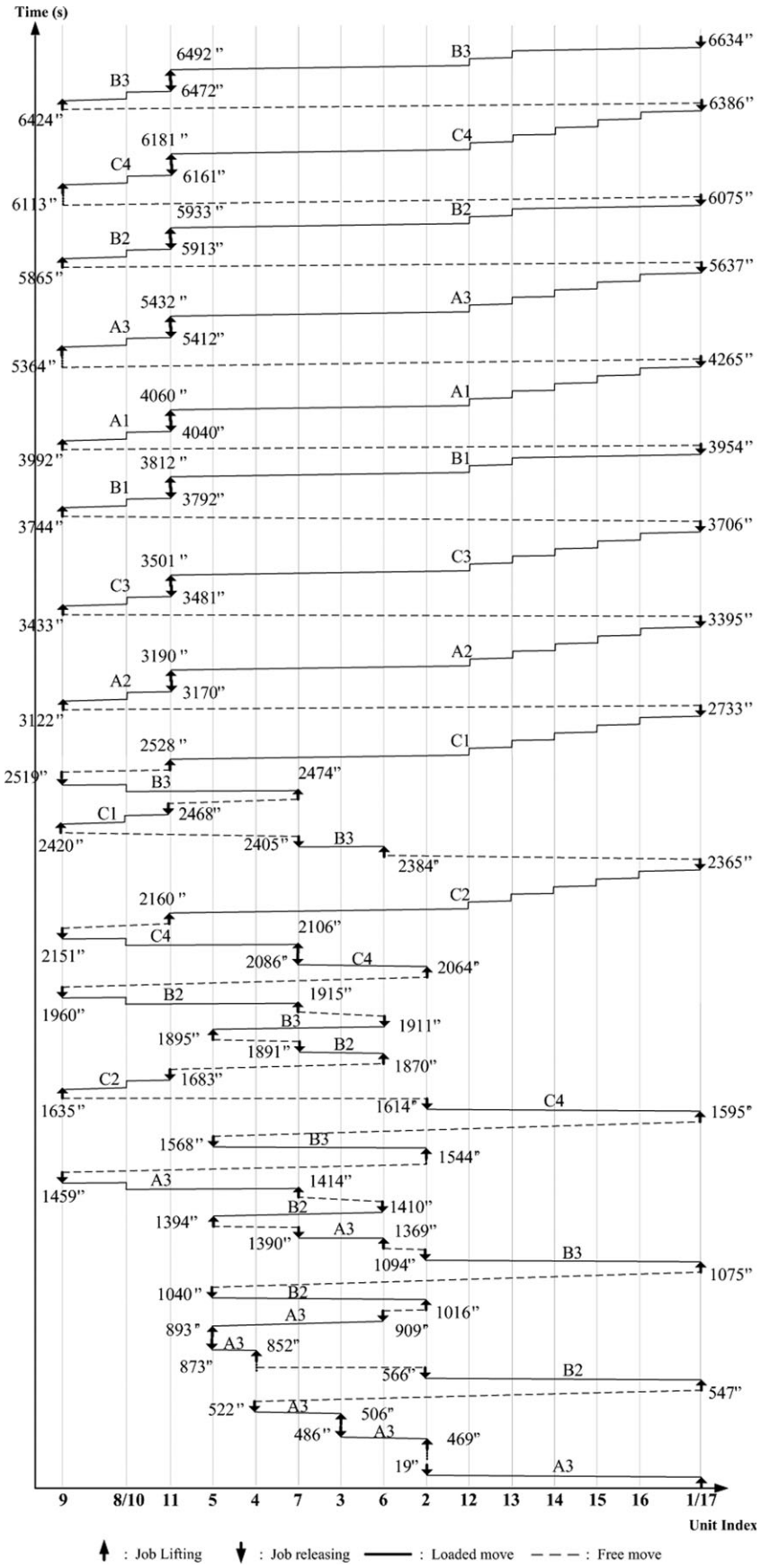


Figure 11. Optimal scheduling results for Case 5.

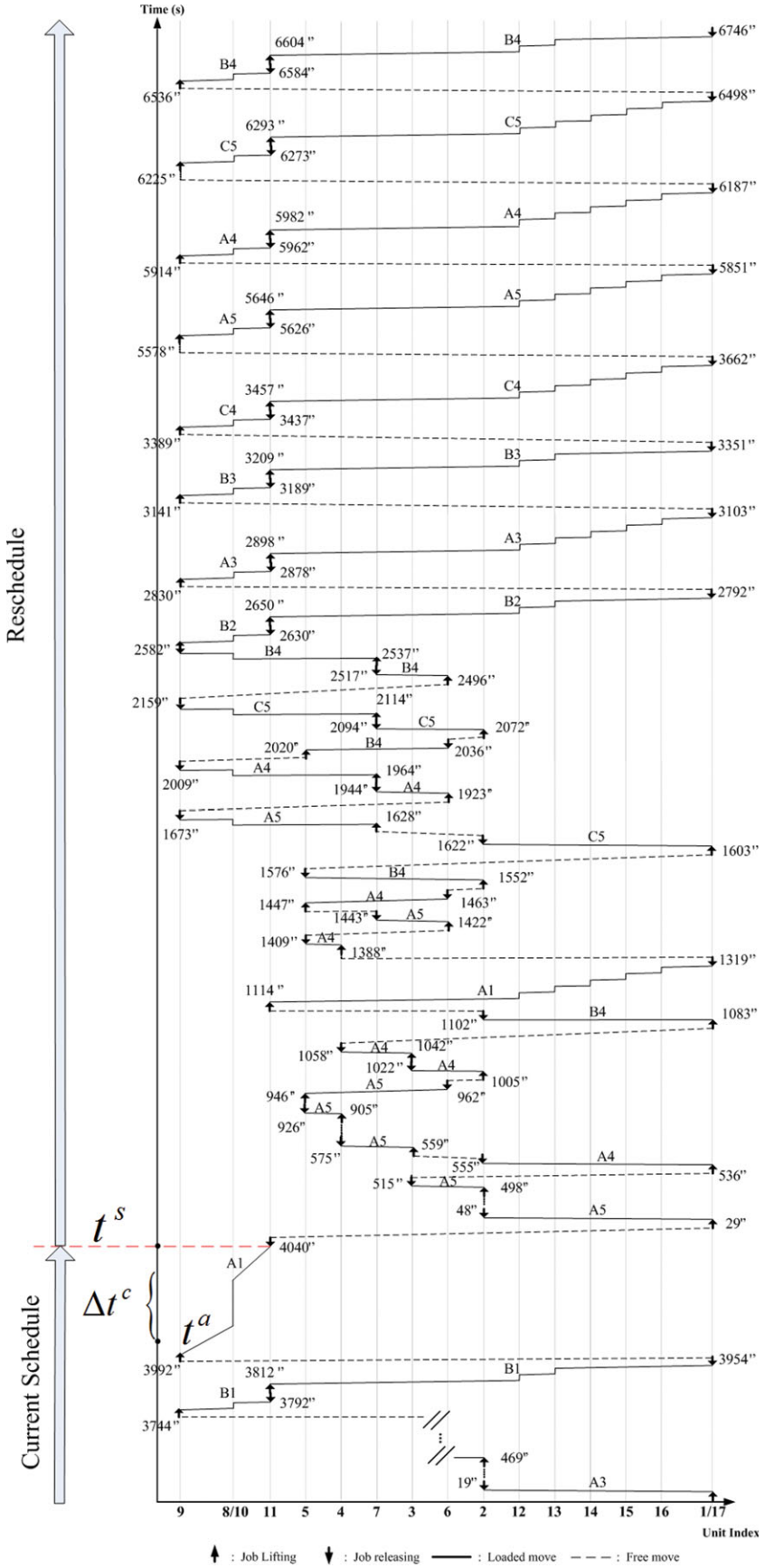


Figure 12. Optimal rescheduling results for Case 5.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

production line simultaneously, the size of this MILP problem becomes larger, which suggests Δt^c should be increased accordingly. This might affect the real time application of reschedules. This should be analyzed through pretest calculation and case by case as well.

Case 5: scheduling of a real-world electroplating problem

The developed methodology has also been used to solve the same electroplating problem from Liu et al.,[11] which is a real hoist scheduling problem from industry. In this electroplating line, three types of unit operation, cleaning, rinsing, and plating, are performed in 16 units. The capacity of unit 9 is 8, and other units are with single capacity. Three different types of jobs will be processed both in current schedule and in reschedule. When new jobs come to this system, it will trigger a rescheduling. At the beginning of the current scheduling problem, there are six jobs under processing in unit 9; meanwhile, four jobs are just released onto the loading zone. Thus, 10 jobs will be processed in the production line of 16 units. The initial jobs in the production line include 3 job A in units 1 and 9; 3 job B in units 1 and 9; and 4 job C in units 1 and 9. The developed MILP model for the current schedule has 2244 binary variables, 3935 continuous variables, and 15,388 constraints, and the solving time for this case is 13.24 s. Figure 11 shows the global optimal scheduling results for the current schedule. The total scheduling time span is 6634 s. Every starting and ending time point of the loaded moves, as well as the job index carried by the hoist for each loaded move are marked in the figure.

As the electroplating line in this case contains more units and jobs than in the previous cases, the solution identification time will increase. Through our pretesting, 30 s can be safely adopted as the computational time (i.e., $\Delta t^c = 30$). Suppose at the time 3995 s, four jobs simultaneously join the loading zone. As shown in Figure 12, the switch time for rescheduling will be set at 4040 s, when the hoist just finishes releasing job A1 to unit 11. And unit 11 will be set as the hoist initial position. In this schedule, nine jobs will be processed in the production line of 16 units. The initial jobs in the production line include 4 job A in units 1, 9, and 11; 3 job B in units 1 and 9; and 2 job C in units 1 and 9. The MILP model for this case has 2022 binary variables, 3631 continuous variables, and 14,442 constraints, and the solving time for rescheduling is 11.98 s. The rescheduling time span for the global optimal solution is 6746 s. Figure 12 shows the global optimal rescheduling results and how the system is switched from the current schedule to the reschedule.

Case 5 demonstrates that the developed methodology still works very well for a large-scale real industrial problem. Certainly, for a very large-scale hoist scheduling problem, the central processing unit solving time will become a big issue for real-time application. However, because generally multihoists will be used for such an application, the scheduling problem can be decomposed into several subproblems and each subproblem will contain a hoist and only a section of a production line, which can still be efficiently solved based on the development of this study.

Concluding Remarks

RDHS targets online generation of optimal schedules for fast handling multiple types of jobs with different recipes that continuously and randomly arrive at a production line. RDHS inevitably needs to overcome the computational complexity for real-time application; and in the meantime, to confront many stringent situations due to uncertainties including job arrival time, type, recipe, and job number. In this article, a novel optimized RDHS methodology is developed, which takes into account the uncertainty of new incoming jobs and targets real-time scheduling optimality and applicability. The RDHS methodology addresses all the major scheduling issues of a multistage material handling process, such as multiple recipes, multiple jobs, multicapacity processing units, diverse processing time requirements, and even optimal processing queues for new coming jobs. The efficacy of the developed methodology is demonstrated by various case studies.

Acknowledgments

This work was supported in part by Research Enhancement Grant and Graduate Student Scholarship from Lamar University.

Notation

Sets

$\mathrm{SN} = \{n | n = 1,\ldots,N\}$ set of jobs in the rescheduling problem
$\mathrm{SI} = \{i | i = \mathrm{BI},\ldots,\mathrm{EI}\}$ set of units of the production line
SI_n = set of leftover processing steps of job n
$\mathrm{SI}_n^{\mathrm{A}}$ = set of units that the hoist can lift job n from
recipe_n = complete processing procedure for manufacturing job n

Parameters

BI = loading zone
Ca_i = number of jobs that unit i can simultaneously process
EI = unloading zone
$F_{i',i}$ = traveling time of the hoist free move from unit i' to unit i
$L_{i,n}$ = traveling time of the hoist loaded move for carrying job n from unit i to the next processing unit according to recipe_n
M = a sufficient large number
$\mathrm{RT}_{i,n}^{\mathrm{lo}}$ = lower time limit for processing job n in unit i
$\mathrm{RT}_{i,n}^{\mathrm{up}}$ = upper time limit for processing job n in unit i
t^{a} = time point when new jobs are added into the production line
Δt^c = the estimated time consumption for solving the rescheduling model

Variables

BI_n = initial location (characterized by a unit index) of job n at the beginning of a reschedule
HI = initial hoist position (characterized by a unit index) at the beginning of a reschedule
$\mathrm{IT}_{i,n}$ = elapsed processing time duration of job n staying in the unit i at the beginning of a reschedule
$p_{i,n}$ = processing time for job n staying in unit i
$\mathrm{SI}_{i,n,i',n'}$ = positive variable to substitute $\left(E_{i'++1,n'} + F_{i'++1,i}\right) w_{i,n,i',n'}$
$t_{i,k}$ = time point that the hoist travels to the unit i for the kth time
t^{s} = switch time point for starting to implement rescheduling results
T = rescheduling time span
$\mathrm{TE}_{i,n}$ = ending time point of a loaded move, when job n is released into unit i
$\mathrm{TS}_{i,n}$ = starting time point of a loaded move, when job n is lifted from unit i
$u_{i,n,n'}$ = binary variable, which is 1 if $\mathrm{TE}_{i,n'} \leq \mathrm{TS}_{i,n}$; otherwise, it is 0
$v_{i,n,n'}$ = binary variable, which is 1 if $\mathrm{TE}_{i,n} \leq \mathrm{TE}_{i,n'}$; otherwise, it is 0
$w_{i,n,i',n'}$ = binary variable, which is 1 if the loaded move that picks up job n' from unit i' occurs ahead of the loaded move that picks up job n from unit i; otherwise, it is 0
$x_{i,n,n'}$ = binary variable, which is 1 if job n' once stays with job n in unit i; otherwise, it is 0

Literature Cited

1. Kumar PR. Scheduling semiconductor manufacturing plants. *IEEE Control System*. 1994;14:33–40.

2. Phillips LW, Unger PS. Mathematical programming solution of a hoist scheduling program. *AIIE Trans*. 1976;8:219–225.
3. Shapiro GW, Nuttle HW. Hoist scheduling for a PCB electroplating facility. *IIE Trans*. 1988;20:157–167.
4. Lei L, Wang TJ. A proof: the cyclic hoist scheduling problem is NP-complete. *Working Paper 89–0016*. Rutgers University, 1989.
5. Baptiste P, Legeard B, Varnier C. Hoist scheduling problem: an approach based on constraints logic programming. In: *Proceedings of IEEE Conference on Robotics and Automation, Nice, France, Vol*. 2. 1992:1139–1144.
6. Lei L, Wang TJ. Determining optimal cyclic hoist schedules on a single-hoist electroplating line. *IIE Trans*. 1994;26:25–33.
7. Armstrong R, Lei L, Gu S. A bounding scheme for deriving the minimal cycle time of a single-transporter N-stage process with time window constraints. *Eur J Oper Res*. 1994;78:130–140.
8. Rodosek R, Wallace MG. A generic model and hybrid algorithm for hoist scheduling problems. In: *Proceedings of the 4th International Conference on Principles and Practice of Constant Programming, Springer, Berlin*. 1998:385–399.
9. Xu Q, Huang YL. Graph-assisted optimal cyclic hoist scheduling for environmentally benign electroplating. *Ind Eng Chem Res*. 2004;43:8307–8316.
10. Kuntay I, Xu Q, Uygun K, Huang YL. Environmentally conscious hoist scheduling for electroplating facilities. *Chem Eng Commun*. 2006;193:273–293.
11. Liu CW, Fu J, Xu Q. Simultaneous mixed-integer dynamic optimization for environmentally benign electroplating. *Comput Chem Eng*. 2011;35:2411–2425.
12. Liu CW, Zhao CY, Xu Q. Integration of electroplating process design and operation for simultaneous productivity maximization, energy saving, and freshwater minimization. *Chem Eng Sci*. 2012;68:202–214.
13. Yin NC, Yih Y. Crane scheduling in a feasible electroplating line: a tolerance based approach. *J Electron Manuf*. 1992;2:137–144.
14. Yih Y. An algorithm for hoist scheduling problems. *Int J Prod Res*. 1994;32:501–516.
15. Lamothe J, Correge M, Delmas J. *Hoist scheduling problem in a real time context*. In: *11ième conference internationale sur l'analyze et l'optimisation des systmès*, Sophia antipolis, 1994.
16. Lamothe J, Correge M, Delmas J. *A dynamic heuristic for the real time hoist scheduling problem*. In: *Symposium on Emerging Technologies and Factory Automation*, Paris October, 1995.
17. Goujon J, Lacomme P. *Computerized industrial surface treatment line control: resolution of hoist scheduling problem*. In: *Simulation in Industry, ESS'96*, Genoa, Italy, Vol. 1. 1996:377–382.
18. Riera D, Yorke-Smith N. An improved hybrid model for the generic hoist scheduling problem. *Ann Oper Res*. 2002;115:173–191.
19. Zhou Z, Li H. A heuristic method for one hoist dynamic scheduling. *Syst Eng-Theory Methodol Appl*. 2002;11:136–140.
20. Xu Q, Huang YL. Real-time dynamic hoist scheduling under uncertainties. In: *AIChE Annual National Meeting*, Cincinnati, OH, 2005.
21. Lei L, Wang TJ. The minimum common-cycle algorithm for cycle scheduling of two material handling hoists with time window constraints. *Manage Sci*. 1991;37:1629–1639.
22. Varnier C, Bachelu A, Baptiste P. Resolution of the cyclic multihoists scheduling problem with overlapping partitions. *INFOR*. 1997;35:309–324.
23. Manier MA, Varnier C, Baptiste P. Constraint-based model for the cyclic multi-hoists scheduling problem. *Prod Planning Control*. 2000;11:244–257.
24. Yang G, Ju DP, Zheng WM, Lam K. Solving multiple hoist scheduling problems by use of simulated annealing. *Transport Res B*. 2001;36:537–555.
25. Leung JMY, Zhang G. Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence. *IEEE Trans Robotics Automation*. 2003;19:480–484.
26. Che A, Chu C. Single-track multi-hoist scheduling problem: a collision-free resolution based on a branch-and-bound approach. *Int J Prod Res*. 2004;42:2435–2456.
27. Leung JMY, Zhang G, Yang X, Mak R, Lam K. Optimal cyclic multi-hoist scheduling: a mixed integer programming approach. *Oper Res*. 2004;52:965–976.
28. Mocchia L, Cordeau JF, Gaudioso M, Laporte G. A branch-and-cut algorithm for the Quay Crane scheduling problem in A container terminal. *Naval Res Logistics*. 2005;53:45–59.
29. Leung JMY, Levner E. An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. *Oper Res Lett*. 2006;34:465–472.
30. Zhou Z, Li L. A solution for cyclic scheduling of multi-hoists without overlapping. *Ann Oper Res*. 2009;168:5–21.
31. Aron I, Genç-Kaya L, Harjunkoski I, Hoda S, Hooker JN. *Factory crane scheduling by dynamic programming*. In: Wood RK, Dell RF, editors. *Operations Research, Computing and Homeland Defense (ICS 2011 Proceedings)*. INFORMS: Monterey, CA, 2010:93–107.
32. Ierapetritou MG, Pistikopoulos EN. *Global optimization for stochastic planning, scheduling and design problems*. In: Grossmann IE, editor. *Global Optimization in Engineering Design*. Kluwer Academic Publishers: Dordrecht, The Netherlands, 1996:231–287.
33. Vin JP, Ierapetritou MG. Robust short-term scheduling of multiproduct batch plants under demand uncertainty. *Ind Eng Chem Res*. 2001;40:4543–4554.
34. Bonfill A, Bagajewicz M, Espuña A, Puigjaner L. Risk management in the scheduling of batch plants under uncertain market demand. *Ind Eng Chem Res*. 2004;43:741–750.
35. Goel V, Grossmann IE. A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Comput Chem Eng*. 2004;28:1409–1429.
36. Balasubramanian J, Grossmann IE. Approximation to multistage stochastic optimization in multiperiod batch plant scheduling under demand uncertainty. *Ind Eng Chem Res*. 2004;43:3695–3713.
37. Lin X, Janak SL, Floudas CA. A new robust optimization approach for scheduling under uncertainty. I. Bounded uncertainty. *Comput Chem Eng*. 2004;28:1069–1085.
38. Jia Z, Ierapetritou MG. Short-term scheduling under uncertainty using MILP sensitivity analysis. *Ind Eng Chem Res*. 2004;43:3782–3791.
39. Janak SL, Lin X, Floudas CA. A new robust optimization approach for scheduling under uncertainty. II. Uncertainty with known probability distribution. *Comput Chem Eng*. 2007;31:171–195.
40. Orçun S, Altinel K, Hortacsu Ö. Scheduling of batch processes with operational uncertainties. *Comput Chem Eng*. 1996;20:S1191–S1196.
41. Petkov SB, Maranas SD. Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty. *Ind Eng Chem Res*. 1997;36:4864–4881.
42. Li W, Hui CW, Li P, Li AX. Refinery planning under uncertainty. *Ind Eng Chem Res*. 2004;43:6742–6755.
43. Mitra K, Ravindra GD, Patwardhan SC, Sardar G. Midterm supply chain planning under uncertainty: a multiobjective chance constrained programming framework. *Ind Eng Chem Res*. 2008;47:5501–5511.
44. You F, Grossmann IE. Mixed-integer nonlinear programming models and algorithms for large-scale supply chain design with stochastic inventory management. *Ind Eng Chem Res*. 2008;47:7802–7817.
45. Balasubramanian J, Grossmann IE. Scheduling optimization under uncertainty—an alternative approach. *Comput Chem Eng*. 2003;27:469–490.
46. Wang J. A fuzzy robust scheduling approach for product development projects. *Eur J Oper Res*. 2004;152:180–194.
47. Petrovic D, Duenas A. A fuzzy logic based production scheduling/rescheduling in the presence of uncertain disruptions. *Fuzzy Sets Syst*. 2006;157:2273–2285.
48. Jia Z, Ierapetritou MG. Generate pareto optimal solutions of scheduling problems using normal boundary intersection technique. *Comput Chem Eng*. 2006;31:268–280.
49. Jia Z, Ierapetritou MG. Uncertainty analysis on the right-hand-side for MILP problems. *AIChE J*. 2006;52:2486–2495.
50. Acevedo J, Pistikopoulos EN. A hybrid parametric/stochastic programming approach for mixed-integer linear problems under uncertainty. *Ind Eng Chem Res*. 1997;36:2262–2270.
51. Acevedo J, Pistikopoulos EN. A multiparametric programming approach for linear process engineering problems under uncertainty. *Ind Eng Chem Res*. 1997;36:717–728.
52. Ryu JH, Dua V, Pistikopoulos EN. A bilevel programming framework for enterprise-wide process networks under uncertainty. *Comput Chem Eng*. 2004;28:1121–1129.
53. Verderame PM, Floudas CA. Operational planning of large-scale industrial batch plants under demand due date and amount uncertainty: I. Robust optimization. *Ind Eng Chem Res*. 2009;48:7214–7231.
54. Rodrigues MTM, Gimeno L, Passos CAS, Campos MD. Reactive scheduling approach for multipurpose chemical batch plants. *Comput Chem Eng*. 1996;20:S1215–S1220.
55. Vin JP, Ierapetritou MG. A new approach for efficient rescheduling of multiproduct batch plants. *Ind Eng Chem Res*. 2000;39:4228–4238.

56. Méndez CA, Cerdá J. Dynamic scheduling in multiproduct batch plants. *Comput Chem Eng*. 2003;27:1247–1259.
57. Méndez CA, Cerdá J. An MILP framework for batch reactive scheduling with limited discrete resources. *Comput Chem Eng*. 2004;28:1059–1068.
58. Novas JM, Henning GP. A reactive scheduling approach based on domain-knowledge. *Comput Aided Chem Eng*. 2009;27:765–770.
59. Kanakamedala KB, Reklaitis GV, Venkatasubramanian V. Reactive schedule modification in multipurpose batch chemical plants. *Ind Eng Chem Res*. 1994;30:77–90.
60. Huercio A, Espuna A, Puigjaner L. Incorporating on-line scheduling strategies in integrated batch production control. *Comput Chem Eng*. 1995;19:S609–S615.
61. Sanmarti E, Huercio A, Espuña A, Puigjaner L. A combined scheduling/reactive scheduling strategy to minimize the effect of process operations uncertainty in batch plants. *Comput Chem Eng*. 1996;20:1263–1268.
62. Roslöf J, Harjunkoski I, Bjorkqvist J, Karlsson S, Westerlund T. An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. *Comput Chem Eng*. 2001;25:821–828.
63. Janak SL, Floudas CA. Production scheduling of a large-scale industrial batch plant. II. Reactive scheduling. *Ind Eng Chem Res*. 2006;45:8253–8269.
64. Floudas CA, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng*. 2004;28:2109–2129.
65. Li ZK, Ierapetritou M. Process scheduling under uncertainty: review and challenges. *Comput Chem Eng*. 2008;32:715–727.
66. Verderame PM, Elia JA, Li J, Floudas CA. Planning and scheduling under uncertainty: a review across multiple sectors. *Ind Eng Chem Res*. 2010;49:3993–4017.
67. GAMS. *GAMS—A User' s Guide*. Washington, DC: GAMS Development Corporation, 2009.
68. CPLEX. *Using the CPLEX Callable Library*. Incline Village, NV: CPLEX Optimization, Inc., 2009.

Manuscript received Nov. 14, 2011, and revision received May 10, 2012.