AIChE

Reactive Scheduling Using Parametric Programming

Zukui Li and Marianthi G. Ierapetritou
Dept. of Chemical and Biochemical Engineering, Rutgers University, Piscataway, NJ 08854

DOI 10.1002/aic.11593
Published online August 26, 2008 in Wiley InterScience (www.interscience.wiley.com).

To address the various disruptive events that occur during process operations, reactive scheduling is commonly used. However, a major limitation of the existing reactive scheduling techniques is the response time, which might cause significant delay while the generation of a new schedule takes place. In this article, a novel approach is proposed to improve the efficiency of reactive scheduling and to avoid the resolution of a complex optimization problem when uncertain event occurs during the scheduling period. In the proposed method, reactive schedule is obtained from the solution of multiparametric programming problem, which is solved ahead of time and covers all possible outcomes of future uncertainty. The multiparametric programming problem is derived from a new reactive scheduling formulation, which integrates disruptive events (rush order and machine breakdown) as uncertain parameters in the process modeling. Several examples are presented to illustrate the effectiveness of the proposed approach. © 2008 American Institute of Chemical Engineers *AIChE J,* 54: 2610–2623, 2008
Keywords: reactive scheduling, multiparametric programming, uncertainty, rush order, machine breakdown

Introduction

Uncertainty is a very important concern in scheduling of process operations in real plants. Uncertainty appears from different sources, such as variability in processing times of different tasks, unexpected machine breakdown, staffing/operator problems, unexpected arrival of new orders, cancellation or modification of existing orders, early or late arrival of raw materials, modification of release and due dates, etc. These uncertainties often make the schedule generated under the deterministic assumption suboptimal or even infeasible. Uncertainty consideration, thus, is very important to preserve plant feasibility and viability during operations.

Based on different treatment of uncertainty, methods for process scheduling under uncertainty can be classified into two groups: preventive scheduling and reactive scheduling. Preventive scheduling generates scheduling policies before uncertainty occurs by taking into account uncertainty in generating schedules that can tolerate parameter variability. The consideration of uncertainty information is used to make the preventive schedule more robust, which means to remain valid and satisfy performance requirements under a wide variety of disturbances. Preventive scheduling is often used when the uncertainty can be quantified in some way. Typical techniques used for preventive scheduling are stochastic programming and robust optimization. Stochastic programming model handles future uncertainty (e.g., demand) through recourse operation according to different uncertainty scenarios[1–3]; whereas robust optimization based methods aim at generating feasible schedule for all possible scenarios.[4–6] If the degree of uncertainty is high or if there is no information about the behavior of uncertainty, a different reactive scheduling approach is more appropriate.

Reactive scheduling, which is also called rescheduling, takes place when the schedule is implemented based on up-to-date information regarding the state of the system. It requires the modification of the existing schedule during the manufacturing process to adapt to the changes (uncertainty)

Correspondence concerning this article should be addressed to M. G. Ierapetritou at marianth@soemail.rutgers.edu.

such as rush order arrivals, order cancellations, or machine breakdowns. For this type of uncertainty, there is not enough information prior to realization of the uncertain parameters that will allow a protective action, so almost all the methods in the literature aim to resolve a rescheduling problem once the disruptive events occur.

The reactive scheduling actions are based on various underlying strategies. It can rely on simple techniques or heuristic rules to seek a quick schedule consistency restoration. One of the earliest efforts in reactive scheduling was reported by Cott and Macchietto[7] who considered fluctuations of processing times and used a shifting algorithm to modify the starting times of processing steps of a batch by the maximum deviation between the expected and actual processing times of all related processing steps. Kanakamedala et al.[8] developed a least-impact heuristic approach with two levels that allows time shifting and unit replacement in multipurpose batch plants. Huercio et al.[9] proposed a reactive scheduling technique to deal with variations in task processing times and equipment availability. They generated a set of decision trees using alternative unit assignments, each based on a conflict in the real production schedule caused by a deviation between the real schedule and the nominal schedule. Branches of the trees are then pruned according to heuristic equipment selection rules. Sanmartí et al.[10] extended this work to cover unexpected equipment failure. Rodrigues et al.[11] also considered uncertain processing times and proposed a rolling-horizon approach that incorporates a look-ahead procedure to avoid possible violations of future due dates. Honkomp et al.[12] proposed a reactive scheduling framework for processing time variations and equipment breakdown by coupling a deterministic schedule optimizer with a simulator that introduces stochastic events where two different formulations of time are considered. A number of rescheduling strategies were proposed and heuristics were used to locate critical tasks, which can be modified to make the nominal schedule less susceptible to the effects of processing time variability.

On the other hand, a number of the techniques presented in the literature involve a full scheduling of the tasks that have to be executed after the unexpected event occurs through mathematical programming approaches relying mostly on mixed integer linear programming (MILP). Roslöf et al.[13] developed an MILP-based heuristic algorithm by iteratively releasing a set of jobs from a nominal schedule and optimally reallocating them, where the complexity of the problem is controlled through the number of simultaneously released jobs. Ruiz et al.[14] presented a fault diagnosis system that interacts with a schedule optimizer for multipurpose batch plants to perform reactive scheduling in the event of processing time variability or unit unavailability. Méndez and Cerda[15] proposed a rescheduling method by first reassigning resources to tasks that still need to be processed and then reordering the sequence of processing tasks for each resource item. They considered start time shifting, local reordering, and unit reallocation of old batches as well as insertion of new batches. This work was extended in Méndez and Cerda[16] to include limited discrete renewable resources where only start-time shifting, local batch reordering, and resource reallocation of existing batches are allowed. Vin and Ierapetritou[17] considered the rescheduling of multiproduct and multipurpose batch plants in the event of machine breakdown or rush order arrival. Full-scale rescheduling of each production schedule is avoided by fixing binary variables for a subset of tasks from the original production schedule. The fixing of tasks results in a reduced computational effort required to solve the resulting MILP problem. Janak and Floudas[18] presented a similar framework where the fixed subset of tasks is determined using a detailed set of rules that reflect the production needs and can be modified for different production facilities. By fixing a subset of tasks, a reduced computational effort is required to solve the resulting MILP problem.

As shown from the literature, a major consideration for reactive scheduling is the response time. If the computation time is large the production may be significantly delayed while the new schedule is developed. In this article, we proposed a framework to solve the reactive scheduling problems using multiparametric programming technique, which will greatly improve the efficiency of the rescheduling approach because the new schedule is obtained from the solution of parametric programming problem which was solved before the occurrences of disruptive events, thus completely avoiding the solution of the rescheduling optimization problem.

Parametric programming serves as an analytic tool by mapping the uncertainties in the optimization problem to optimal alternatives. From this point of view, parametric programming provides the exact mathematical solution of the optimization problem under uncertainty. In the literature, there are not many records on the application of parametric programming in process scheduling problem. Ryu and Pistikopoulos[19] has reported the application of parametric programming to a zero-waiting scheduling problem, where they studied the parametric solution under processing time uncertainty for zero-wait batch processes, but the scheduling formulation does not consider the executed tasks so it is not able to address the reactive scheduling problem. Li and Ierapetritou[20] proposed an efficient multiparametric programming framework and applied it to general scheduling problem to study the effect of uncertain product demand, price and processing time on preventive scheduling problem. In this article, the work is further extended to study the reactive scheduling problem.

The rest of this article is organized as follows. A general reactive scheduling formulation is presented in next section for two kinds of uncertainty: rush order and machine breakdown. Then, a multiparametric programming method, which will be used to solve the parametric reactive scheduling problem, is described. The reactive scheduling for two types of unexpected events is illustrated through examples and the article is finally summarized in the last section.

Reactive Scheduling Formulation

Before the presentation of the reactive scheduling formulation, the deterministic model is first presented in the next subsection. It should be noticed that the proposed methodology for reactive scheduling is not tight to the specific deterministic model. Any schedule modeling framework can be used as long as it can be formulated as a MILP problem, such as the one presented by Floudas and Lin,[21] Méndez et al.,[22] Maravelias and Grossmann.[23]

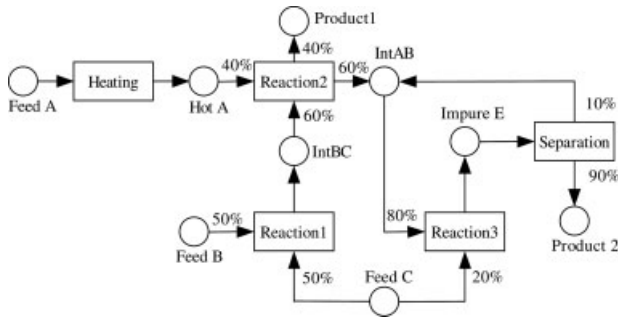

Figure 1. State-task-network (STN) representation of example 1.

Deterministic Scheduling

The deterministic model used for batch process scheduling in this article follows the main idea of the continuous time formulation proposed by Ierapetritou and Floudas.[24] The general model involves the following objective and constraints.

$$\max \sum_{s,n} price_s d_{s,n} \tag{1}$$

s.t.

$$\sum_{i \in I_j} wv_{i,j,n} \le 1 \quad \forall j \in J,\ \forall n \in N \tag{2}$$

$$st_{s,n} = st_{s,n-1} - d_{s,n} - \sum_{i \in I_s} \rho^P_{s,i} \sum_{j \in J_i} b_{i,j,n} + \sum_{i \in I_s} \rho^c_{s,i} \sum_{j \in J_i} b_{i,j,n-1} \quad \forall s \in S,\ \forall n \in N \tag{3}$$

$$st_{s,n} \le st^{\max}_s \quad \forall s \in S,\ \forall n \in N \tag{4}$$

$$v^{\min}_{i,j} wv_{i,j,n} \le b_{i,j,n} \le v^{\max}_{i,j} wv_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{5}$$

$$\sum_n d_{s,n} \ge r_s \quad \forall s \in S \tag{6}$$

$$Tf_{i,j,n} = Ts_{i,j,n} + \alpha_{i,j} wv_{i,j,n} + \beta_{i,j} b_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{7}$$

$$Ts_{i,j,n+1} \ge Tf_{i,j,n} - U(1 - wv_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{8}$$

$$Ts_{i,j,n+1} \ge Tf_{i',j,n} - U(1 - wv_{i',j,n}) \quad \forall i, i' \in I_j, \forall j \in J, \forall n \in N \tag{9}$$

$$Ts_{i,j,n+1} \ge Tf_{i',j',n} - U(1 - wv_{i',j',n}) \quad \forall i, i' \in I_j, i \ne i', \forall j, j' \in J,\ \forall n \in N \tag{10}$$

$$Ts_{i,j,n+1} \ge Ts_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{11}$$

$$Tf_{i,j,n+1} \ge Tf_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{12}$$

$$Ts_{i,j,n} \le H \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{13}$$

$$Tf_{i,j,n} \le H \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{14}$$

The state-task network (STN) concept is employed in the above model, where state nodes represent feeds, intermediates, and final products; task nodes represent the process operations, which transform state(s) to other state(s), and directed edges represent the flow of materials. The model employ the concept of event point that allows different tasks to start at different moments in different units for the same event point, which relaxes the global time point representation.[22,25] In the above formulation, the objective function (1) is the profit although different performance measures like makespan can be used; allocation constraints (2) state that only one of the tasks can be performed in each unit at an event point (n); constraints (3) represent the material balances for each state (s) expressing that at each event point (n) the amount $st_{s,n}$ is equal to that at event point ($n - 1$), adjusted by any amounts produced and consumed between event points ($n - 1$) and (n), and delivered to the market at event point (n); the storage and capacity limitations of production units are expressed by constraints (4) and (5); constraints (6) are written to satisfy the demands of final products; and constraints (7)–(14) represent time limitations due to task duration and sequence requirements in the same or different production units. Detailed description of the symbols in the above formulation is provided in the nomenclature section of the article.

Reactive Scheduling

To apply the parametric programming method on reactive scheduling, it is necessary to develop an effective way to model the disruptive uncertainty into the scheduling formulation (1)–(14). An important fact for formulating the reactive scheduling is that the tasks that have already executed or started cannot be changed. In a previously published work,[17] those binary variables that corresponds to a subset of tasks of

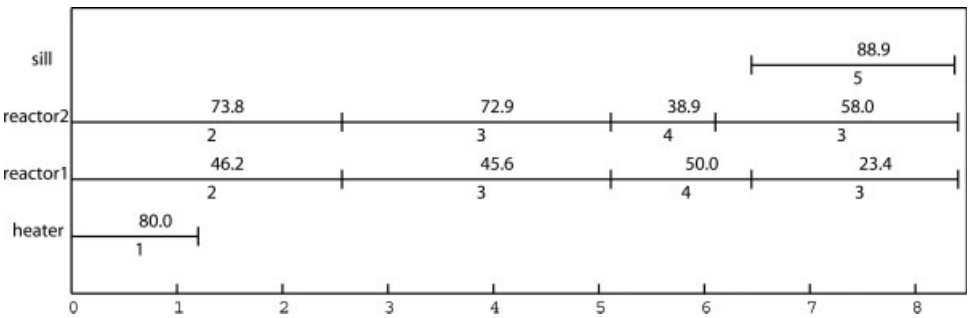

Figure 2. Original schedule of example 1 with nominal demand.

the original production schedule that have been executed are fixed while generating the reactive schedule. However, this method only solves one reactive schedule after the uncertain event occurs.

Our target in this work is to develop a new reactive scheduling formulation to consider all possible uncertain outcomes by formulating the uncertain events as uncertain parameters into the optimization problem. The basic strategy is to generate a complete reschedule but fix the executed tasks with a set of binary indicator variables $y_{i,j,n}$, which denote whether a task is executed ($y_{i,j,n} = 1$) or not ($y_{i,j,n} = 0$). The rules and corresponding constraints to identify these indicator variables will be presented in the next two subsections for the specific disruptive event, rush order or machine breakdown. Here, the constraints that ensure that the executed tasks are fixed using the indicator variables ($y_{i,j,n}$) are described as follows:

$$wv_{i,j,n} = wv^{\text{old}}_{i,j,n}, \quad \text{if } y_{i,j,n} = 1 \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (15)$$

$$b_{i,j,n} = b^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n}, \quad \text{if } y_{i,j,n} = 1 \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (16)$$

$$Ts_{i,j,n} = Ts^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n}, \quad \text{if } y_{i,j,n} = 1 \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (17)$$

$$Tf_{i,j,n} = Tf^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n}, \quad \text{if } y_{i,j,n} = 1 \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (18)$$

Constraint (15) ensures that if a task i assigned to unit j at event point n has been executed, then the corresponding variable $wv_{i,j,n}$ has to be fixed to the value $wv^{\text{old}}_{i,j,n}$ that represents the task in the original schedule. Similarly, constraints (16)–(18) ensure that batch size, task starting and completion time are fixed at the same values as the ones in the original schedule.

The logical constraints (15)–(18) are transformed to mathematical programming constraints as follows.

$$wv^{\text{old}}_{i,j,n} - (1 - y_{i,j,n}) \le wv_{i,j,n} \le wv^{\text{old}}_{i,j,n} + (1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (19)$$

$$b^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - b^{\text{UB}}_{i,j}(1 - y_{i,j,n}) \le b_{i,j,n} \le b^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + b^{\text{UB}}_{i,j}(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (20)$$

$$Ts^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - U(1 - y_{i,j,n}) \le Ts_{i,j,n} \le Ts^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + U(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (21)$$

$$Tf^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - U(1 - y_{i,j,n}) \le Tf_{i,j,n} \le Tf^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + U(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad (22)$$

where $b^{\text{UB}}_{i,j}$ is the upper bound of the batch size, and U is the upper bound of the scheduling time horizon.

Constraint (19) is equivalent to constraint (15). This can be shown as follows: if $y_{i,j,n} = 1$, then constraint (19) becomes $wv^{\text{old}}_{i,j,n} \le wv_{i,j,n} \le wv^{\text{old}}_{i,j,n}$, i.e., $wv_{i,j,n}$ is fixed to $wv^{\text{old}}_{i,j,n}$; if on the other hand $y_{i,j,n} = 0$, then (19) becomes

Table 1. Rush Order Uncertainty for Example 1

	Value	Range
$r^{\text{rush}}_{\text{P1}}$ (New demand of P1)	θ_1	$70 \le \theta_1 \le 90$
T^{rush} (Order arrival time)	θ_2	$2 \le \theta_2 \le 6$

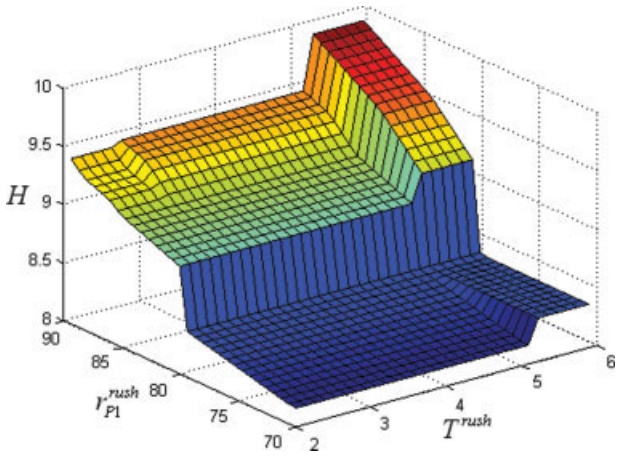

Figure 3. Parametric solution of optimal makespan and the rush order.

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

$wv^{\text{old}}_{i,j,n} - 1 \le wv_{i,j,n} \le wv^{\text{old}}_{i,j,n} + 1$, which is a redundant constraint because it is satisfied for any value of the binary variable $wv_{i,j,n}$. Similarly, constraints (20)–(22) are equivalent to logical constraints (16)–(18), respectively.

To determine the value of $y_{i,j,n}$, additional constraints are required depending on the nature of the disruptive event: rush order or machine breakdown. In the next two subsections, we specify the rules and constraints that determine the value of $y_{i,j,n}$ and present the complete reactive scheduling formulation.

Rush order

Once a rush order arrives during the scheduling execution process, all the tasks that have already started should be identified as executed. When this rule is implemented on the original schedule solution, the value of $y_{i,j,n}$ can be identified. However, this rule should also be implemented on the complete reschedule, so that the reactive schedule does not change the schedule history; otherwise, the reactive schedule can generate "wrong" tasks that start before the disruptive

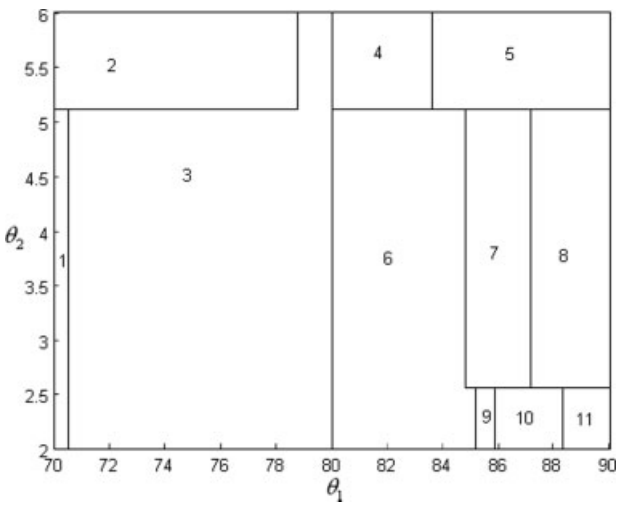

Figure 4. Critical regions of example 1 with rush order.

Table 2. Parametric Objective for Example 1 with Rush Order

Critical Region	Makespan H(h)
1	8.1614
2	8.3723
3	$6.353 + 0.02564\theta_1$
4	$0.6192 + 0.10667\theta_1$
5	$3.962 + 0.06667\theta_1$
6	$5.6353 + 0.041\theta_1$
7	$0.071 + 0.10667\theta_1$
8	$5.7892 + 0.041\theta_1$
9	$0.0467 + 0.10667\theta_1$
10	$5.769 + 0.04\theta_1$
11	$-0.1174 + 0.10667\theta_1$

Table 3. Integer Solution of Critical Region 10

$\{(i,j,n) \mid wv_{i,j,n} = 1\}$	$\{(i,j,n) \mid y_{i,j,n} = 1\}$
heating.heater.n0	heating.heater.n0
heating.heater.n1	heating.heater.n1
heating.heater.n2	rxn1.rtr1.n0
rxn1.rtr1.n1	rxn1.rtr1.n1
rxn1.rtr1.n3	rxn1.rtr2.n0
rxn1.rtr2.n1	rxn1.rtr2.n1
rxn2.rtr1.n2	rxn2.rtr1.n0
rxn2.rtr1.n5	rxn2.rtr2.n0
rxn2.rtr2.n2	rxn3.rtr1.n0
rxn2.rtr2.n5	rxn3.rtr2.n0
rxn3.rtr1.n4	sepn.sill.n0
rxn3.rtr2.n3	sepn.sill.n1
sepn.sill.n5	sepn.sill.n2
	sepn.sill.n3

event, which do not exist in the original schedule. Therefore, we can define the indicator binary variable $y_{i,j,n}$ as follows:

$$y_{i,j,n} = \begin{cases} 1, & \text{if } Ts^{\text{old}}_{i,j,n} < T^{\text{rush}} \text{ and } Ts_{i,j,n} < T^{\text{rush}} \\ 0, & \text{if } Ts^{\text{old}}_{i,j,n} \geq T^{\text{rush}} \text{ and } Ts_{i,j,n} \geq T^{\text{rush}} \end{cases}$$

which can be mathematically formulated as follows:

$$Ts^{\text{old}}_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \leq T^{\text{rush}} \leq Ts^{\text{old}}_{i,j,n} + Uy_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \qquad (23)$$

$$Ts_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \leq T^{\text{rush}} \leq Ts_{i,j,n} + Uy_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \qquad (24)$$

Constraints (23) and (24) correspond to the definition of the binary variables $y_{i,j,n}$. (23) can be verified as follows: if task i in unit j at event point n starts before the rush order arrives ($Ts^{\text{old}}_{i,j,n} < T^{\text{rush}}$), then $y_{i,j,n}$ must take value 1 as a binary variable because if $y_{i,j,n}$ constraint (23) will become $T^{\text{rush}} \leq Ts^{\text{old}}_{i,j,n}$, which contradicts the fact that $Ts^{\text{old}}_{i,j,n} < T^{\text{rush}}$; however, if $y_{i,j,n} = 1$ constraint (23) takes the form $Ts^{\text{old}}_{i,j,n} + \varepsilon \leq T^{\text{rush}} \leq Ts^{\text{old}}_{i,j,n} + U$, which verifies the assumption ($Ts^{\text{old}}_{i,j,n} < T^{\text{rush}}$) because ε is a small positive number and the inequality on the right-hand side is redundant; similarly, if task i in unit j at event point n starts at or after the rush order arrival time ($Ts^{\text{old}}_{i,j,n} \geq T^{\text{rush}}$), $y_{i,j,n}$ must be 0 because in this case constraint (23) satisfies this assumption, whereas the value of 1 results in a contradictory conclusion ($Ts^{\text{old}}_{i,j,n} < T^{\text{rush}}$). Thus, the value of $y_{i,j,n}$ is defined by constraint (23). Constraint (24) defines the variables $y_{i,j,n}$ for the tasks in reactive schedule in the same way.

Furthermore, the demand constraint should be updated as following to account for the new demand in the rush order:

$$\sum_n d_{s,n} \geq r^{\text{rush}}_s \quad \forall s \in S \qquad (25)$$

where r^{rush}_s corresponds to the updated demand after the rush order arrival.

Thus, the reactive scheduling problem considering rush order uncertainty is formulated with the constraints: (2)–(5), (7)–(14), and (19)–(25). The reactive scheduling objective is selected as minimizing the makespan to fulfill the updated order. The complete problem formulation is given in Appendix A. It should be noticed that this formulation covers any case of a rush order arrival including the time of arrival, new orders, and modification or cancellation of existing orders. Also, it is not restricted by the number of different products in the order.

Machine breakdown

To incorporate machine breakdown within the reactive scheduling formulation, the following rules should be included: if a unit j^* breaks down at time T^{break} and requires repair/maintenance time of T^{maint}, then:

1. All the tasks in $j \in J, j \neq j^*$ should be identified as executed if they start before T^{break}.
2. All the tasks in j^* should be identified as executed if they finish at or before T^{break}.

Note that there are different rules for the breakdown units and for the ones that operates normally. For the unit that is broken, we are enforcing rules on task finishing time but not

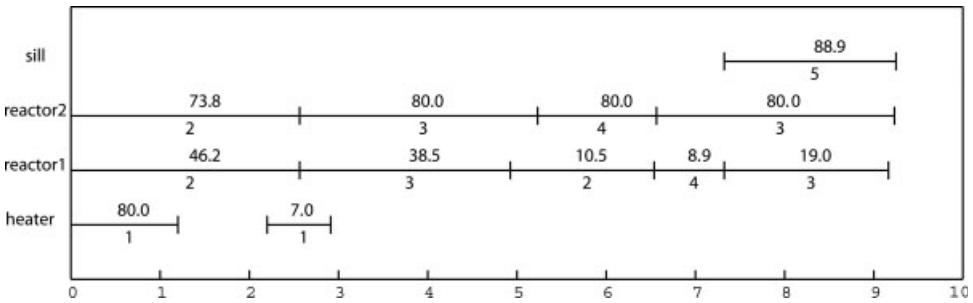

Figure 5. Reactive schedule for example1 with rush order at t = 2.2 h.

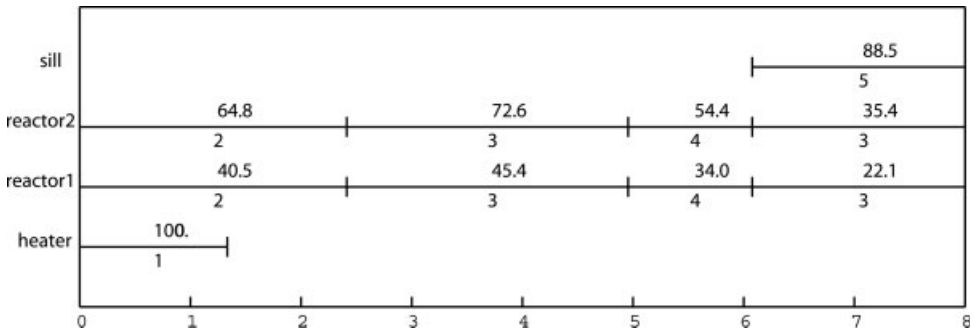

Figure 6. Original schedule of example 1, fixed H = 8 h.

starting time because once the machine is broken, the tasks that have started must be stopped. Therefore, we define the indicator binary variables as follows:

$$y_{i,j,n} = \begin{cases} 1, \text{ if} \begin{cases} Ts^{\text{old}}_{i,j,n} < T^{\text{break}} \text{ and } Ts_{i,j,n} < T^{\text{break}} \\ \quad \text{for} \forall j \in J, j \neq j^*, \text{or} \\ Tf^{\text{old}}_{i,j,n} \leq T^{\text{break}} \text{ and } Ts_{i,j,n} \leq T^{\text{break}} \text{ for} j = j^* \end{cases} \\ 0, \text{ if} \begin{cases} Ts^{\text{old}}_{i,j,n} \geq T^{\text{break}} \text{ and } Ts_{i,j,n} \geq T^{\text{break}} \\ \quad \text{for} \forall j \in J, j \neq j^*, \text{or} \\ Tf^{\text{old}}_{i,j,n} > T^{\text{break}} \text{ and } Ts_{i,j,n} > T^{\text{break}} \text{ for} j = j^* \end{cases} \end{cases}$$

Similar to the rush order case, the rules are implemented in the original schedule to identify the value of $y_{i,j,n}$, and in the reactive schedule to ensure that the complete reschedule does not change the schedule history. The definition of $y_{i,j,n}$ is mathematically equivalent with the following constraints:

$$Ts^{\text{old}}_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \leq T^{\text{break}} \leq Ts^{\text{old}}_{i,j,n} + Uy_{i,j,n}, \quad \forall i \in I, \ \forall j \in J_i, \quad j \neq j^*, \ \forall n \in N \quad (26)$$

$$Tf^{\text{old}}_{i,j^*,n} - U(1 - y_{i,j^*,n}) \leq T^{\text{break}} \leq Tf^{\text{old}}_{i,j^*,n} + Uy_{i,j^*,n} - \varepsilon, \quad \forall i \in I, \ \forall n \in N \quad (27)$$

$$Ts_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \leq T^{\text{break}} \leq Ts_{i,j,n} + Uy_{i,j,n}, \quad \forall i \in I, \ \forall j \in J_i, \quad j \neq j^*, \ \forall n \in N \quad (28)$$

$$Tf_{i,j^*,n} - U(1 - y_{i,j^*,n}) \leq T^{\text{break}} \leq Tf_{i,j^*,n} + Uy_{i,j^*,n} - \varepsilon, \quad \forall i \in I, \ \forall n \in N \quad (29)$$

Constraints (26) are valid for units that do not break down and represent the definition of $y_{i,j,n}$ in the following way: if task i in unit j at event point n starts before the machine breaks down ($Ts^{\text{old}}_{i,j,n} < T^{\text{break}}$), because constraint (26) expresses that $T^{\text{break}} \leq Ts^{\text{old}}_{i,j,n} + Uy_{i,j,n}$, $y_{i,j,n}$ must be 1 as 0 does not satisfy this constraint; similarly, if the task i in unit j at event point n start at or after the machine breakdown time ($Ts^{\text{old}}_{i,j,n} \geq T^{\text{break}}$), because constraint (26) expresses the requirement $Ts^{\text{old}}_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \leq T^{\text{break}}$, $y_{i,j,n}$ must be 0 because 1 will generate contradictory results. Constraints (27) represent the definition of $y_{i,j,n}$ for units that break down and can be verified in similar way. Constraints (28) and (29) are for the tasks in reactive schedule and correspond to the same rules as constraints (26) and (27), respectively and can be verified in the same way.

Furthermore, we should add constraints to change the starting time for tasks that finish after T^{break} in the machine that breaks down:

$$T^{\text{break}} + T^{\text{maint}} \leq Ts_{i,j^*,n} + Uy_{i,j^*,n} \quad \forall i \in I, \quad \forall n \in N \quad (30)$$

Constraint (30) expresses the requirement that if a task has been identified as one that does not finish before breakdown time ($y_{i,j,n} = 0$), it must start after the unit is fixed. For the case that the task finishes before the breakdown occurs ($y_{i,j,n} = 1$), constraint (30) is redundant.

The reactive scheduling formulation incorporating machine breakdown is formulated with the objective of maximizing the profit (or minimizing makespan), using constraints (2)–(14), (19)–(22), and (26)–(30). The complete formulation is given in Appendix B. Note that, this formulation covers all possible machine breakdown events including the breakdown

Table 4. Machine Breakdown Uncertainty for Example 1

	Value	Range
T^{break} (Breakdown time)	θ_1	$1 \leq \theta_1 \leq 7$
T^{maint} (Maintenance time)	θ_2	$0.5 \leq \theta_2 \leq 2.5$

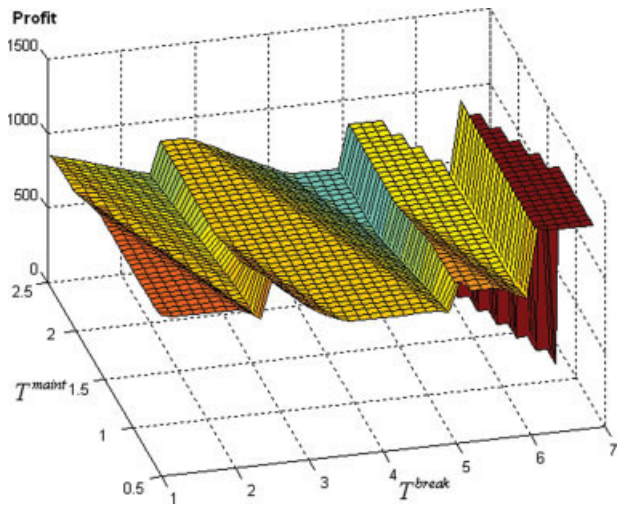

Figure 7. Parametric solution of maximum profit and machine breakdown parameter.

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

Table 5. Parametric Objective for Example 1 with Machine Breakdown

Critical Region	Optimal Profit
1	920.5
2	$2015 - 363.2\theta_1 - 363.2\theta_2$
3	$1674.3 - 240\theta_1 - 240\theta_2$
4	$1403.1 - 150\theta_1 - 150\theta_2$
5	1058.9
6	896.2
7	$1856.2 - 240\theta_1 - 240\theta_2$
8	$2176.2 - 240\theta_1 - 240\theta_2$
9	576.2
10	866.6
11	$2466.6 - 240\theta_1 - 240\theta_2$
12	1008.2
13	$5001 - 706.7\theta_1 - 706.7\theta_2$
14	1356.6
15	$2521.6 - 428.1\theta_1 - 428.1\theta_2$

of any unit at any time during the schedule execution that require any time for repair/maintenance.

Multiparametric Programming

The problem formulations for the reactive scheduling addressing rush order or machine breakdown described in the previous section are represented by MILP problems with uncertain parameters on the right-hand side of the constraints. Thus, it can be exactly addressed using a multiparametric mixed integer linear programming (mpMILP) approach. The details of the approach used in this article can be found in our previous publication,[20] but it is presented here briefly for completeness of the presentation.

For ease in the presentation, the scheduling model can be compactly written as the following MILP problem:

$$\begin{aligned} \min \quad & z = cx \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{P1}$$

where the decision variable vector x is composed by both continuous and binary variables. When uncertainty is incorporated into the scheduling model as with the problem formulations in Appendix A, B, problem (P1) becomes:

$$\begin{aligned} \min \quad & z = cx \\ \text{s.t.} \quad & Ax \geq b + E\theta \\ & x \geq 0 \\ & \theta_j^l \leq \theta_j \leq \theta_j^u, \quad j = 1, .., m \end{aligned} \tag{P2}$$

where θ_j represent uncertain parameter.

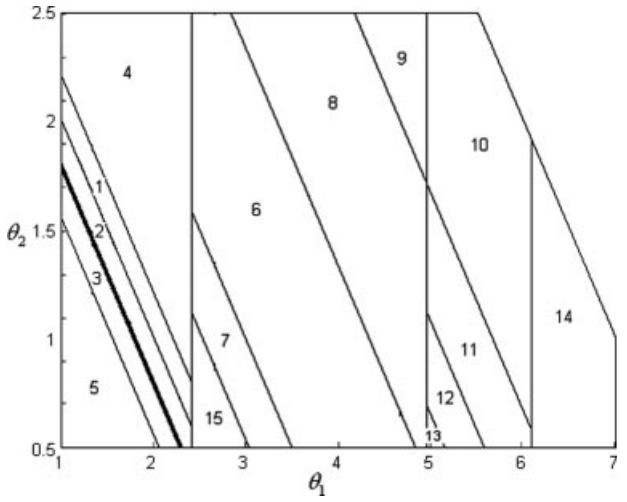

Figure 8. Critical region of the example 1 with machine breakdown.

MILP problem (P2) represents a multiparametric programming problem. The complete solution of a multiparametric programming problem is composed by the complete set of critical regions and optimal objective functions described with respect to uncertain parameters. The critical region is defined as the range of parameter values where the same solution remains optimal. For a mpMILP problem, there is a unique set of integer values and unique parametric objective function of the uncertain parameter in one critical region. The parametric solutions of the continuous variables are also unique parametric function of the uncertain parameter. However, in realistic applications where a large number of continuous variables are involved to avoid storing too much information, we can only store the integer solution in every critical region, and the solution of continuous variable can be retrieved by solving a linear programming problem by fixing the integer solution.

The solution framework is based on the idea of decomposing the original problem into a series of smaller problems. The parametric solution of each subproblem provides the solution around a given parameter value. The framework of determining one critical region involves the following steps:

Step 1. Select a nominal parameter value θ^0 to be studied and solve the deterministic MILP problem by fixing the uncertain parameters at θ^0 to get an integer solution.

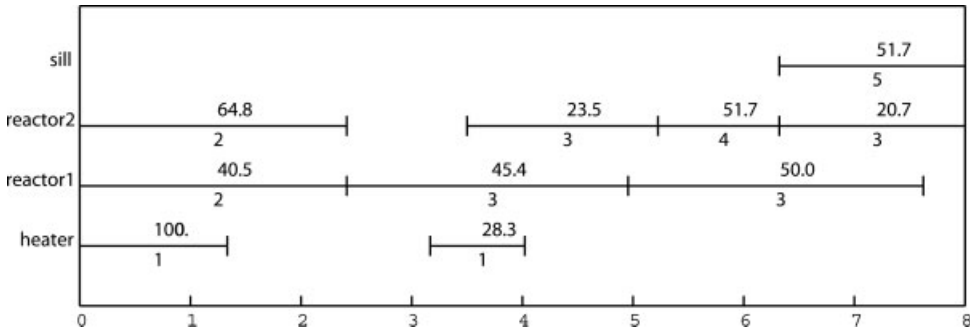

Figure 9. Reactive schedule for reactor 2 breakdown at t = 2.5 h, maintenance time = 1 h.

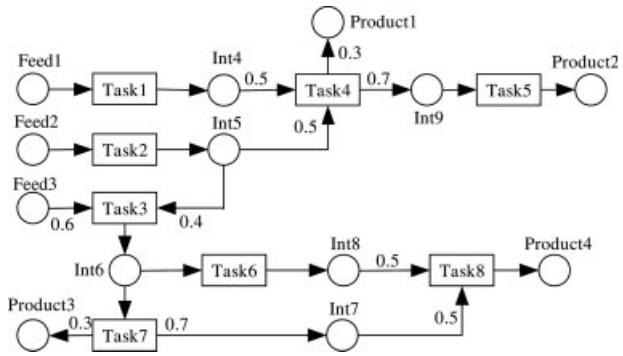

Figure 10. STN representation of example 2.

Step 2. Formulate a mpLP problem by fixing the integer variables at the values obtained from step 1. By solving the mpLP problem, we can get the critical region CR^0 that covers θ^0 and the corresponding optimal objective function z^0. Note that, the complete solution of mpLP for entire uncertain space is not necessary, because we are considering the entire uncertain space by changing the initial point θ^0. The parametric solution is obtained using the following results for a standard linear programming problem $\{\min z = cx|Ax = b(\theta), x \geq 0\}$:

Optimality conditions:

$$\mathrm{A}_B^{-1} b(\theta) \geq 0 \tag{31}$$

$$c_N - c_B \mathrm{A}_B^{-1} \mathrm{A}_N \geq 0 \tag{32}$$

Optimal parametric objective:

$$z^* = c_B \mathrm{A}_B^{-1} b(\theta) \tag{33}$$

Optimal parametric solution:

$$x_B^* = \mathrm{A}_B^{-1} b(\theta) \tag{34}$$

where A_B (c_B) is composed by the columns of A (c) corresponding to the basic variables, and A_N (c_N) is composed by the columns of A (c) corresponding to nonbasic variables. The result of this step is a critical region that covers point θ^0 represented by Eq. 31 and the corresponding optimal parametric objective and parametric solution, which are described by Eqs. 33 and 34, respectively. Because the critical region obtained here is not the real critical region of the original mpMILP problem, the following updating steps are required.

Step 3. Formulate the following MILP problem, which aims at seeking an integer solution with a better objective function in the current critical region:

$$\begin{aligned} \max \quad & \mathrm{err} = z^0 - cx \\ \text{s.t.} \quad & Ax \geq b + E\theta \\ & x \geq 0 \\ & \mathrm{err} \leq \varepsilon \\ & \theta \in CR^0 \end{aligned} \tag{P3}$$

where CR^0 is the current critical region that covers θ^0. Problem (P3) includes all the original constraints of (P2) and the following additional constraints: (i) parametric cut err $\leq \varepsilon$, to seek the integer solution that provides a better objective function, where ε is a small positive number to ensure that only a better but not the best objective is found; (ii) a restriction of the solution space to current critical region CR^0. If the optimal objective err* ≤ 0, the solution procedure stops (no better solution exists in the current region); otherwise, the integer solution and point θ^* are stored and the procedure continues to step 4.

Step 4. Formulate a new mpLP problem by fixing the integer variables at the solution obtained from step 3. Follow the same procedure in step 2, solve mpLP problem to get the optimal objective function z^* and the critical region CR^* that covers θ^*.

Step 5. Update the critical region that covers θ^0 through excluding operation: $CR^0 = CR^0 / CR^{\mathrm{EX}}$, where CR^{EX} represents the region where the new objective z^* is better than z^0 (for a minimization problem, better means smaller). The excluded part CR^{EX} can be determined by comparing the two objective functions in the intersection of the two critical regions: $CR^0 \cap CR^*$ using the following redundancy test problem:

$$\begin{aligned} \max \quad & \mathrm{err} = z^* - z^0 \\ \text{s.t.} \quad & \theta \in CR^0 \cap CR^* \end{aligned} \tag{P4}$$

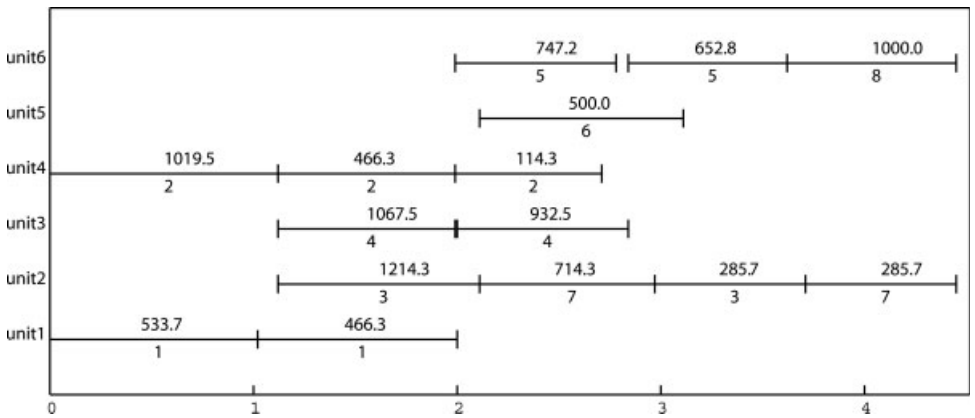

Figure 11. Original schedule for example 2.

Table 6. Rush Order Uncertainty for Example 2

	Value	Range
r_{P1}^{rush} (New order of P1)	θ_1	$500 \le P_1 \le 800$
r_{P2}^{rush} (New order of P2)	θ_2	$1200 \le P_2 \le 1600$
T^{rush} (Order arrival time)	θ_3	$1 \le \theta_2 \le 4$

If err* ≤ 0, it means that $z^* \le z^0$ is redundant in $CR^0 \cap CR^*$ (z^* is always better for a minimization problem) and $CR^{\text{EX}} = CR^0 \cap CR^*$, whereas if err* < 0, it means that $z^* \le z^0$ is not redundant and $CR^{\text{EX}} = CR^0 \cap CR^* \cap \{\theta | z^* \le z^0\}$. Thus, the region CR^{EX} is excluded from the original critical region CR^0 that covers θ^0. The exclusion is implemented by testing and selecting one constraint in CR^{EX} that θ^0 violates and adding the corresponding constraint into CR^0. For example, if the selected constraint in CR^{EX} has the form of $\sum_i b_i\theta_i + c \le 0$, the constraint added into CR^0 has the following form: $-\sum_i b_i\theta_i - c \le 0$. After the critical region CR^0 is updated, go back to step 3.

The proposed framework only determines one critical region around one point. To get a complete map of the original problem, the rest of the parameter space should be explored. To achieve this, a uniformly distributed testing method is used. This involves the generation of a number of uniformly distributed points to cover the uncertainty space which are then tested to determine whether they are already covered by the identified critical regions, if not, the proposed procedure can be used. This method is effective because it only requires function value evaluations. Also, note that the random selection of samples does not affect the correctness of the resulted parametric solution as they are derived from the optimality condition of LP problem. The number of points selected is based on the dimension of the uncertain space and the range: a higher dimension space and a relative bigger range needs more points to be tested so that the space is covered as much as possible. Also, note that the number of points used to derive critical regions is less than the number of testing points because once a point is covered it is not been reconsidered. The final solution of the problem might involve overlapped critical regions because they belong to a larger nonconvex critical region.[20]

As shown in the reactive scheduling formulations (Appendix A, and B), for the reactive scheduling with rush order, the uncertain parameters are the updated order amount r_s^{rush} and order arrival time T^{rush}, whereas for the case of machine breakdown, the uncertain parameters consist of machine breakdown time T^{break} and machine maintenance time T^{maint}. Both cases correspond to mpMILP problems with RHS uncertainty. The result of the multiparametric programming provides direct mapping information between the uncertain event and the optimal reactive scheduling solution. Thus, the reactive schedule can be obtained directly from the parametric solution of the two different formulations. Furthermore, sensitivity information can be easily obtained given that the parametric solution provides the complete information of an optimization problem under uncertainty.

Although the proposed reactive scheduling formulation does not consider the case where several consecutive uncertain events happen (e.g., a rush order arrival after another or a rush order arrival and machine breakdown after another rush order, etc.), the proposed formulation can be applied to solve this type of problems in the following way: the parametric problem considering an upcoming rush order or machine breakdown event is solved before the schedule execution as proposed in the article. Once an uncertain event occurs, the parametric solution is applied and a new parametric problem is solved based on the updated schedule. This process can be repeated to accommodate multiple unexpected events.

We can also formulate the smooth operation requirement into the reactive scheduling formulation through the use of a penalty term in the objective function. Details can be referred to our previous publication.[17] This, however, does not affect the applicability of the proposed multiparametric algorithm because mpMILP problem can still be formulated.

In the next section, several examples are solved to illustrate the application of the proposed reactive scheduling solution procedure.

Examples

Two examples are solved to study the application of the proposed method on reactive scheduling. The problems are solved with CPLEX 10.1 as the LP/MILP solver in a Pentium PC (2.8 GHz, 1 G RAM).

Example 1

Example 1 involves the production of two products using three raw materials. The state-task-network (STN) representation of this example is shown in Figure 1, and the problem data can be found in the Appendix C.

Rush Order. To study the reactive scheduling problem considering an unexpected rush order, we assume the original deterministic schedule is generated first to satisfy the nominal demand of products P1 and P2, which are both set as 80 units. The deterministic scheduling problem consisting of constraints (1)–(14) is solved with the objective function of minimizing the makespan. The resulted schedule is shown in Figure 2.

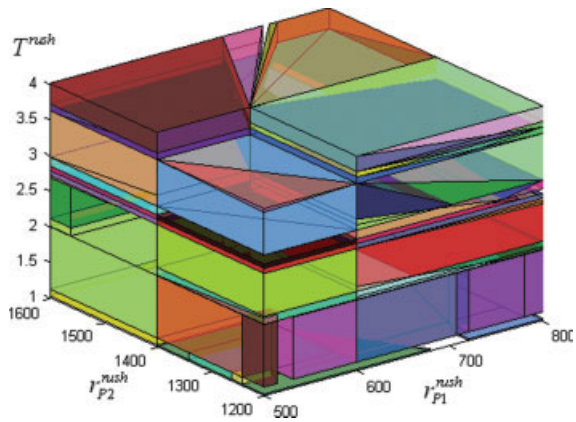

Figure 12. Critical region of example for rush order uncertainty.

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

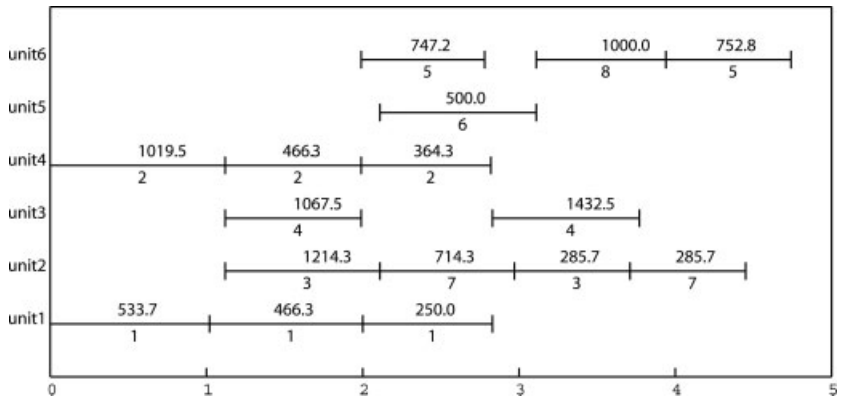

Figure 13. Reactive schedule for rush order at $t = 1.5$ h.

A rush order of product P1 is investigated with an uncertain demand and an uncertain order arrival time as described in Table 1.

For the reactive scheduling formulation with rush order uncertainty, because the demand of product is increased to satisfy the additional order, the number of event points used for the original deterministic scheduling might not be enough and can cause the problem to be infeasible. Therefore, we need to find the appropriate number of event points for reactive scheduling formulation with the maximum demand using the deterministic formulation and then fix the event point number and solve the parametric problem. For this example, the original number of event points for deterministic formulation is 7, which is the number required for the maximum demand of product P1 (90 units), so the number of event points are fixed at 7 for the reactive scheduling formulation during the multiparametric programming solution process. Then, the corresponding multiparametric programming problem is solved and the parametric results are obtained. Figure 3 illustrates the exact relationship between the uncertain parameter and the optimal makespan. Figure 4 shows the critical regions of the solution, and Table 2 shows the detail parametric objective in different critical regions.

As shown in Figures 3 and 4, the parametric result gives the exact relationship between the uncertain parameter and the scheduling solution, thus the reactive schedule can be obtained explicitly from the parametric solution once the rush order arrives. For example, if a rush order arrives at $t = 2.2$ h ($\theta_2 = 2.2$) with 7 units additional demand of P1, thus the new demand of P1 is 87 units ($\theta_1 = 87$) and the reactive schedule can be obtained from the parametric solution for $(\theta_1, \theta_2) = (87, 2.2)$, which corresponds to critical region 10 with a parametric objective of $5.769 + 0.04\theta_1$ and the integer solution is shown in Table 3. Thus, the optimal makespan can be evaluated by $5.769 + 0.04 \times 87 = 9.249$ and the corresponding schedule is obtained as shown in Figure 5.

Machine Breakdown. To study the case when machine breakdown occurs, we consider the problem of maximizing the profit in a given makespan (8 h), and there is no requirement on the demand of product P1 and P2. Note that, these assumptions are not necessary and any other optimization objective and demand constraints can be used. The original schedule is obtained as shown in Figure 6, which results in a maximum profit of 1498.2. Therefore, we assume that reactor 2 breaks down during the scheduling execution period and the breakdown time and the time of maintenance are assumed uncertain as shown in Table 4.

To illustrate the parametric solution, the parametric objective is shown in Figure 7 and the details of the solution are given in Table 5. The critical regions of the solution are shown in Figure 8. Computational data for the solution process is shown in Table 9. From the parametric solution, it can be observed that the final optimal profit will decrease with the increase of the maintenance time if the same unit breaks down at the same time. Also, it should be noticed that the problem will become infeasible if the machine breakdown time and the maintenance time increase beyond certain limit.

Once the parametric solution is obtained, the reactive schedule can be directly determined once the event occurs. For example, a reactive schedule for the reactor 2 breakdown at $t = 2.5$ h with 1 h maintenance can be obtained by mapping the parameter value $(\theta_1, \theta_2) = (2.5, 1)$ in the critical region 15. The corresponding reactive schedule is shown in Figure 9. The resulted profit is $2521.6 - 428.1 \times 2.5 - 428.1 \times 1 = 1023.3$, which corresponds to a big decrease compared to the original profit (1498.2) because of the machine breakdown.

Example 2

In example 2, four products are produced through eight tasks from three feeds, and there are nine intermediates in the system. In all, six different units are required for the whole process. The STN representation of this process is shown in Figure 10, and the problem data can be found in Appendix C. Through this example, we are studying the application of the proposed method on consecutive uncertainties for reactive scheduling. Specifically, we assume that the first disruptive event is a rush orders for products P1 and P2,

Table 7. Machine Breakdown Uncertainty for Example 2

	Value	Range
T^{break} (Breakdown time)	θ_1	$3 \le \theta_1 \le 5$
T^{maint} (Maintenance time)	θ_2	$0.5 \le \theta_2 \le 2.5$

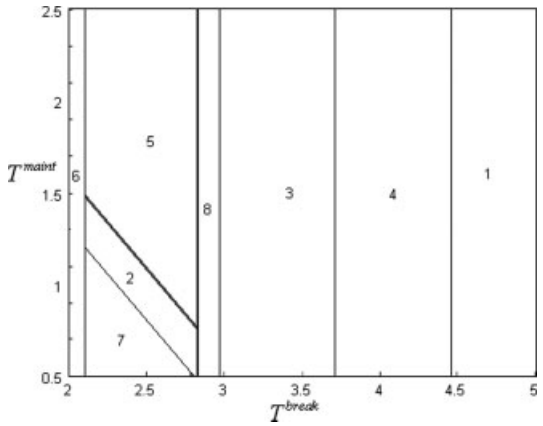

Figure 14. Critical region of example 2 with machine breakdown.

and the second disruptive event is that unit 2 breaks down and needs maintenance.

First, we solve the deterministic scheduling problem with the objective of minimizing makespan to satisfy the nominal demand of products as: P1 = 600, P2 = 1400, P3 = 300, P4 = 1000. The original deterministic schedule is solved with seven event points and the schedule is shown in Figure 11, which has a minimum makespan of 4.45 h.

To address the upcoming rush order uncertainty, we can start solving the parametric problem for rush order soon after we get the original schedule. Using the maximum demand of the new order, nine event points are identified to be necessary for the reactive scheduling formulation. The uncertain event is described as shown in Table 6. Then, the multiparametric programming problem is solved with the critical regions illustrated in Figure 12.

Having obtained the parametric solution, we can generate a reactive schedule as soon as the rush order arrives. For example, if the demand of product P1 increases to 750 (θ_1 = 750), and the demand of P2 increases to 1500 (θ_2 = 1500) at time t = 1.5 h (θ_3 = 1.5), the parametric solution for (θ_1, θ_2, θ_3) = (750,1500,1.5) can be found directly from the parametric result. Figure 13 illustrates the new schedule which has a makespan of 4.74 h.

Table 8. Parametric Objective for Example 2 with Machine Breakdown

Critical Region	Makespan H(h)
1	4.74
2	$3.74 + 0.062\theta_1 + 0.062\theta_2$
3	$1.49 + \theta_1 + \theta_2$
4	$0.74 + \theta_1 + \theta_2$
5	$2.37 + \theta_1 + \theta_2$
6	$4.102 + 0.0656\theta_1$
7,8	$2.48 + \theta_1 + \theta_2$

Soon after the reactive schedule is executed, a new parametric reactive scheduling problem is solved to deal with future unexpected events. The machine breakdown uncertainty considered here is defined in Table 7. The critical regions of the parametric solution are shown in Figure 14, whereas the detail parametric objectives are shown in Table 8.

After we obtain the parametric solution, we can address the upcoming machine breakdown. For example, if unit 2 breaks down at T^{break} = 3 h, and requires T^{maint} = 1.5 h, it corresponds to (θ_1, θ_2) = (3, 1.5), so the reactive schedule can be obtained from the parametric result and it is shown in Figure 15.

Following this dynamic way of addressing uncertainty, multiple disruptive uncertainties in the scheduling process can be addressed. The only requirement is that upon the arrival of a new disruptive event, the corresponding parametric solution that covers this uncertain event has been retrieved. In this example, the first parametric problem is solved in 2300 CPU s, and the second parametric problem is solved in 1442 CPU s. In both cases, we test 5000 points uniformly distributed in the uncertain space and the parametric solution can cover all the given uncertain space (Figures 12 and 14) except the infeasible operation areas. Detail computational statistics are given in Table 9. Furthermore, during the process of solving the parametric programming problem, the uncertain space that represents the near future can be solved

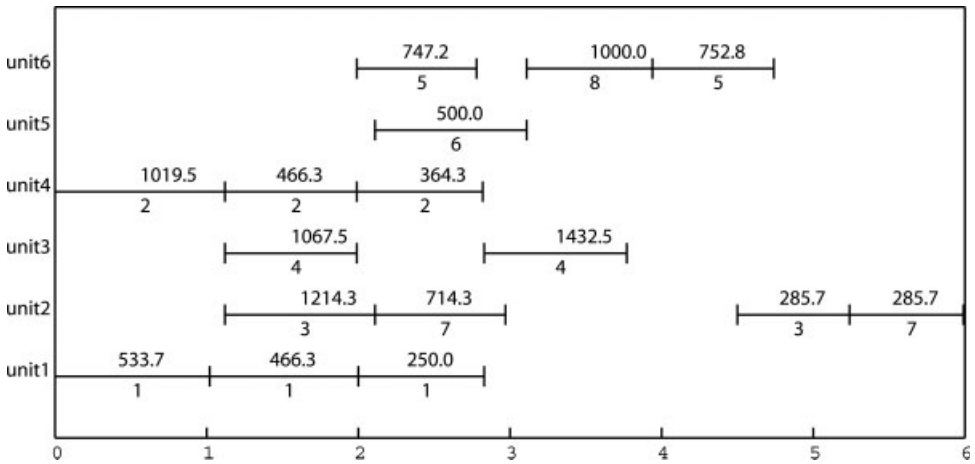

Figure 15. Reactive schedule for unit 2 breakdown at t = 3 h, maintenance time = 1.5 h.

Table 9. Computational Statistics for the Examples

	Example 1		Example 2	
	Rush Order	Machine Breakdown	Rush Order	Machine Breakdown
Reactive scheduling model				
Constraints	1360	1596	2278	2358
Continuous	576	673	1316	1370
Binaries	280	320	864	864
Number of testing points	5000	5000	5000	5000
Number of critical regions*	35	49	89	18
Average iterations per point	3	3	3	2
Average CPU time per point (s)**	7	5	6	5

*Equal to the number of points used to apply the multiparametric programming, because some of these critical regions might belong to a larger nonconvex critical region, the final critical region illustrated in the article is postprocessed result after union operation.
**The average time is for multiparametric programming solution process for a point.

at the beginning, so that the earlier disruptive events can be covered by the parametric solution.

Summary

A new methodology for efficient reactive scheduling is proposed in this article. Different to any existing method, this article provides a direct mapping approach to generate the reactive schedule with the parametric solution. It greatly improves the efficiency of reactive scheduling because the reactive schedule is obtained by checking from a set of parametric solutions, which is solved ahead of time but not solve a rescheduling problem after the uncertainty occurs. The proposed methodology is designed to address single disruptive event. However, consecutive uncertainties can be addressed through the repetitive application of the method. It is worthwhile to note here that the number of critical regions of mpMILP problem increases with the size of the uncertain space (number of the uncertain parameters), so complete coverage of the uncertain space needs considerable computational effort. However, the parametric solution generated using the proposed method provides a way to derive the possible reactive decision with existing computational ability before the uncertain event, which make it possible to save time in making reactive decision. Once the realized uncertainty is not covered by the current solution, the reactive schedule can be directly solved through the developed reactive scheduling formulation. Further improvements on the proposed method lie on developing parallel algorithm to solve the multiparametric programming problem to further save the computation time.

Acknowledgments

The authors gratefully acknowledge financial support from the National Science Foundation under Grants CTS 0625515 and 0224745.

Notation

$i \in I$ = task index and sets
I_s = tasks which produce or consume state s
I_j = tasks which can be performed in unit j
$j \in J$ = unit index and sets
J_i = units which are suitable for performing task i
$n \in N$ = event points representing the beginning of a task
$s \in S$ = state index and sets
$price_s$ = price of state s
$d_{s,n}$ = amount of state s delivered to the market at event point n
$wv_{i,j,n}$ = binary, whether or not task i in unit j start at event point n
$st_{s,n}$ = continuous, amount of state s at event point n
$\rho^P_{s,i}$, $\rho^C_{s,i}$ = proportion of state s produced, consumed by task i, respectively
$b_{i,j,n}$ = continuous, batch size of task i in unit j at event point n
$st_s^{\max}$ = available maximum storage capacity for state s
$v_{i,j}^{\min}$, $v_{i,j}^{\max}$ = minimum amount, maximum capacity of unit j when processing task i
r_s = market demand for state s at the end of the time horizon
$Tf_{i,j,n}$ = continuous, completion time of task i in unit j that starts at event point n
$Ts_{i,j,n}$ = continuous, time at which task i starts in unit j at event point n
$\alpha_{i,j}$, $\beta_{i,j}$ = constant, variable term of processing time of task i in unit j respectively
H = scheduling time horizon
$y_{i,j,n}$ = binary, whether task i in unit j starts at event point n has been executed
U = upper bound of scheduling time horizon
ε = small positive constant
T^{rush} = rush order arrival time
r_s^{rush} = the demand of state s in the rush order
T^{break} = the machine breakdown time
T^{maint} = the maintenance time for the broken unit
$b_{i,j}^{\text{UB}}$ = upper bound of the batch size
old = superscript that represents the solution of original schedules

Literature Cited

1. 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.
2. Bonfill A, Espuna A, Puigjaner L. Addressing robustness in scheduling batch processes with uncertain operation times. *Ind Eng Chem Res*. 2005;44:1524–1534.
3. Sand G, Engell S. Modeling and solving real-time scheduling problems by stochastic integer programming. *Comput Chem Eng*. 2004;28:1087–1103.
4. 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.
5. Vin JP, Ierapetritou MG. Robust short-term scheduling of multiproduct batch plants under demand uncertainty. *Ind Eng Chem Res*. 2001;40:4543–4554.

6. Jia Z, Ierapetritou MG. Generate Pareto optimal solutions of scheduling problems using normal boundary intersection technique. *Comput Chem Eng*. 2006;31:268–280.
7. Cott BJ, Macchietto S. Minimizing the effects of batch process variability using online schedule modification. *Comput Chem Eng*. 1989;13:105–113.
8. Kanakamedala KB, Reklaitis GV, Venkatasubramanian V. Reactive schedule modification in multipurpose batch chemical plants. *Ind Eng Chem Res*. 1994;30:77–90.
9. Huercio A, Espuña A, Puigjaner L. Incorporating on-line scheduling strategies in integrated batch production control. *Comput Chem Eng*. 1995;19;S609–S615.
10. Sanmarti E, Huercio A, Espuna 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.
11. Rodrigues MTM, Gimeno L, Passos CAS, Campos MD. Reactive scheduling approach for multipurpose chemical batch plants. *Comput Chem Eng*. 1996;20;S1215–S1220.
12. Honkomp SJ, Mockus L, Reklaitis GV. A framework for schedule evaluation with processing uncertainty. *Comput Chem Eng*. 1999;23:595–609.
13. Roslöf J, Harjunkoski I, Björkqvist J, Karlsson S, Westerlund T. An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. *Comput Chem Eng*. 2001;25:821–828.
14. Ruiz D, Cantón J, Nougués JM, Espuña A, Puigjaner L. On-line fault diagnosis system support for reactive scheduling in multipurpose batch chemical plants. *Comput Chem Eng*. 2001;25:829–837.
15. Méndez CA, Cerdá J. Dynamic scheduling in multiproduct batch plants. *Comput Chem Eng*. 2003;27:1247–1259.
16. Méndez CA, Cerdá J. An MILP framework for batch reactive scheduling with limited discrete resources. *Comput Chem Eng*. 2004;28:1059–1068.
17. Vin JP, Ierapetritou MG. A new approach for efficient rescheduling of multiproduct batch plants. *Ind Eng Chem Res*. 2000;39:4228–4238.
18. Janak SL, Floudas CA. Production scheduling of a large-scale industrial batch plant. II. Reactive scheduling. *Ind Eng Chem Res*. 2006; 45:8253–8269.
19. Ryu JH, Pistikopoulos EN. A novel approach to scheduling of zero-wait batch processes under processing time variations. *Comput Chem Eng*. 2007;31:101–106.
20. Li Z, Ierapetritou MG. Process scheduling under uncertainty using multiparametric programming. *AIChE J*. 2007;53:3183–3203.
21. Floudas CA, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng*. 2004;28:2109–2129.
22. Méndez CA, Cerdá J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput Chem Eng*. 2006;30:913–946.
23. Maravelias CT, Grossmann IE. On the relation of continuous- and discrete-time state-task network formulations. *AIChE J*. 2006;52: 843–849.
24. Ierapetritou MG, Floudas CA. Effective continuous-time formulation for short-term scheduling. I. Multipurpose batch processes. *Ind Eng Chem Res*. 1998;37:4341–4359.
25. Kondili E, Pantelides CC, Sargent RH. A general algorithm for short-term scheduling of batch operations. I. MILP formulation. *Comput Chem Eng*. 1993;17:211–227.

Appendix A: Reactive Scheduling Formulation for Rush Order Uncertainty

$$\min H \tag{A.1}$$

s.t.

$$\sum_{i\in I_j} wv_{i,j,n} \le 1 \quad \forall i \in I \tag{A.2}$$

$$st_{s,n} = st_{s,n-1} - d_{s,n} - \sum_{i\in I_s}\rho^P_{s,i}\sum_{j\in J_i} b_{i,j,n} + \sum_{i\in I_s}\rho^c_{s,i}\sum_{j\in J_i} b_{i,j,n-1} \quad \forall s \in S,\ \forall n \in N \tag{A.3}$$

$$st_{s,n} \le st^{\max}_s \quad \forall s \in S,\ \forall n \in N \tag{A.4}$$

$$v^{\min}_{i,j} wv_{i,j,n} \le b_{i,j,n} \le v^{\max}_{i,j} wv_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.5}$$

$$\sum_n d_{s,n} \ge r^{\text{rush}}_s \quad \forall s \in S \tag{A.6}$$

$$Tf_{i,j,n} = Ts_{i,j,n} + \alpha_{i,j} wv_{i,j,n} + \beta_{i,j} b_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.7}$$

$$Ts_{i,j,n+1} \ge Tf_{i,j,n} - U(1 - wv_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.8}$$

$$Ts_{i,j,n+1} \ge Tf_{i',j,n} - U(1 - wv_{i',j,n}) \quad \forall i, i' \in I_j,\ \forall j \in J,\ \forall n \in N \tag{A.9}$$

$$Ts_{i,j,n+1} \ge Tf_{i',j',n} - U(1 - wv_{i',j',n}) \quad \forall i, i' \in I_j,\ i \ne i',\ \forall j, j' \in J,\ \forall n \in N \tag{A.10}$$

$$Ts_{i,j,n+1} \ge Ts_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.11}$$

$$Tf_{i,j,n+1} \ge Tf_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.12}$$

$$Ts_{i,j,n} \le H \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.13}$$

$$Tf_{i,j,n} \le H \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.14}$$

$$wv^{\text{old}}_{i,j,n} - (1 - y_{i,j,n}) \le wv_{i,j,n} \le wv^{\text{old}}_{i,j,n} + (1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.15}$$

$$b^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - b^{\text{UB}}_{i,j}(1 - y_{i,j,n}) \le b_{i,j,n} \le b^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + b^{\text{UB}}_{i,j}(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.16}$$

$$Ts^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - U(1 - y_{i,j,n}) \le Ts_{i,j,n} \le Ts^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + U(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.17}$$

$$Tf^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - U(1 - y_{i,j,n}) \le Tf_{i,j,n} \le Tf^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + U(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.18}$$

$$Ts^{\text{old}}_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \le T^{\text{rush}} \le Ts^{\text{old}}_{i,j,n} + U y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.19}$$

$$Ts_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \le T^{\text{rush}} \le Ts_{i,j,n} + U y_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \tag{A.20}$$

Appendix B: Reactive Scheduling Formulation for Machine Breakdown Uncertainty

$$\max \sum_{s,n} price_s d_{s,n} \quad \text{or} \quad \min H \tag{B.1}$$

s.t.

$$\sum_{i\in I_j} wv_{i,j,n} \le 1 \quad \forall i \in I \tag{B.2}$$

$$st_{s,n} = st_{s,n-1} - d_{s,n} - \sum_{i\in I_s} \rho^P_{s,i} \sum_{j\in J_i} b_{i,j,n} + \sum_{i\in I_s} \rho^c_{s,i} \sum_{j\in J_i} b_{i,j,n-1} \quad \forall s \in S,\ \forall n \in N \quad \text{(B.3)}$$

$$st_{s,n} \le st^{\max}_s \quad \forall s \in S,\ \forall n \in N \quad \text{(B.4)}$$

$$v^{\min}_{i,j} wv_{i,j,n} \le b_{i,j,n} \le v^{\max}_{i,j} wv_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.5)}$$

$$\sum_n d_{s,n} \ge r_s \quad \forall s \in S \quad \text{(B.6)}$$

$$Tf_{i,j,n} = Ts_{i,j,n} + \alpha_{i,j} wv_{i,j,n} + \beta_{i,j} b_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.7)}$$

$$Ts_{i,j,n+1} \ge Tf_{i,j,n} - U(1 - wv_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.8)}$$

$$Ts_{i,j,n+1} \ge Tf_{i',j,n} - U(1 - wv_{i',j,n}) \quad \forall i, i' \in I_j,\ \forall j \in J,\ \forall n \in N \quad \text{(B.9)}$$

$$Ts_{i,j,n+1} \ge Tf_{i',j',n} - U(1 - wv_{i',j',n}) \quad \forall i, i' \in I_j,\ i \ne i',\ \forall j, j' \in J,\ \forall n \in N \quad \text{(B.10)}$$

$$Ts_{i,j,n+1} \ge Ts_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.11)}$$

$$Tf_{i,j,n+1} \ge Tf_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.12)}$$

$$Ts_{i,j,n} \le H \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.13)}$$

$$Tf_{i,j,n} \le H \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.14)}$$

$$wv^{\text{old}}_{i,j,n} - (1 - y_{i,j,n}) \le wv_{i,j,n} \le wv^{\text{old}}_{i,j,n} + (1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.15)}$$

$$b^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - b^{\text{UB}}_{i,j}(1 - y_{i,j,n}) \le b_{i,j,n} \le b^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + b^{\text{UB}}_{i,j}(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.16)}$$

$$Ts^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - U(1 - y_{i,j,n}) \le Ts_{i,j,n} \le Ts^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + U(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.17)}$$

$$Tf^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} - U(1 - y_{i,j,n}) \le Tf_{i,j,n} \le Tf^{\text{old}}_{i,j,n} wv^{\text{old}}_{i,j,n} + U(1 - y_{i,j,n}) \quad \forall i \in I,\ \forall j \in J_i,\ \forall n \in N \quad \text{(B.18)}$$

$$Ts^{\text{old}}_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \le T^{\text{break}} \le Ts^{\text{old}}_{i,j,n} + U y_{i,j,n} \quad \forall i \in I,\ \forall j \in J_i, j \ne j^*,\ \forall n \in N \quad \text{(B.19)}$$

$$Tf^{\text{old}}_{i,j^*,n} - U(1 - y_{i,j^*,n}) \le T^{\text{break}} \le Tf^{\text{old}}_{i,j^*,n} + U y_{i,j^*,n} - \varepsilon \quad \forall i \in I,\ \forall n \in N \quad \text{(B.20)}$$

$$Ts_{i,j,n} + \varepsilon - U(1 - y_{i,j,n}) \le T^{\text{break}} \le Ts_{i,j,n} + U y_{i,j,n}, \quad \forall i \in I,\ \forall j \in J_i, j \ne j^*,\ \forall n \in N \quad \text{(B.21)}$$

$$Tf_{i,j^*,n} - U(1 - y_{i,j^*,n}) \le T^{\text{break}} \le Tf_{i,j^*,n} + U y_{i,j^*,n} - \varepsilon, \quad \forall i \in I,\ \forall n \in N \quad \text{(B.22)}$$

$$T^{\text{break}} + T^{\text{maint}} \le Ts_{i,j^*,n} + U y_{i,j^*,n} \quad \forall i \in I,\ \forall n \in N \quad \text{(B.23)}$$

Appendix C: Process Data for the Examples

Table C1. Process Data for Example 1

Unit	Capacity	Suitability	Processing Time
Heater	100	Heating	10
Reactor 1	50	Reaction 1, 2, 3	2.0, 2.0, 1.0
Reactor 2	80	Reaction 1, 2, 3	2.0, 2.0, 1.0
Sill	200	Separation	1 for product 2, 2 for IntAB

State	Storage Capacity	Initial Amount	Prices
Feed A	Unlimited	Unlimited	0
Feed B	Unlimited	Unlimited	0
Feed C	Unlimited	Unlimited	0
Hot A	100	0.0	0
IntAB	200	0.0	0
IntBC	150	0.0	0
impure	200	0.0	0
Product 1	Unlimited	0.0	10
Product 2	Unlimited	0.0	10

Table C2. Process Data for Example 2

Unit	Capacity	Suitability	Processing Time
Unit1	1000	Task 1	1
Unit2	2500	Task 3,7	1
Unit3	3500	Task 4	1
Unit4	1500	Task 2	1
Unit5	1000	Task 6	1
Unit6	4000	Task 5, 8	1

State	Storage Capacity	Initial Amount	
Feed 1, 2, 3	Unlimited	0.0	0
Int4	1000	0.0	0
Int5	1000	0.0	0
Int6	1500	0.0	0
Int7	2000	0.0	0
Int8	0	0.0	0
Int9	3000	0.0	0
Products 1, 2, 3, 4	Unlimited	0.0	18, 19, 20, 21

Manuscript received Feb. 25, 2008, and revision received May 28, 2008.