Efficient Scheduling Method of Complex Batch Processes with General Network Structure via Agent-Based Modeling

Yunfei Chu
Dept. of Chemical and Biological Engineering, Northwestern University, Evanston, IL 60208, USA

John M. Wassick
The Dow Chemical Company, Midland, MI 48674

Fengqi You
Dept. of Chemical and Biological Engineering, Northwestern University, Evanston, IL 60208, USA

DOI 10.1002/aic.14101
Published online April 15, 2013 in Wiley Online Library (wileyonlinelibrary.com)

A novel efficient agent-based method for scheduling network batch processes in the process industry is proposed. The agent-based model is based on the resource-task network. To overcome the drawback of localized solutions found in conventional agent-based methods, a new scheduling algorithm is proposed. The algorithm predicts the objective function value by simulating another cloned agent-based model. Global information is obtained, and the solution quality is improved. The solution quality of this approach is validated by detailed comparisons with the mixed-integer programming (MIP) methods. A solution close to the optimal one can be found by the agent-based method with a much shorter computational time than the MIP methods. As a scheduling problem becomes increasingly complicated with increased scale, more specifications, and uncertainties, the advantages of the agent-based method become more evident. The proposed method is applied to simulated industrial problems where the MIP methods require excessive computational resources. © 2013 American Institute of Chemical Engineers *AIChE J*, 59: 2884–2906, 2013
Keywords: agent-based modeling, efficient scheduling, batch processes, general network structure, mixed-integer programming

Introduction

Scheduling is crucial in the operations of batch processes, and it has a significant impact on production efficiency, and, therefore, profitability. Thus, extensive research efforts in both academia and industry have been pursued to develop different techniques to solve various scheduling problems over the last decade.[1–5]

Mathematical programming methods provide a systematic approach to solve scheduling problems, and a great variety of scheduling methods based on mixed-integer programming (MIP) have been developed.[6–14] The scheduling decisions are determined by solving the MIP problems based on the discrete-time formulation or the continuous-time formulation.[3,15,16] The main advantage of the MIP-based methods is that the optimality of the solutions can be guaranteed. However, the computational complexity of the MIP problem has slowed their adoption in complex scheduling problems, especially for scheduling processes with many types of uncertainties.

A real-world manufacturing process is subject to various uncertainties,[17,18] for example, breakdown of equipment units, variation in processing times, or change in order demands. Due to uncertainties, the sequence of production operations will soon deviate from the original offline schedule. Online rescheduling according to updated information is necessary. Unlike offline scheduling, online rescheduling has a much more stringent requirement for computational efficiency, because the scheduling decisions must be determined in a short time frame. Therefore, a fast but reliable method is the key to online rescheduling.

To circumvent the computational complexity in traditional MIP-based methods, many alternatives have been presented.[19–26] One that receives wide attention is the agent-based modeling and simulation.[27,28] The agent-based approach can provide an efficient solution to the scheduling problem, and it can tackle uncertainties in a real process. As a promising alternative for solving scheduling problems, a great number of agent-based techniques have been presented (see examples[29–38]). Correspondingly, various toolkits and frameworks have been developed to facilitate the software development and administration of agent-based applications.[39,40]

However, most agent-based scheduling methods concentrate on discrete manufacturing and few on batch manufacturing. Generally, batch scheduling problems have more complicated structures than scheduling problems in discrete manufacturing. A principal difference is that material splitting and mixing are allowed in batch manufacturing so that a

Correspondence concerning this article should be addressed to F. You at you@northwestern.edu.

batch scheduling problem consists of more constraints such as maintaining a material balance, enforcing equipment and storage capacities, and various storage policies.[41] Because of the general network structure, the product routing paradigm for discrete manufacturing is inadequate to represent a batch process. More appropriate representations are the state-task network (STN),[9] or equivalently, the resource-task network (RTN).[11] The complexity of the batch scheduling problem prohibits the application of existing agent-based methods, which have been developed for scheduling problems in discrete manufacturing. A new agent-based model is required for scheduling a batch process, including new types of agents and new logic rules that drive the agent-based system under all constraints present in the process.

Another important issue, which is often neglected in exiting agent-based methods, is the solution quality. An agent-based method typically uses local information to make scheduling decisions. Global information, such as the performance of the schedule, quantified in an objective function value is difficult to obtain. Consequently, the solution quality is not guaranteed as measured by an objective function value. This problem becomes more severe for the batch scheduling problem. Due to the complicated problem structure, it is difficult to find a direct relationship between a local scheduling criterion and the objective function. To validate the performance of an agent-based method, comparisons with MIP-based methods have been used in this study. Furthermore, as the agent-based methods and the MIP-based method have their own advantages and disadvantages, it is valuable to know which method is more suitable for a specific problem. A comparison of the two types of methods can provide such a guideline for selecting an appropriate method.

The objective of this work is to propose a novel agent-based scheduling method applicable to network batch processes. The novel features of the proposed agent-based method include

1. New agent model based on the general RTN representation, which is able to model various specifications of batch scheduling problems, including material splitting and mixing, equipment and storage capacities, task changeover times, batch size dependent processing times, parallel and multipurpose equipment units, and different kinds of storage policies.

2. Innovative scheduling algorithm using both local and global information, which can balance computation efficiency and solution optimality. Complexity analysis proves that it is a polynomial time algorithm. The returned solution is close-to-optimal by comprehensive comparisons with MIP methods based on both discrete-time RTN and continuous-time RTN.

3. Effective rescheduling approach analogous to model predictive control (MPC), which can tackle real-world complex scheduling problems under various uncertainties, for example, equipment breakdown, variations in processing times, or changes in order demands.

The remainder of this article is organized as follows. The background section "Network Batch Scheduling Problem" presents an overview of agent-based modeling and batch process scheduling. The agent-based model is proposed in the "Agent Model" section, followed by the scheduling algorithms in the "Scheduling Algorithms" section. The online implementation is presented in the "Online Scheduling" section. Two case studies based on process models derived in The Dow Chemical Company are presented in the "Case studies" section where detailed comparisons with MIP-based methods are conducted. Advantages and disadvantages of the agent-based scheduling methods are summarized in the "Summary of the Comparisons" section according to the comparison results. Finally, some conclusions are given in the "Conclusions" section.

Network Batch Scheduling Problem

The aim of process scheduling is to optimally determine the sequence of operational tasks and to allocate limited resources over time to each task. In batch manufacturing, commonly encountered tasks are chemical reactions, separations, material transfers, maintenance activities, mixing, heating, and so forth. Resources required by a task typically include equipment units, storage tanks, raw materials or intermediates, utilities, and operators, and so forth.

A batch process is effectively represented by the RTN[11] that has three types of nodes: (1) the state nodes that are associated with raw materials, intermediates, and final products, (2) the task nodes that represent operations transforming the materials in the input states to those in the output states, and (3) the equipment nodes that represent units where a task can execute. The RTN representation is equivalent to another popular STN[9] representation. The proposed agent-based model will be built based on the RTN representation. Due to the equivalence of RTN and STN, the model can also be applied to the STN representation.

In the RTN, if there are parallel equipment units that can process the same task, the task is split into multiple new tasks according to the units. One task is then linked to only one unit. In this way, selection of a task to execute implies assignment of the associated equipment to the task. There is no explicit equipment assignment in a scheduling model based on the RTN representation. We should note that the task splitting procedure is just a way to simplify the model representation. The equipment assignment decisions are still coupled with the task sequencing decisions.[11]

A typical example of the RTN representation is shown in Figure 1. The state nodes are denoted by a circle, the task nodes are denoted by a rectangle box, and the equipment nodes are denoted by an eclipse. The arc linking a task node and a state node indicates the flow of materials. The fraction of the material in a state that is consumed or produced by a task is marked beside the arc. For example, the task T_3 has two input states and the fractional values of S_2 and S_3 are 0.3 and 0.7, respectively. For clarity, the value of the fraction is often omitted if it is one.

In the network process displayed in Figure 1, the task T_1 can execute in the identical equipment of E_1 or E_2. According to the formulation rule of the RTN, the task is split into two new tasks, defined by $T_1\_E_1$ and $T_1\_E_2$. The new task $T_1\_E_1$ represents the original task T_1 executed in the equipment E_1, whereas the task $T_1\_E_2$ represents the task T_1 in the equipment E_2. The combination of the equipment to the task transforms the equipment assignment decision to the task selection. For example, if the task $T_1\_E_1$ is selected to execute, the associated equipment E_1 is assigned to the original task T_1. Similarly, if the equipment E_2 is assigned to the task T_1, then the derived task $T_1\_E_2$ is selected.

Scheduling problems are generally difficult to solve, because discrete decision variables are required for task sequencing or resource assignment. From the view of the

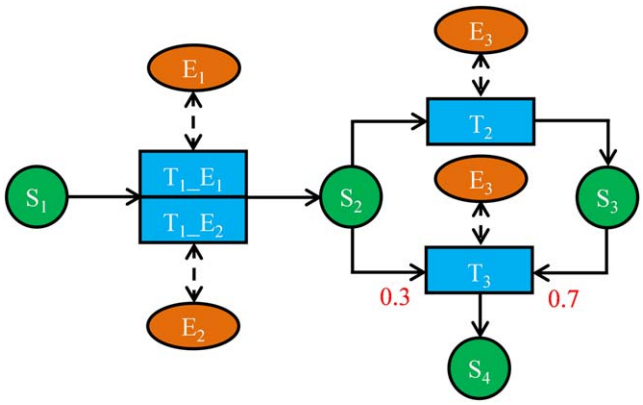

Figure 1. Illustration of the RTN.

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

computational complexity, the majority of scheduling problems are NP-hard.[42] The computational resources consumed by MIP methods can increase rapidly beyond an acceptable range as the scale of a scheduling problem grows. Moreover, the dynamic nature of the scheduling environment, which involves various uncertainties and disturbances, makes it extremely difficult to follow an offline schedule for very long.

Agent-based methods are promising alternatives for coping with real-world complex scheduling problems. Unlike the MIP-based methods that are centralized and take all information into account simultaneously, agent-based methods distribute the scheduling decisions among the interactions of agents, each using local information. The distributed decisions make an agent-based method fast for a complex problem and adaptable to the changing environment.

However, existing agent-based scheduling methods have been primarily developed for discrete manufacturing, and they are not applicable to network batch processes. In discrete manufacturing, the intermediate products are not divided during the production procedure. Although parts are processed through multiple stages, the identity of each part is preserved. Thus, an agent is usually built to track the product. In contrast, batch manufacturing processes contain continuous materials where material splitting, mixing, and resizing are common. An agent cannot be assigned to track the batch, because the batch integrity is not preserved.

Material splitting, mixing, and resizing also entail more constraints in the batch scheduling problem. For instance, the scheduling system needs to model the material transfer from one unit to another to enforce the material balance constraint. It needs to determine the batch size, as input materials can be split. The batch size is constrained by the equipment capacity, the amount of materials, and the available storage after the processing. The batch size can also be coupled with the task processing time. The scheduling system must take all these factors into account. Due to the complexity in the scheduling problem, a new agent-based model is required for batch manufacturing.

Agent-Based Model for Scheduling a Network Process

In this section, we propose a novel agent-based model applicable for efficient scheduling of batch processes with general network structures. The main features of a network process, which we need to model using agents, are listed in Table 1.

To model a network batch process with features listed in Table 1, we developed the agent-based model based on the RTN representation shown in Figure 1. The class diagram of the agent-based model is exhibited in Figure 2. There are six types of agents. The state agents, the task agents, and the equipment agents correspond to the states, tasks, and equipment in the RTN, respectively. Operations that can be executed in different equipments are represented as separate tasks in the RTN, and each task has an assigned task agent. Besides, the three types of agents identified from the RTN, the agent-based model also includes storage agents, order agents, and the scheduler agent.

The main functions of each agent are summarized:

- *Order agents*: Record the order demand and the order due date for each product. The number of order agents is equal to the number of product orders.
- *Scheduler agent*: Implement a scheduling algorithm and select task agents to execute according to a selection criterion. The scheduler agent is unique.
- *Task agents*: Monitor the whole execution procedure of a task. The number of task agents is equal to the number of tasks in RTN.
- *State agents*: Record the amount of raw materials, intermediate products, and final products. The number of state agents is equal to the number of states in RTN.
- *Equipment agents*: Manage the equipment units. The number of equipment agents is equal to the number of equipment in RTN.
- *Storage agents*: Manage the storage tanks. The number of storage agents is equal to the number of storage tanks.

The task agents play the key role in the agent-based model. The attributes of a task agent are summarized in Figure 2. The attributes are cast into three categories. The attributes in the first category include the structure information of the RTN representation, including the identification of the task agent, the input state set, the output state set, and the equipment, which are linked to the task. The second category of attributes contains the recipe data for the task execution: processing times, the range of the batch size, the task changeover information, and the fractional values of the input states and the output states. The third category consists of attributes recording the runtime information, that is, the task status, the batch size, the remaining processing time, and the remaining changeover time. These attributes are

Table 1. Features of the Network Batch Scheduling Problem

Feature	Specification
Structure	Network represented by RTN or STN
Material	Allow splitting, mixing, and resizing
Resource capacity	Limited
Batch size	Variable
Storage policy	UIS
	FIS
	NIS
	SIS
Processing time	Batch size dependent (linear or nonlinear dependence)
Changeover time	Sequence dependent
Equipment	Parallel and multipurpose
Equipment availability	Continuously available or periodically available
Objective function	Production cost, makespan, and tardiness

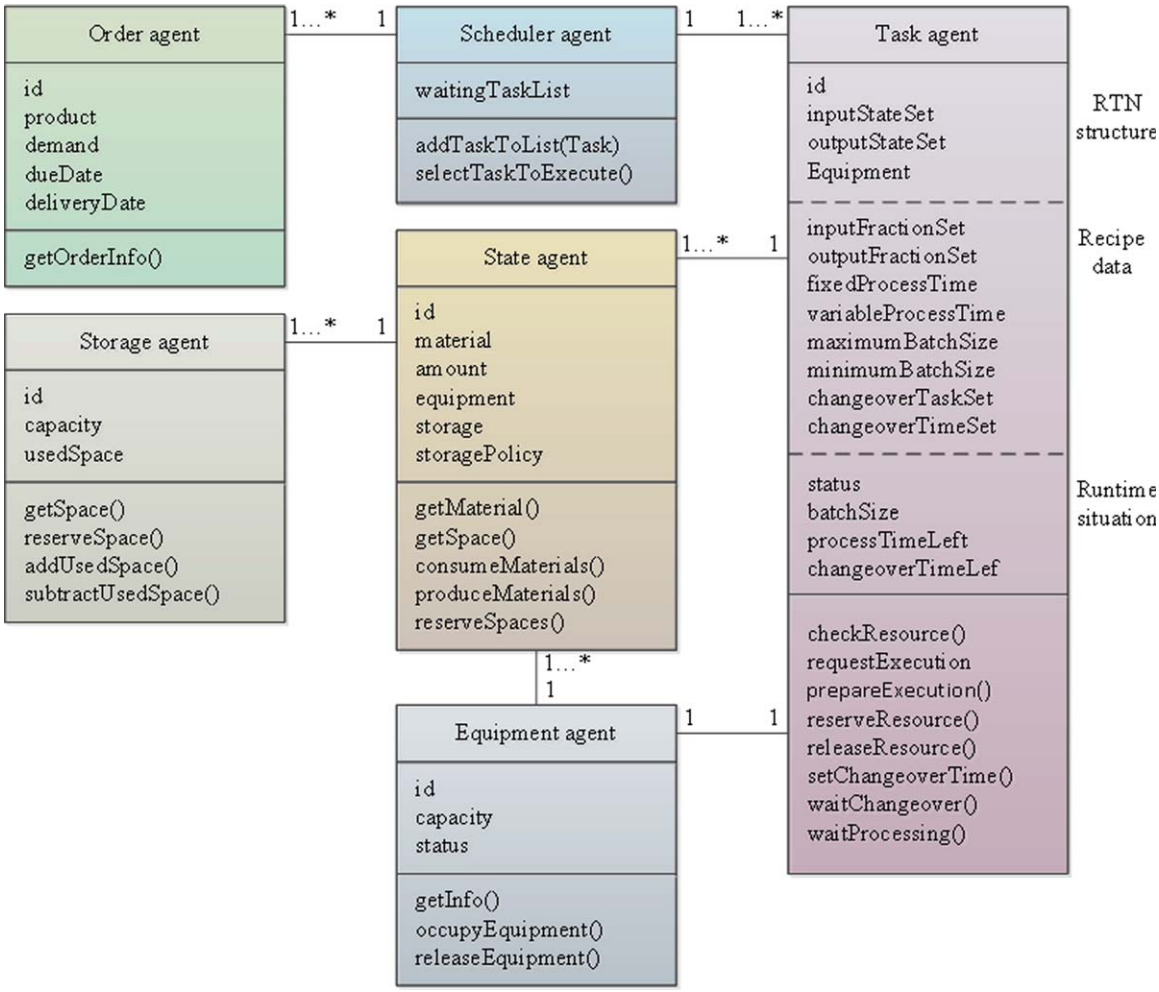

Figure 2. The class diagram of the agent-based model for scheduling a network batch process.

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

determined by the current running environment of the agent system, and they vary as the time evolves.

The agent-based model is driven by time clicks. The time interval is partitioned uniformly by a set of discrete time points. Each time point represents a time click. Agents only update their information and take actions at the discrete time points. When the actions of all agents are completed, a time click is triggered and the simulation time of the agent-based model advances by one time step. The time representation is similar to that in the discrete-time RTN. However, the agent-based model can use a much smaller time step for higher resolution in the time domain.

The sequence diagram that describes the execution of a task agent is exhibited in Figure 3, and the state diagram of the task agent is shown in Figure 4. To distinguish "state" in the state diagram from "state" in the state agent or in the RTN representation, we use the word "status" to denote the term "state" in the state diagram. The task agent is initialized in the idle status. Before it can start executing, there are multiple statuses it must pass through. A necessary condition for executing a task is that the task has all the required resources. Specifically, the resources include the equipment, the materials of the input states, and the storage spaces of the output states. To check conditions of the required resources, the task agent communicates with the associated equipment agent and state agents. If all resources are ready, the task agent will calculate the batch size according to the amounts of available resources. The focus here is placed on agent interactions, and the detailed calculation of the batch size will be discussed in the following (see Eq. 1).

Once all the conditions to execute the task are met, the task agent asks the scheduler agent for permission to execute. The scheduler agent maintains a list of tasks agents waiting to execute, and it selects a task agent to release to execute according to a defined criterion. Although only one task is selected at a time, the scheduler agent can repeat the selection procedure at a simulation time point. Then, multiple tasks can be selected at a time point, so that they can begin executing in parallel. The detailed selection criterion of the scheduler agent will be presented in the next section. When the task agent is notified, it begins executing. At this time, the task agent communicates with all the appropriate state and equipment agents to seize all required resources. It notifies the equipment agent that the equipment is held and not available to other tasks. The task agent also asks the input state agents to log a consumption of materials and requests the output state agents to reserve storage spaces. Unlike the equipment agent, the input or output agents are not totally occupied by the task agent. The task agent only reserves a fraction of resources. The remaining resources, if any, are also available to other tasks. Thus, it is possible that an input state agent can feed multiple task agents

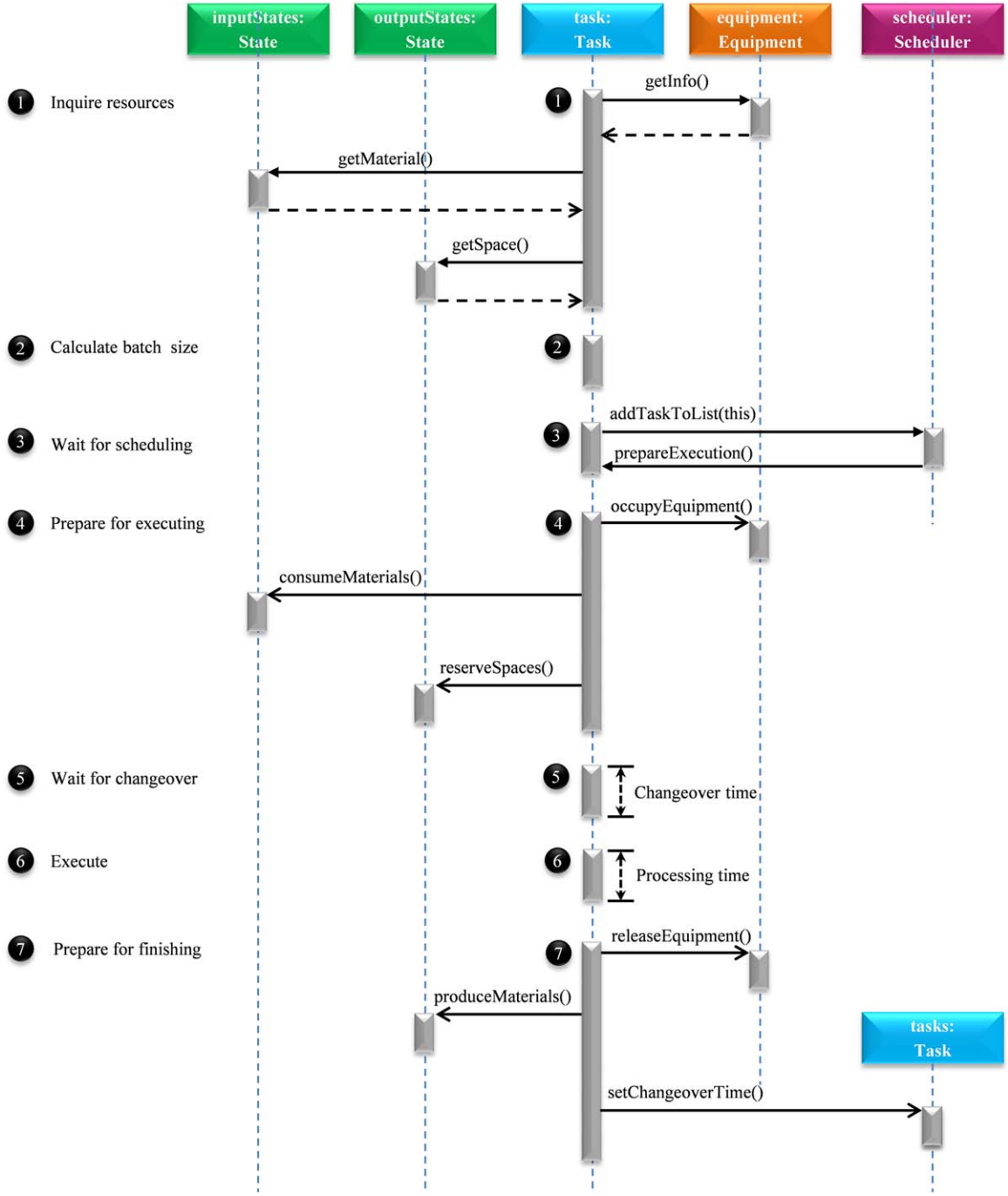

Figure 3. Sequence diagram of executing a task.

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

simultaneously, and an output agent can provide storage spaces to different task agents at the same time. After the resources are seized, they cannot be used by other task agents.

If there is a changeover time, the task agent needs to wait until the changeover time is up. When the changeover completes, the task agent starts execution. A timer is used to track the processing time. The task agent finishes its execution, when the processing time is equal to the value set by the production recipe or the value determined by the batch size. After executing, the task agent informs the equipment agent that the equipment is available to other tasks and communicates with output state agents to log the produced amount. Finally, the task agent returns to the initial idle status and starts a new production cycle. In the idle status, the task agent constantly checks the resources. Once the resources are available, the task agent will enter the next status, repeating the same procedure discussed earlier.

As pointed out earlier, a batch scheduling problem consists of various constraints. The detailed procedures that transform the constraints of the network scheduling problem into logic rules of the agent system are reviewed next.

Allocation constraint

A processing unit can only be used to execute one task at a time. This constraint is realized by introducing an attribute,

status, in the equipment agent. The state diagram of the equipment agent is shown in Figure 5. When a task occupies the equipment, the task agent signals the equipment agent to set the status to "occupied," so that other tasks have no access to the equipment. After the task completes, the task agent signals the equipment agent again to set the status to "idle," and the equipment is then available to other tasks. The status of the equipment can also be denoted "breakdown" to account for the time the equipment is out of service. If the status is set to breakdown, the equipment cannot be used by any task, and all task agents associated with it will wait in the idle status until the equipment unit becomes available.

Resource limitation and batch size

The amount of resources, for example, materials and storage spaces, is limited. The agent-based model enforces resource constraints by manipulating the batch size. The task agent obtains the amount of resources it can use by inquiring all resource agents before it executes. Based on the information, the task agent sets its batch size to the smallest resource constraints, specifically

$$B_S = \min\left\{B_S^{\max}, C_E, S_i/\rho_i^{in}, P_j/\rho_j^{out}\right\} \quad (1)$$

where B_S is the task batch size, $B_S^{\max}$ is the maximum value of the allowed batch size, C_E is the capacity of the equipment, S_i is the material value in an input state of the task, ρ_i^{in} is the unit ratio of the input state, P_j is the available storage space in an output state of the task, and ρ_j^{out} is the unit ratio of the output state.

Figure 4. State diagram of the task agent.

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

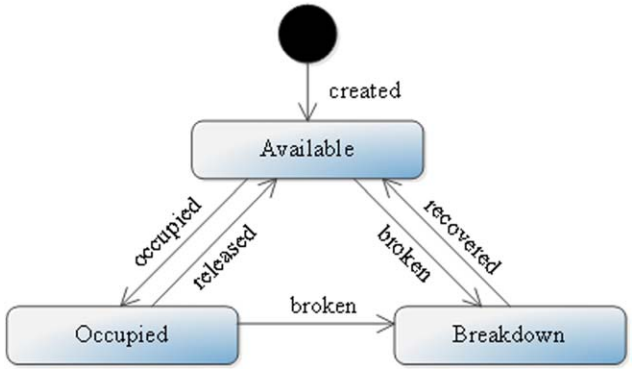

Figure 5. State diagram of the equipment agent.

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

Although the batch size is generally a variable in the network process, in some situations, the batch size of some tasks may be fixed. To deal with the fixed batch size, an addition condition is added to calculate the batch size as

$$B_S = \begin{cases} B^{\text{fixed}}, & \text{if } B^{\text{fixed}} \le \min\left\{B_S^{\max}, C_E, S_i/\rho_i^{in}, P_j/\rho_j^{out}\right\} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where B^{fixed} is the fixed batch size. The fixed batch size is first compared with the value calculated according to Eq. 1. If the fixed batch size is less than that value, the batch size is set as the fixed size. In this case, all other constraints on the batch size are satisfied. Otherwise, the batch size is set to zero, because the batch size smaller than the fixed value is not allowed. When the batch size is zero, the task will not be executed. Therefore, an executable task always has the batch size equal to the fixed value.

Material balance

The change in materials associated with a state agent is equal to the amount of materials produced by the task agents minus the amount of materials consumed by task agents. The material change is monitored whenever a task starts or finishes. At the beginning of the execution period, a task consumes materials in the input states, and the remaining material value in a state becomes

$$S_i = S_i - \rho_i^{in} B_S, \text{ at the task starting point} \quad (3)$$

After the execution finishes, the task produces materials to the output states, and the amount of materials in an output state changes to

$$S_j = S_j + \rho_j^{out} B_S, \text{ at the task finishing point} \quad (4)$$

Thus, material balance constraints are satisfied.

Storage policies

Batch manufacturing includes different storage policies for raw materials, intermediates, and products. A very common one is finite intermediate storage (FIS) where a dedicated tank is assigned to an intermediate product. The dedicated tank has a finite capacity, and the unused capacity is used to determine the batch size as discussed earlier. The sequence diagram about the FIS is shown in Figure 6a. Before

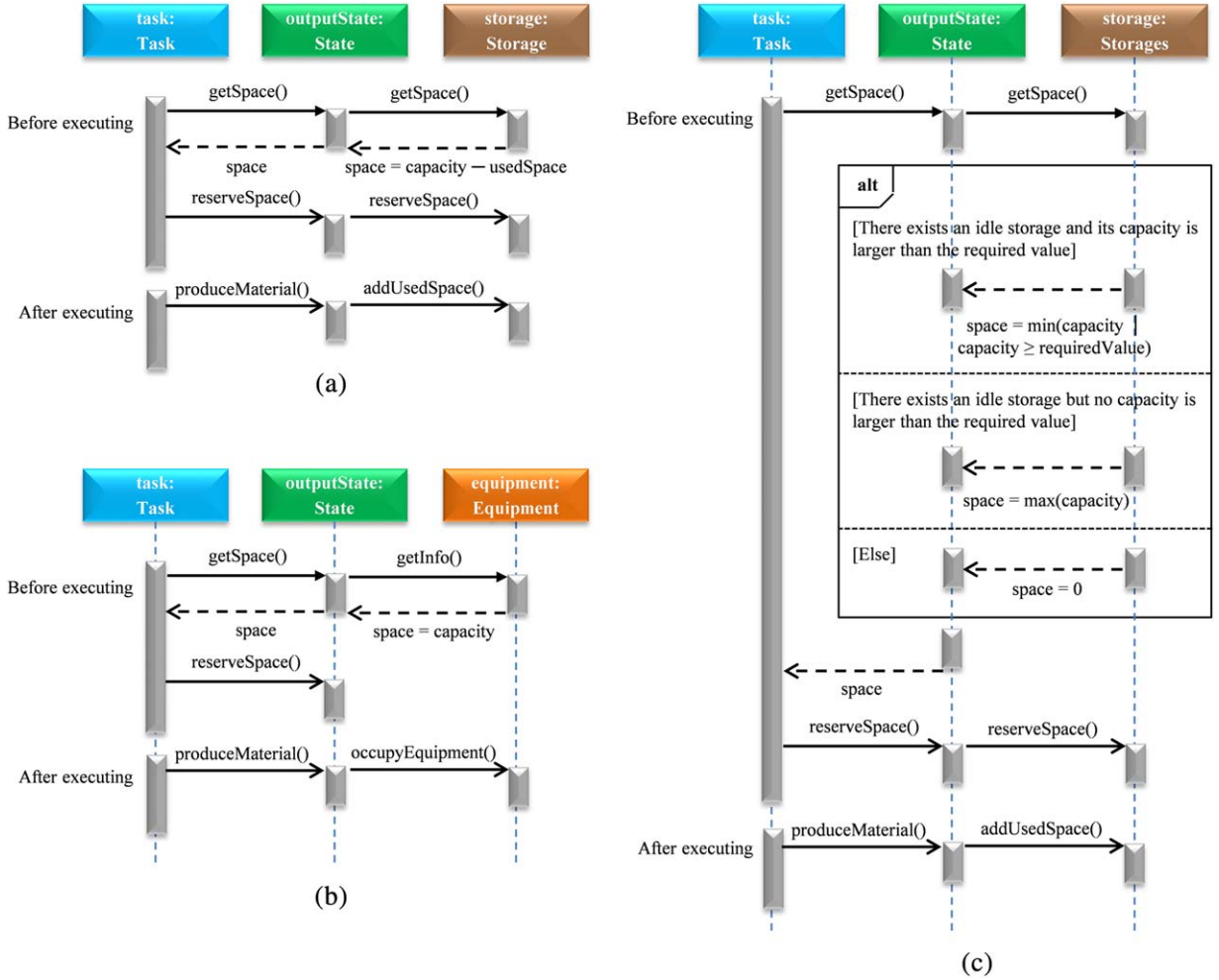

Figure 6. Sequence diagram of storage policies: (a) FIS; (b) NIS; and (c) SIS.

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

executing, the task agent inquires about storage space from the output state agent. Then, the state agent communicates with the associated storage agent for the information. The space is equal to the capacity of the storage tank minus the amount of material already contained in the tank. For this storage policy, there is a one-to-one relationship between the state agent and the storage agent. The state agent transfers the messages between the task agent and the storage agent. The unlimited intermediate storage (UIS) policy is a special case of the FIS where the capacity has no practical limit and, thus, it can be dealt with in the same way as the FIS.

For the policy of no intermediate storage (NIS), there is no storage agent linked to the state agent. The materials can be temporarily stored in the equipment, so that the state agent communicates with the equipment agent. The sequence diagram of the NIS policy is shown in Figure 6b. The storage space is equal to the capacity of the equipment. When the task agent sends the message to reserve the storage space, the state agent does not need to reserve the equipment, because the task agent will do so when it starts executing. After executing, the state agent seizes the equipment for storing the produced materials. The equipment will remain occupied until the stored materials are completely consumed by downstream tasks.

The implementation of shared intermediate storage (SIS) is more complicated, and the sequence diagram is shown in Figure 6c. The state agent is linked to a set of storage agents, one of which is responsible for a tank able to store the materials. More than one tank can be used to store the materials. The demand on the storage space is initiated by the task agent to the state agent that requests status information from the connected storage agents. If there are idle storage agents that have spaces larger than the required value, the smallest storage capacity that is no less than the required size is found, and the corresponding storage agent is selected by the state agent. If there are no idle storage agents having capacities larger than the required size (according to Eq. 1 or 2), the storage agent with the largest storage space is selected and the available space is equal to the capacity of the storage. If there are no idle storage agents, the available space is zero.

The different storage policies are realized by different logical rules among the state agent, the equipment agent, and the storage agents. However, the different storage policies are transparent to the task agent, and it communicates with the state agent in the same way as shown in Figure 6. This approach simplifies the logic rules for task selection and execution.

The policy of the zero-wait intermediate storage (ZIS) is not realized by the proposed agent-based model, because the policy imposes strong interactions among agents as well as correlated events at different time points. Therefore, global

information is required, which is difficult for the local agents to obtain. The agent-based method has limited ability to deal with constraints requiring global information. Although a general approach to realize the ZIS policy is difficult to construct, the policy can still be dealt with by taking advantage of specific domain knowledge, for example, the model structure or the problem specification. An agent-based model could be formulated with logical rules that represent the required domain knowledge.

Batch size dependent processing time

When the task processing time T_P is dependent on the batch size B_S, it is expressed as

$$T_P = \Delta_T \cdot \text{ceil}\left(\frac{T_P^f + T_P^v B_S}{\Delta_T}\right) \quad (5)$$

where Δ_T is the simulation time step of the agent-based model, T_P^f is the fixed processing time that is independent of the batch size, and T_P^v is the factor in the variable processing time dependent on the batch size. The ceiling function is applied to round the resulted processing time to an integer multiplied by the time step. There is an approximation error in the rounding procedure; however, the simulation step of the agent-based model can be small enough that the rounding error is negligible.

Sequence-dependent changeover time

It is common in batch manufacturing that a processing unit requires cleaning or other preparation procedures before it can be used to execute a new task. A changeover time is required when a task executes in the same equipment after another task and the changeover time is frequently dependent on the sequence of the tasks. The constraint of the changeover time is satisfied by interactions among task agents. After the execution, a task agent will set the attribute that denotes the remaining changeover time for all task agents including itself. The values of the changeover time are different according to the task sequence. A task will wait in the status of "waiting changeover" until the changeover time is up.

Scheduling Algorithms for the Agent-Based Model

The key feature of the agent-based method is that the scheduling decisions are made by a collection of distributed agents. It has been seen that the constraints in the scheduling problem of a network process can be easily satisfied by interactions among the task agents and other agents. When a task is ready to execute, the task agent enters the waiting list of the scheduler agent. The job of the scheduler agent is then reduced to solving the task selection problem.

Agent-based methods commonly use local information to make scheduling decisions. However, to improve the solution quality, global information such as an objective function value can be used. In this section, we propose a novel scheduling algorithm, which predicts the objective function value and uses the predicted value to select a task to execute when idle equipment is available. Complexity analysis proves that it is a polynomial time algorithm, and it improves the solution quality by providing a balance between efficiency and schedule performance.

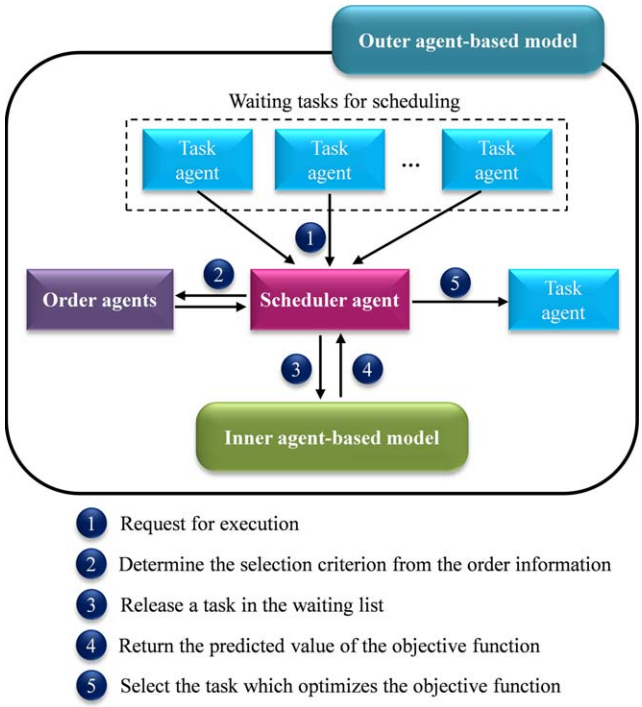

Figure 7. The structure of the two-level agent-based model where the inner agent-based model provides a prediction of the objective function value for the outer agent-based model to select the tasks.

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

Scheduling via simulation-based prediction

Global information is not straightforward to obtain in the agent-based model. The objective function value, for example, the total cost, the makespan, or the tardiness, is only known when the whole schedule is determined. Obviously during the scheduling procedure, when the scheduler selects a task to execute, it cannot use the objective function as a criterion. To obtain global information, we propose a simulation-based approach for predicting the objective function value.

The simulation-based approach relies on a second agent-based model to predict the objective function, and it is referred to as the "inner agent-based model." Correspondingly, the original agent system is referred to as the "outer agent-based model." This results in an embedded agent system, whose structure is shown in Figure 7. The embedded agent system contains two levels of agent-based models, so we call it the "two-level agent-based model." To distinguish, the common agent system with only one level is referred to as the "single-level agent-based model."

At each simulation step when the scheduler agent in the outer agent-based model needs to select a task from the waiting list, it cooperates with the inner agent-based model to determine the best selection. The scheduler agent evaluates the impact of a waiting task by copying the current running environment of the outer agent-based model to the inner agent-based model with the selected task set to execute. The inner agent-based model then continues the simulation from the current condition of the outer agent-based model using the selected task.

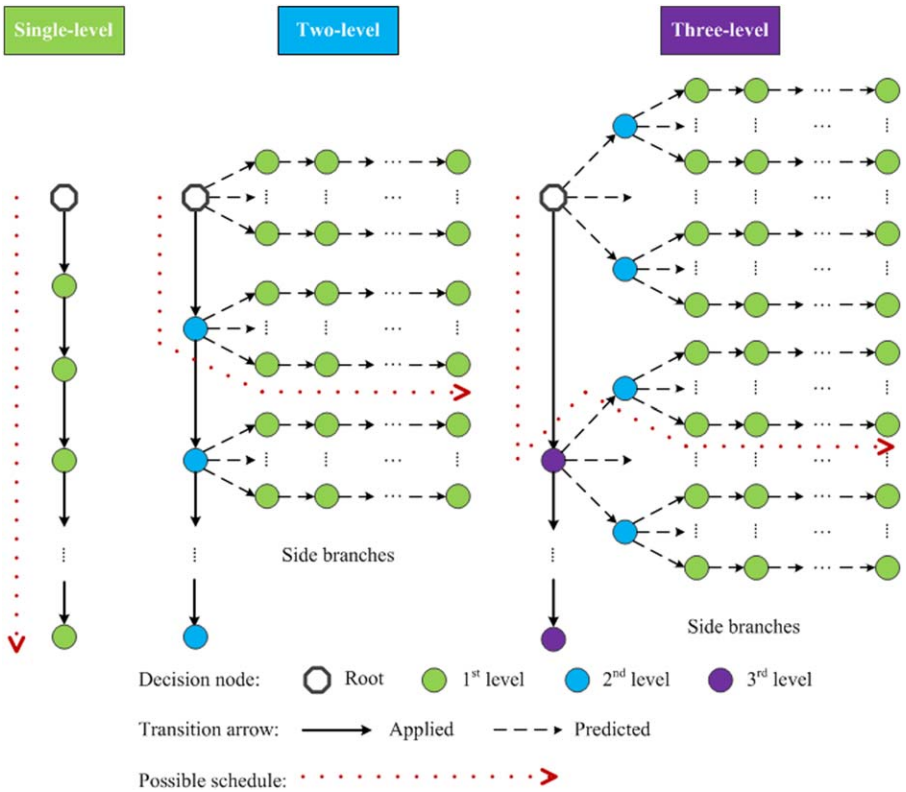

Figure 8. The tree search structure for multilevel agent-based models.

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

The inner agent-based model uses the local information of the batch size to complete the remaining schedule. The total amount of materials that need to be processed is often fixed by the order demands. Therefore, if a task has a large batch size, more materials can be processed at one time and the number of batches can be reduced. The reduction in the number of batches in turn can reduce the operation cost and the makespan. According to this rule, the scheduler agent selects a task having a large batch size over a task having a small batch size.

It is possible to use other types of information. For example, the task processing time is an alternative to the batch size. A task that has a small processing time is favored, because the equipment will be occupied for a shorter period and it will be ready sooner for executing other tasks. We applied both criteria in examples for case study. The criterion based on the batch size outperformed that based on the processing time. Therefore, in this work, the scheduler agent adopts this heuristic rule and selects the task with the largest batch size.

At each simulation step, the task with the largest batch size is selected by the inner agent-based model. The simulation continues until all order demands are satisfied. At the end of the simulation, the inner agent-based model records the objective function value and returns it to the scheduler agent in the outer agent-based model. By this "try-and-see" procedure, the scheduler agent can assess the impact of its current selection and select the task with the best predicted objective function value. The advantage of the simulation-based approach is that all the logic rules for deriving the prediction are encapsulated inside the inner agent-based model.

The scheduler agent enumerates all tasks that are ready for executing at the current time point. This does not, however, mean the whole schedule is determined by the brute force enumeration, which is extremely time consuming. The inner agent-based model uses the local information to schedule the process and returns the predicted objective function value. The scheduling algorithm is actually a heuristic tree search, which is analyzed in more detail in the following section about the complexity analysis.

Complexity analysis of agent-based scheduling algorithm

The computational time of a scheduling method is critical for its online performance. A efficient scheduling method should have a computational time less than the resolution of the scheduling problem. In this section, we investigate the computational complexity of a general n-level agent-based model extended from the two-level model shown in Figure 7. It will be shown that the number of task selection operations is bounded by a polynomial function on the number of batches. A polynomial time algorithm is a promising feature for efficient applications.

The two-level agent-based model uses the embedded single-level agent-based model to predict the objective function. Such an embedded structure can be extended to general multilevel agent-based models as exhibited in Figure 8. The three-level agent-based model is built by embedding a two-level agent-based model. It predicts the objective function value by simulating the embedded two-level agent-based model. Then, the two-level agent-based model, in turn, invokes its own embedded single-level agent-based model to simulate the process. In the same way, a general n-level agent-based model can be constructed based on an $n-1$-level agent-based model.

The scheduling decisions as well as the search paths for each agent-based model are represented by a tree structure in

Figure 8. The root node is expressed by an open octagon, which is the initial condition for scheduling. At each node, a task is selected. Thus, a branch from the root to a leaf represents a possible schedule, which is illustrated by the dotted line in Figure 8. For the single-level model, the tree is reduced to a vertical line. This is the only schedule, and it is applied to the process. For a multilevel model, besides the vertical line, there are horizontal side branches. The side branches represent the simulation procedure conducted to obtain the prediction of the objective function value. These simulated steps are not applied to the process. The decisions that are really applied are expressed by the nodes in the vertical line.

The complexity of an n-level agent-based scheduling algorithm is characterized by the number of all nodes in the searching paths of the model. Strictly speaking, we need to count the nodes in an agent-based model as displayed in Figure 8. The counting starts from the single-level agent-based model. In the model, all nodes are in the vertical line. Because a node corresponds to the selection of a task and when a task is selected to execute a bar will appear in the Gantt chart, the number of the nodes is equal to the number of bars in the Gantt chart, or to the number of all batches in the schedule, that is

$$N_{\text{node}}^{(1)} = N_{\text{batch}} \tag{6}$$

where $N_{\text{node}}^{(1)}$ denotes the number of nodes for the single-level agent-based model, and N_{batch} is the number of batches.

For a multilevel model, there can be multiple branches so the nodes in all branches should be counted. Each branch represents a possible schedule, and the number of all batches in the schedule is assumed to be bounded by U_{batch} such that $N_{\text{batch}} \leq U_{\text{batch}}$. When scheduling a batch process with a general network structure, the number of batches is dependent on the scheduling algorithm. However, the variation in the number of batches is usually not significant as the number is related to filling customer orders, so it is not difficult to find the upper bound U_{batch}.

Using the upper bound of batches, we can count the nodes in the two-level model. At the root node, there are different side branches. Each side branch denotes a trial of selecting a task. So the number of side branches is equal to the number of tasks in the waiting list of the scheduler agent, that is

$$N_{\text{branch}} = N_{\text{waiting}} \leq N_{\text{task}} \tag{7}$$

where N_{branch} is the number of side branches, and N_{waiting} is the number of waiting tasks which is bounded by the number of all tasks denoted by N_{task}. The number of nodes in all side branches starting from the root node is bounded by $N_{\text{task}}(U_{\text{batch}}-1)$. Because the decision in the root node has been made, the number of batches is decreased by one. Similarly, for the second node in the vertical line, the corresponding side branches contain at most $N_{\text{task}}(U_{\text{batch}}-2)$ nodes. The number of nodes in the side branches linking to the ith node in the vertical line is $N_{\text{task}}(U_{\text{batch}}-i)$. To sum up, the number of searching nodes in the two-level model is bounded by

$$\begin{aligned} N_{\text{node}}^{(2)} &\leq N_{\text{task}}\left[(U_{\text{batch}}-1)+(U_{\text{batch}}-2)+\cdots+1\right]+U_{\text{batch}} \\ &= \frac{1}{2}N_{\text{task}}U_{\text{batch}}(U_{\text{batch}}-1)+U_{\text{batch}} \leq \frac{1}{2}N_{\text{task}}U_{\text{batch}}^2 \end{aligned} \tag{8}$$

The last term of U_{batch} in the summation aims to add up the number of nodes in the vertical line. Therefore, the number of searching nodes has the same order with the quadratic function of the batch size, that is

$$N_{\text{node}}^{(2)} = O\left(N_{\text{batch}}^2\right) \tag{9}$$

For the three-level agent-based model, the number of nodes in the side branches that connect to the root node is bounded by

$$N_{\text{task}}\left[\frac{1}{2}N_{\text{task}}(U_{\text{batch}}-1)^2\right] = \frac{1}{2}N_{\text{task}}^2(U_{\text{batch}}-1)^2 \tag{10}$$

according to the Eq. 8. The total number of searching nodes is bounded by

$$\begin{aligned} N_{\text{node}}^{(3)} &\leq \frac{1}{2}N_{\text{task}}^2\left[(U_{\text{batch}}-1)^2+(U_{\text{batch}}-2)^2+\cdots+1\right]+U_{\text{batch}} \\ &\leq \frac{1}{4}N_{\text{task}}^2U_{\text{batch}}^3 \end{aligned} \tag{11}$$

and it has the same order with the cubic function of the batch size as

$$N_{\text{node}}^{(3)} = O\left(N_{\text{batch}}^3\right) \tag{12}$$

To repeat the same procedure, the number of searching nodes in an n-level model is

$$N_{\text{node}}^{(n)} = O\left(N_{\text{batch}}^n\right) \tag{13}$$

The number of searching nodes determines the computational time of the agent-based method, and the computational time is proportional to the number of nodes. The number of batches is mainly determined by the order demand, and it characterizes the size of the scheduling problem. Therefore, we can conclude that the multilevel agent-based scheduling algorithm has a polynomial computational time with the size of the scheduling problem. This is a main advantage of the agent-based methods over the MIP-based methods, because the computational time of an MIP-based method can increase exponentially.

The number of levels is a design parameter for the agent-based methods, which balances the solution optimality and the computation efficiency. As more levels are included in a model, the returned schedule is improved at the expense of more computational resources. Because the main purpose for applying agent-based methods is the computational efficiency, the focus of this work is placed on the single- and the two-level agent-based models. Application of the agent-based model containing more levels is left for the future research.

We should note that the complexity analysis builds the relation between the number of searched nodes with the number of batches. This analysis is similar to the one conducted for the exact solution.[43] For example, the complexity analysis on a jobshop problem in discrete manufacturing

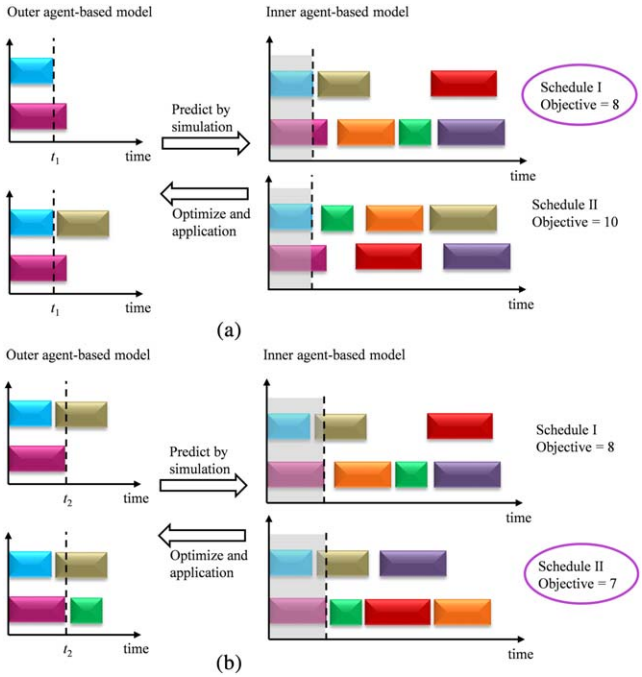

Figure 9. The scheme of prediction and optimization for the scheduling method based on the two-level agent-based model.

A task is selected at the time point of (a) t_1 and (b) t_2. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

studies the relation between the number of iterations and the number of job operations. Complexity analysis focuses on the worst-case scenario and does not directly determine the actual computational time of an algorithm, which is dependent on a particular scheduling problem and specific implementation of the algorithm. For the agent-based modeling, there are various frameworks developed to improve the performance of the software implementation.[39,40] However, optimizing the software implementation is beyond the scope of this work.

Online Scheduling Under Uncertainties

Using the global information of the objective function, the prediction-based scheduling algorithm is different from the conventional agent-based methods, which uses local information only. In fact, the scheduling algorithm has a receding horizon procedure similar to classical MPC, which is illustrated in Figure 9. This procedure is composed of the following steps:

1. State update: At each sampling point, the MPC updates the current value of the state variables to establish the initial conditions for the prediction through the time horizon in the next step. Similarly, at each simulation point, the proposed scheduling algorithm obtains the state of the current environment. For example, at the time point $t = t_1$ in Figure 9a, the scheduler agent knows that a task is finished and the equipment can be used to execute another task. Using the updated system state is crucial in dealing with uncertainties.

2. Prediction: For receding horizon methods, decisions are based on their predicted future effects, which are obtained via dynamic model predictions. In MPC, the decisions involve the trajectory of the control variables through the time horizon and the future effects are the output values (controlled variables) in the time horizon. The predicted output values are calculated by a dynamic model of the system. In the scheduling algorithm, the scheduler agent's decision is task selection, and which task is selected is based on its predicted future effect on the process. The predicted future effect is provided by simulating the inner agent-based model. Although the decision variables, the future effects, and the dynamic models are different, the purpose of prediction is the same, that is, providing data for the decision making. The predictions are both made based on the updated state of the system.

At the time point $t = t_1$ in Figure 9a, the simulation-based prediction is conducted to determine which tasks are selected to execute in the available equipment. There are two possible selections of tasks. For each selection, the inner agent-based model simulates the process from the current state of the outer agent-based model, and the historical range (covered in gray) is excluded from the simulation. The simulation generates two schedules, and they represent the effects caused by the two tasks, respectively.

3. *Optimization*: In receding horizon methods, the best decision is identified as the one that results in an optimal value of a quantitative measure of the systems performance. In the MPC, a linear or nonlinear programming problem is formulated, and the optimal trajectory of the control variables is obtained by solving the problem to obtain the optimal objective function. A sophisticated solver is often required to solve the optimization problem in the MPC. In the scheduling algorithm, the scheduler agent selects the task which results the optimal objective function value, a measure of schedule performance. Because the possible schedules are finite and the number is often small, the optimal solution can be simply found by enumerating all possible solutions. Although it is as simple as an enumeration, the optimization procedure still exists in the scheduling algorithm. For the example in Figure 9a, the objective function value of Schedule I is 8, whereas the value of Schedule II is 10. If the goal is to minimize the objective function, then Schedule I is the optimal solution and, thus, the corresponding task is selected.

4. *Application*: When the optimization procedure determines the optimal solution, the solution is applied to the system or the process. A special feature of receding horizon method is that only the immediate decision is applied. In the MPC, only the first value in the optimal trajectory is applied, whereas in the embedded agent-based scheduling algorithm only the first task of the schedule is applied. For example, at the time point $t = t_1$ in Figure 9a, only the task at the current time point is selected to execute.

Such a procedure of applying only the first scheduling decision at a time has two advantages. First, because the prediction is based on simulation of the inner agent-based model using the local information, the predicted schedule is not guaranteed to be optimal. Thus, applying only the first scheduling decision instead of the whole schedule increases the chance of improving the solution at subsequent time points. Second, the scheduling environment includes various uncertainties, and it is difficult to follow a predetermined schedule over a long-time horizon. It is more effective to only apply the current decision and then solve the prediction-based optimization again at the next point so that the updated information of the environment can be taken into account.

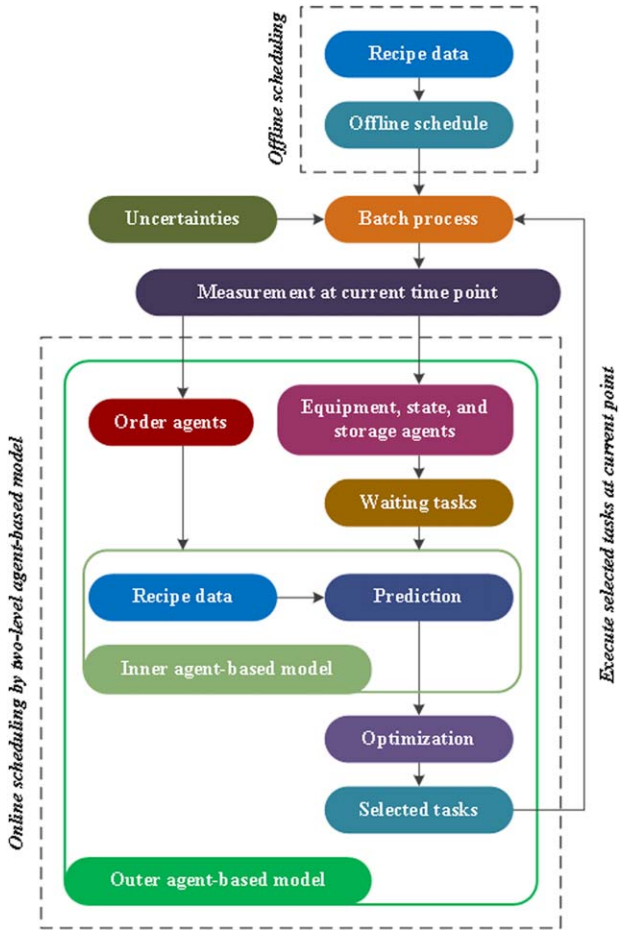

Figure 10. Flow diagram of online rescheduling under uncertainties.

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

At the next time point $t = t_2$, the same procedure of prediction and optimization is repeated in Figure 9b. The scheduling decision applied at this time point is also the first decision from the optimal schedule.

There are different rescheduling strategies and different approaches to handle uncertainties. Due to its efficiency, the agent-based method can be applied in a reactive way. The flow diagram of the online implementation is shown in Figure 10. The offline schedule is determined based on the process model and applied to the batch process at the beginning of the production. The batch process is subject to the uncertainties. The two-level agent-based model updates the measurement at each time point. The measurement consists of data about the equipment status, the material amount, the storage space, and the order information. These data are then used to update the attributes of the corresponding agents. In this way, the change occurring in the process is reflected on the resource agents and the order agents. According to the resource availability, the task agents determine if the associated tasks can be executed. The ready task agents enter the waiting list of the schedule agent. The order information is conveyed to the scheduler agent directly through the order agents. The scheduler agent then invokes the inner agent-based model to predict the objective function. The task optimizing the objective function is selected. The scheduler agent can select multiple tasks at one time point, if these tasks do not require conflicting resources. The selected tasks are then executed. It should be noted that, like MPC, only the tasks selected at current time points are executed. When an equipment unit becomes idle in the future, the same procedure is repeated according to the updated information so that uncertainties are taken into account.

Case Studies

To demonstrate the agent-based scheduling methods, we present two case studies in this section. Both examples represent real-world batch processes at The Dow Chemical Company. The first example is a batch plant where thorough comparisons with the MIP methods based on the discrete-time model[9] and the continuous-time model[10] are made. The two scheduling methods are very popular for scheduling batch processes.[3] The second example is a challenging large-scale scheduling problem where the MIP-based methods require excessively long computational times and a great amount of computer resources.

Small batch plant

The batch plant manufactures four products, P1–P4, from four raw materials, M1–M4. It consists of one raw material preparation unit, two batch reactors, one finishing system, and one drumming line. The RTN representation is shown in Figure 11, including 11 tasks (rectangle nodes), 14 states (circle nodes), and 5 processing units (at the bottom). As each reaction task can take place in either of the two identical reactors, a reaction task is split into two new tasks in the RTN, one corresponding to each reactor. Besides the four raw materials and the four final products, there are six intermediate states involved in the production. The storage capacity for the states associated with the raw materials and the final products is unlimited. The FIS policy is applied to the intermediate states where the capacity of each storage tank is 100 kg. The initial value of all states is zero.

The processing time and the maximum batch size for each task are listed in Table 2. The sequence-dependent changeover times for reaction tasks in a reactor and the packing tasks in the finishing system are given in Tables 3 and 4, respectively. There are no changeover times for the drumming tasks. It is noted that the changeover time is not symmetric for each pair of tasks.

The time step for the agent-based models is 1 min. The length of the time period in the discrete-time model is set as 6 min which is the largest common factor of the task processing times and the changeover times. In this setting, the discrete-time model accurately describes all time events, and the results are comparable to those based on the continuous-time model and those calculated by the agent-based scheduling methods.

Because the performance of a method is problem dependent, for example, the problem scale, the type of constraints, or the value of the parameters, comparisons of the agent-based scheduling methods with the RTN methods are conducted with different cases ranging from a simple problem in a short scheduling horizon without task changeover times to a complicated problem with changeover times, variable processing times, and large demand volume.

Agent-based scheduling methods using both the single-level model and the two-level model were applied to each. For the single-level model, the scheduling agent uses the simple criterion of selecting the task with the largest batch

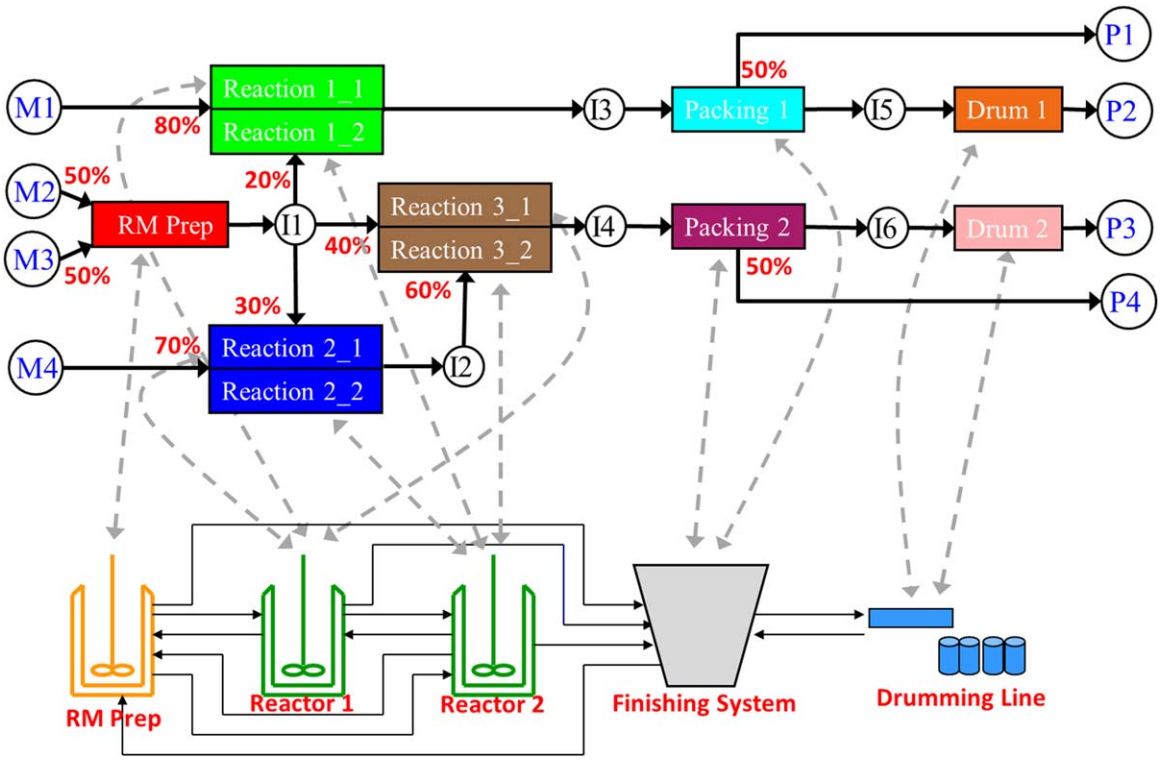

Figure 11. RTN of the scheduling problem for the small batch plant.

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

size that is executable at each time point. The two-level model, using the predictive capability provided by simulating the embedded agent system, selects tasks based on the minimum value of the predicted makespan. The two-level agent-based model returns a smaller makespan than the single-level model but requires more computational resources. When a solution is returned by the agent-based scheduling methods, the associated makespan, cost, and computational time are recorded for comparing with the MIP-based methods.

MIP methods based on the discrete-time RTN and the continuous-time RTN were applied to validate the results returned by the agent-based scheduling methods. To check the efficiency of the agent-based methods, the scheduling problem is optimized to the extent where the calculated solution is not worse than that calculated by the two-level agent-based model. The solution is obtained by setting the scheduling horizon to the makespan returned by the two-level agent-based model, and the optimality gap is adjusted so that when the solver stops the returned cost value is the closest one to, while not larger than, the cost value obtained by the agent-based method. The computational time to achieve this result is the smallest one for the MIP-based methods to return a solution having the same quality as the agent-based method, and it is recorded for comparison purposes. The efficiency of the agent-based method is exhibited by comparing the computational time with those of the MIP methods required for the equivalent solutions.

To check the quality of solutions returned by the agent-based methods, the MIP methods are applied to calculate the minimum cost. The time horizon is fixed to the same as that returned by the two-level agent-based method, and the cost is minimized to the optimality gap of 1%. Then, the objective function is changed to the makespan, and the gap is set as 1% as well. The optimal value returned by the MIP methods verifies the quality of the solution returned by the agent-based methods.

The summary of the comparison results are listed in Table 5. The setting of each case study and the detailed procedure to obtain the data listed in the table will be presented in the following sections. Conclusions and discussions are given at the end of the article.

In this work, the agent-based model is implemented in Java using Eclipse. The data about the scheduling problem are recorded in spreadsheets in an excel file, including the RTN representation, recipe data, order information, plant information, and finished partial schedule. The structure of the agent-based model, including the classes and the interfaces, are predefined. However, the number of each type of agents

Table 2. Task Information for the Small Batch Plant

Task	Processing Time (min)	Maximum Batch Size (kg)	Fixed Cost ($)	Variable Cost ($/kg)
RM Prep	72	100	1000	50
Reaction 1	162	80	3000	250
Reaction 2	138	50	1500	150
Reaction 3	162	80	2000	100
Packing 1	108	100	500	20
Packing 2	108	100	500	20
Drumming 1	90	50	200	50
Drumming 2	90	50	200	50

Table 3. Changeover Times of the Reaction Tasks (min) of the Small Batch Plant

	Reaction 1	Reaction 2	Reaction 3
Reaction 1	12	30	30
Reaction 2	30	12	6
Reaction 3	30	30	12

Table 4. Changeover Times of the Packing Tasks (min) of the Small Batch Plant

	Packing 1	Packing 2
Packing 1	0	6
Packing 2	6	0

and their attribute values are determined from the scheduling data. For different batch scheduling problems, the user only needs to input the scheduling data in the excel file. Based on the data, the agent model can be automatically constructed. All agents are created at the beginning when the data are imported. For a multilevel agent-based model, the user only builds the top-level agent model, whereas bottom-level agent models are built by the top-level agent model according to the tree search algorithm. Similarly, the online rescheduling is implemented. The data file is updated when a change occurs in the scheduling system. The data are then read into the agent-based model to update attributes of the agents. Thus, the change of the environment can be reflected in the agent-based model and taken into account by the scheduling algorithm.

The MIP problems are modeled in GAMS 23.8.1 and solved by CPLEX 12.4. All computations are conducted on a PC with Intel(R) Core(TM) i5–2400 CPU @ 3.10 GHz, 8 GB RAM, and Window 7 64-bit operating system. The CPU has four cores. Only one core is used to execute the Java codes for the agent-based scheduling methods, whereas three cores are used to run CPLEX, which takes advantage of the parallel computation capability of CPLEX. The maximum computational time is set as 12 h (43,200 s).

Case 1: Small Value of Order Demands Without Task Changeover Times. The comparison starts from a simple scheduling problem where the order demands are set to be P1 100 kg, P2 100 kg, P3 50 kg, and P4 50 kg. The task changeover times are not included.

The makespan returned by the single-level agent-based model is 948 min, and the corresponding cost is \$107,500. The two-level agent-based model reduces the makespan to 894 min, while the computational time increases from 0.6 to 5.4 s. The Gantt charts of the agent-based methods are plotted in Figures 12a,b.

For comparison, the MIP methods are applied. First, a suboptimal solution that is equivalent to that returned by the two-level agent-based method is obtained. The MIP method based on the discrete-time RTN is applied to minimize the cost when the makespan (894 min) is the same with the two-level agent-based model. The length of each time period is 6 min according to the largest common factor of task processing times and changeover times. Thus, the number of time points is equal to 150 (894/6 + 1). The optimization procedure stops when the same cost (\$107,700) with the two-level agent-based method is found. The computational time is 5.8 s, and the optimality gap is 4.1%. To obtain the solution with the same quality, the MIP method based on the discrete-time model is as fast as the agent-based scheduling method.

When we use the continuous-time RTN model, the computational time for calculating an equivalent solution is 29.8 s. The solution quality and the computational time of the continuous-time model are highly dependent on the number of time points. The number of time points was determined by a "trial-and-error" approach. The approach starts from a small number of time points where the minimum cost is larger than the value returned by the two-level agent-based model. Then, the number of time points is increased by one. Increase in the number of time points can further improve the optimal value of the cost while requiring longer computational time. The trial-and-error approach stops when a solution is calculated which is not worse than the one returned by the agent-based method. This trial-and-error approach determines the smallest number of time points required by the continuous-time model. The smallest number in this case study is 11 as listed in Table 5. The computational time (29.8 s) listed in the table is just the time associated with the 11 time points not the entire trial-and-error approach. In comparison, the continuous-time model needs more computational resources than the discrete-time model and the agent-based models in this simple case.

The solution returned by the two-level agent-based model is suboptimal, though it is better than the solution of the single-level agent-based model. An improved solution can be obtained by the MIP-based methods. The minimum cost

Table 5. Comparisons of the Agent-Based Methods with the MIP-Based Methods for Scheduling the Small Batch Plant

		Agent-Based		Discrete Time			Continuous Time		
Case		Single-Level	Two-Level	Equivalent	Optimal Cost	Optimal MS	Equivalent	Optimal Cost	Optimal MS
1	# Points	949	895	150	150	150	11	11	12
	MS (min)	948	**894**	894	894	**870**	894	894	**870**
	Cost (k\$)	107.5	**107.7**	107.7	**107.5**	–	107.7	**107.5**	–
	CPU (s)	0.6	**5.4**	**5.8**	45.7	69.8	29.8	33.4	414.8
	Gap (%)	–	–	4.1	1.0	1.0	2.5	1.0	1.0
2	# Points	1303	1219	204	204	204	15	15	15
	MS (min)	1302	**1218**	1218	1218	**1188**	1218	1218	1218
	Cost (k\$)	211.9	**208.9**	208.5	**207.5**	–	208.5	**207.5**	–
	CPU (s)	0.7	**12.8**	**51.9**	56.0	400.8	8723.6	9361.5	36,152.1
	Gap (%)	–	–	1.4	1.0	1.0	1.9	1.0	1.0
3	# Points	1435	1291	216	216	216	–	–	–
	MS (min)	1434	**1290**	1290	1290	**1254**	–	–	–
	Cost (k\$)	211.9	**208.9**	208.7	**207.5**	–	–	–	–
	CPU (s)	0.8	**13.4**	**1315.5**	1330.8	43,200	–	–	–
	Gap (%)	–	–	1.1	1.0	2.8	–	–	–
4	# Points	809	733	–	–	130	–	–	12
	MS (min)	808	**732**	–	–	762	–	–	**731**
	CPU (s)	0.6	**4.3**	–	–	43,200	–	–	19,964.8
	Gap (%)	–	–	–	–	3.2	–	–	1.0

The numbers in bold fonts show the important comparison results among different methods.

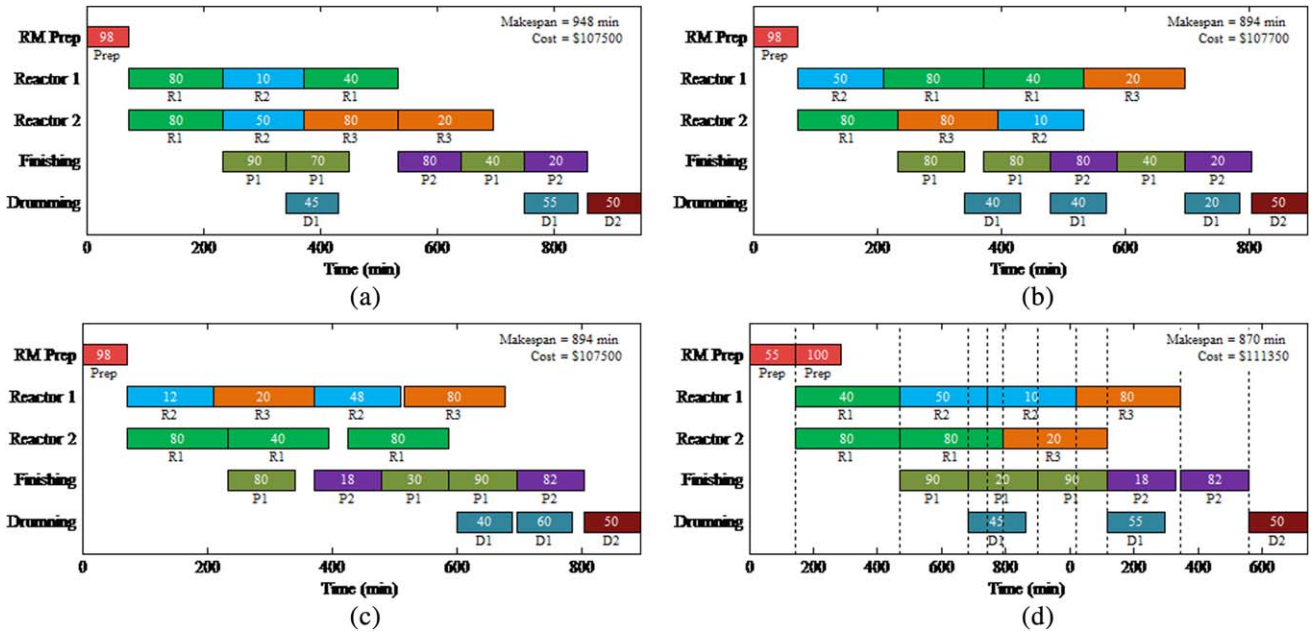

Figure 12. Gantt charts of Case 1 returned by (a) the single-level agent-based model; (b) the two-level agent-based model; (c) the discrete-time model via minimizing the cost; and (d) the continuous-time model via minimizing the makespan.

Demands are P1 100 kg, P2 100 kg, P3 50 kg, and P4 50 kg, and task changeover times are not included. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

calculated based on the discrete-time RTN with the optimality gap of 1.0% is $107,500, and the Gantt chart is shown in Figure 12c. MIP based on the continuous-time RTN returns the same optimal cost. It is seen that the cost returned by the two-level agent-based model is very good, as it is larger than the optimal value by only 0.2%.

The makespan returned by the agent-based method is also a suboptimal one and the optimal one calculated by the discrete-time RTN is 870 min. The minimum makespan returned by the continuous-time RTN greatly depends on the number of time points, which are listed in Table 6. There are no feasible solutions when the number of time points is less than or equal to 8. When nine time points are used, the minimum makespan is 948 min that is larger than the one calculated by the discrete-time RTN, or even larger than the one returned by the two-level agent-based method. To calculate the same optimal solution as the discrete-time RTN, 12 time points are needed and the computational time is 414.8 s. This is the smallest computational time required for the continuous-time RTN, and it is listed in Table 5. The schedule calculated by the continuous-time RTN is shown in Figure 12d. The vertical dash lines denote the location of the time points. Based on this comparison, the makespan calculated by the two-level agent-based method is only 2.8% larger than the optimal one.

Case 2: Increase in Order Demands Without Task Changeover Times. The second case study compares the agent-based methods with the MIP-based methods when a longer scheduling horizon is required to satisfy increased order demands. The demands for the products are doubled, that is, P1 200 kg, P2 200 kg, P3 100 kg, and P4 100 kg. The task changeover times are not taken into account in this case. The scheduling results are summarized in Table 5. The results are calculated in the same way as those in the first case study, so detailed explanations of the computational procedure are omitted.

When the continuous-time RTN is used to minimize the makespan, it can only obtain the same makespan obtained by the two-level agent-based model. The optimal value is obtained with 15 time points, because the MIP solver CPLEX stops unexpectedly for the 16 time points due to running out of memory. The solution at the stopping point for the 16 time points is worse than the solution returned using the 15 time points. The continuous-time RTN encounters more computational difficulties than the discrete-time RTN. These results have been encountered in Dow[44,45] where the discrete-time RTN is preferred.

In comparison to the optimal solutions found with the RTN, the solution quality of the two-level agent-based model is still very good. The optimal cost from the RTN is only 0.8% smaller than that returned by the two-level agent-based model while the optimal makespan is 2.5% smaller. As the problem scale increases, the computational times for all methods increase. However, those of the MIP-based methods increase more rapidly than the agent-based methods. The advantage in the computational efficiency of the agent-based method becomes more evident for the larger scheduling. The discrete-time RTN can still calculate a solution in a reasonable time while the continuous-time RTN apparently reaches its computational limit.

Case 3: Task Changeover Times. This case study increases the complexity of the scheduling problem by

Table 6. Minimum Makespan Obtained by the Continuous-Time Model Under Different Numbers of Time Points in Case 1

Time Points	8	9	10	11	12
MS (min)	Inf	948	918	894	870
CPU (s)	0.2	1.6	18.7	93.9	443.4

The optimality gap is equal to zero.

taking task changeover times into account. The MIP-based methods deal with task changeover times by introducing more constraints on the starting time and the ending time of tasks. Although the representation of these constraints in an RTN model is straight forward, these constraints considerably complicate the scheduling problem and increase the computational time for the MIP-based methods.

The scheduling results subject to task changeover times are summarized in Table 5. The demands are the same as Case 2, that is, P1 200 kg, P2 200 kg, P3 100 kg, and P4 100 kg. The continuous-time model already reached its computational limit in the previous case study, so it is not evaluated in this more complicated case with task changeover times.

The agent-based models in dealing with task changeover times require slightly more computational time than without the changeover times. However, inclusion of the changeover times has a considerable impact on the computational time of the discrete-time RTN. The computational time for calculating an equivalently good solution increases to 1315.5 s. A much more significant increase in the computational time is observed in calculating the minimum makespan of 1254 min where the computational time is 43,200 s, reaching the resource limit.

As the scheduling problem becomes increasingly complicated, the solution by the MIP-based methods becomes more and more computationally intensive. It is here that the computational efficiency of the agent-based methods is more pronounced.

Case 4: Irregular and Variable Processing Times. The processing times and the changeover times used in previous case studies are constant, having the largest common factor of 6 min. This situation facilitates the discrete-time RTN. If the length of the time period is set as 6 min, there is no modeling error. However, in practice the processing times and changeovers are frequently irregular and the largest common factor has a very small value which would require an excessive number of time points. Moreover, the processing times of some tasks are dependent on the batch sizes. These more realistic requirements expose the limitations of the discrete-time model, and an approximated model is used. The approximation results in suboptimal or even infeasible solutions.

Case 4 compares results when irregular processing times and batch size dependent processing times are encountered. The continuous-time model has modeling advantages in this scenario, so it is used in the comparison. To reduce the computational burden of the continuous-time model, the task changeover times are not included in the scheduling problem and the order demands are assumed to be as small as those in Case 1, that is, P1 100 kg, P2 100 kg, P3 50 kg, and P4 50 kg.

The data for the task processing times are changed to those listed in Table 7. There are two components in a processing time that is expressed as

$$T_P = T_P^f + T_P^v B_S \tag{14}$$

Table 7. Data of Task Processing Times in Case 4

	Prep	R1	R2	R3	P1	P2	D1	D2
Fixed (min)	69	160	134	157	18	18	30	30
Variable factor								
Linear	0	0	0	0	1	1	1	1
Quadratic	0	0	0	0	1/90	1/90	1/60	1/60
Maximum (min)	69	160	134	157	108	108	90	90

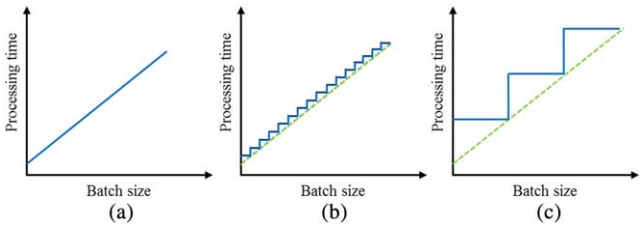

Figure 13. Representation of the batch size dependent processing time by (a) the continuous-time model, (b) the agent-based model, and (c) the discrete-time model.

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

where T_P is the total processing time, T_P^f is the fixed processing time, which is independent of the batch size, and T_P^v is the factor that produces a variable processing time dependent on the batch size B_S. The largest common factor in the fixed processing time is only 1 min. The agent-based model can use it as the simulation step. However, the value is too small for a time step in the discrete-time RTN, and a rounding approximation is required. In the approximation, the length of the time period is set as 6 min, which is the same as the value in the previous case studies. Due to the finite resolution in the time domain of the agent-based model and the discrete-time RTN, there are approximation errors in the batch size dependent processing time according to the rounding procedure in Eq. 5. However, the time step in the agent-based model is smaller than that in the discrete-time RTN, so the agent-based model has a more accurate approximation. Because the continuous-time RTN has a continuous representation of time, there is no need to partition the time domain by a set of predefined time points so the batch size dependent processing time can be represented accurately in the continuous-time RTN. The representations of the batch size dependent processing time in the three models are visualized in Figure 13.

In addition to the inaccurate approximation due to the large step size, the discrete-time RTN also needs to introduce additional tasks to model the batch size dependent processing time. Each new task represents a horizontal bar in Figure 13c so that the task can have a constant processing time. For example, the processing time for the packing task ranges from 18 to 108 min in Table 7. The time step of the discrete-time RTN is 6 min. Thus, there are 15 (=(108 − 18)/6) stairs in the piecewise function of the processing time. The original packing task is then replaced by 15 new tasks. The first task has a constant processing time of 24 min (the function value in the first segment). Similarly, the processing time is 30 min for the second task, 36 min for the third task,…, and 108 min for the 15th task. This is a disadvantage of a discrete-time model, as a more complicated model is derived. Similar to the packing task, there are 10 new tasks introduced for a drumming line. In contrast, there is no need to change the model structure for the continuous-time RTN and the agent-based model. The 1-min resolution of the agent-based model is sufficiently small.

As the discrete-time model is a rough approximation, in the case study, we only consider the results by the MIP-based methods via minimizing the makespan. The scheduling results are summarized in Table 5. The minimum makespan returned by the two-level agent-based model is 732 min, and

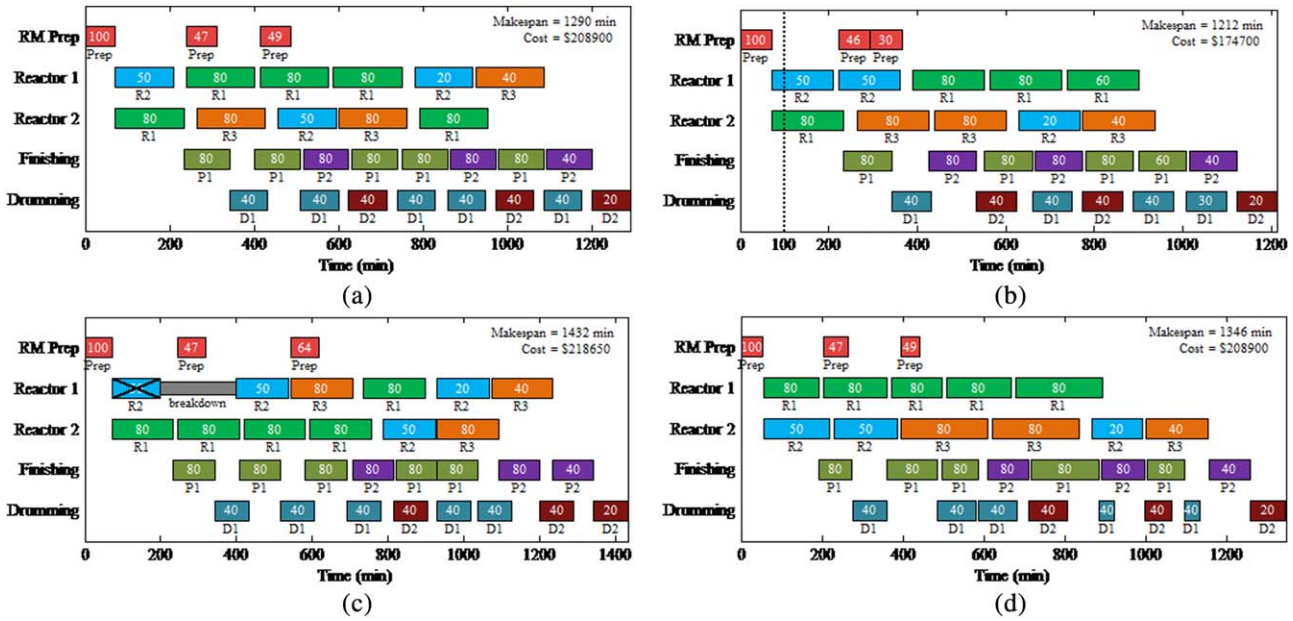

Figure 14. Scheduling results in Case 5 by the two-level agent-based model: (a) the original schedule, (b) the order cancellation, (c) the equipment breakdown, and (d) the uncertain processing times.

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

the computational time is only 4.3 s. In contrast, the computational times of the discrete-time RTN reach the limit of 43,200 s due to the increased complexity in the model. The computational time of the continuous-time RTN is 19,964.8 s, which is also much larger than that of the agent-based model. The optimal solution of the continuous-time RTN is only 0.1% less than that returned by the agent-based model. The discrete-time RTN returns a poorer solution than the agent-based model by 4.1%. The suboptimality stems from the inaccurate approximation of the processing time shown in Figure 13c.

The function of the processing time dependent on the batch size can be more complex. Generally, it can be a power function as[46]

$$T_P = T_P^f + T_P^v B_S^d \tag{15}$$

where the power d is non-negative. A nonlinear relationship of the processing time imposes severe difficulties into the MIP-based methods, because the mixed-integer linear programming (MILP) problem becomes the mixed-integer nonlinear programming (MINLP) problem. The MINLP problem is much more difficult to solve than the MILP problem. However, the inclusion of the nonlinear function has no effect on the computation of the agent-based methods. The two-level agent-based model requires 4.3 s when the variable processing time is a linear function ($d = 1$) and 4.4 s when the variable processing time is a quadratic function ($d = 2$).

Case 5: Online Scheduling Under Uncertainties. In the previous cases, the agent-based scheduling method is much faster than the MIP-based methods. This speed makes the agent-based method more appropriate to implement online and, thus, deal with various uncertainties in the production process. This case study explores the potential online implementation of the agent-based method. The scheduling results are calculated based on the two-level agent-based model. The order demands and the task changeover times are the same with those in Case 3. The scheduling results without uncertainties are shown in Figure 14a.

Online implementation has a stringent requirement on the scheduling method. The new schedule should be determined within a short time interval, for example, less than the time resolution of the scheduling model. The time step in the agent-based method is 1 min. The computational time of the offline scheduling is only 13.4 s from Table 5. Thus, the agent-based method is efficient enough for online implementation. In contrast, the MIP-based methods are rarely applied online. For example, the time step of the discrete-time RTN is 6 min. Although this resolution is so large that it poorly approximates the varying processing time in the previous case, a 6-min computational time is still not enough for the discrete-time RTN. Computational inefficiency prevents online implementation of the MIP-based methods. They are usually applied offline to provide a short-term schedule.

Three types of commonly encountered uncertainties are investigated: sudden change in demand, an equipment breakdown, and uncertain processing times. The online rescheduling under uncertainties follows the procedure displayed in Figure 10.

First, a sudden change in demand is tested. Suppose a sudden cancellation arrives at the time point of 100 min and the demands for P1 and P2 decrease from 200 to 150 kg. The Gantt chart is displayed in Figure 14b. The initial schedule before the time of 100 min follows the original solution shown in Figure 14a. When the cancellation in the order is received, the process is rescheduled starting from 100 min. As the reaction tasks Reaction 1 and Reaction 2 have already taken place, the rescheduling does not interrupt their execution. However, the Gantt chart at a later time significantly differs from the original solution, as the change in the order demand is taken into account.

Second, equipment breakdown and recovery is considered. Suppose Reactor 1 breaks down at 200 min and it is available again at 400 min. The rescheduling results are shown in Figure 14c. The schedule before the reactor breakdown is

the same with the original solution shown in Figure 14a. When the reactor breaks down, the task of Reaction 2 is executing in the reactor. So the materials that have been loaded in the reactor are wasted, and the task needs to be executed again. The process is rescheduled at two time points: one point for the equipment breakdown and the other for the equipment recovery. The breakdown period is marked in gray during which no task is allowed to execute in the equipment.

Third, uncertain processing times are taken into account. In fact, the uncertainty in the processing time of a task is very common in batch manufacturing. The processing time of the same product can vary between batches and the change in the processing time is unpredictable.

Varying processing times cause the process operations to quickly drift from a preoptimized schedule. The schedule quickly becomes unusable and requires reoptimization. Schedule drift imposes a significant burden on the person responsible for maintaining the schedule, because it requires them to continually monitor the operation to detect schedule drift and then to produce a new schedule. A promising solution to this problem is to have the schedule continuously updated in real time. Such an approach would be a huge burden for the MIP-based methods. However, the efficient agent-based methods are well suited for reactive scheduling as discussed in the previous section.

The scheduling results are shown in Figure 14d. The offline schedule is initially applied. After execution of any task, the processing time of the task is updated from process measurements. In this simulated example, the processing time is set as a value that is randomly sampled from a set of real process data. Then, the process is rescheduled according to the updated processing time. In the rescheduling, the deterministic processing times are used for the future tasks. The process is rescheduled repeatedly at the end of every task execution. Obviously, the uncertain processing times have significant effects on the scheduling results by comparing with the original schedule in Figure 14a.

Large-scale scheduling problem

To demonstrate the scalability of the agent-based scheduling methods, a challenging large-scale industrial scheduling problem was investigated. The problem comes from a manufacturing operation in The Dow Chemical Company. There are 10 processing units including seven batch reactors, two drumming lines, and one bulk-loading station. The case study considers 30 products that are manufactured via multiple stages. The number of stages required is product dependent. There are 21 storage tanks, which can be used to store the intermediate products. Among them, 12 are dedicated and 9 are shared tanks. The capacities of the tanks vary.

The diagram of the process is shown in Figure 15. Intermediate materials are prepared in Reactor 1 and stored in the intermediate tanks. If a tank is dedicated to a product, the storage policy is the FIS. If a product has no dedicated tank, it uses a shared tank and the storage policy is SIS. Depending on the final product, an intermediate product receives further processing in one or several of the reactors labeled Reactor 2 to Reactor 7. The routing is defined by the production recipe. There is no storage for the intermediate products, as they pass through these reactors so the storage policy is the NIS. All downstream reactors can be used as temporary storage tanks. After passing through all required reactions, the final product is stored in a tank. The final products are

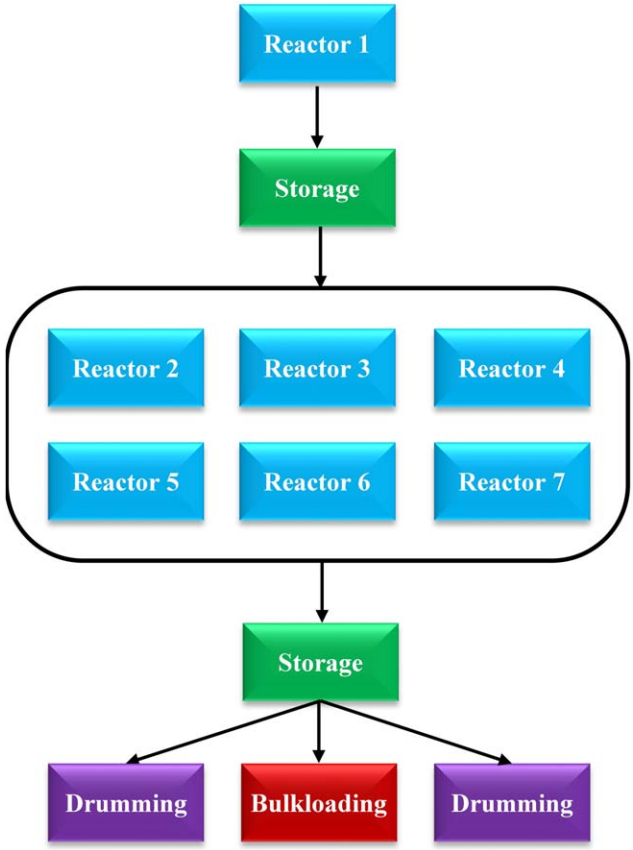

Figure 15. Diagram of the large-scale Dow scheduling example.

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

packed by the two drumming lines or the bulk-loading station. The final products are stored in large warehouse and the storage capacity is considered unlimited (UIS). The drumming lines and the bulk-loading station operate only 16 h a day, and they do not operate on weekends.

To maintain confidentiality, the RTN representation and the production data are not provided. The RTN consists of 93 tasks, 71 states, and 10 processing units. There are changeover times between the tasks. The processing time of the drumming lines and the bulk-loading station is dependent on the batch size. For the agent-based scheduling methods, all processing times and changeover times are rounded to the nearest hour and the simulation step is set as 1 h.

In this example, another objective function is investigated. The advantage of the two-level agent-based model over the single-level agent-based model is that it can predict the objective function value. The addition of global information for the scheduling agent can improve the solution quality as illustrated in the previous case studies. It also allows the agent-based scheduling method to easily adapt to different scheduling objectives.

For this case study orders for products have specific due dates. The due dates for each order can be different and, thus, one objective of production is to minimize delivery delays as much as possible. If the delivery date is after the due date, the product is tardy and tardiness is expressed as

$$T_i^{\text{tardiness}} = \max\left\{0, T_i^{\text{delivery}} - T_i^{\text{due}}\right\} \tag{16}$$

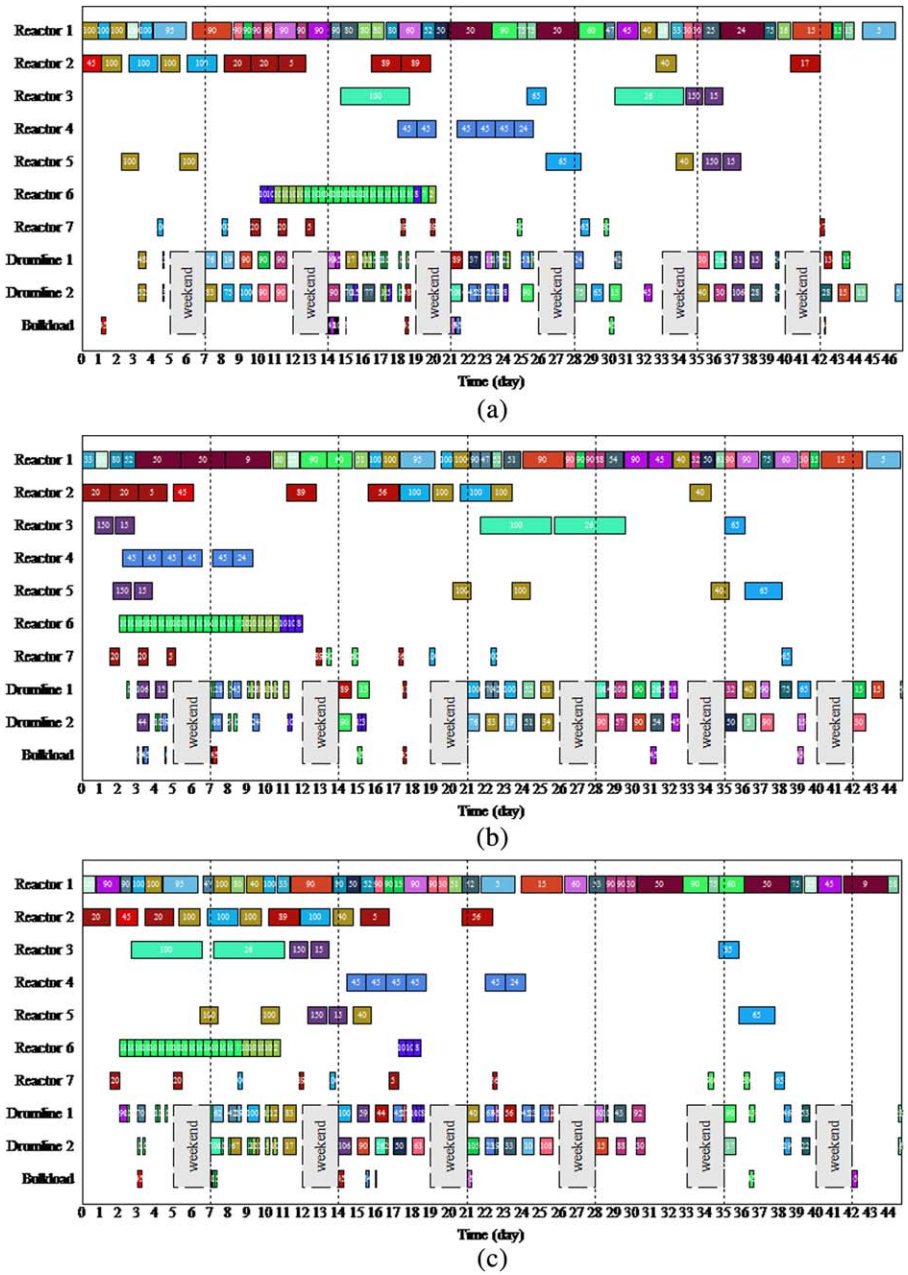

Figure 16. Scheduling results of the large-scale Dow example by (a) the single-level agent-based model, (b) the two-level agent-based model that minimizes the makespan, and (c) the two-level agent-based model that minimizes the tardiness.

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

where $T_i^{\text{tardiness}}$ is the tardiness of product i, T_i^{delivery} is the delivery date, and T_i^{due} is the due date. The objective function is to minimize the penalty function of the tardiness

$$\min \sum_i T_i^{\text{tardiness}} \tag{17}$$

To compare different approaches, the single-level model, the two-level model that minimizes the makespan, and the two-level model that minimizes the tardiness penalty function (17) are applied. The scheduling results are shown in Figure 16. The makespan returned by the single-level agent-based model is 1096 h, and the computational time is only 2.6 s. Both two-level agent-based models return the makespan of 1072 h. In this example, the two-level agent-based model does not make a large improvement in the solution quality because of the bottleneck in Reactor 1. In the Gantt chart, the line associated with the processing unit is fully occupied throughout the scheduling horizon.

The computational time for minimizing the makespan is 1213.1 s, and the computational time for minimizing the tardiness is 1309.6 s for the two-level models. Although the objective functions are distinct, the computational times are similar. A feature of the agent-based methods is that they determine the scheduling decisions by simulation instead of an iterative optimization search. Therefore, the computational time (or the simulation time) is mainly dependent on the length of the simulation (makespan) and less on the

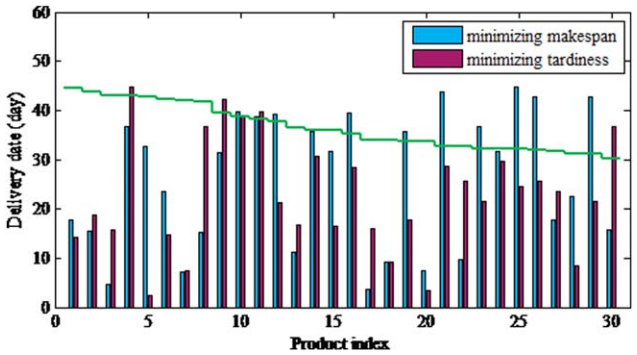

Figure 17. Delivery dates by two scheduling objective functions. The green line denotes the due date of the products.

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

specifications of the scheduling problem, for example, the objective function. Although they are much larger than the computational time of the single-level model, the computational times of the two-level models are still acceptable for an online implementation since the practical resolution in the time domain is 1 h for the process and the computational times are merely about one third of the resolution value. Even for such a large industrial-size problem, the agent-based methods are still very efficient.

The two two-level models use different objective functions to schedule the process, and the scheduling results are different as shown in Figure 16. To provide insight into how the objective functions affect the scheduling results, the delivery dates of all products are visualized in Figure 17. The due dates of 30 products range from 30 days to 45 days. In the schedule determined by minimizing the makespan, there are 10 tardy products, and the total tardiness is 59.5 days. In the schedule determined by minimizing the tardiness, the number of the tardy products reduces to 4 and the total tardiness reduces to 12.6 days. It must be noted that the demands of this case study exceeded the capacity of the process as evidenced by the operation of Reactor 1. Obviously, the objective functions have a significant impact on the scheduling results. The ability to use global information about the objective function is an advantage of the scheduling algorithm based on the two-level agent-based model while the conventional single-level agent-based model only uses the local information and, thus, has difficulty dealing with different objective functions. For both objective functions, the single-level agent-based model returns the same schedule as shown in Figure 16.

In contrast to the agent-based scheduling methods, the MIP-based methods are very computationally expensive for such a complicated real-world problem. Even in the previous much simpler example, the continuous-time model reaches its computational limits when the number of time points is 15. However, there are 41 batches in Reactor 1, which are calculated by both two-level agent-based models. This means that the number of time points of the continuous-time model would be at least 41, as the beginning of a batch should be associated with a time point. The number of 41 is just an underestimate calculated from the single unit. More time points may be required for all units. And this number based on only one processing unit is too large to be handled by the continuous-time model.

If the discrete-time RTN is applied, it will first encounter the rounding problems because the processing times are irregular and variable. The time step of 1 h used in the agent-based methods is too small to be adopted in the discrete-time model. An alternative is to choose a larger value of 6 h. If the time step is chosen as 6 h, there will be significant rounding errors as many of task changeover times are just 1, 2, or 3 h. The large time step would make it difficult to capture the events with small durations. However, even for such a large time step, the resulting model has a very large scale. When the scheduling horizon is equal to 1074 min which is close to that of the two-level agent-based models, the formulated model contains 126,912 equations and 46,518 variables (16,484 among them are discrete variables), and this does not include task changeover times, the shared tanks, and the packaging. Even with these simplifications, we have a very complicated model.

Summary of the comparisons

The features of the agent-based scheduling methods are compared to the MIP-based methods in Table 8. The principal advantage of the MIP-based methods is that the optimality of the scheduling solution can be guaranteed. However, the existence of the optimal solution in theory does not imply that it is achieved in practice. The dynamic environment of a manufacturing plant contains many barriers to implement an optimal schedule.

In a real-world scheduling problem, the solution optimality might not be attained due to three factors. First, the MIP method is applied using a mathematical model, which is really an approximation of a real process. Strictly speaking, the resulting schedule is only optimal for the simplified model rather than for the real process. As demonstrated in Case 4, the discrete-time RTN can involve significant rounding errors for the irregular and variable processing times. This model discrepancy reduces the solution quality. The optimal solution to the simplified model after the rounding procedure is no longer optimal for the real process. The solution is less optimal than that returned by the agent-based method, because the model of the agent-based method is more accurate.

Table 8. Comparison of MIP-Based Methods and Agent-Based Methods

Feature	MIP-Based	Agent-Based
Solution approach	Modeling and optimization	Programming and simulation
Model formulation	Variables and constraints	Agents and logic rules
Advantage	Optimality	Efficiency
Disadvantage	Calculation	Programming
Computational time	Long and unpredictable	Short and predictable
Handling uncertainty	Difficult	Easy
Implementation	Off line and predictive	On line and reactive
Maturity	Well established	Under developed

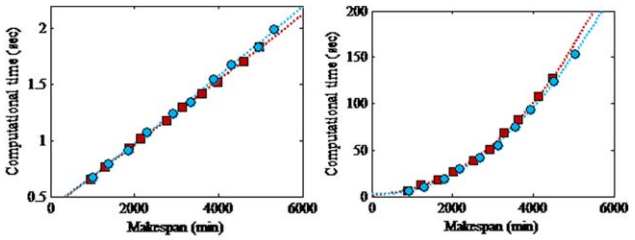

Figure 18. The dependence of the computational time on the makespan for (a) the single-level agent-based model and (b) the two-level agent-based model.

The red square denotes the data from models without task changeover times while the blue circle denotes the data from models with task changeover times. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Second, due to the limited computational resources, the optimal schedule present in theory might not be reached in practice. As demonstrated in Case 2, the continuous-time RTN only returns the same makespan with the agent-based model. Because the MIP solver runs out of memory, a better solution cannot be found in practice even if it exists in theory. Both discrete-time RTN and continuous-time RTN suffer from the computational complexity as a majority of MIP-based scheduling problems are NP-hard.[2,42] The computational time can increase exponentially as the scale of the scheduling problem expands. For a large industrial-size scheduling problem, the computational time can quickly exceed the affordable range and even an offline solution becomes very difficult.

Third, the process will soon drift from optimal schedule due to process variability. The scheduling environment in a real production process is dynamic, and it is subject to uncertain disruptions like order cancelation and equipment breakdown. It also must deal with process uncertainty such as varying batch times and minor delays in operator interventions. However, the most used and tractable MIP methods ignore uncertain factors and perform a deterministic optimization. When order demands, equipment availability, and task processing times are realized in the operating process, the predetermined optimal schedule is no longer optimal, and potential infeasible due to the strong interactions in a network batch process.

As compared to the MIP-based methods, the main benefit gained from the agent-based methods is its practicability for a complex scheduling problem. The agent-based methods determine scheduling decisions from simulation driven by the logic rules. The solution is much easier to obtain than the MIP-based methods where the solution is calculated by an iterative search. Therefore, agent-based methods need a much shorter computational time as illustrated in the case studies, especially for complicated scheduling problems.

In addition to shorter computational time, the agent-based approach has other advantages. One is the predictable computational time that varies with the length of the scheduling horizon. The computational time as a function of the makespan is visualized in Figure 18. The data are generated by continually increasing the order demands from P1 100 kg, P2 100 kg, P3 50 kg, P4 50 kg to P1 1000 kg, P2 1000 kg, P3 500 kg, and P4 500 kg. The incremental step for P1 and P2 is 100 kg, whereas it is 50 kg for P3 and P4. The computational time of the single-level agent-based model is a linear function with the makespan. This makes sense, because there are no iterative steps in the simulation and the simulation time is proportional to the length of the simulation interval, which is the makespan in a scheduling problem. The computational time of the two-level agent-based model is a quadratic function with the makespan, because it involves an embedded single-level agent-based model to predict the objective function and the computational time is squared.

Another advantage is that the computational time is not affected by the problem specifications like task changeover times. As seen in Figure 18, the inclusion of task changeover times in the scheduling problem has little impact on the computational time. However, as demonstrated in Case 3, the task changeover times significantly increased the computational time of the MIP-based methods by adding a great number of constraints in the mathematical model.

The shorter, predictable, specification-independent computational time makes the agent-based method very suitable for the online implementation where stringent demands are placed on computational time. For the MIP-based methods, the computational time is longer and unpredictable, being dependent on both process data and model specifications. The agent-based methods as efficient scheduling approaches were illustrated in Case 5.

The main disadvantage of the agent-based methods is that the solution quality is difficult to guarantee, because the scheduling decisions are made by local interactions among agents that do not use an objective function to drive global performance. However, the solution quality can be improved by the proposed two-level agent-based model, which uses both local and global information to balance solution quality and computational efficiency. As illustrated in the case studies, the solution returned by the two-level agent-based model is very close to the optimal value. The improvement in the solution quality can also be seen from Figure 18. On average, the makespan returned by the two-level model is 9.1% smaller than the value returned by the single-level model when task changeover times are not included. When the changeover times are taken into account, the average reduction in the makespan is 7.5%.

In addition to the comparison with the MIP-based methods, it is interesting to make a comparison with other heuristic methods. A great variety of heuristic scheduling methods have been developed to overcome the computational complexity of the MIP-based methods. Agent-based methods can be regarded as a kind of heuristic approach, because the aim is to provide a reasonably good solution in a much more efficient way than the MIP-based methods. However, the agent-based methods have essential differences from other heuristic methods.

There are deterministic heuristic methods, for example, the LP-based heuristic,[47] the heuristic-based decomposition,[48] and rolling horizon algorithms.[49] Stochastic meta-heuristic methods are also popular to deal with discrete decision variables, including the genetic algorithm,[50] the simulated annealing,[51] the tabu search,[52] and the swarm optimization.[53] The principle difference between the agent-based methods and these methods stems from the modeling strategies. Most heuristic methods use a centralized modeling strategy like the MIP-based methods while using different solution approaches. However, agent-based methods use a decentralized modeling strategy.

The different modeling strategies use different solution approaches and produce different performance. The computational time of the agent-based methods is predictable from the scheduling horizon and a bound exists. The computational times for other heuristic methods are unpredictable, and the upper bound is difficult to know.

Conclusions

A novel agent-based scheduling method applicable to efficient scheduling of batch processes with general network structure has been proposed. The agent-based model is different from the conventional ones designed for discrete manufacturing, because the operations of splitting, mixing, and resizing materials are allowed in the network process. The agent-based model is built from the RTN representation, and it can model various specifications, for example, task changeover times, batch size dependent processing times, and different storage policies.

To improve the solution quality, a scheduling algorithm using a two-level agent-based model was presented. The outer agent-based model can predict the objective function value by simulating the inner agent-based model. The global information of the objective function is used to determine the task selection in the outer agent-based model. This provides good balance between the solution quality and efficiency. The scheduling algorithm overcomes the drawbacks of conventional agent-based methods where agents act based on the local information alone.

The advantages of the proposed agent-based scheduling methods are demonstrated by case studies on two industrial problems including a large-scale scheduling problem. Detailed comparisons are made with the MIP methods based on the discrete-time RTN and the continuous-time RTN. The computational time of the single-level agent-based model is a linear function of the scheduling horizon whereas the computational time of the two-level agent-based model is a quadratic function. In comparison, the computational times of the MIP-based methods are much longer and unpredictable. Although the agent-based methods cannot attain a guaranteed optimal solution, the returned solution is very close to the optimal one (within 3%). Furthermore, the capability of obtaining the optimal solution for an MIP-based method fades as the scheduling problem becomes complicated. The agent-based efficient scheduling methods can be applied online to complex real-world problems containing various uncertainties, where the MIP-based methods usually fail.

Literature Cited

1. Floudas CA, Lin XX. Mixed integer linear programming in process scheduling: modeling, algorithms, and applications. *Ann Oper Res*. 2005;139(1):131–162.
2. Kallrath J. Planning and scheduling in the process industry. *OR Spectrum*. 2002;24(3):219–250.
3. Mendez CA, Cerda 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(6–7):913–946.
4. Ottino JM. Chemical engineering in a complex world: grand challenges, vast opportunities. *AIChE J*. 2011;57(7):1654–1668.
5. Wassick JM, Agarwal A, Akiya N, Ferrio J, Bury S, You F. Addressing the operational challenges in the development, manufacture, and supply of advanced materials and performance products. *Comput Chem Eng*. 2012;47:157–169.
6. You F, Castro PM, Grossmann IE. Dinkelbach's algorithm as an efficient method to solve a class of MINLP models for large-scale cyclic scheduling problems. *Comput Chem Eng*. 2009;33:1879–1889.
7. Giannelos NF, Georgiadis MC. A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. *Ind Eng Chem Res*. 2002;41(9):2178–2184.
8. Ierapetritou MG, Floudas CA. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Ind Eng Chem Res*. 1998;37(11):4341–4359.
9. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations—I. MILP formulation. *Comput Chem Eng*. 1993;17(2):211–227.
10. Maravelias CT, Grossmann IE. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2003;42(13):3056–3074.
11. Pantelides CC. Unified frameworks for optimal process planning and scheduling. Foundations of Computer-Aided Process Operations. New York: Cache Publications, 1994:253–274.
12. Sundaramoorthy A, Karimi IA. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chem Eng Sci*. 2005;60(10):2679–2702.
13. Mockus L, Reklaitis GV. Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization. *Comput Chem Eng*. 1997;21(10):1147–1156.
14. Chu Y, You F. Integration of scheduling and control with online closed-loop implementation: Fast computational strategy and large-scale global optimization algorithm. *Comput Chem Eng*. 2012;47:248–268.
15. Floudas CA, Lin XX. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng*. 2004;28(11):2109–2129.
16. Yue D, You F. Sustainable scheduling of batch processes under economic and environmental criteria with MINLP models and algorithms. *Comput Chem Eng*. 2013. Accepted. DOI: 10.1016/j.compchemeng.2013.03.013.
17. Aytug H, Lawley MA, McKay K, Mohan S, Uzsoy R. Executing production schedules in the face of uncertainties: a review and some future directions. *Eur J Oper Res*. 2005;161(1):86–110.
18. Verderame PM, Elia JA, Li J, Floudas CA. Planning and scheduling under uncertainty: a review across multiple sectors. *Ind Eng Chem Res*. 2010;49(9):3993–4017.
19. Novas JM, Henning GP. Reactive scheduling framework based on domain knowledge and constraint programming. *Comput Chem Eng*. 2010;34(12):2129–2148.
20. Mendez CA, Cerda J. Dynamic scheduling in multiproduct batch plants. *Comput Chem Eng*. 2003;27(8–9):1247–1259.
21. Yue D, You F. Planning and scheduling of flexible process networks under uncertainty with stochastic inventory: MINLP models and algorithm. *AIChE J*. 2013. Accepted. DOI: 10.1002/aic.13924.
22. Chu Y, You F. Integrated scheduling and dynamic optimization of sequential batch processes with online implementation. *AIChE Journal*. 2013. Accepted. DOI: 10.1002/aic.14022.
23. Panek S, Engell S, Subbiah S, Stursberg O. Scheduling of multi-product batch plants based upon timed automata models. *Comput Chem Eng*. 2008;32(1–2):275–291.
24. Katare S, Caruthers JM, Delgass WN, Venkatasubramanian V. An intelligent system for reaction kinetic modeling and catalyst design. *Ind Eng Chem Res*. 2004;43(14):3484–3512.
25. Kanakamedala KB, Reklaitis GV, Venkatasubramanian V. Reactive schedule modification in multipurpose batch chemical plants. *Ind Eng Chem Res*. 1994;33(1):77–90.
26. Romero J, Puigjaner L, Holczinger T, Friedler F. Scheduling intermediate storage multipurpose batch plants using the S-graph. *AIChE J*. 2004;50(2):403–417.
27. Macal CM, North MJ. Tutorial on agent-based modelling and simulation. *J Simul*. 2010;4(3):151–162.
28. Shen WM, Wang LH, Hao Q. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *IEEE Trans Syst Man Cybern C: Appl Rev*. 2006;36(4):563–577.
29. Archimede B, Coudert T. Reactive scheduling using a multi-agent model: the SCEP framework. *Eng Appl Artif Intell*. 2001;14(5):667–683.
30. Leitao P, Restivo F. A holonic approach to dynamic manufacturing scheduling. *Robot Comput-Integr Manuf*. 2008;24(5):625–634.
31. Cowling PI, Ouelhadj D, Petrovic S. A multi-agent architecture for dynamic scheduling of steel hot rolling. *J Intell Manuf*. 2003;14(5):457–470.
32. Wong TN, Leung CW, Mak KL, Fung RYK. Dynamic shopfloor scheduling in multi-agent manufacturing systems. *Expert Syst Appl*. 2006;31(3):486–494.

33. Metzger M, Polakow G. A survey on applications of agent technology in industrial process control. *IEEE Trans Ind Inform.* 2011;7(4):570–581.
34. Julka N, Srinivasan R, Karimi I. Agent-based supply chain management-1: framework. *Comput Chem Eng*. 2002;26(12):1755–1769.
35. Tatara E, Birol I, Teymour F, Cinar A. Agent-based control of autocatalytic replicators in networks of reactors. *Comput Chem Eng*. 2005;29(4):807–815.
36. Palombarini J, Martinez E. SmartGantt—an intelligent system for real time rescheduling based on relational reinforcement learning. *Expert Syst Appl*. 2012;39(11):10251–10268.
37. Argente E, Julian V, Botti V. Multi-agent system development based on organizations. *Electron Notes Theor Comput Sci.* 2006;150(3):55–71.
38. Verstraete P, Saint Germain B, Valckenaers P, Van Brussel H, Belle J, Hadeli H. Engineering manufacturing control systems using PROSA and delegate MAS. *Int J Agent-Oriented Softw Eng*. 2008;2(1):62–89.
39. Jennings NR. On agent-based software engineering. *Artif Intell*. 2000;117(2):277–296.
40. Vallejo D, Albusac J, Mateos JA, Glez-Morcillo C, Jimenez L. A modern approach to multiagent development. *J Syst Softw*. 2010;83(3):467–484.
41. Sundaramoorthy A, Maravelias CT. A general framework for process scheduling. *AIChE J*. 2011;57(3):695–710.
42. Toptal A, Sabuncuoglu I. Distributed scheduling: a review of concepts and applications. *Int J Prod Res*. 2010;48(18):5235–5262.
43. Brucker P. Scheduling Algorithms. Berlin: Springer, 2007.
44. Wassick JM. Enterprise-wide optimization in an integrated chemical complex. *Comput Chem Eng*. 2009;33(12):1950–1963.
45. Wassick JM, Ferrio J. Extending the resource task network for industrial applications. *Comput Chem Eng*. 2011;35(10):2124–2140.
46. Modi AK, Karimi IA. Design of multiproduct batch processes with finite intermediate storage. *Comput Chem Eng*. 1989;13(1–2):127–139.
47. Blomer F, Gunther HO. LP-based heuristics for scheduling chemical batch processes. *Int J Prod Res.* 2000;38(5):1029–1051.
48. You F, Grossmann IE. Design of responsive supply chains under demand uncertainty. *Comput Chem Eng*. 2008;32:3090–3111.
49. Dimitriadis AD, Shah N, Pantelides CC. RTN-based rolling horizon algorithms for medium term scheduling of multipurpose plants. *Comput Chem Eng*. 1997;21:S1061–S1066.
50. Goncalves JF, Mendes J, Resende MGC. A hybrid genetic algorithm for the job shop scheduling problem. *Eur J Oper Res.* 2005;167(1):77–95.
51. Vanlaarhoven PJM, Aarts EHL, Lenstra JK. Job shop scheduling by simulated annealing. *Oper Res.* 1992;40(1):113–125.
52. Ben-Daya M, Al-Fawzan M. A tabu search approach for the flow shop scheduling problem. *Eur J Oper Res*. 1998;109(1):88–95.
53. Liao CJ, Tseng CT, Luarn P. A discrete version of particle swarm optimization for flowshop scheduling problems. *Comput Oper Res.* 2007;34(10):3099–3111.

Manuscript received Aug. 24, 2012, and revision received Jan. 15, 2013.