An Intuitive and Efficient Approach to Process Scheduling with Sequence-Dependent Changeovers Using Timed Automata Models

ARTICLE *in* INDUSTRIAL & ENGINEERING CHEMISTRY RESEARCH · FEBRUARY 2011

Impact Factor: 2.59 · DOI: 10.1021/ie101652d

3 AUTHORS:

Subanatarajan Subbiah
ABB Corporate Research Center, Germany
17 PUBLICATIONS **137** CITATIONS
SEE PROFILE

Christian Schoppmeyer
Technische Universität Dortmund
18 PUBLICATIONS **29** CITATIONS
SEE PROFILE

Sebastian Engell
Technische Universität Dortmund
506 PUBLICATIONS **3,243** CITATIONS
SEE PROFILE

I&EC research
Industrial & Engineering Chemistry Research

ARTICLE

pubs.acs.org/IECR

An Intuitive and Efficient Approach to Process Scheduling with Sequence-Dependent Changeovers Using Timed Automata Models

Subanatarajan Subbiah,* Christian Schoppmeyer, and Sebastian Engell

Process Dynamics and Operations Group, Department of Biochemical and Chemical Engineering, Technische Universität Dortmund, 44221 Dortmund, Germany

ABSTRACT: In the process industries where multiple products have to be produced in the batch mode, the optimal assignment of the operations to the available resources and their sequencing can contribute considerably to economic success. Among the several methods proposed to model and solve batch scheduling problems, techniques based on a reachability analysis of timed automata (TA) models have gained attention recently. The appeal of the approach is the modular, intuitive, and straightforward graphical modeling of complex scheduling problems, and an efficient solution technique based upon reachability algorithms. In this contribution, we present an introduction to the TA-based approach to scheduling and specifically address the problem of batch scheduling with sequence-dependent setup and changeover times. In the TA-based approach, the resources, recipes, and additional timing constraints are modeled independently as sets of (priced) timed automata. The sets of individual automata are synchronized by means of synchronization labels and are composed by parallel composition to form a global automaton. A cost-optimal symbolic reachability analysis is performed on the composed automaton to derive schedules with the objective of minimizing makespan. The TA models of the recipes are extended here to include setup times as well as sequence-dependent changeovers. The performance of the approach to model and to solve real-world scheduling problems with sequence-dependent changeovers is demonstrated for two different case studies. A comparative study on the TA-based approach with various MILP formulations is performed on a famous case study from the literature, and the results are discussed.

1. INTRODUCTION

Multiproduct and multipurpose batch plants are widely used in the production of high valued fine or special chemicals and in the food industries whenever a variety of different products have to be produced. In such plants, one of the several challenges that plant managers face is to schedule the recipe operations such that the resources are utilized optimally. Most of the solution approaches proposed in recent years solve such problems by modeling them as a mathematical program (mostly MILP) and applying commercial solvers to solve them.

1.1. Mixed-Integer Programming Approaches. A tremendous amount of effort has been invested in recent decades in model building and solution approaches to handle scheduling and medium-term planning problems in the batch industries. The standard approach to tackling a scheduling problem is to transform the problem into a (usually large) set of equations with integer and real variables and to obtain the solution with MILP or MINLP solvers. The research has led to the development of several problem formulations that range from models that are problem specific to very generic ones. Significant contributions to and reviews[1−3] of different mathematical formulations based on continuous-time,[4−10] discrete-time,[11,12] global events, unit-specific events, and slot-based[13,14] formulations have been published in the past decade. Many of the recent mathematical formulations are based on a continuous-time representation; however, both discrete-time formulations and continuous-time formulations have their limitations and strengths.

One of the problem characteristics that makes batch scheduling challenging is the presence of sequence-dependent changeover procedures. Processing units which process different materials usually require setup and changeover operations to switch the production from one product to another. In most contributions to batch scheduling, the setup times and the changeover times are either neglected or added to the durations of the production steps. In the presence of a significant sequence dependency of the changeover times, the utilization times of the processing units are strongly influenced by the sequence in which the products are produced. To tackle this widespread problem, it is necessary to use efficient scheduling models and solution techniques.

In ref 15, the authors present a comprehensive review on production scheduling problems with setup times. They classify the scheduling problems based on the type of the process, as batch or continuous, and categorize the various approaches based on the production environments. The authors in ref 16 present an extensive review on scheduling problems with setup procedures particularly focusing on production environments with a flow shop structure. Brucker and Thiele[17] proposed a branch-and-bound method based on a disjunctive graph representation to solve makespan minimization problems in job shop and open shop environments which include sequence-dependent setup times. They demonstrated the applicability of the method proposed by testing it on a large set of problems. Artigues and Feillet[18] also proposed a branch-and-bound method based on a disjunctive

Special Issue: Puigjaner Issue

Received: August 2, 2010
Accepted: December 20, 2010
Revised: December 14, 2010
Published: February 21, 2011

graph representation to solve job shop scheduling problems with sequence-dependent setup times and with the objective of minimizing the makespan. The branch-and-bound method proposed was tested on the problems discussed in ref 17 showing that the optimality gap was reduced in comparison to that in ref 17.

Kelly and Zyngier[19] proposed a MILP formulation in discrete time where the key feature is to use specific variables for each unit to track the unit-specific events. The formulation is applicable to both batch and continuous processes. Several mathematical formulations based on continuous-time formulations were also proposed. Hui et al.[20] presented a MILP formulation for short-term scheduling of multistage, multiproduct batch plants. Erdirik-Dogan and Grossmann[21] proposed two MILP formulations that include the impact of the changeovers which are sequence-dependent. One of the MILP formulations is based on underestimating the impact of changeover durations on the production plan, and the other is based on an exact estimation of the impact on the changeovers. However, the latter formulation mentioned above led to larger problems and increased the computational effort in comparison to the former formulation. Following this work, the same authors proposed another method[22] to handle the planning and the scheduling problem by decomposing the original problem into two levels. The first level consists of a planning model where the mass balances are aggregated over time, and the second level consists of a detailed scheduling model based on slots. The planning model in the upper level is solved to predict the number of slots required to solve the scheduling model on the lower level. In contrast to continuous-time formulations based on a single-time grid, MILP formulations based on multiple-time grids were proposed in refs 23 and 24. While most contributions focused on makespan minimization problems, the authors in ref 25 presented a MILP formulation in continuous-time to handle short-term scheduling of multistage batch plants with intermediate due dates. The formulation proposed is based on general precedence and uses different sets of binary variables to handle the decisions of allocating and sequencing the operations on the resources. In ref 26, the authors present a rule-based heuristic using a genetic algorithm to solve single-stage multiproduct batch processes with up to 200 jobs.

1.2. Timed Automata-Based Scheduling. In contrast to the equation-based modeling of scheduling problems using mathematical programs, graph-theoretic approaches have also been used to solve combinatorial problems in process scheduling. One such technique is the S-graph approach introduced in refs 27−29. Another relatively new method used to model and solve batch scheduling problems using a graph-based search is by means of timed automata (TA). Finite state machines are a modeling framework to model systems with discrete behaviors. Timed automata are extensions of finite state automata with a set of clocks that can be used to model and to analyze timed systems with discrete dynamics.[30] The clocks measure the time between events and enable modeling of the timing behavior of the system. It is simple and straightforward to model a complex system in a decomposed fashion as a set of often small and individually defined timed automata. In addition to the advantages for modeling, TA can serve as a computational model which can be analyzed rigorously by reachability analysis. Efficient tools such as IF,[31] Uppaal,[32] and Kronos[33] have been developed for modeling and analyzing systems modeled as networks of TA. Recent contributions have extended the range of reachability analysis from a qualitative to a quantitative evaluation of the behavior of the system, thereby making it applicable in areas such as controller synthesis, optimization, and production scheduling. Scheduling based on timed automata and reachability analysis started with the seminal papers of Fehnker[34] and Abdeddaim and Maler.[35]

The extension of TA to priced timed automata or weighted timed automata[36] with the notion of costs that include the costs for transitions and cost rates for the periods of stay in locations further motivated the use of an approach for scheduling problems, as the concept of priced TA enables the formulation of more complex objective functions. The introduction of priced TA led to the extension of the reachability analysis of TA to cost-optimal reachability analysis. Cost-optimal reachability analysis aims at finding a path from the initial state to a target state with optimal cost.[32,37,38] Specialized software tools like Uppaal-CORA[32] and TAOpt[39,40] are available and perform a cost-optimal reachability analysis of priced TA.

Solutions of benchmark problems proposed by the operations research community on makespan minimization problems in job shops using the TA-based approach were reported in refs 35 and 40. An extension of the approach to handling job-shop problems with uncertainties in task durations is discussed in ref 41. Similar to many branch-and-bound approaches to handling scheduling problems, the TA-based approach suffers from the problem of combinatorial explosion, calling for efficient pruning procedures to reduce the search space. In order to handle job shop problems, efficient reduction techniques that reduce the state-space were proposed in refs 35 and 40. Panek et al.[39] introduced the idea of computing lower bounds for makespan minimization problems in job shops by embedding linear programming (LP) into the cost-optimal reachability analysis to reduce the search-space. This technique improved the overall performance particularly for large-scale problems, as a considerable part of the search tree was pruned due to the lower bounds computed by the LP. This however is achieved at the expense of solving many embedded LPs. The state-space reduction methods for job shops were recently extended to handle problem classes with more complex constraints[42] such as production processes with diverging and converging material flows, blending of products, cyclic material flows, and nonintermediate storage policies. In ref 42, the performance of the TA-based approach with various state-space reduction methods in comparison to a state-of-the-art MILP formulation was discussed for a well-known case study proposed by Kallrath.[1] The results from refs 40 and 42 showed that the performance of the approach is comparable or superior to that of state-of-the-art MILP approaches.

The main aim of this paper is two-fold: (1) to give a comprehensive introduction to the modeling of scheduling problems by timed automata and the techniques to solve the resulting optimization problems by reachability analysis and (2) to propose an approach to modeling batch scheduling problems with sequence-dependent changeovers. The outline of the paper is as follows: In section 2, the idea of modeling a scheduling problem using timed automata in a modular fashion is explained using an illustrative example. In section 3, the technique of composing the sets of automata is explained; then the solution approach to solving the problem based on reachability analysis is presented. In section 4, an approach to modeling scheduling problems with sequence-dependent changeovers is proposed. In section 5, the design of the prototypic tool Timed Automata Optimizer (TAOpt) developed by the Process Dynamics and Operations group, which realizes the TA-based approach to scheduling along with a short overview on the different search strategies to guide the exploration and various reduction techniques to enhance the search, is described.

In section 6, the performance of the TA-based approach is tested on two real-world case studies, and the results are presented, followed by a comparative study with various MILP formulations on a well-known example from the literature. Finally, we present a summary of the work and main conclusions in section 7.

2. MODEL BUILDING USING TA

In this section, a brief introduction to timed automata (TA) is presented first. Model building using timed automata (TA) and the solution approach based on a reachability analysis of TA are explained by means of an illustrative example.

Timed automata (TA) are extensions of finite state automata with the notion of clocks to model discrete-event systems with timing behaviors. For formal definitions of the syntax and semantics of TA, see refs 30, 43, and 44, and for priced-TA, see refs 32, 36, 45, and 46. A timed automaton consists of a finite set of discrete *locations* and *transitions* between the locations. To express the behavior of the system with respect to time, a finite set of real-valued variables called *clocks*, $c_i \in \mathbb{R}^{\geq 0}$, are used. The values of the clocks increase continuously at a rate $\dot{c}_i = 1$ and cannot be stopped but can be reset to zero. At any instant, the value of a clock indicates the time elapsed since the last time when the clock was reset. The *state* of a TA is composed of a discrete location and vectors of clock values. Discrete transitions in TA switch the state from one location to another one and may reset the clocks. Time transitions in TA increase the values of all clocks by the same value while the system stays in a location. A run of a TA is defined as a sequence of states and state transitions. The state transitions in a run are alternating time transitions and discrete transitions. A run of a TA characterizes a possible evolution of the system.

In a TA, only a single location can be active at any point of time, and a discrete transition indicates the change of the active location. Transitions between the locations must be enabled before they can be taken, and to enable a transition, all enabling conditions associated with the transition must evaluate to true. Enabling conditions for transitions are called *guards*. The guards are conjunctions of constraints on the clock valuations of the form $c_i \otimes n$ or $(c_i - c_j) \otimes n$, where $\otimes \in \{<, \leq, ==, \neq, >, \geq\}$ and $n \in \mathbb{R}^{\geq 0}$. A different type of conditions, called *invariants*, must be satisfied for a location to become and remain active. The invariants are conjunctions of clock constraints such as $c_i \leq n$ where $n \in \mathbb{R}^{\geq 0}$. The automaton is forced to leave the location when the invariant condition evaluates to false. If no transition is enabled at this state, the TA is blocked. The state of passive components (e.g., the current configuration of a resource, amount of materials present in a storage tank) can be represented by *shared variables* denoted by $s_1, s_2, ..., s_i$. The term *shared* refers to the fact that these variables can be manipulated by any automaton which is a part of the network of TA that represents the system. Guards on the transitions can include conjunctions of arbitrary arithmetic formula involving such shared variables.

One of the main aspects of the TA framework is that the models can be created in a modular fashion. Interactions between automata can be realized using synchronized transitions (i.e., two or more automata change their locations simultaneously by taking the labeled individual transitions in the automata). When a transition is executed, the actions associated with it are performed (e.g., resets of clocks, changes of the values of the shared variables, sending of synchronization labels). The following types of actions are defined in most TA implementations:

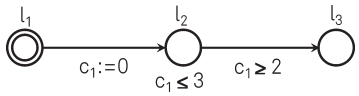

Figure 1. Example of a simple timed automaton.

- Clock resets on transitions: When a transition is taken, a clock reset $c_i := 0$ action attached to it assigns the value 0 to the clock c_i. Note that the clock is not stopped; i.e. the clock rate $\dot{c} = 1$ remains unchanged.
- Actions on shared variables: Each action $s_i := f(s_1, s_2, s_3, ..., s_k)$ evaluates an arbitrary function f of the shared variables and assigns the result to the variable s_i.

Figure 1 shows a simple TA. The locations are depicted as circles; the initial location of the automaton is depicted by double circles; and the transitions are depicted as arcs. It has a clock c_1; three discrete locations, l_1, l_2, and l_3; and two transitions, one from location l_1 to l_2 and another from l_2 to l_3. Initially, the TA is active in location l_1. The initial location l_1 has no invariant condition; hence, the TA can spend an arbitrary amount of time staying in this location. When the transition from location l_1 to l_2 is taken, a clock reset action is performed on clock c_1, which is reset to zero. After the transition is taken, location l_2 becomes active and the clock value increases at the rate $\dot{c} = 1$. The value of the clock now shows the time elapsed since the occurrence of the transition to state l_2. Location l_2 has an outgoing transition with the guard condition $c_1 \geq 2$. This states that the transition can only be taken if the time elapsed since the last clock reset is greater than or equal to 2. The invariant condition $c_1 \leq 3$ associated with the location l_2 specifies that the automaton can stay there for at most 3 time units since the last transition which reset the clock was taken, and the transition from l_2 must be taken before the invariant condition is violated. As a consequence, the transition has to occur in the interval $2 \leq c_1 \leq 3$.

In the next subsection, the modeling of batch scheduling problems is demonstrated by means of an illustrative example.

2.1. Toy Example Process. In the considered example process, two products A and B are manufactured according to recipes in a production environment with two units U_1 and U_2. The recipe to produce one batch of product A consists of two operations: op_{1A} followed by op_{2A}. In the production step op_{1A}, the raw material S_{Raw-A} is processed in unit U_1 for 5 time units to produce the intermediate material S_{Int-A}. Following this step, in the next operation op_{2A}, the intermediate material S_{Int-A} is processed in unit U_2 for 5 time units to produce the final end product S_A. Similarly, the recipe for one batch of product B consists of two operations: op_{1B} succeeded by op_{2B}. The raw material S_{Raw-B} is processed in U_1 for 7 time units to produce the intermediate material S_{Int-B}, and the intermediate material is further processed in U_2 for 2 time units to produce the final end product S_B. There exist individual storage units for the raw materials, the intermediate materials, and the end products, and the capacity of all storage units is assumed to be unlimited. The goal is to produce one batch of product S_A and one of product S_B within a minimum period of time. It is assumed that the durations of the operations are deterministic, and the availability of the units and the production orders are known at the beginning of the scheduling horizon.

A standard method used to represent a batch process is to use the framework of state-task networks (STN)[12] or resource-task networks (RTN).[47] According to the RTN framework, the process and the plant are represented graphically as networks consisting of three types of nodes, namely, *states*, *tasks*, and *resources*, and the nodes are connected by directed *arcs*. The raw

materials, the intermediate materials, the end products, and the byproduct are represented by the node type defined as *state*. The production steps and the operations are represented by the node type defined as *task*. The processing units, the storage units, and the pieces of equipment are represented by the node type defined as *resource*. The *arcs* connect the different nodes with each other and represent either the utilization of equipment or material flows. Weights on the arcs with fractional numbers describe the percentage of the material transported to the successor node.

The resource-task network (RTN) of recipe A and recipe B is shown in Figure 2. In the RTN, the raw materials (S_{Raw-A}, S_{Raw-B}), the intermediate materials (S_{Int-A}, S_{Int-B}), and the final products (S_A, S_B) are represented as circles. The operations (op_{1A}, op_{2A}, op_{1B}, and op_{2B}) are represented as rectangles and connected to their respective resources U_1 and U_2.

When modeling the example process described above by a network of TA, the following restrictions are taken into account:

- Each operation of a recipe can only be in one of the states—waiting to execute in a resource (or) executing in a resource (or) finished.
- Each resource can only be in one of the states—idle and not performing an operation or busy executing an operation assigned to it.
- A recipe operation may not be interrupted, and thus after starting its execution, the resource occupied by it can only be released after the operation has been completed.
- At any point of time, a recipe operation cannot be executed in more than one resource.
- At any point of time, a resource cannot execute more than one recipe operation.

The above example process can be modeled as a network of timed automata. The following sections describe the modeling of the example process introduced above using TA. The process is modeled in a modular fashion. The recipes and the resources are modeled individually as TA and represented as *recipe automata* and *resource automata*, respectively. The modular TA model of the example process is shown in Figure 3.

2.1.1. Recipe Automata. Automata R_1 and R_2 represent the *recipe automata* of recipes A and B. Each operation of a recipe is represented by two locations, *wait*—where the operation is waiting to be executed in the corresponding resource—and *execute*—where the corresponding resource is executing the operation. The *wait* and the *execute* locations of the operation op_{1A} are labeled as *wait* op_{1A} and *exec* op_{1A}, respectively. Since the state in which operation op_{1A} has finished its execution is equivalent to the state in which operation op_{2A} is waiting to execute, the former can be omitted. Similarly, the state in which operation op_{2A} has finished execution is equivalent to the termination of recipe A. So, the former is omitted, and a location called *finish* A is defined to indicate the completion of recipe A.

Transitions that represent starting an operation and finishing an operation are represented with labels *start*(op) and *finish*(op), respectively, with the operation index as a suffix. Starting the execution of operation op_{1A} in resource U_1 is represented by a transition labeled *start*(op_{1A}), and finishing the execution of operation op_{1A} is represented by a transition labeled *finish*(op_{1A}).

The precedence constraint between the operations op_{1A} and op_{2A} in recipe A is modeled by the transition from location *exec* op_{1A} to *wait* op_{2A} with label *finish*(op_{1A}). This ensures that operation op_{2A} can start only after executing operation op_{1A}. Each recipe automaton contains a clock to model the timing of

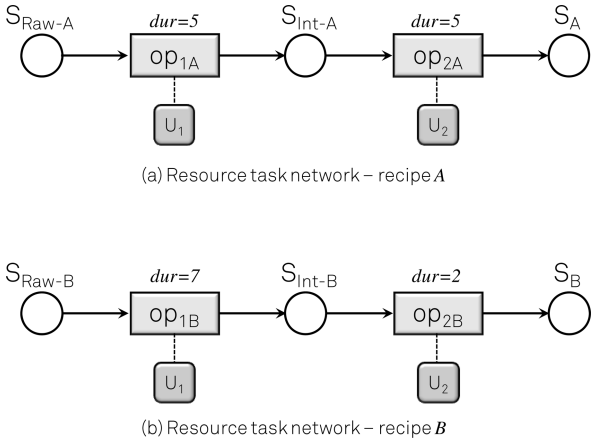

Figure 2. Resource task network of recipe A and recipe B.

the operations. In the recipe automaton R_1, clock c_1 measures the duration of the operations with respect to recipe A; similarly, in recipe automaton R_2, clock c_2 is used. Every transition that represents the starting of an operation contains the action of resetting the clock of the corresponding recipe automaton to 0. Every *exec op* location of an operation has an invariant condition which states that the location must be active only for an elapsed duration which is less than or equal to the given processing duration of the corresponding operation. Every transition from an *exec op* location that represents finishing of an operation has a guard condition that states that the operation should be executed for at least the given processing duration before starting the next operation. The guard conditions and the invariant conditions ensure that the operations are executed for the corresponding durations only. The invariant $c_1 \leq 5$ in location *exec* op_{1A} of operation op_{1A} forces recipe automaton R_1 to leave location *exec* op_{1A} once the operation duration of 5 units has expired. The guard $c_1 \geq 5$ on the finishing transition of operation op_{1A} ensures that the operation is executed for exactly 5 time units.

2.1.2. Resource Automata. For each processing unit, a separate resource automaton is created. Each resource automaton consists of a single *idle* location, which represents a case where the resource is not occupied, and a *busy* location for each of the recipe operations that it could execute, representing the execution of the corresponding operation. For each operation that can be performed in a resource, a transition from *idle* to *busy* and a transition from *busy* to *idle* exists. The *idle* to *busy* transition represents the allocation of the resource in order to start executing an operation, and the *busy* to *idle* transition represents the release of the resource when the corresponding operation has been finished. The actions on the resources (allocation and release) are synchronized with the corresponding transitions of the recipe automata. For the example considered, the *idle* location of the resource automaton U_1 is represented by $idle_1$, and the *busy* locations of operations op_{1A} and op_{1B} which this resource can execute are represented by locations *busy* op_{1A} and *busy* op_{1B}, respectively. The allocation of resource U_1 when starting operation op_{1A} of recipe A is modeled by synchronizing the transition from *wait* op_{1A} to *exec* op_{1A} in automaton R_1 and the transition from $idle_1$ to *busy* op_{1A} in automaton U_1 with the synchronization label *start*(op_{1A}). Similarly, the release of resource U_1 after finishing operation op_{1A} is modeled by synchronizing the transitions *exec* op_{1A} to *wait* op_{2A} and *busy* op_{1A} to $idle_1$ with the synchronization label *finish*(op_{1A}).

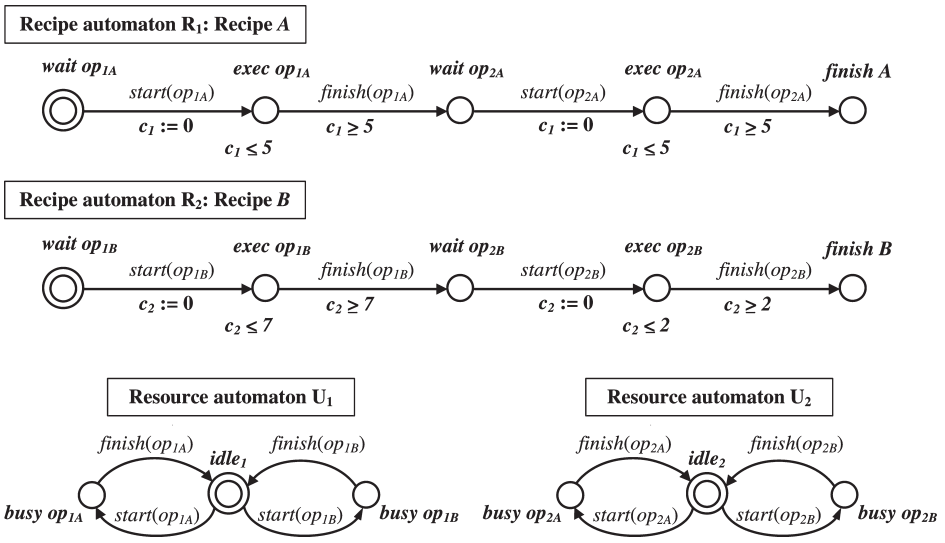

Figure 3. Timed automata submodels of the example process.

3. SOLUTION OF SCHEDULING PROBLEMS BY REACHABILITY ANALYSIS

Once the individual sets of recipe automata and resource automata have been created in a modular fashion, the scheduling problem can be solved using the *reachability analysis*. As a first step of the reachability analysis, the sets of automata are combined using a technique known as *parallel composition*[30,48] to form a composed global automaton that represents the complete scheduling problem as a directed graph.

3.1. Parallel Composition. The technique of composing the individual automata of the recipes and of the resources, shown in Figure 3, into a global automaton (GA) is explained in this section. The global automaton (GA) is constructed by creating combinations of all locations, transitions, and conditions of the individual automata (R_1, R_2, U_1, and U_2). The parallel composition of the automata for the example process is performed step by step. Initially, the recipe automata R_1 and R_2 are composed together, which is represented as $R_1||R_2$. The initial location of the composition $R_1||R_2$, represented by l_0, is a combination of the initial locations of R_1 and R_2: l_0:= (*wait* op_{1A}, *wait* op_{1B}). The invariant inv_0 of the initial location l_0 of $R_1||R_2$ results as an intersection of the individual invariants: $inv_0 := inv_{op1A} \cap inv_{op1B}$. Location l_1 of the composed automaton $R_1||R_2$ combines the second location of automaton R_1 and the initial location of R_2: l_1:= (*exec* op_{1A}, *wait* op_{1B}). Similarly, the other locations of the composed automaton $R_1||R_2$ are obtained: l_2:= (*wait* op_{2A}, *wait* op_{1B}), l_3:= (*exec* op_{2A}, *wait* op_{1B}), l_4:= (finish *A*, *wait* op_{1B}), l_5:= (*wait* op_{1A}, *exec* op_{1B}), hellip, l_{24}:= (finish *A*, finish *B*).

The transitions of the composed automaton $R_1||R_2$ are combinations of the transitions of the individual automata R_1 and R_2. The sets of transitions of the individual automata R_1 and R_2 are extended by an empty transition. The transitions in the automaton $R_1||R_2$ are constructed by combining either a transition of R_1 with the empty transition of R_2 or the empty transition of R_1 with a transition of R_2. The transition from location l_0 to l_1 is constructed as $trans_{0\rightarrow1} := start(op_{1A}) \times \{\}$. Similarly the other transitions of the composed automaton are created, and the automaton that results after composing R_1 and R_2 is shown in Figure 4. For the sake of clarity, the guards, the invariants, and the actions of the composed automaton are not shown.

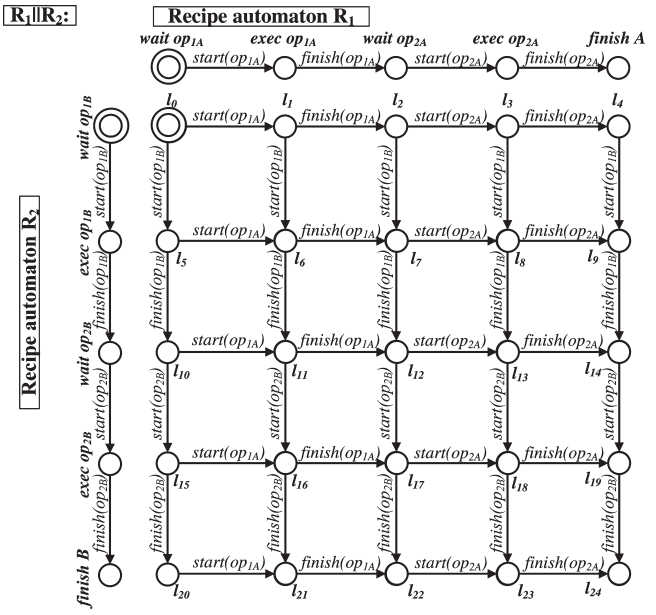

Figure 4. The composed automaton $R_1||R_2$.

To obtain the global automaton $GA := R_1||R_2||U_1||U_2$, the resource automata U_1 and U_2 have to be composed with the previously generated automaton $R_1||R_2$ shown in Figure 4. Similar to the description above, all of the locations and the transitions of the automata U_1 and U_2 are combined with the composed automaton $R_1||R_2$. This yields the global automaton with the initial location l_0 as (*wait* op_{1A}, *wait* op_{1B}, $idle_1$, $idle_2$). The synchronization labels on the transitions of the recipe automata and the resource automata enable modeling of the mutual exclusion property of the resources and exclude certain combinations of locations from the global automaton; e.g., the transition from location l_1:= (*exec* op_{1A}, *wait* op_{1B}, $busy_{1A}$, $idle_2$) to location l_6:= (*exec* op_{1A}, *exec* op_{1B}, $busy_{1B}$, $idle_2$) is not allowed, since resource automaton U_1 is in location *busy* op_{1A} at l_1 and a transition from *busy* op_{1A} to *busy* op_{1B} does not exist in this automaton. Following this scheme, invalid combinations are excluded from the global automaton, and locations without any

dx.doi.org/10.1021/ie101652d |*Ind. Eng. Chem. Res.* 2011, 50, 5131–5152

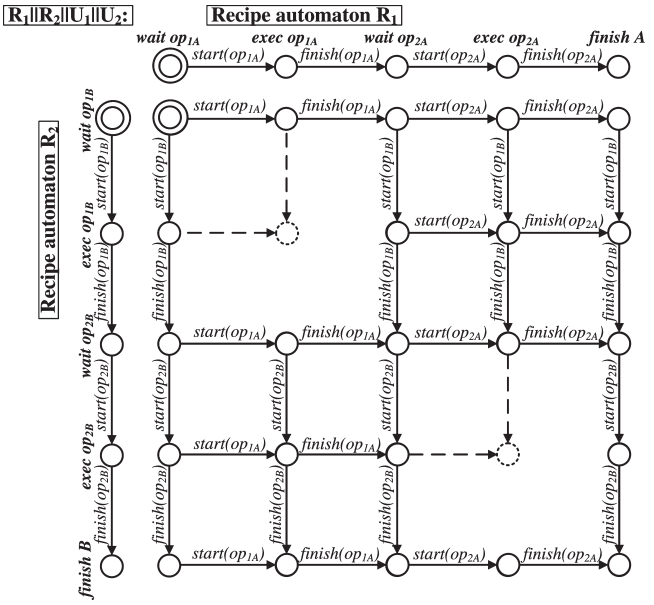

Figure 5. Global automaton $GA := R_1||R_2||U_1||U_2$, invalid locations represented by dotted lines.

incoming transitions, except of the initial location, are also removed. The complete global automaton is depicted in Figure 5. The invalid locations and their incoming transitions are marked by dotted lines.

The global automaton shown in Figure 5 has a single initial location ($wait\ op_{1A}$, $wait\ op_{1B}$, $idle_1$, $idle_2$) which represents the case where the recipe operations op_{1A} and op_{1B} are waiting to execute and the resources U_1 and U_2 are idle. The final target location of the global automaton is ($finish\ A$, $finish\ B$, $idle_1$, $idle_2$), which represents the case where all of the operations with respect to recipes A and B have finished their execution and resources U_1 and U_2 are back to the idle state.

Every run through the locations of the global automaton from the initial location where all operations are waiting to start their execution to the final target location where all recipe operations have finished their execution represents a feasible production schedule. Certain locations in the global automaton do not have conditions that limit the period for which the location can be active, and hence the automaton can spend an arbitrary amount of time staying in such locations (e.g., for the initial location). Due to the continuous nature of the clock variables and nonempty staying intervals in locations, an infinite number of possible delays exist. This feature makes the set of runs of the global automaton infinite; thus an enumerative approach to the cost-optimal reachability problem is infeasible. One of the methods to make this problem tractable is to use the technique of *zone abstraction*.[49−51] A *zone* Z is defined as the conjunction of simple inequalities such as $c_i - c_j \leq b$ or $c_i \leq b$ or $c_i \geq b$ where $c_i, c_j \in C$ and $b \in \mathbb{R}^{\geq 0}$. Zones are polyhedra in the clock space, and their form depends solely on the number and the type of guard conditions and invariants. A zone of a TA might contain infinitely many clock valuations $u \in Z$. For priced TA, the zones are extended by linear cost functions that assign a cost to each clock valuation in it.

Combinations (l, Z) of locations and zones form *symbolic states* q^S which are elements of the *symbolic state space* Q^S. The symbolic semantics are defined in terms of a symbolic transition system in which zones Z replace the single clock valuations u. A symbolic run of a TA is a sequence of symbolic states from Q^S and symbolic transitions in which zones are transformed by intersections of guards and invariants, time delay operators $Z^{\uparrow}$, and clock reset operators. Each symbolic transition from one symbolic state to another transforms the current zone to a new zone by applying a sequence of such operations. Each run contains an infinite number of single paths with different clock valuations $u \in Z$ but which traverse the same sequence of locations. The main advantage of zone abstraction is that the symbolic state space Q^S is enumerable (and possibly finite). It forms the *directed symbolic reachability graph* which can be explored using well-known graph search algorithms. If the search aims at finding the cheapest run from the initial state q_0 to the target state q_{tar}, shortest-path algorithms for weighted graphs can be used. The resulting shortest run is the optimal symbolic run. The shortest run can then be derived from the symbolic run by picking the cheapest clock valuations u_i^* from the zones Z_i^* of the symbolic states along the path. In the context of scheduling problems with the objective of minimizing makespan, the cost of the run represents the makespan and is given by the time elapsed on a global clock which is started at the start of the run from the initial state and never reset to zero until the time it reaches the final target state.

3.2. Reachability Analysis Algorithm. The zone abstraction technique helps to overcome the problem of infinite runs and enables one to obtain the symbolic reachability graph in which the possible runs from the initial state to the target state are finite. To obtain the optimal schedule that corresponds to the run with minimal cost among all runs in the reachability graph, a (cost-optimal) reachability analysis is performed. This enumerative technique explores the symbolic state space step by step by evaluating the symbolic successor relation on the fly. Several efficient reachability algorithms exist to perform the task of algorithmically checking logic properties of TA.[30,51] This section presents and explains the reachability algorithm from ref 42, which is used here for the analysis of scheduling problems modeled as TA:

Algorithm 1. A cost-optimal reachability algorithm for minimization problems

$cost^* = \infty$
$\mathscr{W} := \{q_0\}; \mathscr{P} := \varphi$
while $\mathscr{W} \neq \varphi$ **do**
  $q = selectRemove(\mathscr{W})$
  if $q \notin \mathscr{P}$ **then**
    $\mathscr{P} := \mathscr{P} \cup \{q\}$
    if $cost(q) < cost^*$ **then**
      if final(q) **then**
        $cost^* := cost(q)$
      else
        $\mathcal{S} := succ(q)$
        $\mathcal{S}' : reduce(\mathcal{S})$
        $\mathscr{W} := \mathscr{W} \cup \mathcal{S}'$
      end if
    end if
  end if
end while

The global automaton is nondeterministic, and the reachability analysis has to explore a large number of different

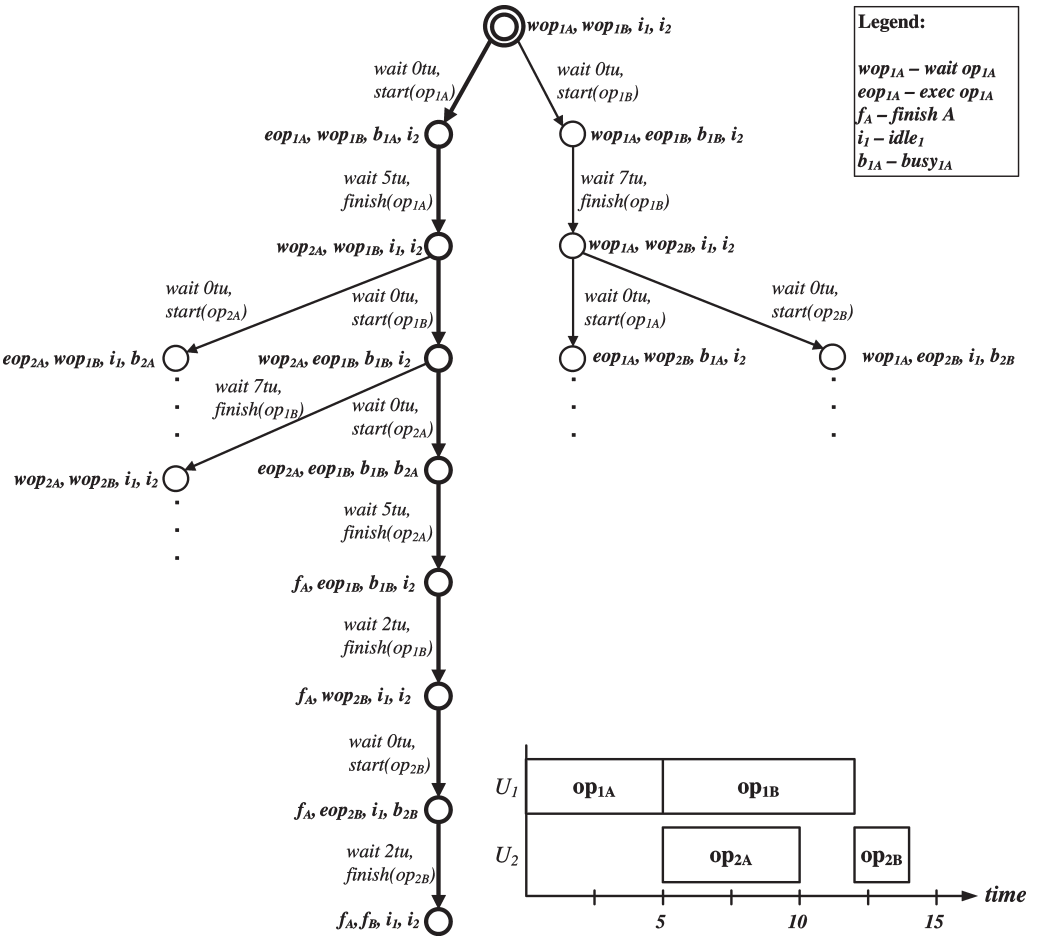

Figure 6. Reachability tree of the global automaton.

evolutions. The algorithm operates on the nodes of the reachability graph which contain the information on the locations, clock valuations, accumulated cost, and additional data about the values of the shared variables. *cost** represents the cost of the best path found so far. The list of nodes that have to be explored is represented by the set $\mathscr{W}$, called the *waiting list*. The set $\mathscr{P}$ represents the *passed list*, a list of nodes that were already explored. The function *selectRemove*($\mathscr{W}$) selects and removes a node from the waiting list $\mathscr{W}$. The selection of the node is determined according to an ordering defined by a graph search strategy; see section 5 for the different graph search techniques that can be implemented to guide the search. q corresponds to the candidate node that is currently explored. $\mathcal{S}$ is the set of immediate successors that can be reached by a single transition from the node q. The function *succ*(q) computes the immediate successor nodes of q and stores them temporarily in $\mathcal{S}$. The function *reduce*($\mathcal{S}$) is used to identify and remove those nodes that cannot lead to a better path than the best path found so far; see section 5 for various reduction techniques that can be implemented to prune the search tree.

At the start of the algorithm, the waiting list $\mathscr{W}$ contains only the initial node q_0, which corresponds to the state where no jobs have been started. Initially, the passed list $\mathscr{P}$ is empty since no node has been explored, and *cost**, which represents the cost of the best path found so far, is initialized with a high value that defines a safe upper bound of the objective function. In the first iteration of the loop, the only candidate node to be explored, q_0, is removed from the waiting list and explored by creating its immediate successors using the function *succ*(q). The successor nodes of the initial node q_0 which are to be explored later are moved to the waiting list, and the explored node q_0 is moved to the passed list. The second iteration starts by selecting a candidate node q from the waiting list, using the *selectRemove*($\mathscr{W}$) function. In order to avoid visiting and exploring the same node repeatedly, the test $q \notin \mathscr{P}$ is performed. If the test fails, then the current candidate node q is removed and a new candidate node from the waiting list is selected using the *selectRemove*($\mathscr{W}$) function. On the other hand, if the test is passed, then the candidate node q is added to the passed list. In the next step, the test *cost*(q) < *cost** compares the cost value of the currently explored node (i.e., the accumulated cost to reach q from the initial node q_0) with the best cost value found, thereby avoiding exploration of nodes that lead to nonoptimal runs if the comparison fails. The function *final*(q) checks whether node q is a target node or not, and if the check is positive, then the cost of the best path found is updated with the cost of the current node. On the other hand, if the current explored node q is not the target node, then the successor nodes of the currently explored node q are computed. The function *reduce*($\mathcal{S}$) is used to identify and to remove those nodes that cannot lead to a better path than the best path found so far. The remaining successor nodes of q are appended to the waiting list $\mathscr{W}$. The next iteration starts by selecting a new candidate node from the waiting list, and this

Changeover matrix of U_1		
To / From	A	B
Global	set_{1A}	set_{1B}
A	0	CO_{1AB}
B	CO_{1BA}	0

Changeover matrix of U_2		
To / From	A	B
Global	set_{2A}	set_{2B}
A	0	CO_{2AB}
B	CO_{2BA}	0

Figure 7. Changeover matrices of resources U_1 and U_2.

exploration scheme is repeated as long as the waiting list is not empty or in other words until the whole solution space is explored.

A part of the *reachability tree* of the global automaton of the example problem is shown in Figure 6. The optimal trace from the initial node to the target node is marked with bold lines. The corresponding optimal schedule with a makespan of 14 time units is shown in the bottom-right part of the figure.

4. EXTENSION OF THE MODEL TO CHANGEOVER PROCEDURES

In this section, a technique to model scheduling problems with sequence-dependent changeover procedures by TA is introduced. The example process described in section 2.1 is extended by setup and changeover times. The resources U_1 and U_2 are utilized and shared in the production process by both recipes *A* and *B*. To avoid contamination of the materials in the resources, a changeover procedure has to be performed whenever the operation waiting to be executed in the resource is different from the last operation that was executed in the same resource. The duration of the changeover procedure may depend on the sequence of the operations performed on the resource. We refer to the state in which the resource is currently prepared to perform a recipe operation as the *configuration* of the resource. The resource can be in one of the following configurations: *Global*, *A*, or *B*. Initially, all resources are assumed to be in the configuration *Global*, meaning that the resource could start executing an operation either of recipe *A* or of recipe *B* after undergoing the corresponding setup operation. Depending on the setup operation performed, the configuration of the resource is updated to either state *A* or state *B*. The changeover matrices of the resources U_1 and U_2 are shown in Figure 7. They define the changeover times; e.g., from the changeover matrix of U_1, a minimum setup time period of set_{1A} is required to prepare resource U_1 to execute an operation of recipe *A*, and a minimum changeover time period of CO_{1AB} is required to change resource U_1 from configuration *A* to configuration *B*.

The presence of sequence-dependent changeovers in resources U_1 and U_2 leads to additional operations for setup and changeovers which can be introduced in the recipe automata. The extended TA model of recipe *A* with changeovers is shown in Figure 8. Compared to the previous situation where the recipe is modeled as a single automaton, the model is extended such that for every recipe operation which is executed on a resource that requires a setup or a changeover, a separate TA with an individual clock is introduced. The automaton that models the setup and the changeover operations of the resource U_1 with respect to recipe *A* is shown as *Element 2 of Recipe Automaton* R_1 in Figure 8. Similarly, the automaton that models the setup and the changeover operations of resource U_2 with respect to recipe *A* is shown as *Element 3 of Recipe Automaton* R_1 in Figure 8.

The state where a resource is waiting to perform a setup or a changeover operation in order to prepare the resource to start a recipe operation is defined by a common *wait-prep* location. The state where a resource is executing a setup is defined by an *exec setup* location, and the state where a resource is executing a changeover operation is defined by an *exec CO* location. The common wait location of resource U_1 with respect to recipe *A* is labeled as $wait\text{-}prep_{1A}$. Executing the setup operation, the changeover operation from *A* to *A* and the changeover operation from *B* to *A* are labeled as *exec* $setup_{1A}$, *exec* CO_{1AA}, and *exec* CO_{1BA}, respectively. In the case where more than one batch of product *A* has to be produced subsequently in U_1, *exec* CO_{1AA} is required to model the changeover. Since in the example process only one batch of product *A* is produced, this location is redundant. The location $prep_{1A}$ represents the case where a resource has finished performing a setup or a changeover operation and is prepared to execute the recipe operation op_{1A}. Similar to the execute locations of the recipe operations, the execute locations of the setup and of the changeover operations have invariant conditions. The guard conditions and invariant conditions ensure that the setup operations and the changeover operations are executed for the corresponding durations only. Since the changeover from *A* to *A* has null duration, the *exec* CO_{1AA} location is defined as an *urgent* location depicted with the letter "**U**" meaning that the location is not active for more than 0 time units and the transition leaving this location must be taken immediately. Starting and finishing the execution of the setup operation is modeled by the transitions labeled $start(setup_{1A})$ and $finish(setup_{1A})$, respectively. Starting the execution of a changeover operation in resource U_1 from *B* to *A* is represented by transitions labeled $start(CO_{1BA})$, and finishing it by releasing the resource is represented by a transition labeled $finish(CO_{1BA})$. The transition from the location $prep_{1A}$ to $exec_{1A}$ in element 2 of the recipe automaton is synchronized with the transition $start\text{-}(op_{1A})$ in element 1 of the recipe automaton. This ensures that operation op_{1A} can be executed only after the resource U_1 has undergone a setup or a changeover procedure. A similar set of labels is introduced for the setup operations and the changeover operations of resource U_2.

To represent the current configuration of the resources, shared variables are assigned to each resource. In the extended example process, the shared variable S_1 is assigned to resource U_1 and can have values 1, 2, and 3 which represent the configurations *Global*, *A*, and *B*, respectively. The guards referring to the shared variables ensure that only the required setup or changeover operations are executed, and the shared variables are updated with the new configuration after executing the corresponding operation; e.g., the guard condition $S_1 == 1$ on the transition labeled $start(setup_{1A})$ from location $wait\text{—}prep_{1A}$ to *exec* $setup_{1A}$ ensures that this transition can only take place if resource U_1 is in its *Global* configuration, and after executing the setup operation, the shared variable S_1 is updated with the value 2 by the action $S_1 := 2$ on the transition from $prep_{1A}$ to $exec_{1A}$ with label $start\text{-}(op_{1A})$, where value 2 here represents that the resource is in configuration *A* now. The clocks c_2 and c_3 model the timing behavior of the changeover operations of resources U_1 and U_2, respectively.

The resource automata are extended by additional *busy* locations for the setup and changeover operations that they can execute. Transitions from *idle* to *busy* and from *busy* to *idle* exist with the corresponding synchronization labels to synchronize the resource automata with the corresponding recipe automata. The automaton model of resource U_1 is shown in Figure 9.

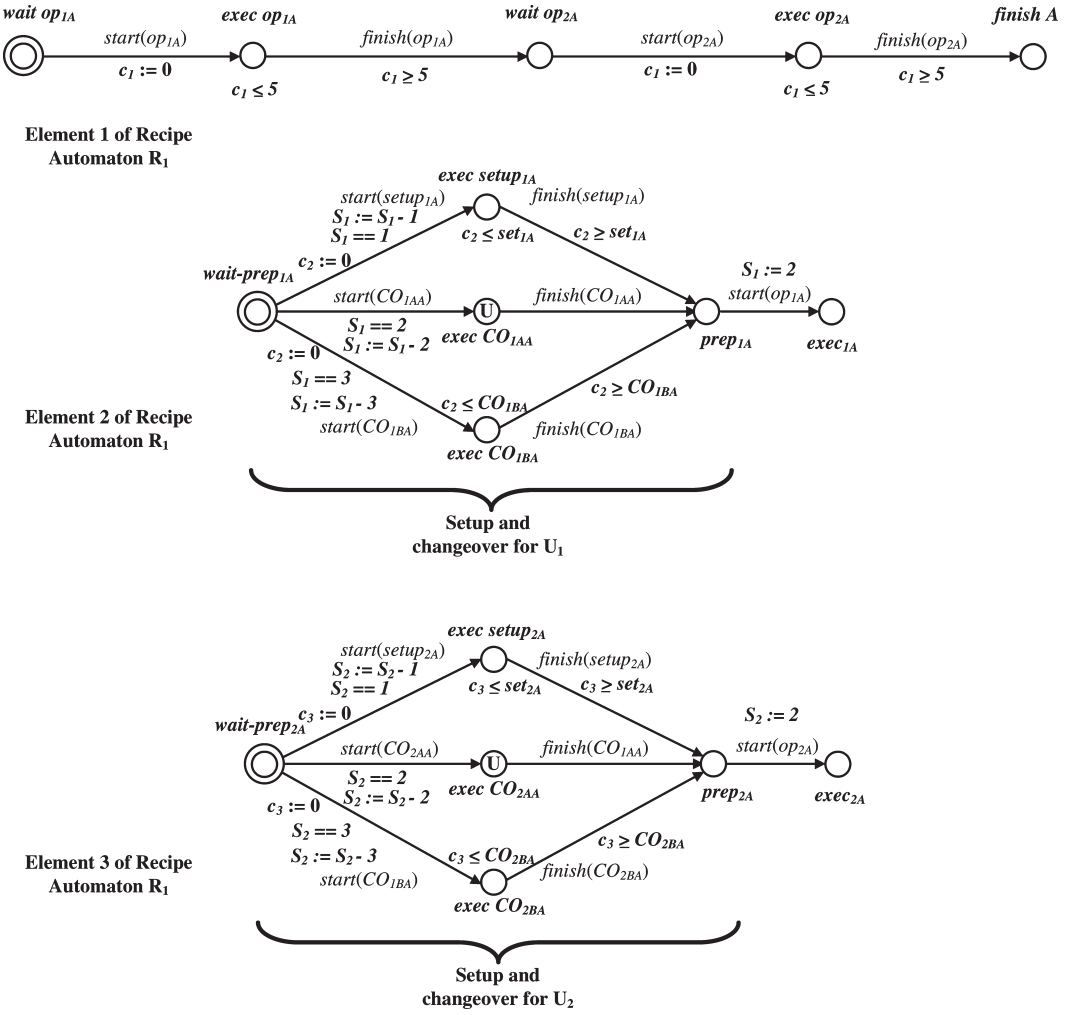

Figure 8. Automata networks of recipe *A* with changeovers on resources U_1 and U_2.

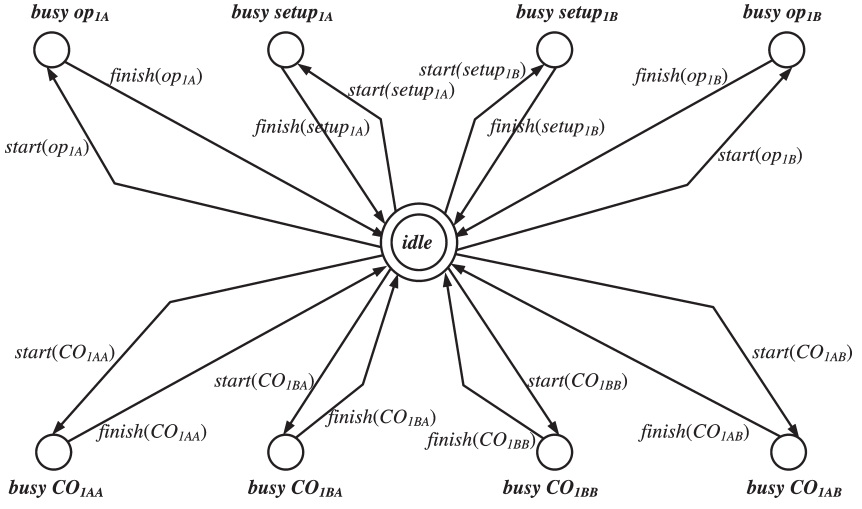

Figure 9. Automaton of resource U_1.

4.1. Extensions to More General Scheduling Problems. The TA-based approach can be extended to scheduling problems with other features such as alternative production paths, timing constraints, and material balances. Problems with objective functions other than makespan, such as minimization of total tardiness, minimization of total weighted tardiness, minimization of total earliness, minimization of total weighted earliness, and minimization of changeover costs can also be modeled by means

dx.doi.org/10.1021/ie101652d |*Ind. Eng. Chem. Res.* 2011, 50, 5131–5152

of priced TA. The techniques used to model complex scheduling problems are explained briefly below:

- *Alternative production paths*: Alternative resources and alternative production paths in the process can be represented in the TA model by introducing additional transitions that branch from the waiting location of the corresponding operation to locations that represent possible operation-resource pairs.
- *Timing constraints*: The maximal waiting time constraint for starting an operation can be represented by introducing an invariant in the waiting location of the corresponding operation in the recipe automaton. Similarly, the minimum wait timing constraints before starting an operation can be represented by introducing a guard on the outgoing transition from the waiting location of the operation. For example, the timing constraint on starting a production order which is usually termed the *release date* can be modeled by introducing a guard on the outgoing transition from the waiting location of the first operation of the production order that refers to a global clock, thus constraining the production order to start only after the release date has elapsed.
- *Material balances*: Each raw material, intermediate material, and final product can be associated with a shared variable. The values of the shared variables represent the amount of material present and they are part of the state of the TA. Consumption of a material by an operation is represented by decreasing the value of the corresponding shared variable while taking the transition that represents starting of the operation. Similarly, producing a material by an operation is represented by increasing the value of the corresponding shared variable when taking the transition that represents finishing the operation. Capacity constraints of the resources can be modeled by introducing guards on the corresponding transitions.
- *Other objective functions*: Objective functions other than makespan can be modeled by using priced TA. As mentioned earlier, priced TAs are extensions of TA with the notion of costs. In addition to the syntax and semantics of TA, priced TA has additional functions that assign cost rates to locations and fixed costs to transitions. The cost incurred for staying in a location is calculated by the product between the time period for which the location is active and the cost rate assigned to that location. Tardinesss minimization problems can be modeled by introducing an additional automaton to model the due date for each of the production orders. This additional automaton contains a location with a cost rate equal to one that becomes active once the due date has passed and remains active until the production order is finished. On the other hand, if the production order is finished earlier, the active period of this location is zero. The cost incurred thus represents the tardiness of the production order. Problems with the objective of minimizing weighted tardiness can be modeled by following the above idea of modeling tardiness with cost rates of the locations that compute the tardiness of the production orders assigned to values that correspond to the priority of the production orders. Similarly, problems with the objective function of minimizing changeovers can be modeled by assigning cost rates to the locations that represent changeover operations.

For a successful implementation and investigation of the TA-based approach to modeling problems with material balances, see ref 42, and for problems with release dates, due dates and with the objective of minimizing tardiness, see ref 52.

The extensions listed above enable one to model a broad variety of scheduling problems by networks of TA. However, the TA approach has limitations with respect to modeling decision variables with continuous degrees of freedom such as batch sizes (i.e., the amount of material processed by an operation). In an approximate fashion, variable batch sizes can be modeled by defining a number of alternative operations with different batch sizes that are encoded by different guards and actions.

5. TIMED AUTOMATA OPTIMIZER - TAOPT

The conceptual idea on TA-based modeling and the solution approach by reachability analysis has been realized by implementing the framework in the tool TAOpt developed at the Process Dynamics and Operations Group.[39] Other tools which have been developed to model and to analyze TA are *Uppaal CORA*,[53] *KRONOS*,[35] and *IF*.[31]

A schematic representation of the structure of TAOpt is shown in Figure 10. TAOpt consists of three main modules, the *user interface*, the *modeling framework*, and the *solution technique*. The problem definition is entered by a representation as RTN with information about recipes, production steps, resources, and timing constraints by means of the *user interface* as the input to the *modeling framework*. From the RTN formulation, the sets of timed automata models of the recipes and of the resources are created automatically in a modular fashion. Once the individual TA models are created, they are given as input to the module—*solution technique*. The sets of automata are combined by *parallel composition* to form a *global automaton* that represents the complete scheduling problem as a directed graph. The optimal schedule is obtained by performing the reachability analysis on the global automaton. Search heuristics and reduction techniques are integrated in the solution approach to improve the performance of the cost-optimal reachability algorithm. The optimal or the best production schedule found until some resource constraint has been reached is derived by reachability analysis and is delivered back to the *user interface* where it can be visualized as a Gantt chart.

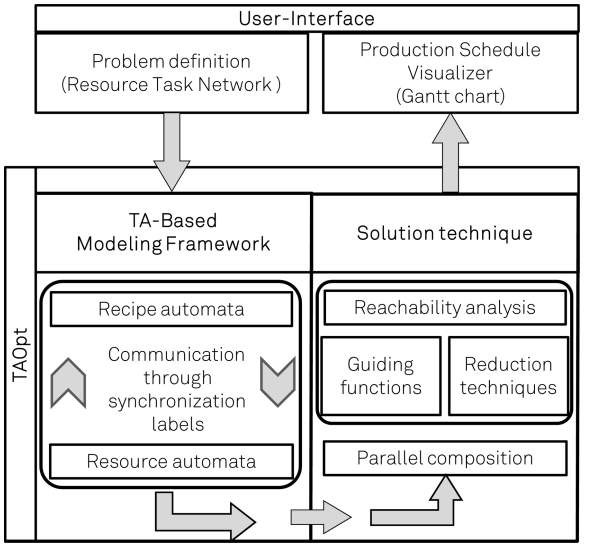

Figure 10. Schematic diagram of the structure of the TA-based scheduling tool TAOpt.

The formal representation of the scheduling problem as a RTN serves as input to the *modeling framework* of TAOpt. The graphical formulation of the scheduling problem as RTN can be constructed using the RTN model builder introduced in ref 54. Unlike the approach of mathematical programming where the problem formulation has to be created by the users from the RTN representation and where experience and knowledge about the effect of using different possible model constructs on the performance of the solvers is required, in the tool TAOpt, the TA models of the recipes, resources, and other problem specific constraints are generated *automatically* from the RTN network that represents the scheduling problem.

The sets of TA models created are then passed to the *solution technique* module of TAOpt to create the composed automaton and to search the resulting graph for traces that correspond to valid schedules. For problems with material balances, e.g., the benchmark problem proposed by Westenberger and Kallrath in ref 1, where an operation consumes only a fraction of a material and produces only a fraction of new material and the same operation can be performed more than one time, the parallel composition is done on the fly only on the sets of resource automata to form the global automaton.[55] In this case, the TA models of the recipes were not created, as the precedence relation between the operations of the recipes was encoded in shared variables of the resource automata.[55] For problems without material balances where an operation consumes the entire material and produces the entire new material and where each operation is performed only once, e.g., for job shop scheduling, the TA models of the recipes are created to take care of the precedence relation, and the parallel composition is done on the fly on sets of resource and recipe automata.

In the tool TAOpt, several techniques are implemented that make the reachability analysis more efficient. These include guiding techniques to guide the exploration of nodes toward the optimum and reduction techniques to prune suboptimal or redundant branches of the reachability graph. These techniques are briefly explained below.

An important factor in obtaining a good solution performance is to guide the exploration of the search tree toward the optimum quickly.[35] Within the context of the reachability analysis, this corresponds to deciding on the next node to be explored from the list of candidate nodes. The selection rules implemented in the tool TAOpt are based on different attributes of nodes (breadth, costs, depth, priority), and with different directions, minimum (min) or maximum (max).[40] The set of all possible combinations γ is defined as $\{min, max\} \times \{breadth, costs, depth, priority\}$. The search strategy with the combination of *max* and *depth* defines that among the nodes in the waiting list the node with maximal depth is selected first to be explored. The selection strategies select nodes from the waiting list for which one of the listed attributes is maximal or minimal, e.g., depth-first search or breadth-first search. These strategies select nodes with maximal or minimal depth, respectively. The minimum-cost search (best-first search) strategy selects nodes according to minimal cost values of the cost to reach the particular node. Other strategies, e.g. priority search or random search, are based on different priority selection rules or select the nodes in a random fashion. Priority selection rules that TAOpt supports are shortest processing time (SPT) first, earliest release date (ERD) first, and earliest due date (EDD) first. In most cases, it is useful to combine several of the heuristic selection rules. When using the depth-first strategy, it often happens that there exists more than one node in the waiting list with the same value of depth. In this case, additional decision criteria that involve other attributes of the nodes are useful. A combined selection rule is defined as $\gamma_1, \gamma_2, \gamma_3, ..., \gamma_n$. The combined strategy selects a node from the waiting list with minimal (maximal) γ_1. If several nodes with this property exist, the decision is made according to γ_2, and so on. The *selectRemove*($\mathscr{W}$) function of the reachability algorithm presented in section 3.2 enables one to use the above-mentioned techniques to guide the exploration of the reachability tree.

Due to the combinatorial explosion of the reachability analysis, several reduction techniques are implemented which avoid the exploration of nonoptimal or redundant branches of the reachability tree. The reduction techniques implemented in the tool TAOpt focus on reducing the set of successor nodes while exploring a node. The nonlaziness reduction scheme and the sleep-set method are reduction techniques in TAOpt that compare different attributes of the successor nodes pairwise.

The safe *nonlaziness* reduction scheme was first introduced by Abdeddaim and Maler[35] for pruning parts of the search tree which lead to suboptimal traces. Such traces include unnecessary waiting times of resources in the corresponding schedule. The reduction scheme compares pairs of successor nodes and marks certain nodes as candidates to be removed which are part of the traces that lead to slots in the corresponding schedule where other operations could fit, thereby leading to a better schedule. The term "safe" nonlaziness is used since this reduction technique does not prune the optimal trace. A more aggressive but not safe version of the nonlaziness reduction technique, called greedy nonlaziness, is also implemented in the tool TAOpt and was explained in more detail in ref 40. Greedy nonlaziness forbids any waiting time of the resources if another operation can possibly be executed. It is a heuristic, as this reduction technique may prune the optimal trace. In job shop theory, the difference between the safe and the greedy nonlaziness is also known as the *scheduling anomaly*. The solution obtained by TAOpt when applying any of the various reduction techniques always corresponds to a feasible schedule.

The *sleep-set method*, originally introduced by Godefroid[56] for partial order reduction in reachability analysis, aims at avoiding the exploration of runs corresponding to the same schedule in the reachability graph. In the symbolic reachability graph, different traces which correspond to the same schedule may exist, and such schedules include operations which are executed at the same absolute time, but the corresponding transitions in the reachability graph are traversed in different logical orders resulting in different traces. Such redundant traces are removed by the sleep-set method. Two different alternatives of the sleep-set method are implemented in TAOpt, a safe and a greedy version. Similar to the greedy nonlaziness scheme, the greedy version of the sleep-set method prunes a large part of the search space but at the cost of possibly pruning the optimal trace. The application of TAOpt with the sleep-set reduction to job-shop scheduling problems was discussed in ref 40.

Embedded LP-models can be used to compute lower bounds on the cost to go for nodes which are explored in the reachability analysis. The implementation of such embedded LPs was introduced and discussed in refs 39 and 57. These lower bounds help to prune parts of the search tree that only consist of suboptimal traces, thus corresponding to suboptimal schedules. The *reduce*($\mathscr{S}$) function in the reachability algorithm presented in section 3.2 enables one to use the above-mentioned reduction techniques to prune parts of reachability tree.

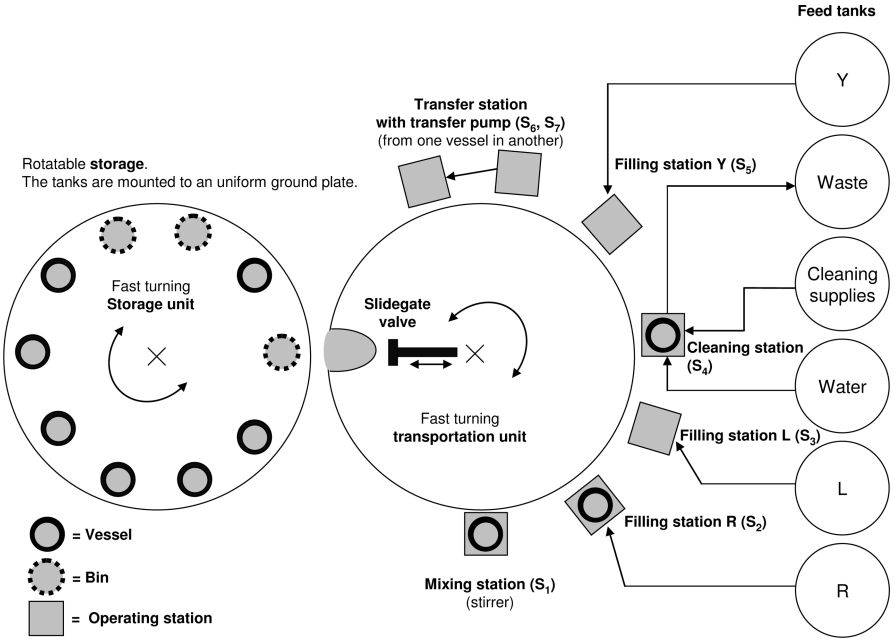

Figure 11. Sketch of the pipeless plant.

6. EXPERIMENTAL RESULTS

In this section, three different case studies are investigated to demonstrate the performance of the TA-based scheduling approach. All of the tests were performed with the scheduling tool TAOpt. The time to generate the scheduling problem as sets of TA for the case studies are included in the CPU times reported in the experimental results since the TA models are formulated automatically by the tool TAOpt from the RTN description of the process to be scheduled. The computational equipment used for all of the tests performed was a dual core Xeon machine with a speed of 3 GHz, 2 GB of main memory, and operating with a Linux platform.

6.1. Pipeless Plant Case Study. *6.1.1. Description.* The first case study is a multiproduct pipeless plant demonstrator which was developed at the Process Dynamics and Operations Group.[54] In traditional multiproduct batch plants, the connections between the units are realized by fixed pipes and the transport of the materials between the units is performed by pumping or draining them through the pipes. In contrast to classical multiproduct batch plants, in a pipeless plant, the materials stay in the vessels and are transported using a movable container to the stationary processing units where they are processed. The demonstration plant is currently used as a laboratory experiment for students of (bio)chemical engineering in order to familiarize them with scheduling problems and serves as a test bench for scheduling and automation.

The demonstration plant is used to produce water-based substances in various colors and shades from three different raw materials, **L**, **R**, and **Y**. A sketch of the pipeless plant is shown in Figure 11. The plant consists of seven different stationary processing stations. The materials are transported in vessels between the stations using a single transportation unit. The processing stations include a mixing station (S_1), three filling stations (S_2, S_3, and S_5), a transfer station (S_6, S_7), and a cleaning station (S_4). One filling station per raw material (**L**, **R**, and **Y**) is present to pump the content from the raw material storage to the vessels. The mixing station provides the functionality of blending the contents in the vessels. The transfer station is used to pump the content from one vessel into another vessel. In the cleaning station, the contents of a vessel are removed by pumping them out, and the vessel is cleaned by rinsing it with the cleaning agent and water, thus preparing it to produce the next batch.

A circular bidirectional rotating plate with bins is used to store the vessels. It has a maximum capacity of 10 vessels. A bidirectional rotating transportation unit with an extendable arm transports the vessels between the processing stations and between a processing station and the storage unit (see Figure 12). The arm of the transportation unit can be extended or contracted by linear motion, and the transportation unit can be moved radially. Transporting a vessel from a source station to a destination station is performed by a series of actions which consists of extending the arm by linear motion, grabbing the vessel from the source station followed by pulling the vessel, moving the arm to the destination station, pushing the vessel to the destination station followed by releasing the vessel, and retracting the arm.

6.1.2. Problem Formulation. Three different end products **A**, **B**, and **C** have to be produced from three raw materials **L**, **R**, and **Y**. The recipe structure of products **A**, **B**, and **C** is shown in Figure 13.

The detailed recipe steps to produce one batch of products **A**, **B**, and **C** are as follows:

- Recipe to produce one batch of product **A**: 75 mL of raw material **Y** and 30 mL of material **R** are mixed in the mixing station for 45 s to produce the intermediate product $\mathbf{A_1}$. Another vessel is filled with 195 mL of material **L**, and this is indicated as the intermediate product $\mathbf{A_2}$. The intermediate material $\mathbf{A_1}$ is then added to the other intermediate material $\mathbf{A_2}$ in the transfer station, and the content is mixed to produce one batch of the final end product **A**.
- Recipe to produce one batch of product **B**: 30 mL of material **R** and 195 mL of material **L** are mixed in the mixing station for 45 s to produce the final end product **B**. The raw material **R** is filled initially, and then the material **L** is added.

dx.doi.org/10.1021/ie101652d |*Ind. Eng. Chem. Res.* 2011, 50, 5131–5152

- Recipe to produce one batch of product **C**: 100 mL of material **L** and 200 mL of material **R** are mixed in the mixing station for 45 s to produce the final end product **C**. The raw material **L** is filled initially, and then material **R** is added.

Each batch of the end products **A**, **B**, and **C** has to be moved to the storage bin for ripening, which takes at least 10 s. After the ripening process is finished, the vessels are cleaned in the cleaning station, which takes 95 s, and then moved back to the storage. Filling the vessels with the raw materials at the filling stations is performed at a rate of 5 mL/s. Table 1 shows the periods of time required to rotate the transportation unit from the starting stations to the destination stations. Irrespective of the stations, the duration of pulling the vessel from the starting station and of pushing the vessel to the destination station is 7 s each.

The problem instance considered here is to compute a production schedule to produce three batches of the three end products, resulting in a batch scheduling problem with a total demand of nine batches. The objective is to produce these nine batches with minimal makespan. The complexity of the considered scheduling problem is due to the following features of the plant and the production process:

- The durations of the transportation operations to transport a vessel between the stations depend on the current position of the arm. One technique to model this characteristic feature of the plant is to consider the transportation operations as changeover operations and the current position of the arm as the current configuration. This leads to modeling the transportation unit with *sequence-dependent changeover times*.
- The absence of intermediate buffers between the stations may lead to *blocking*. A faulty decision on the transportation of the vessels leads to mutual blocking of the stations and to deadlocks.
- For the recipe with transfer operations, in order to begin the transfer task, the source position and the target position must be occupied with the vessels containing the respective materials. Hence, the sets of operations that produce the source material and the sets of operations that produce the material to which it is added have to be performed in parallel to perform the transfer task optimally.

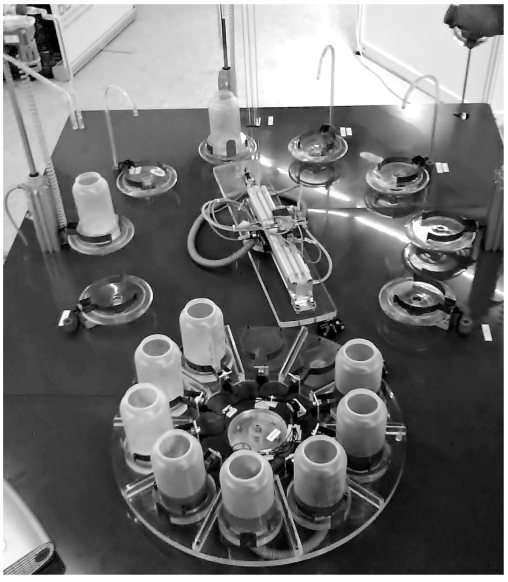

Figure 12. Photo of the pipeless plant located in the laboratory of the Process Dynamics and Operations Group.

6.1.3. Experimental Setup and Results. The recipes in the form of RTN, the information on the resources, and the total number of production orders were given as input to TAOpt. From this information, the sets of recipe automata and resource automata were automatically created in a modular fashion, and the reachability analysis was performed on the fly by TAOpt to derive the production schedule. The exploration of the search space by TAOpt was limited to 3600 CPU seconds. The timed automata based model for the case study consists of 35 single automata with a total of 2757 locations, 5149 transitions, and 36 clocks. To search the reachability graph, a combination of depth-first and best-first search techniques was used; i.e., the criterion to choose the next node to be explored from the list of unexplored nodes is that the node with maximum depth is chosen first, and if more than one node with the same maximum depth exists, then the node with minimal cost is chosen. The state-space reduction techniques, sleep-set method, and nonlaziness scheme as explained in ref 42 were used. During the search process in the reachability tree, within 3600 CPU seconds, 3 314 683 nodes were explored. The first feasible schedule with a makespan of 1556 s was found after 11 CPU seconds, and the best solution with a makespan of 1503 s was found after 78 CPU seconds. However, the solution quality did not improve after 78 CPU seconds until the termination criterion was reached. The Gantt chart of the best solution with 111 tasks is depicted in Figure 14. As the graph could not be explored fully within the predefined period, no certificate of optimality or optimality gap was obtained.

6.2. Food Manufacturing Case Study. *6.2.1. Description.* The second case study considered is derived from a real-world, medium-sized, ice cream manufacturing plant described in ref 58. The case study considered is a multistage multiproduct batch plant where eight different ice creams (A−H) are produced. The production process is performed such that the raw materials stored in the warehouse are transported to the processing department where they are mixed according to the recipes for the

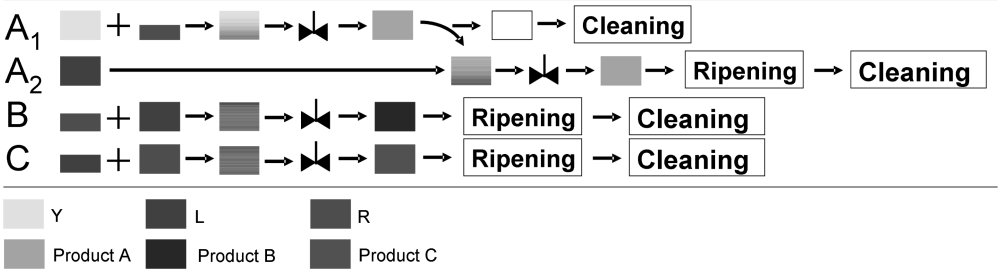

Figure 13. General recipe structures.

Table 1. Transportation Times (s)

	to							
from	storage bin	station S1	station S2	station S3	station S4	station S5	station S6	station S7
storage bin	0	8	7	6	5	4	3	2
station-S1	8	0	2	3	4	5	6	7
station-S2	7	2	0	2	3	4	5	6
station-S3	6	3	2	0	2	3	4	5
station-S4	5	4	3	2	0	2	3	4
station-S5	4	5	3	3	2	0	2	3
station-S6	3	6	5	4	3	2	0	2
station-S7	2	7	6	5	4	3	2	0

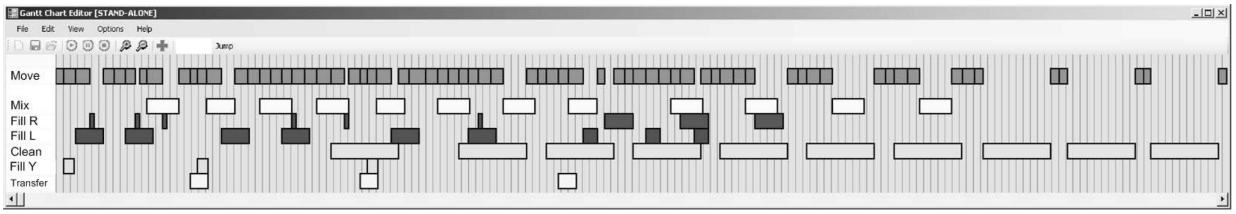

Figure 14. Gantt chart of the schedule with the best makespan of 1503 s.

different products and transferred to the intermediate storage units. In the storage units, the intermediate products are cooled down, and after a minimum waiting period has elapsed, the frozen products are transferred to the packing department. The end products are then packed in the packing lines and delivered to the customers. The intermediate products obtained from the processing department have a maximum shelf time period and perish if they are not packed before the shelf time period has elapsed. This property of the intermediate products with minimum waiting and maximum shelf life imposes a timing constraint on the storage periods in the vessels. The intermediate products can be prepared and kept over the weekend; however the maximum shelf time period has to be obeyed.

A schematic diagram of the process is shown in Figure 15. The production plant consists of two stages, where the first stage comprises the processing line and the storage vessels and the second stage consists of the packing lines. A single processing line with a production rate of 4500 kg/h is used to process the raw materials irrespective of the recipes. Six intermediate storage vessels (S1−S6) are available to freeze the processed materials. Vessels S1 and S2 have maximum capacities of 8000 kg each, and vessels S3−S6 have maximum capacities of 4000 kg each. The second stage consists of two packing lines P1 and P2 with production rates that depend on the product packed. The connectivity between the resources is such that vessels S1 and S2 are coupled only to packing line P1, and vessels S3−S6 are coupled only to packing line P2. This imposes a routing constraint that intermediate materials stored in S1 or S2 can be packed only in packing line P1 and intermediate materials stored in S3−S6 can be packed only in packing line P2. An additional constraint exists such that the products A−D can be packed only in packing line P1 and products E−H can be packed only in packing line P2. Thus, products A−D must be stored in storage vessels S1 and S2 only, and products E−H must be stored in storage vessels S3−S6 due to the constraints stated above. The complexity of the scheduling problem results from (a) the high

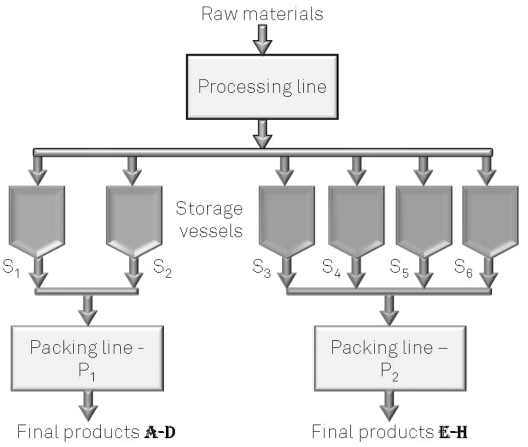

Figure 15. Schematic diagram of the food manufacturing process.

utilization of the equipment in each stage (>70%), (b) the presence of bottlenecks in both stages of the process, and (c) the shared storage vessels. The problem data for the case study can be found in the Appendix.

6.2.2. Problem Formulation. One batch of product comprises 8000 kg, and the task durations are given in minutes. The problem instance proposed by the authors in ref 58 is to meet a demand of 91 orders within a scheduling horizon of 7200 min, where a production order refers to one batch. The demand of 91 orders includes different amounts of the different products.

Previously, Bongers and Bakker[58] solved the problem instance of producing 91 orders by implementing the problem using commercial software. However, applying the available solvers did not result in a feasible schedule, and hence a manual intervention with heuristics was used to create a feasible schedule. The heuristics used are as follows: (a) The sequence of the batches was preordered such that the rate of packing the products in packing line P1 approximately matched the rate of packing in

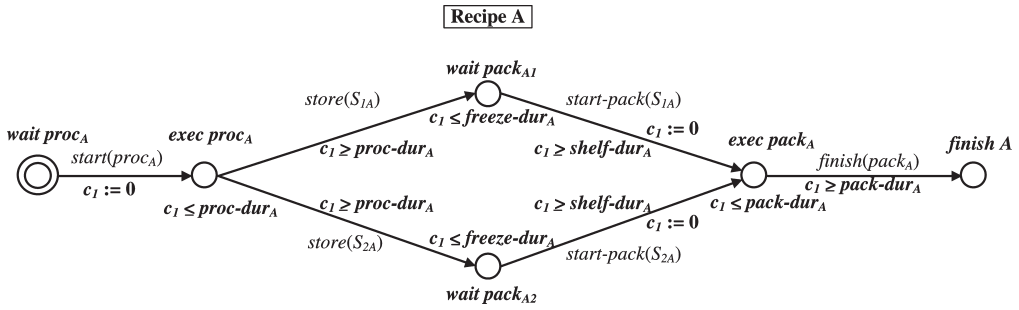

Figure 16. Element 1 of the recipe automaton of product *A*.

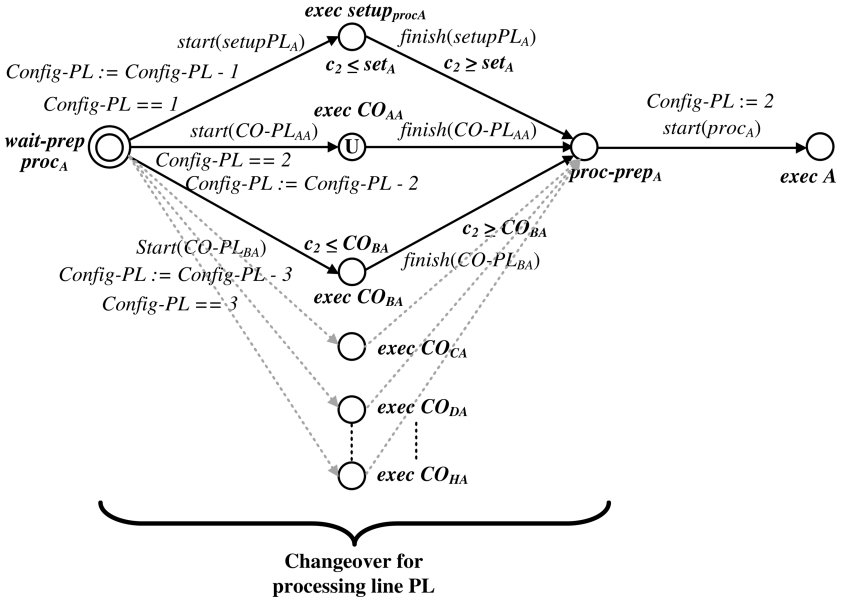

Figure 17. Element 2 of the recipe automaton of product *A* modeling the changeovers in processing line PL.

packing line P2. (b) Slack times between the batches were introduced in the packing lines to create flexibility in the processing line. (c) The products in the packing lines were packed in a campaign fashion in the sequence D−B−A−C in P1 and G−H−F−E in P2. Inspection of the resulting production schedule showed that it is not the best possible schedule with respect to production times and changeover times.[58] In addition to the original instance proposed in ref 58, different additional instances of the case study were solved by TAOpt. In the additional instances, production orders with different amounts of various products have to be produced within a minimum amount of time. The main features of the case study which make the problem instances challenging are as follows: (a) The processing unit in the first stage and the packing lines in the second stage are subject to sequence-dependent setup times and changeover procedures. (b) The intermediate storage units are limited and have to be shared by various products. (c) The intermediate products are unstable and have a minimum waiting time and a maximum shelf life period, introducing a timing constraint between the processing task and the packing task. (d) The demand should be met within the scheduling horizon of 7200 min.

The recipes and the resources were modeled using the TA-based modeling approach described in section 4. From the recipes in the form of RTN, the information on the resources, and a table of production orders, the scheduling tool TAOpt automatically created sets of recipe automata and resource automata in a modular fashion, and the reachability analysis was performed on the fly. For each of the problem instances investigated, one recipe automata network per production order was created since a production order is equal to one batch of the corresponding recipe. In addition, nine resource automata were generated to model the processing line, the two packing lines, and the six storage vessels.

The network of TA models created by TAOpt for recipe A is shown in Figures 16, 17, and 18. The recipe operations modeled as TA are shown in Figure 16. The setup and the changeover operations in the processing line modeled as a TA are shown in Figure 17. Similarly, the setup and the changeover operations in packing line P1 modeled as a TA are shown in Figure 18. For the sake of clarity, only a few of the locations and a few of the transitions that represent the changeover operations are shown. The shared variable *Config−PL* represents the configuration of the processing line, and the values of the shared variable 1−9 correspond to the configurations *Global*, *A*, *B*, ..., *H*, respectively. Similarly, the shared variable *Config−P1* represents the current configuration of packing line P1, and the values of the shared variable 1−5 correspond to the configurations *Global*, *A*, *B*, *C*, and *D*, since only these products can be packed in packing line P1. All resources have the initial state of the configuration *Global*,

dx.doi.org/10.1021/ie101652d |*Ind. Eng. Chem. Res.* 2011, 50, 5131–5152

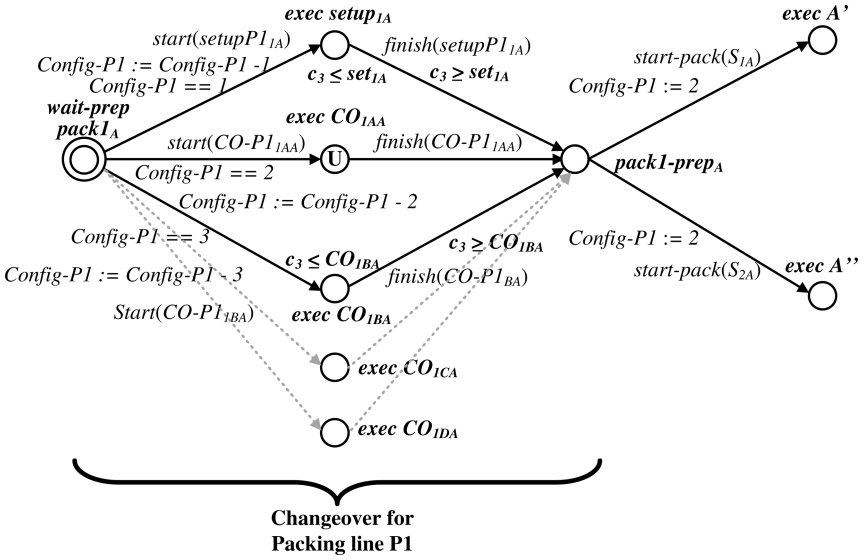

Figure 18. Element 3 of the recipe automaton of product *A* modeling the changeovers in packing line P1.

Table 2. Result of the Tests on the Set A Instances of the Food Manufacturing Case Study

Set A				
number of production orders	number of nodes created	number of nodes explored	CPU time needed to find the best solution (in seconds)	makespan of the best solution (in minutes)
10	271	114	0.12	1064.27
15	434	160	0.27	1498.54
20	551	195	0.23	1990.50
25	844	262	0.29	2357.82
30	848	264	0.39	2602.07
35	896	278	0.40	3073.78
40	971	304	0.43	4063.77
45	985	295	0.44	4391.38
50	1041	354	0.46	4939.94

and hence all of the shared variables have initial values of 1. Clock c_1 models the timing behavior of the recipe operations, and clocks c_2 and c_3 model the timing behavior of the changeover operations of the processing line and packing line P1, respectively.

6.2.3. Experimental Setup and Results. To search the reachability graph, a combination of depth-first and best-first search techniques was used. In the case when there exists more than one node with the same depth and cost, then one of them is chosen randomly. The state-space reduction techniques, safe sleep-set method, and safe nonlaziness scheme as explained in ref 42 were used. The reachability analysis is terminated if one of the following condition becomes valid: (a) the waiting list which contains the nodes to be explored is empty or (b) the computation time has reached the maximum limit of 3600 CPU seconds or (c) the number of nodes explored has reached the maximum limit of 3 million.

The additional instances considered for numerical investigations consist of two sets of problem instances (set A and set B) with different numbers of orders. Set A consists of instances with smaller numbers of orders (ranging from 10 to 50), and set B consists of instances with larger numbers of orders (ranging from 60 to 91). Table 2 shows results of the tests conducted on the instances of set A. The second and the third column show the number of nodes created and the number of nodes of the reachability tree that were explored to reach the best feasible solution for various instances of set A when solved using TAOpt. The fourth column and the fifth column show the computation time in CPU seconds needed to reach the best feasible solution, and the objective value. In the tests performed, for all problem instances, a feasible schedule was derived by TAOpt within a fraction of CPU seconds. The optimality of the best solution obtained could not be proven for any of the instances since the search was terminated when the number of nodes explored reached its maximum limit of 3 million.

Table 3 shows results of the tests conducted on the instances of set B. The problem instances of set B consist of larger numbers of orders. To handle the complexity of the problem instances, a plausible heuristic was implemented to reduce the model size and the size of the reachability tree. In the reduced formulation, the recipe automata are modeled such that in the packing lines only the transitions that represent the starting of changeover operations with the shortest changeover times among all of the changeover operations are allowed, and all other transitions are excluded. However, it should be noted that the problem instance

Table 3. Result of the Tests on the Set B Instances of the Food Manufacturing Case Study

Set B				
number of production orders	number of nodes created	number of nodes explored	CPU time needed to find the best solution (in seconds)	makespan of the best solution (in minutes)
60	1077	358	0.30	5116.62
70	1360	434	1.02	5767.52
80	1609	490	1.47	6419.46
91	**124462**	**48203**	**13.13**	**7152.8**

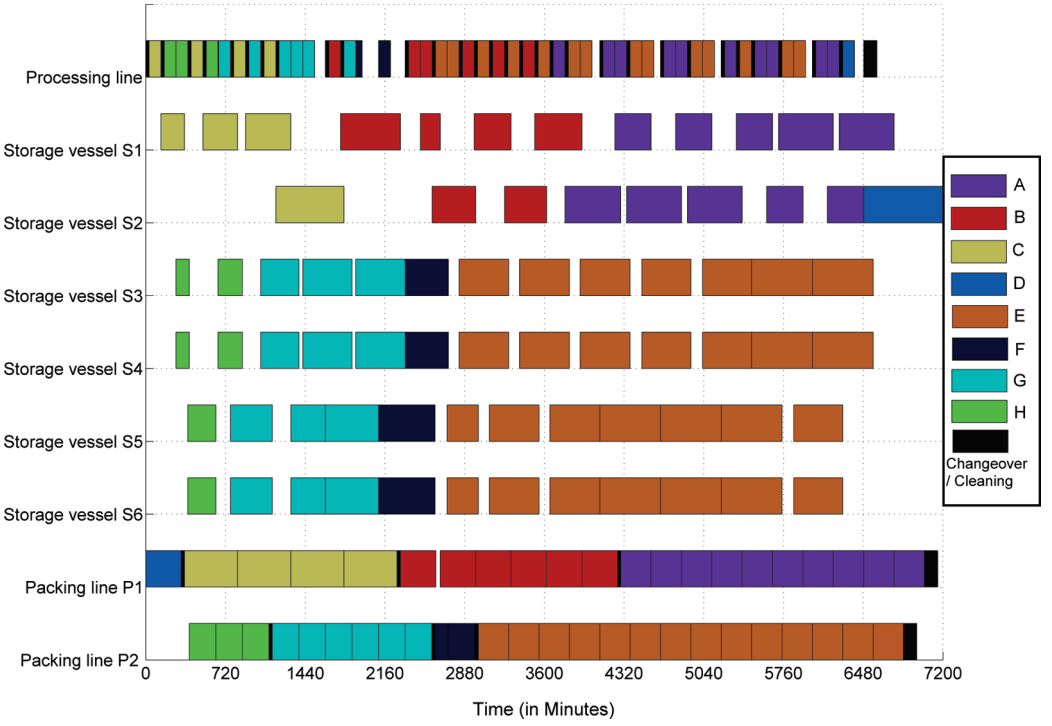

Figure 19. Weekly production schedule computed by TAOpt for 91 orders.

is still complex, as none of the transitions that represent the starting of changeover operations in the processing line are excluded. For all of the instances of set B, the recipes are modeled with the reduced formulation and scheduled on the packing lines in a campaign fashion in the sequence D−C−B−A in P1 and H−G−F−E in P2 since the changeover times in the packing lines are minimal in this sequence (see ref 58). The sequencing on all other units and the complete timing were computed by TAOpt.

While it cannot be ruled out that the heuristic employed might prune the optimal solution, it is remarkable that a feasible solution which schedules 91 orders within a makespan of approximately 7153 min including the cleaning time was computed by TAOpt within less than 15 CPU seconds. A comparison of the production schedule derived from TAOpt with the one in ref 58 for the instance with 91 orders shows that the schedule computed by TAOpt has a shorter makespan and the utilization rate of the processing line is reduced by 10%. TAOpt could synthesize a feasible and better production schedule in comparison to the tool used in ref 58, which required manual intervention to obtain a feasible schedule. The production schedule computed by TAOpt for the instance with 91 orders is shown in Figure 19. In the production schedule for 91 orders, it should be noted that, similar to the solution in ref 58, the packing line P1 starts operating at time 0 due to the fact that the intermediate product for recipe D is prepared earlier and stored in storage vessel S1 during the weekend.

6.3. Plastic Compounding Case Study. *6.3.1. Description.* The third case study is a plastic compounding problem which was introduced by Henning and Cerdá in ref 59 and has been discussed extensively by many researchers from the process systems engineering community.[25,60−64] The case study considered is a single-stage multiproduct batch plant with four extruders in parallel. The extruders differ in the setup times, which are unit-dependent, and in the processing rate and capacities, which are dependent on the product being manufactured. The problem is to schedule a total of 12 single-batch orders within a time horizon of 30 days. Each of the 12 orders has a specific due date, and an early production of the orders before the due date is penalized with inventory costs. Since the plant has enough capacity to schedule all 12 orders without any tardiness, the objective is to minimize the overall summed up earliness. The challenging part of the scheduling problem is introduced by considering limited discrete resources (manpower to operate the extruders), as usually solving MILP models using a continuous time representation with discrete resource constraints is computationally

dx.doi.org/10.1021/ie101652d |*Ind. Eng. Chem. Res.* 2011, 50, 5131–5152

Table 4. Model Sizes and Computation Results for the Plastic Compounding Case Study

model	case	binary variables	continuous variables	constraints	T_{CPU}	optimal solution
Janak et al.[63]	A	150	513	1389	0.07	1.026
	B	458	2137	10382	6.53	1.895
	C	444	2137	10382	236.87	7.334
Méndez and Cérda[62]	A	82	24	241	0.03	1.026
	B	127	24	622	2.93	1.895
	C	115	24	490	13.28	7.334
	D	151	24	886	3600[a]	6.234[a]
	E	139	24	754	3600[b]	11.12[b]
Marchetti and Cérda[64]	A	82	24	214	0.03	1.026
	B	223	24	622	0.66	1.895
	C	223	24	622	7.39	7.334
	D	223	72	897	6.63	5.276
	E	223	72	897	8.59	11.12
		created nodes	explored nodes	T_{best}	T_{CPU}	optimal solution
TAOpt	A	715	474	0.04	0.08	1.026
	B	778	509	0.06	0.09	1.895
	C	46711	27403	1.08	1.16	7.334
	D	2025	1315	0.12	0.14	5.276
	E	8465	4859	0.23	0.33	11.12

[a] Suboptimal solution. [b] Optimality was not proved.

expensive.[62] On the basis of the manpower required to operate the extruders, five different cases (cases A−E) of the problem instance are considered. In the first three cases A, B, and C of the problem instance, it is assumed that a single operator is required to operate each extruder, and in the last two cases D and E, it is assumed that the number of operators required to operate each extruder is dependent on the unit. The total number of operators available is assumed to be unlimited for case A, at most three workers for case B, and at most two workers for case C, thus limiting the number of extruders that can work in parallel to three and two in cases B and C, respectively. Similarly, the total number of operators available is limited to five in case D and to four in case E. The problem data considered for all of the instances were taken from ref 64.

6.3.2. Experimental Setup and Results. All of the above-mentioned cases A−C were modeled as RTN and solved using TAOpt using the conceptual idea explained in section 4. Along with the recipe automaton, additional automata are introduced to model the due date for each of the production orders. These additional automata contain a location with a cost rate equal to one that becomes active once the production orders are finished and remains active until the due date is reached. The cost incurred in a run represents the earliness of the production order. The discrete resources (operators) which are shared by all production orders are modeled by introducing an additional discrete shared variable. The number of operators available is indicated by the value of the shared variable. The value of the shared variable is then updated on the respective transitions that either start an operation, thereby allocating an operator, or that end an operation, thereby releasing an operator.

For all of the tests performed in TAOpt, the search strategy used was a combination of depth-first and best-first search techniques. The state-space reduction techniques, safe sleep-set method, and safe nonlaziness scheme as explained in ref 42 were used. The reachability analysis is terminated if one of the following conditions becomes valid: (a) the waiting list which contains the nodes to be explored is empty or (b) the computation time has reached the maximum limit of 3600 CPU seconds or (c) the number of nodes explored has reached the maximum limit of 1 million.

Table 4 shows the model statistics (number of binary variables, number of continuous variables, number of constraints), the computation time to prove optimality, and the objective value for three different continuous time MILP formulations which can handle resource-constrained scheduling problems. The MILP formulation by Janak et al.[63] uses unit-specific event points to detect the starting time of a task and utility related tasks. The results found with the mathematical formulation in ref 63 were obtained using a 3.0 GHz Linux workstation, which is comparable to the computational equipment we used for our tests, and the model was built and solved using the commercial tool GAMS 2.5 and CPLEX 8.1 with a relative optimality tolerance of 0.01%. The MILP formulation of Méndez and Cérda[62] uses preordering rules embedded in the MILP framework, and ordering of the batches at any resource is handled by a common set of sequencing variables. The results found with the mathematical formulation in ref 62 were obtained using a 1.8 GHz Pentium IV workstation, and the model was built and solved using the commercial tool ILOG OPL Studio 3.6 embedded with CPLEX 8.0. The MILP formulation of Méndez and Cérda[62] is also a continuous time formulation with a general precedence-based representation that uses separate sets of binary variables to handle allocation and sequencing decisions on the resources. The computational equipment and the solver used were the same as those used by Marchetti and Cerda in ref 64. The results obtained using TAOpt are shown at the bottom part of the table. The third and the fourth column shows the number of nodes created in the search tree and the number of nodes explored to prove optimality, respectively. The fifth column

shows the computation time required to reach the optimal solution and the sixth column shows the time taken to prove optimality.

As shown in Table 4 for all of the problem instances tested, using TAOpt, the optimal solution was found and optimality was proved. Similarly, the MILP formulation of Janak et al.[63] could find the optimal solution for the cases A−C. The computation times to prove optimality of the best solutions obtained by TAOpt on cases A−C in comparison to the MILP formulation by Janak et al.[63] shows that the former is at least as efficient as the later for case A, where the discrete resources are unconstrained, and for the constrained cases, TAOpt is significantly faster by a substantial factor. The MILP formulation by Méndez and Cérda[62] found the optimal solution for the cases where the discrete resources are not unit-dependent. However, for the cases D and E, the formulation by Méndez and Cérda[62] optimality could not proved, and in case D, only a suboptimal solution could be found. Though the computational platform used by Méndez and Cérda[62] is not comparable with the one used in our work, it should be noted that for cases D and E TAOpt found the optimal solution within fractions of seconds, whereas the former resulted in only suboptimal solutions. It is a fact that as the problem instances get more constrained the solution-space is reduced, but interestingly, it can be seen that the number of nodes explored by TAOpt increases as the problem instances get more constrained. The reason for this is that the search strategy guides the search initially to a suboptimal region of the solution space, which is far away from the neighborhood of optimal solution, which can be noted from the large number of backtrackings that the algorithm performs after finding the first feasible solution.

7. CONCLUSIONS

In this paper, we have proposed an efficient and intuitive approach to model and to solve multistage, multiproduct batch scheduling problems with sequence-dependent changeovers. The approach is based on the framework of timed automata (TA). The recipes and the resources are modeled individually as TA, and the timing behaviors of the operations are modeled with the help of clocks. The modular nature of the automata framework is exploited by modeling the changeover operations separately as sets of TA which are synchronized with the recipe automata. The automata models of the recipes and of the resources are combined on the fly to yield a global automaton that represents the complete scheduling problem. Feasible production schedules correspond to paths from the initial node of the automaton to the desired target node. The optimal solution can be obtained by searching for the path with the minimal cost among all of the paths, and this particular approach of searching for the optimal solution is realized using reachability analysis.

The performance of the approach was investigated for two different case studies from real life using the TA-based scheduling tool TAOpt, and experimental results were presented. In the first case study, a scheduling problem in a multiproduct pipeless plant demonstrator was modeled and solved. The TA-based approach can compute a first feasible solution very quickly and a very good, if not optimal, solution within 2 min of CPU time. This shows its applicability to handle scheduling problems with complex constraints in real time. The second case study concerns a real-world ice-cream manufacturing plant, and various instances of the case study were investigated. The results from the second case study show the ability of the approach to handle large-scale problems with production demands close to 100 orders. It was shown that the solution technique is flexible, and heuristics can be implemented easily in the reachability algorithm. The approach is able to handle problems of industrial size.

The modeling approach is flexible in the sense that a wide range of problem characteristics of batch scheduling (such as NIS and FIS storage policies, changeovers) can be included without difficulties. The efficiency of the solution approach by reachability analysis is achieved by search space reduction techniques and efficient algorithms from graph theory and from automata theory. The graphical approach to model building using TA and the modular definition of the elements of the problem lead to a transparent and user-friendly approach to handling scheduling problems. The solution technique by means of reachability analysis is able to obtain good feasible solutions quickly; however, for large-scale problems, the optimality of the solution can often not be guaranteed. Despite this disadvantage from a theoretical point of view, in practical situations where decisions have to be made within a limited amount of time, and where feasibility is more critical than optimality, the approach provides good feasible schedules quickly.

The above-mentioned features of the modeling approach and the effective solution technique make the TA-based approach a suitable candidate to be applied in real production environments where changes of the demands are frequent and good feasible solutions are needed in relatively small amounts of time. From the point of view of the time taken to specify the problem and to solve it, TA-based methods are competitive with the more established MILP techniques where determining a problem formulation that can be efficiently solved by MIP tools can be a time-consuming task.

In order to enhance the efficiency of the TA-based scheduling approach, the reachability algorithm can be augmented with bounding procedures to compute lower bounds that would help to prune the reachability tree. Current work includes the investigation of embedding LP-based bounding procedures and minimum remaining processing-time-based bounding routine to compute lower bounds for scheduling problems with changeovers.

APPENDIX

A. Problem Data for the Food Manufacturing Case Study. The processing line has a processing rate of 4500 kg/h irrespective of the product being processed.

Table 5. Data for the Packing Lines

product	packing rate	packing line
A	1750	P1
B	1500	P1
C	1000	P1
D	1500	P1
E	1750	P2
F	2000	P2
G	2000	P2
H	2000	P2

Table 6. Data for Minimum Waiting Time and Maximum Shelf Time Period

product	minimum waiting time [min]	maximum shelf time [min]
A	60	4320
B	180	4320
C	180	4320
D	0	4320
E	120	4320
F	120	4320
G	120	4320
H	120	4320

Table 7. Changeover Table of the Processing Line

	to								
from	Global	A	B	C	D	E	F	G	H
Global	0	120	120	120	120	120	120	120	120
A	120	0	30	30	30	30	30	30	30
B	120	30	0	30	30	30	30	30	30
C	120	30	30	0	30	30	30	30	30
D	120	30	30	30	0	30	30	30	30
E	120	30	30	30	30	0	15	15	15
F	120	30	30	30	30	5	0	15	15
G	120	30	30	30	30	5	5	0	15
H	120	30	30	30	30	5	5	5	0

Table 8. Changeover Table of the Packing Line P1

	to				
from	Global	A	B	C	D
Global	0	120	120	120	120
A	120	0	60	60	60
B	120	30	0	60	60
C	120	30	30	0	60
D	120	30	30	30	0

Table 9. Changeover Table of the Packing Line P2

	to				
from	Global	E	F	G	H
Global	0	120	120	120	120
E	120	0	60	60	60
F	120	30	0	60	60
G	120	30	30	0	60
H	120	30	30	30	0

Table 10. Production Demand

product	quantity in kg
A	80000
B	48000
C	32000
D	8000
E	112000
F	12000
G	48000
H	24000

AUTHOR INFORMATION

Corresponding Author

*E-mail: {s.subbiah|c.schoppmeyer|s.engell}@bci.tu-dortmund.de.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support from the NRW Graduate School of Production Engineering and Logistics at Technische Universitaet, Germany to S. Subbiah.

REFERENCES

(1) Kallrath, J. Planning and Scheduling in The Process Industry. *OR Spectrum* **2002**, *24*, 219–250.

(2) Floudas, C. A.; Lin, X. Continuous-Time Versus Discrete-Time Approaches for Scheduling of Chemical Processes: A Review. *Comput. Chem. Eng.* **2004**, *28*, 2109–2129.

(3) Mendez, C. A.; Cerda, J.; Grossmann, I. E.; Harjunkoski, I.; Fahl, M. State-of-The-Art Review of Optimization Methods for Short-Term Scheduling of Batch Processes. *Comput. Chem. Eng.* **2006**, *30*, 913–946.

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

(5) Ierapetritou, M. G.; Floudas, C. A. Effective Continuous-Time Formulation for Short-Term Scheduling. 2. Continuous and Semi-Continuous Processes. *Ind. Eng. Chem. Res.* **1998**, *37*, 4360–4374.

(6) Ierapetritou, M. G.; Floudas, C. A. Short-Term Scheduling: New Mathematical Models vs. Algorithmic Improvements. *Comput. Chem. Eng.* **1998**, *22*, 419–426.

(7) Kallrath, J. Combined Strategic and Operational Planning - AnMILP Success Story in Chemical Industry. *OR Spectrum* **2002**, 315–341.

(8) Maravelias, C. T.; Grossmann, I. E. New General Continuous-Time State-Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.* **2003**, *42*, 3056–3074.

(9) Castro, P. M.; Barbosa-Povoa, A.; Matos, H.; Novais, A. Simple Continuous-Time Formulation for Short-Term Scheduling of Batch and Continuous Processes. *Ind. Eng. Chem. Res.* **2004**, *43*, 105–118.

(10) Shaik, M. A.; Janak, S. L.; Floudas, C. A. Continuous-Time Models for Short-Term Scheduling of Multipurpose Batch Plants: A Comparative Study. *Ind. Eng. Chem. Res.* **2006**, *45*, 6190–6209.

(11) Mockus, L.; Reklaits, G. V. Mathematical Programming Formulation for Scheduling of Batch Operations Based on Non-uniform Time Discretization. *Comput. Chem. Eng.* **1997**, *21*, 1147–1156.

(12) Kondili, E.; Pantelides, C.; Sargent, R. A General Algorithm for Short-term Scheduling of Batch Operations - I. MILP Formulation. *Comput. Chem. Eng.* **1993**, *17*, 211–227.

(13) Lim, M.; Karimi, I. A. Resource-Constrained Scheduling of Parallel Production Lines Using Asynchronous Slots. *Ind. Eng. Chem. Res.* **2003**, *42*, 6832–6842.

(14) Sundaramoorthy, A.; Karimi, I. A. A Simpler Better Slot-Based Continuous-Time Formulation for Scheduling in Multipurpose Batch Plants. *Chem. Eng. Sci.* **2005**, *60*, 2697–2702.

(15) Allahverdi, A.; Gupta, N. D. J.; Aldowaisan, T. A Review of Scheduling Research Involving Setup Considerations. *Int. J. Manage. Sci.* **1999**, *27*, 219–239.

(16) Cheng, T. C. E.; Gupta, N. D. J.; Wang, G. A Review of Flowshop Scheduling Research with Setup Times. *J. Prod. Operations Manage.* **2000**, *9*, 262–282.

(17) Brucker, P.; Thiele, O. A Branch-and-Bound Method for The General-Shop Problem with Sequence-Dependent Setup-Times. *OR Spectrum* **1996**, *18*, 145–161.

(18) Artigues, C.; Feillet, D. A Branch-and-Bound Method for The Job Shop Problem with Sequence-Dependent Setup Times. *Ann. Operations Res.* **2008**, *159*, 135–159.

(19) Kelly, D. J.; Zyngier, D. An Improved MILP Modeling of Sequence-Dependent Switchovers for Discrete-Time Scheduling Problems. *Ind. Eng. Chem. Res.* **2007**, *46*, 4964–4973.

(20) Hui, C.-W.; Gupta, A.; Muelen, v. d. H. A. J. A Novel MILP Formulation for Short-Term Scheduling of Multi-Stage Multi-Product Batch Plants with Sequence-Dependent Changeovers. *Comput. Chem. Eng.* **2000**, *24*, 2705–2717.

(21) Erdirik-Dogan, M.; Grossmann, E. I. Planning Models for Parallel Batch Reactors with Sequence-Dependent Changeovers. *J. Am. Inst. Chem. Eng.* **2007**, *53*, 2284–2300.

(22) Erdirik-Dogan, M.; Grossmann, E. I. Slot-Based Formulation for The Short-Term Scheduling of Multi-Stage, Multi-Product Batch Plants with Sequence-Dependent Changeovers. *Ind. Eng. Chem. Res.* **2008**, *47*, 1159–1183.

(23) Castro, M. P.; Grossmann, E. I.; Novais, Q. A. Two New Continuous-Time Models for The Scheduling of Multistage Batch Plants with Sequence-Dependent Changeovers. *Ind. Eng. Chem. Res.* **2006**, *45*, 6210–6226.

(24) Castro, M. P.; Novais, Q. A. Scheduling of Multistage Batch Plants with Sequence-Dependent Changeovers. *J. Am. Inst. Chem. Eng.* **2009**, *55*, 2122–2136.

(25) Marchetti, A. P.; Cerda, J. A Continuous-Time Tightened Formulation for Single-Stage Batch Scheduling with Sequence-Dependent Changeovers. *Ind. Eng. Chem. Res.* **2009**, *48*, 483–498.

(26) He, Y.; Hui, C.-W. A Rule-Based Genetic Algorithm for The Scheduling of Single-Stage Multi-Product Batch Plants with Parallel Units. *Comput. Chem. Eng.* **2008**, *32*, 3067–3083.

(27) Sanmarti, E.; Friedler, F.; Puigjaner, L. Combinatorial Technique for Short-Term Scheduling of Multi-Purpose Batch Plants Based on Schedule-Graph Representation. In *Proceedings of the 8th European Symposium on Computer Aided Process Engineering*; Elsevier: New York, 1998; pp 847−850.

(28) Sanmarti, E.; Puigjaner, L.; Holczinger, T.; Friedler, F. Combinatorial Framework for Effective Scheduling of Multi-Purpose Batch Plants. *J. Am. Inst. Chem. Eng.* **2002**, *48*, 2557–2569.

(29) Majozi, T.; Friedler, F. Maximization of Throughput in a Multi-Purpose Batch Plant Under a Fixed Time Horizon: S-Graph Approach. *Ind. Eng. Chem. Res.* **2006**, *45*, 6713–6720.

(30) Alur, R.; Dill, D. L. *Real-Time: Theory in Practice*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 1992; Vol. 600; Chapter The Theory of Timed Automata, pp 45−73.

(31) Mounier, L.; Graf, S.; Bozga, M. A Validation Environment for Component-Based Real-Time Systems. *Proc. Comput. Aided Verification* **2002**, *2404*, 630–640.

(32) Behrmann, G.; Larsen, K.; Rasmussen, J. I. Optimal Scheduling Using Priced Timed Automata. *ACM Sigmetrics, Performance Evaluation Rev.* **2005**, *32*, 34–40.

(33) Yovine, S. Kronos: A Verification Tool for Real-Time Systems. *Int. J. Software Tools Technol. Transfer* **1997**, *1*, 123–133.

(34) Fehnker, A. Scheduling A Steel Plant with Timed Automata. In *Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications*; IEEE Press: Los Alamitos, CA, 1999; pp 280−286.

(35) Abdeddaim, Y.; Maler, O. *Computer Aided Verification*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2001; Vol. 2102, Chapter Job-Shop Scheduling Using Timed Automata, pp 478−492.

(36) Behrmann, G.; Fehnker, A.; Hune, T. S.; Petterson, P.; Larsen, K. G.; Romijn, J. *Tools and Algorithms for The Construction and Analysis of Systems*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2001; Vol. 2031, Chapter Efficient Guiding Towards Cost-Optimality in Uppaal, pp 174−188.

(37) Rasmussen, J. I.; Larsen, K. G.; Subramani, K. *Tools and Algorithms for the Construction and Analysis of Systems*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2004; Vol. 2988, Chapter Resource-Optimal Scheduling Using Priced Timed Automata, pp 220−235.

(38) Behrmann, G.; Brinksma, E.; Hendriks, M.; Mader, A.Scheduling Lacquer Production By Reachability Analysis - A Case Study. *Proceedings of the 16th IFAC World Congress*; Elsevier Science: New York, 2005.

(39) Panek, S.; Stursberg, O.; Engell, S. Job-Shop Scheduling By Combining Reachability Analysis with Linear Programming. *Proceedings of 7th Int. IFAC Workshop on Discrete Event Systems*; International Federation of Automatic Control: Laxenburg, Austria, 2004; pp 199−204.

(40) Panek, S.; Stursberg, O.; Engell, S. Efficient Synthesis of Production Schedules By Optimization of Timed Automata. *Control Eng. Practice* **2006**, *14*, 1183–1197.

(41) Abdeddaiem, Y.; Asarin, E.; Maler, O. Scheduling with Timed Automata. *Theor. Comput. Sci.* **2006**, *354*, 272–300.

(42) Panek, S.; Engell, S.; Subbiah, S.; Stursberg, O. Scheduling of Multi-Product Batch Plants Based Upon Timed Automata Models. *Comput. Chem. Eng.* **2008**, *32*, 275–291.

(43) Alur, R.; Henzinger, T. A. Modularity for Timed and Hybrid Systems. In *Proceedings of the 9th International Conference on Concurrency Theory*; Springer: New York, 1997; pp 74−88.

(44) Johan, B.; Wang, Y. *Lectures On Concurrency and Petri Nets*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2004; Vol. 3098, Chapter Timed Automata: Semantics, Algorithms and Tools, pp 87−124.

(45) Larsen, K. G.; Behrmann, G.; Brinksma, E.; Fehnker, A.; Hune, T.; Pettersson, P.; Romijn, J. *Computer Aided Verification*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2001; Vol. 2102, Chapter As Cheap As Possible: Efficient Cost-Optimal Reachability for Priced Timed Automata, pp 493−505.

(46) Larsen, K. G. Priced Timed Automata: Theory and Tools. In *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*; Schloss Dagstuhl: Wadern, Germany, 2009; Vol 4, pp 417−425.

(47) Pantelides, C. Unified Frameworks for Optimal Process Planning and Scheduling. In *Proceeding of 2nd Conference on Foundations on Computer-Aided Process Optimization*; Elsevier: New York, 1994; pp 253−274.

(48) Yovine, S. *Embedded Systems*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 1998; Vol. 1494, Chapter Model-Checking Timed Automata, pp 114−152.

(49) Daws, C.; Tripakis, S. *Tools and Algorithms for Construction and Analysis of Systems*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 1998; Vol. 1384, Chapter Model Checking of Real-Time Reachability Properties Using Abstractions, pp 313−329.

(50) Behrmann, G.; Bengtsson, J.; David, A.; Larsen, K. G.; Pettersson, P.; Yi., W. *Formal Techniques in Real-Time and Fault Tolerant Systems*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2002; Vol. 2469, Chapter Uppaal Implementation Secrets, pp 3−22.

(51) Behrmann, G.; Larsen, K. G.; Pearson, J.; Weise, C.; Yi, W. *Computer Aided Verification*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 1999; Vol. 1633, Chapter Efficient Timed Reachability Analysis Using Clock Difference Diagrams, pp 682−695.

(52) Subbiah, S.; Tometzki, T.; Engell, S.; Panek, S. Multi-Product Batch Plants Scheduling with Intermediate Due Dates Using Priced Timed Automata Models. *Comput. Chem. Eng.* **2009**, *33*, 1661–1676.

(53) Behrmann, G.; Bouyer, P.; Larsen, K. G.; Pelanek, R. *Tools and Algorithms for The Construction and Analysis of Systems*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2004; Vol. 2988, Chapter Lower and Upper Bounds in Zone Based Abstractions of Timed Automata, pp 312−326.

(54) Tometzki, T.; Neymann, T.; Steimel, J.; Subbiah, S.; Engell, S. An Efficient and User-friendly Optimization Framework for Batch Process Scheduling. *Proceedings of the IFAC Conference on Management and Control of Production Logistics*; International Federation of Automatic Control: Laxenburg, Austria, 2010.

(55) Panek, S.; Stursberg, O.; Engell, S. In *Logistics of Chemical Production Processes*; Engell, S., Ed.; Wiley-VCH: New York, 2006;

Chapter Scheduling Based on Reachability Analysis of Timed Automata, pp 215−235.

(56) Godefroid, P. *Computer-Aided Verification*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 1991; Vol. 531, Chapter Using Partial Orders To Improve Automatic Verification Methods, pp 176−185.

(57) Panek, S.; Stursberg, O.; Engell, S. *Formal Modeling and Analysis of Timed Systems*; Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, 2004; Vol. 2791, Chapter Optimization of Timed Automata Models Using Mixed-Integer Programming, pp 73−87.

(58) Bongers, M. M. P.; Bakker, B. H. Application of Multi-Stage Scheduling. In *Proceedings of the 16th European Symposium on Computer Aided Process Engineering*; Elsevier: New York, 2006; pp 1917−1922.

(59) Henning, G.; Cerdá, J. ESA: An Expert Scheduling Assistant for Batch Processes. *Foundations Comput.-Aided Proc. Operations* **1994**, 403–408.

(60) Pinto, J. M.; Grossmann, I. E. A Continuous Time MILP Model for Short-Term Scheduling of Batch Plants. *Ind. Eng. Chem. Res.* **1995**, 3037–3051.

(61) Pinto, J. M.; Grossmann, I. E. A Logic-Based Approach To Scheduling Problems with Resource Constraints. *Comput. Chem. Eng.* **2009**, *21*, 801–818.

(62) Méndez, C. A.; Cérda, J. An MILP Framework for Short-Term Scheduling of Single-Stage Batch Plants with Limited Discrete Resources. In *Proceedings of the 12th European Symposium on Computer Aided Process Engineering*; Elsevier: New York, 2002; pp 721−726.

(63) Janak, S. L.; Lin, X.; Floudas, C. A. Enhanced Continuous-Time Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multipurpose Batch Processes: Resource Constraints and Mixed Storage Policies. *Ind. Eng. Chem. Res.* **2004**, *43*, 2516–2533.

(64) Marchetti, A. P.; Cerda, J. A General Resource-Constrained Scheduling Framework for Multi-Stage Batch Facilities with Sequence-Dependent Changeovers. *Comput. Chem. Eng.* **2009**, *33*, 871–886.