Hybrid Time Slots Sequencing Model for a Class of Scheduling Problems

Pedro M. Castro
Unidade de Modelação e Optimização de Sistemas Energéticos, Laboratório Nacional de Energia e Geologia, 1649-038 Lisboa, Portugal

Luis J. Zeballos
Universidad Nacional del Litoral—Facultad de Ingeniería Química, Santa Fe, Argentina

Carlos A. Méndez
INTEC (Universidad Nacional del Litoral—CONICET), Santa Fe, Argentina

DOI 10.1002/aic.12609
Published online March 31, 2011 in Wiley Online Library (wileyonlinelibrary.com).

This article presents a new model for the short-term scheduling of multistage batch plants with a single unit per stage, mixed storage policies, and multiple shared resources for moving orders between stages. Automated wet-etching stations for wafer fabrication in semiconductor plants provide the industrial context. The uncommon feature of the continuous-time model is that it relies on time grids, as well as on global precedence sequencing variables, to find the optimal solution to the problem. Through the solution of a few test cases taken from the literature, we show that new model performs significantly better than a pure sequencing formulation and better than a closely related hybrid model with slightly different sequencing variables. We also propose a new efficient heuristic procedure for extending the range of problems that can effectively be solved, which essentially solves relaxed and constrained versions of the full-space model. © 2011 American Institute of Chemical Engineers *AIChE J,* 58: 789–800, 2012

Keywords: optimization, mixed-integer linear programming, mathematical modeling, continuous-time, robot

Introduction

The key element of scheduling is ensuring that equipment resources do not handle multiple competing tasks simultaneously. In other words, tasks executed on the same unit must be sequenced, which can be done in a variety of ways.[1] Sequencing can be done explicitly, through the use of immediate or general precedence sequencing variables, or implicitly, by assigning tasks to different slots of a time grid. Sequencing variables models are more appropriate for specific plant topologies, such as multistage plants, where a given equipment unit is associated to a single production stage and product identity is kept. On the other hand, time grid based models are typically employed for simultaneous batching (number and size of batches) and scheduling in multipurpose plants with complex production recipes, where there is also the need to keep track of material resources over time. Time grid models are also linked to problems featuring renewable shared resources other than equipment, such as manpower[2–4] or utilities.[4–7] Nevertheless, the sequencing variables approach of Méndez and Cerdá[3] has been shown[1] a better alternative for a single stage plant with limited manpower.

Under the context of multistage, multiproduct plants, Gupta and Karimi[8] have used immediate precedence sequencing variables to schedule product orders subject to both release and due dates on a set of distinct parallel units with sequence-dependent setups. An important aspect of the

Correspondence concerning this article should be addressed to P. M. Castro at pedro.castro@lneg.pt.

mixed integer linear programming (MILP) model is that it features big-M constraints to sequence different orders assigned to the same unit, which are known to yield poor relaxations. The same occurs with the concept of general precedence,[9,10] which leads to more efficient models primarily due to the reduction by half of the number of binary sequencing variables. The drawback is a minor loss of accuracy that may lead to suboptimal solutions as noted by Castro et al.[11]

Sequencing approaches can also address simultaneous batching and scheduling problems, potentially leading to better solutions than if both decision levels are decoupled. Maravelias and coworkers[12,13] have proposed MILP models that involve the selection of one or more batches (from a postulated set) and corresponding size, toward satisfaction of the given production orders. The general[12] or immediate[13] precedence sequencing variables feature five indices (order, batch, order, batch, stage) leading to a rapid increase in the number of binary variables with problem size. For single stage plants under the restriction of a single batch size,[14] the number of indices can be reduced to just 3 (order, order, unit), if a new set of integer variables is added to determine the number of batches required for each order on a particular unit. However, the resulting model was orders of magnitude slower than one using multiple time grids to keep track of events taking place. One possible reason for this behavior is the absence of big-M constraints in the latter.

If one chooses to rely on time grids instead of sequencing variables there are a few more possibilities. Some of those that are suitable for multistage plants under an unlimited intermediate storage policy and without shared resources (other than equipments) have recently been evaluated by Castro and coworkers.[11,15–17] If one focuses on continuous-time approaches, the most important insight is that it is undoubtedly better to rely on multiple time grids, also known as unit-specific, than on a single grid with global events. With the former, one can assume without loss of generality that a single time slot is enough for every processing task,[15] whereas in the latter, tasks will span across multiple slots.[6] Note that unit-specific models need also to allow tasks to span over multiple event points[18] when dealing with multipurpose plants. The other important insight is that there are[16] at least three good, conceptually different unit-specific approaches, more so if sequence-dependent changeovers are involved.[11,17] The one by Castro and Grossmann[15] has the important property of being a minimum event point approach, while the ones by Castro and Novais[16] and Shaik and Floudas[18,19] have a wider scope (they allow for both batching and scheduling).

The most serious drawback of time grid based approaches is the uncertainty in the number of event points, which typically leads to the requirement of a few iterations before finding the global optimal solution. To address this issue, Li and Floudas[20] have recently proposed a fairly accurate prediction method for unit-specific models.[4,18,19] On the other hand, sequencing models need to be solved only once and can find very good solutions in the early nodes of the branch-and-bound tree despite sometimes being difficult to close the optimality gap. Besides mathematical programming, other approaches can be used to tackle scheduling problems in multistage plants such as constraint programming,[10,11,15,21] timed automata,[22] and genetic algorithms.[23]

The general scheduling problem of multistage multiproduct plants with resource constraints has recently been addressed by Sundaramoorthy et al.[24] The plant topology features nonidentical processing units in each stage with intermediate and final storage vessels, where each unit is associated to a single stage. Utilities are resources shared by processing tasks provided that maximum availability is not exceeded. The robot resources considered in this article are discrete entities that are shared by transportation tasks that move the orders from one stage to the next. While there are obvious similarities between the robot units and the transportation vessels, the main distinction is that a particular robot can handle transportation tasks belonging to different production stages and not just one. This more complex plant topology leads to major changes in the mathematical formulation, and cannot be considered a special case of the model by Sundaramoorthy et al.,[24] which also considers assignment decisions. Furthermore, the proposed model relies on a continuous instead of a discrete-time representation.[24]

Automated wet-etching stations

In semiconductor plants, wafer fabrication is the most important process.[25] Following imprinting of integrated circuits, the wafers must go through a series of chemical and deionizing baths. This step is known as etching and involves automated transfers of wafer lots between baths with strict requirements in exposure times. An automated wet-etching station (AWS) facility can be classified as a multistage plant with no intermediate storage (NIS) between baths and a zero-wait (ZW) policy on the chemical baths. The most challenging aspect is however the incorporation of the material handling constraints arising from automated transfers. In this article, we consider that automated transfers are performed by multiple robots that can be viewed as shared resources.

Geiger et al.[26] developed a heuristic algorithm for AWS based on tabu search, for scheduling transfers and processing for a given sequence of jobs on baths. Bhushan and Karimi[27] obtained better results using other heuristic procedures such as simulated annealing and heuristic algorithms for determining the initial sequencing and timing of jobs. For small instances, the same authors have proposed[28] a MILP model for makespan minimization that is able to determine the optimal sequence of jobs in the baths as well as robot transfers. The model consists of a slot based formulation for scheduling under unlimited transportation resources coupled with typical big-M constraints from sequencing models to ensure that the available robot executes one transfer at a time. It was also used as the basis of a two step heuristic for larger problems, where the unlimited robot model first defines the best job sequence and the constrained one robot model then generates a feasible solution. Combining the strengths of the slot based formulation, which does not require big-M constraints for sequencing jobs within a unit, with the advantage of the sequencing model of not requiring the definition of a large number of slots to tackle problems where many tasks are involved, was a major breakthrough. Somewhat related but on a higher level, Sundaramoorthy and Maravelias,[29] have recently proposed a unified framework for processes comprising network and sequential subsystems that expresses the latter using a material-like

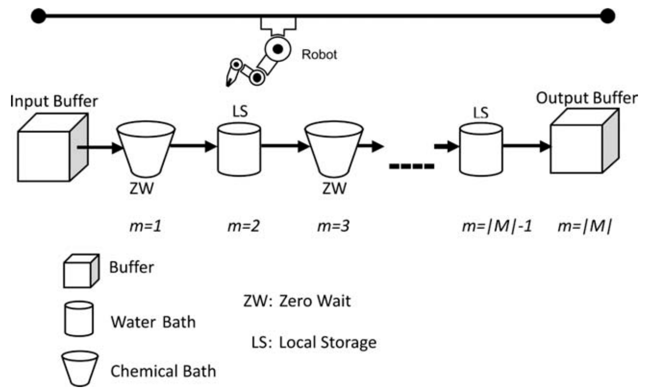

Figure 1. Schematic of an AWS.

formalism typical of the former but with special features, under the context of a discrete-time formulation.

This article proposes a hybrid time grid sequencing model for the optimal scheduling of AWS. The main novelty is the incorporation of the concept of general precedence, whereas Bhushan and Karimi[28] sequencing variables are defined for a narrower neighborhood. The new model is shown to be more efficient and faster at finding good solutions, particularly with the increase in the number of baths. A two-step heuristic strategy is also proposed that instead of relying on a single solution from the unlimited robot model in the first step, generates a few different job sequences that are within a specified optimality gap, with each being then used to constrain the one robot model in the second step. The insight gained is that the best sequence for the one robot model is sometimes suboptimal for the unlimited case. In other words, better solutions may result with more or less the same computational effort when compared with the previous approach.

Problem Definition

An AWS deals with groups of wafers of the same type, named wafer lots. Lots in the input buffer come from the upstream process and are destined to the output buffer, which follows processing in the last bath. In this type of technology, wafer lots are moved from one chemical or water bath to the next by means of automated material handling devices, referred to as robots. All lots follow the same recipe in sequence, see Figure 1. Overexposure in the chemical baths can damage the wafers, so the contact time must be controlled strictly. This is known as a ZW policy. On the other hand, overexposure to water is safe, so a wafer can stay in a water bath beyond its processing time. Water baths can thus act as local storage (LS). Indeed, wafers can only wait at water baths, since there is NIS and robots cannot be used as a temporary buffer. Each robot moves a single wafer lot at a time and needs to remove a lot from a bath before moving another lot to that same bath. In other words, baths hold at most one lot at a time to avoid contamination.

Given an AWS consisting of $m \in M$ units divided into $|M|$-1 baths and an output buffer, $i \in I$ wafer lots and multiple robots $r \in R$ for moving lots across baths, the objective is to determine the sequencing of lots at the units, the assignment of transportation tasks to robots and their sequence so as to minimize the makespan. Fixed processing times, $p_{i,m}$, and transportation times to a given unit/buffer, π_m, the latter not depending on robot positioning, are assumed. Note that the output buffer is not an actual processing stage ($p_{i,|M|}$ is not defined) but one must ensure robot availability and also account for the transportation times of all lots to it.

It is important to highlight that this can be seen as a $2\cdot(|M|\text{-}1)+1$ stages problem, featuring tasks requiring dedicated equipment units in the even stages (the chemical and water baths), and requiring shared resources in the odd stages (the robots).

General Precedence Sequencing Variables Model

Aguirre and Méndez[30] have recently proposed a MILP based on global precedence sequencing variables for AWS with multiple robots. The model given next is essentially their model despite minor changes in the definition of the sequencing variables. Three sets of binary variables are employed, being two sequencing variables:

$$X_{i,i'} = \begin{cases} 1 & \text{if lot } i \text{ is sequenced before } i' \\ 0 & \text{otherwise} \end{cases} \quad \forall i,i' \in I, i' > i \tag{1}$$

$$Y_{i,m,i',m'} = \begin{cases} 1 \text{ if lot } i \text{ in unit } m \text{ is sequenced before lot } i' \text{ in } m' \\ 0 \text{ otherwise} \end{cases}$$
$$\forall i,i' \in I, i' > i, m,m' \in M, m \neq m' \tag{2}$$

Binaries $W_{i,m,r}$ indicate that the transfer of lot i to unit m is assigned to robot r and there are two sets of continuous variables, $Ts_{i,m}$ and $Tf_{i,m}$, which give the starting and ending times of lot i in unit m, respectively.

The ZW and LS policies are enforced through Eqs. 3 and 4. Equation 5 then ensures that the robot chosen for transfer does not hold lots longer than it should, i.e. the starting time in unit $m+1$ is equal to the ending time in m plus the transfer time to $m+1$. Naturally, the starting time in the first chemical bath must be greater than the transfer time to it, Eq. 6. Sequencing constraints between different jobs on the same bath are enforced through Eqs. 7 and 8, where H is a parameter representing the time horizon. Notice that the former is enforced if lot i is sequenced before i' whereas the latter is enforced otherwise.

$$Tf_{i,m} = Ts_{i,m} + p_{i,m} \quad \forall i \in I, m \in ZW \tag{3}$$

$$Tf_{i,m} \geq Ts_{i,m} + p_{i,m} \quad \forall i \in I, m \in LS \tag{4}$$

$$Ts_{i,m+1} = Tf_{i,m} + \pi_{m+1} \quad \forall i \in I, m \in M, m \neq |M| \tag{5}$$

$$Ts_{i,1} \geq \pi_1 \quad \forall i \in I \tag{6}$$

$$Ts_{i',m} \geq Tf_{i,m} + \pi_m + \pi_{m+1} - H \cdot (1 - X_{i,i'}) \forall i,i' \in I, i' > i, \; m \in M, m \neq |M| \tag{7}$$

$$Ts_{i,m} \geq Tf_{i',m} + \pi_m + \pi_{m+1} - H \cdot X_{i,i'} \forall i,i' \in I, i' > i, \; m \in M, m \neq |M| \tag{8}$$

If the transfer of any two lots to different units is handled by the same robot, they cannot occur simultaneously, see Eqs

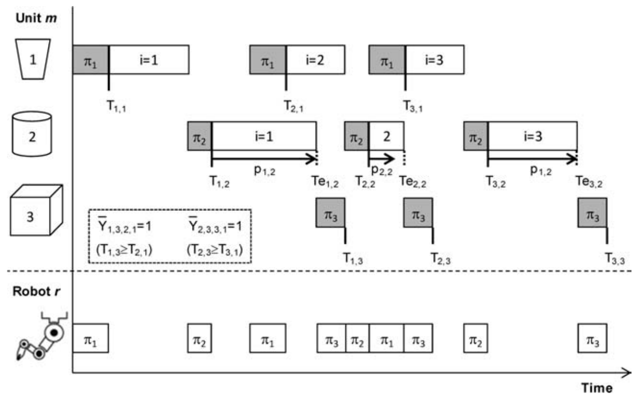
Unit m
1
2
3
Robot r
Time
i=1
i=2
i=3
2
$T_{1,1}$ $T_{2,1}$ $T_{3,1}$
$T_{1,2}$ $p_{1,2}$ $Te_{1,2}$ $T_{2,2}$ $p_{2,2}$ $Te_{2,2}$ $T_{3,2}$ $p_{1,2}$ $Te_{3,2}$
$T_{1,3}$ $T_{2,3}$ $T_{3,3}$
$\bar{Y}_{1,3,2,1}=1$ $(T_{1,3} \geq T_{2,1})$
$\bar{Y}_{2,3,3,1}=1$ $(T_{2,3} \geq T_{3,1})$
π_1 π_2 π_3

Figure 2. Diagram for understanding timing variables of the new hybrid model.

9,10. Naturally, all transfers must be assigned to one robot, Eq. 11. Equation 12 defines the makespan, *MS*, the variable to minimize (Eq. 13). It is important to highlight that there are two new terms on the LHS. Although not strictly necessary (not considered by Aguirre and Méndez[30]), since the transition time to the output buffer is accounted for in the determination of $Ts_{i,|M|}$ (see Eq. 5 when applied to unit m=|M|-1), these terms make the formulation tighter by including for all lots i' sequenced after i, the processing time in the last water bath plus the transfer time to the output buffer. The one robot version of this model will be referred to as O-AM.

$$Ts_{i,m} + \pi_{m'} \leq Ts_{i',m'} + H \cdot (3 - Y_{i,m,i',m'} - W_{i,m,r} - W_{i',m',r}) \quad \forall i, i' \in I, i' > i, m, m' \in M, m \neq m', r \in R \tag{9}$$

$$Ts_{i',m'} + \pi_m \leq Ts_{i,m} + H \cdot (2 + \bar{Y}_{i,m,i',m'} - W_{i,m,r} - W_{i',m',r}) \quad \forall i, i' \in I, i' > i, m, m' \in M, m \neq m', r \in R \tag{10}$$

$$\sum_{r \in R} W_{i,m,r} = 1 \quad \forall i \in I, m \in M \tag{11}$$

$$Ts_{i,|M|} + \sum_{\substack{i' \in I \\ i' > i}} X_{i,i'} \cdot (p_{i',|M|-1} + \pi_{|M|}) + \sum_{\substack{i' \in I \\ i > i'}} (1 - X_{i',i}) \cdot (p_{i',|M|-1} + \pi_{|M|}) \leq MS \quad \forall i \in I \tag{12}$$

$$\min MS \tag{13}$$

Remarks

The model is implicitly assuming that the robots are allocated to different physical areas around the lineal configuration of baths meaning that their paths do not overlap. This is apparent in Eqs. 7,8, where transfer times to and from unit m are included to ensure that the robot can only pick up lot i' from unit m-1 on the way back from moving lot i to unit m+1. If, on the other hand, one assumes that the robot paths can overlap, lower makespans can be attained as will be seen later on. This has been the approach of Zeballos et al.[21] who have proposed a constraint programming model coupled with an efficient search strategy. The necessary adjustments in the model constraints involve replacing Eqs. 7,8 with Eqs. 14–17 and Eq. 12 with Eq. 18. This model will be referred to as AM.

$$Ts_{i',m} \geq Tf_{i,m} - H \cdot (1 - X_{i,i'}) \; \forall i, i' \in I, i' > i, m \in M, m \neq |M| \tag{14}$$

$$Ts_{i,m} \geq Tf_{i',m} - H \cdot X_{i,i'} \quad \forall i, i' \in I, i' > i, m \in M, m \neq |M| \tag{15}$$

$$Ts_{i',m} \geq Tf_{i,m} + \pi_m + \pi_{m+1} - H \cdot (3 - X_{i,i'} - W_{i,m,r} - W_{i',m,r}) \quad \forall i, i' \in I, i' > i, m \in M, m \neq |M|, r \in R \tag{16}$$

$$Ts_{i,m} \geq Tf_{i',m} + \pi_m + \pi_{m+1} - H \cdot (2 + X_{i,i'} - W_{i,m,r} - W_{i',m,r}) \quad \forall i, i' \in I, i' > i, m \in M, m \neq |M|, r \in R \tag{17}$$

$$Ts_{i,|M|} + \sum_{\substack{i' \in I \\ i' > i}} X_{i,i'} \cdot p_{i',|M|-1} + \sum_{\substack{i' \in I \\ i > i'}} (1 - X_{i',i}) \cdot p_{i',|M|-1} \leq MS \; \forall i \in I \tag{18}$$

New Hybrid Time Grid/Sequencing Variables Model

The AWS scheduling problem has some features that can be considered a special case of the typical multistage plant scheduling problem. In terms of plant topology, and neglecting transfer constraints, AWS feature a single unit per stage meaning that we only need to be concerned with lot sequencing. As a consequence, multiple time grid approaches become even more attractive, since there is now no uncertainty in the number of slots to specify for each unit, meaning that there is no need to implement an iterative search procedure over the number of slots to find the global optimal solution.[1] In fact, the number of slots to specify for each unit is equal to the number of lots.[28]

Unlimited robot model (URM)

Starting from the model introduced by Castro and Grossmann,[15] we derive the MILP model for the unlimited robot case by adding new variables and constraints to model the features of AWS associated with the NIS and ZW policies. Consider a time grid with $t \in T$ slots with $|T| = |I|$. Binary variables $N_{i,t}$ assign lot i to slot t, with the latter index implicitly giving the position of lot i in the sequence. Notice that unit index m has been dropped[15] since the initial sequence is kept throughout the processing stages due to the NIS and ZW policies. Continuous variables $T_{t,m}$ indicate the starting time of slot t in unit m, while $Te_{t,m}$ give the time at which processing ends in water bath m ($m \notin$ ZW). Figure 2 provides a Gantt chart with the location of the model variables for a simple example.

Equation 19 states that the duration of slot t in unit m must be greater than the processing time of the lot assigned to the slot. While this is true for all units, the constraint is written for chemical baths only, since for the water baths the relation is with the end of processing in the previous slot, Eq. 20. The starting time in water bath unit m+1 must be equal to the start of the same slot t in the previous unit plus processing and transfer times, Eq. 21. In contrast, the end of processing in water bath units may go beyond the lot processing time, see Eq. 22 and Figure 2 for $i = 2$ in $m = 2$. Still, the starting time of the lot in the next unit must be equal to its ending plus transfer times, Eq. 23. Equation 24 ensures that the starting time of the first slot in the first unit must be greater than the transfer time to it.

$$T_{t+1,m} \geq T_{t,m} + \sum_{i \in I} N_{i,t} p_{i,m} \quad \forall m \in ZW, t \in T, t \neq |T| \tag{19}$$

$$T_{t+1,m} \geq Te_{t,m} \quad \forall m \in LS, t \in T, t \neq |T| \tag{20}$$

$$T_{t,m+1} = T_{t,m} + \sum_{i \in I} N_{i,t} p_{i,m} + \pi_{m+1} \quad \forall m \in ZW, t \in T \tag{21}$$

$$Te_{t,m} \geq T_{t,m} + \sum_{i \in I} N_{i,t} p_{i,m} \quad \forall m \in LS, t \in T \tag{22}$$

$$T_{t,m+1} = Te_{t,m} + \pi_{m+1} \quad \forall m \in LS, t \in T \tag{23}$$

$$T_{1,1} \geq \pi_1 \tag{24}$$

Equations 25,26 enforce that a slot holds exactly one lot. The makespan is defined through Eq. 27, while Eq. 28 makes the formulation more efficient. It simply adds the lot processing times in unit m and subsequent units to their transfer times. The unlimited robot model, featuring no big-M constraints, is completed with Eq. 13.

$$\sum_{t \in T} N_{i,t} = 1 \quad \forall i \in I \tag{25}$$

$$\sum_{i \in I} N_{i,t} = 1 \quad \forall t \in T \tag{26}$$

$$T_{|T|,|M|} \leq MS \tag{27}$$

$$T_{t,m} + \sum_{i \in I} N_{i,t} \cdot \sum_{\substack{m' \in M \\ m' \geq m \wedge m' \neq |M|}} (p_{i,m'} + \pi_{m'+1}) \leq MS$$
$$\forall m \in M, m \neq |M|, t \in T \tag{28}$$

Multiple robot model (MRM)

In order to model the transportation tasks of the robot resources, we use the concept of general precedence. When compared with the approach of Aguirre and Méndez[30] we are relating different unit-slot pairs like in Bhushan and Karimi[28] instead of lot-unit pairs. The sequencing variables are the following:

$$\bar{Y}_{t,m,t',m'} = \begin{cases} 1 & \text{if slot } t \text{ in unit } m \text{ starts after slot } t' \text{ in } m' \\ 0 & \text{otherwise} \end{cases}$$
$$\forall t, t' \in T, t' > t, m, m' \in M, m \neq m' \tag{29}$$

We will also be needing robot assignment variables for the transfer tasks, $\bar{W}_{t,m,r}$.

Equations 30,31 are active whenever the same robot is assigned to transfer tasks of consecutive lots into and out of the same unit. Then, Eqs. 32,33 are the equivalents of Eq. 9,10. Finally, all transfers must be assigned to one robot, Eq. 34. In summary, the multiple robot model comprises Eqs. 13, 19–28, and 30–34.

$$T_{t+1,m} \geq T_{t,m} + \sum_{i \in I} N_{i,t} p_{i,m} + \pi_{m+1} + \pi_m$$
$$- H \cdot (2 - \bar{W}_{t+1,m,r} - \bar{W}_{t,m+1,r}) \quad \forall m \in ZW, t \in T,$$
$$t \neq |T|, r \in R \tag{30}$$

$$T_{t+1,m} \geq Te_{t,m} + \pi_{m+1} + \pi_m - H \cdot (2 - \bar{W}_{t+1,m,r} - \bar{W}_{t,m+1,r})$$
$$\forall m \in LS, t \in T, t \neq |T|, r \in R \tag{31}$$

$$T_{t,m} \geq T_{t',m'} + \pi_m - H \cdot (3 - \bar{Y}_{i,m,t',m'} - \bar{W}_{t,m,r} - \bar{W}_{t',m',r})$$
$$\forall t, t' \in T, t' > t, m, m' \in M, m \neq m', r \in R \tag{32}$$

$$T_{t',m'} \geq T_{t,m} + \pi_{m'} - H \cdot (2 + \bar{Y}_{t,m,t',m'} - \bar{W}_{t,m,r} - \bar{W}_{t',m',r})$$
$$\forall t, t' \in T, t' > t, m, m' \in M, m \neq m', r \in R \tag{33}$$

$$\sum_{r \in R} \bar{W}_{t,m,r} = 1 \quad \forall t \in T, m \in M \tag{34}$$

One robot model (ORM)

While the multiple robot model can deal with just a single robot, its complexity can be significantly reduced for such special case. First, we can avoid the big-M constraints associated to transfers of consecutive lots into and out of the same unit (Eqs. 35,36 replace Eqs. 30,31). Equations 35,36 can also replace Eqs. 19,20 in model URM to get a lower bound on the solution that ORM can obtain. Such relaxed one robot model will be named R-ORM. Second, and more importantly, the domain of the sequencing variables can drastically be reduced.

$$T_{t+1,m} \geq T_{t,m} + \sum_{i \in I} N_{i,t} p_{i,m} + \pi_{m+1} + \pi_m$$
$$\forall m \in ZW, t \in T, t \neq |T| \tag{35}$$

$$T_{t+1,m} \geq Te_{t,m} + \pi_{m+1} + \pi_m \quad \forall m \in LS, t \in T, t \neq |T| \tag{36}$$

The sequencing values that take the value of 1 can be seen in Figure 2 for a simple example comprising a single robot. The fact that there are just two equal to 1 makes us wonder if the domain in Eq. 29 can be reduced. Indeed, it can be significantly reduced as will be shown in the next subsection. For now, let us assume that subsets *YD0* and *YD1* represent the cases where variables $\bar{Y}_{t,m,t',m'}$ are not necessarily equal to 0 and 1, respectively. Furthermore, sets *YE0* and *YE1* hold the quartets that are equal to 0 and 1.

The sequencing constraints that prevent simultaneous transfer of lots by the robot are given by Eqs. 37,38, which replace Eqs. 32,33. Equation 34 can also be avoided. Notice that if $\bar{Y}_{t,m,t',m'} = 0$, Eq. 37 becomes irrelevant and does not need to be written. If one knows a priori that the value must be equal to 1, then the big-M term is not required to make the starting time of slot t in unit m greater than the starting time of slot t' in unit m' plus the transfer time to m. If, on the other hand, the binary is equal to zero, slot t in unit m starts earlier than t' in m', which is ensured by Eq. 38.

$$T_{t,m} \geq T_{t',m'} + \pi_m - \left[H \cdot (1 - \bar{Y}_{t,m,t',m'})\right]\Big|_{(t,m,t',m') \notin YE1}$$
$$\forall (t, m, t', m') \in YD0 \tag{37}$$

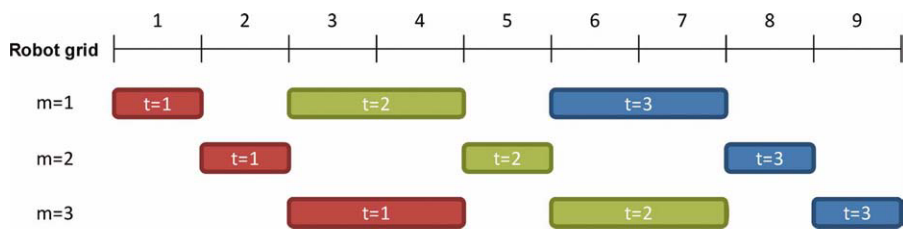

Figure 3. Domain of transportation tasks for a simple example featuring |*I*|=3 and |*M*|=3.
[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com].

$$T_{t',m'} \geq T_{t,m} + \pi_{m'} - \left[H \cdot \bar{Y}_{t,m,t',m'}\right]\big|_{(t,m,t',m')\notin YE0} \quad \forall (t,m,t',m') \in YD1 \qquad (38)$$

Sequencing variables domain reduction for ORM

We now focus on tightening the domain of the sequencing variables for the one robot model using information about the process. More specifically, the robot needs first to remove a lot from a bath before picking up another lot and bringing it to that same bath. Consider the lot that will be assigned to the first position in the sequence. Clearly, the first transportation tasks performed by the robot will involve moving such lot into the first chemical bath and into the first water bath (m=2). Only then, can the second lot in the sequence be picked from the input buffer. This analysis can be systematized rather easily if the transportation tasks are reported on a hypothetical time grid that accounts for the robot events. Such grid would require a total of $|I|\cdot|M|$ events.

Recall the example in Figure 2, featuring 3 lots and 3 units. The domain for the transportation tasks on the robot grid is given in Figure 3 as a function of the time slot (bath grids) the lot is assigned to, and the equipment unit. One can see that the domain is quite narrow, with most transportation tasks being confined to a single slot. Only pairs (t = 1, m = 3), (2,1), (2,3), and (3,1) have two possibilities. Based on this diagram, the sets involved in Eqs. 23 and 24 are easy to determine. For instance, $YD0$={(1,3,2,1),(2,3,3,1)} and $YE1$ = Ø. Notice that the schedule in Figure 2, features slot 1 in unit 3 starting after slot 2 in unit 1, as well as slot 2 in unit 3 following slot 3 in unit 1, so the two sequencing variables that could be different than 0 are actually equal to 1.

Assuming that the $LB_{m,t}$ and $UB_{m,t}$ give the lowest/highest possible interval in the robot grid for transferring to unit m the lot in position t in the bath sequence, we have:

$$LB_{m,t} = \sum_{\substack{t' \in T \\ t' \leq t}} \sum_{\substack{m' \in M \\ m' \leq m+t-t'}} 1 \quad \forall m \in M, t \in T \qquad (39)$$

$$UB_{m,t} = |I| \cdot |M| + 1 - \sum_{\substack{t' \in T \\ t' \geq t}} \sum_{\substack{m' \in M \\ m' \geq m+t-t'}} 1 \quad \forall m \in M, t \in T \qquad (40)$$

With these parameters, the elements of sets $YE0$, $YD0$, $YE1$, and $YD1$ can be calculated through Eqs. 41–44. Notice that $UB_{m,t} \leq LB_{m',t'} \Rightarrow T_{t,m} < T_{t',m'}$ and $\bar{Y}_{t,m,t',m'} = 0$. Likewise, if $UB_{m',t'} \leq LB_{m,t} \Rightarrow T_{t,m} > T_{t',m'}$ and $\bar{Y}_{t,m,t',m'} = 1$. Such binary variables can thus be removed from the formulation.

$$YE0 = \{(t,m,t',m') : t,t' \in T, t' > t, m, m' \in M, m \neq m', UB_{m,t} \leq LB_{m',t'}\} \qquad (41)$$

$$YE1 = \{(t,m,t',m') : t,t' \in T, t' > t, m, m' \in M, m \neq m', UB_{m',t'} \leq LB_{m,t}\} \qquad (42)$$

$$YD0 = \{(t,m,t',m') : t,t' \in T, t' > t, m, m' \in M, m \neq m', UB_{m,t} > LB_{m',t'}\} \qquad (43)$$

$$YD1 = \{(t,m,t',m') : t,t' \in T, t' > t, m, m' \in M, m \neq m', UB_{m',t'} > LB_{m,t}\} \qquad (44)$$

Remarks

A full time grid based, one robot model, using slot assignment variables for both bath and robot grids, was also developed. The latter grid used $|I|\cdot|M|$ slots while all former grids kept the $|I|$ slots. Unfortunately, the model was orders of magnitude slower primarily because the domains of the additional (bath slot, unit, robot slot) variables are larger than that of the global precedence sequencing variables. This result is consistent with findings by Bhushan and Karimi.[28]

The sequencing variables used by Bhushan and Karimi[28] are somewhat between the concepts of immediate and general precedence. The exact definition using this article's nomenclature is given in Eq. 45. The authors then employed sequencing constraints similar to Eqs. 37,38, for a tighter domain than the one in Eq. 45, coupled with one mandatory and two performance enhancing sets of constraints involving sequencing variables only. A comparison between their approach and ours is provided later on in terms of computational statistics.

$$\bar{Y}_{t,m,t',m'} = \begin{cases} 1 & \text{if slot } t \text{ in unit } m \text{ starts after slot } t' \text{ in } m' \\ & \text{and before slot } t' \text{ in } m'+1 \\ 0 & \text{otherwise} \end{cases} \quad \forall t,t' \in T, t' < t, m, m' \in M, m < m' \qquad (45)$$

New Heuristic Method

Increasing the number of robots makes the MRM model considerably more difficult to solve due to the large increase in size resulting from additional binary sequencing and assignment variables and associated constraints. We now propose an efficient heuristic method for the solution of the one robot problem that can naturally be extended to the multiple robot case.

One way to reduce the complexity of model ORM, is to fix the lot-slot assignment variables with the values from the one robot model without the sequencing constraints for the

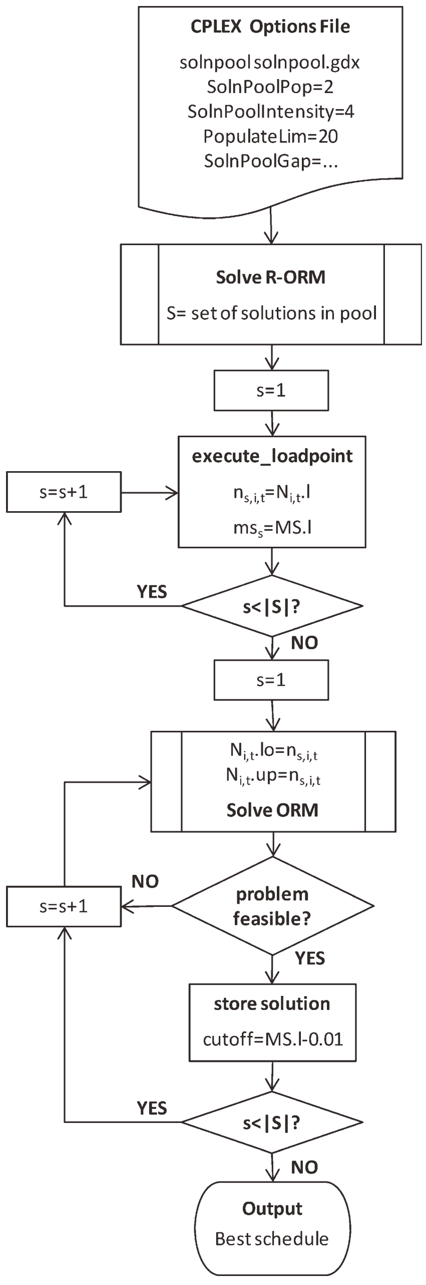

Figure 4. Two-stage heuristic algorithm for AWS.

transfers. This relaxed model was named R-ORM. This two-step heuristic method was proposed by Bhushan and Karimi[28] who reported a maximum deviation in makespan of only 0.42% when compared with the optimal solution, in problem sizes up to 14 lots in 4 baths and 10 lots in 6 baths. Following experimental observations that: (i) R-ORM is highly degenerate; (ii) solutions with the same objective function but different lot sequences in the baths will typically lead to a range of makespans when considering the one robot model; (iii) the optimal bath sequence from ORM may be suboptimal for R-ORM; we propose solving the one robot model a few times, starting from bath sequences that are very good, but not necessarily optimal.

The new heuristic method relies on four options of the CPLEX MILP solver.[31] With *Solnpoolpop*=2 we can call the populate procedure for storing multiple solutions to the problem in a solution pool. With *Solnpoolintensity*=4, the solver is instructed to enumerate all practical solutions. Since this number may be huge even for small problems, we control the number of solutions generated for the solution pool with parameter *Populatelim*. We have used the default value (20), which is reasonable and avoids consuming a large quantity of memory. Finally, with *Solnpoolgap*, the user ensures that solutions in the solution pool are within a relative tolerance of the optimal value. Note that this setting should be selected carefully since the solver will not generate the best solutions within the given tolerance. If equal to 0, only optimal solutions will be generated, which may be too few. If greater than zero, the solver may generate the optimal solution and roughly *Populatelim*-1 suboptimal solutions, despite possibly having *Populatelim* degenerate optimal solutions, which will most likely be better starting points for the one robot model. Overall, *Solnpoolgap* should be selected by trial and error and will thus be problem dependent.

The search algorithm is illustrated in Figure 4. Following the generation of the solution pool, we extract the value of the binary variables and makespan for each solution $s \in S$ of the relaxed one robot model and store it in parameters $n_{s,i,t}$ and ms_s, respectively. The binary variables for the one robot model are then fixed to the former values by specifying the lower and upper bound attributes (.lo, .up). The first solution provides the sequence that is optimal for the relaxed scenario. When adding the robot constraints, the makespan will typically increase slightly. This will be the first feasible solution. Afterwards, the focus is on generating better solutions while minimizing computational effort. To achieve this goal, a cutoff value is specified to eliminate the parts of the branch-and-bound search tree with an objective equal or worse to current best solution. This will make almost all subsequent problems infeasible. If better solutions are indeed found, one can: (i) check if the sequence was optimal for the R-ORM case by comparing the values of ms_s; (ii) quantify the gain in considering more than just (one of) the optimal sequence(s) for R-ORM.

Computational Results

The heuristic algorithm and scheduling models were implemented in GAMS 23.5 using CPLEX 12.2 as the MILP solver and taking advantage of the two parallel threads of the Intel Core2 Duo T9300 2.5 GHz laptop (Windows Vista Enterprise, 4 GB of RAM). The relative optimality gap was set to 10^{-6} and the maximum computational time to 3600 CPUs. The remaining settings were the defaults, except those indicated in Figure 4 for the two-stage algorithm.

The performance of the optimization methods is illustrated through the solution of 13 test problems built from the data given by Bhushan and Karimi.[27] These range from 18 lots in 4 baths to 15 lots in 12 baths, which is roughly what the unlimited robot model can handle in less than 1 h of computational time. The required data is given in Table 1, with the lots and units to consider being the firsts according to problem specification. Five different scenarios will be considered:

Table 1. Processing and Transfer Times for Problems P1–P13

	Unit												
Lot	1	2	3	4	5	6	7	8	9	10	11	12	13
1	4.3	6.7	11.3	6.3	2.5	6.9	8.1	7.5	4.2	7.1	3.9	6.8	–
2	5.8	6.7	8.2	6.5	4.9	6.5	12.8	6.8	10.4	6.7	11.8	6.7	–
3	10.6	6.7	2.6	6.4	2.7	7.3	13.0	6.6	11.4	6.8	9.2	6.6	–
4	2.7	6.9	6.9	7.6	3.5	7.4	3.9	6.6	7.2	6.7	3.9	6.8	–
5	4.1	6.7	11.0	6.8	7.4	6.2	3.1	6.3	3.7	6.2	9.4	6.9	–
6	3.7	6.9	2.5	6.4	6.5	6.6	2.5	6.6	2.6	6.5	2.7	6.3	–
7	10.5	6.7	3.7	6.6	11.9	6.6	2.6	6.2	6.9	6.5	3.9	6.8	–
8	3.9	6.8	6.6	6.4	3.3	6.9	3.4	6.4	11.3	6.7	5.8	7.5	–
9	2.5	7.5	1.4	7.6	6.6	6.8	11.0	6.9	12.9	6.5	5.2	7.8	–
10	10.8	6.7	10.1	6.5	2.5	6.6	2.7	7.1	4.6	6.5	11.4	6.3	–
11	8.7	6.2	4.2	7.2	6.1	6.2	5.9	6.5	4.6	6.7	8.8	6.6	–
12	7.0	6.3	7.2	6.6	2.7	6.7	8.9	7.1	2.9	6.7	6.4	6.8	–
13	9.1	6.8	2.8	6.4	5.9	6.4	5.9	6.9	10.4	6.9	8.8	6.5	–
14	2.7	6.1	11.4	6.9	7.7	6.4	5.1	6.2	4.7	6.9	10.0	6.8	–
15	2.8	6.8	6.8	6.3	4.2	6.7	8.5	6.6	5.7	6.5	4.3	6.9	–
16	5.7	6.9	2.8	7.1	4.7	6.1	3.9	6.9	4.4	6.4	2.7	6.3	–
17	2.5	7.6	6.7	6.5	2.6	6.4	3.4	7.2	2.9	6.7	7.8	6.4	–
18	3.9	6.8	12.1	6.8	2.7	6.3	9.3	6.2	4.7	6.3	2.6	6.8	–
π_m	1.2	0.6	0.8	1.0	0.4	0.6	1.0	1.0	0.8	0.4	0.8	1.0	1.2

(i) unlimited robot availability; (ii) single robot availability while assuming that transfer tasks are always possible (iii-iv) one robot with full-space and decomposition approach; (v) two robots.

Unlimited robot availability

We start by analyzing the performance of model URM that provides a lower bound for the multiple robot model MRM. As can be seen in Table 2, only P5 and P9 cannot be solved to global optimality, both with an out of memory termination before reaching the maximum computational limit defined. It clearly reflects the increasing complexity with the increase in the number of lots and units. Interestingly, the largest problem, featuring 15 lots in 12 baths, can still be solved in less than an hour.

Single robot neglecting transfer tasks

To compare the performance of models with and without big-M constraints, we consider the relaxed one robot models for the time grids (R-ORM) and the general precedence sequencing concept used by Aguirre and Méndez[30] and earlier described (R-AM). As can be seen in Table 3, R-ORM outperforms R-AM by orders of magnitude despite needing more than twice the number of binary variables. Notice that the absence of big-M constraints makes R-ORM considerably tighter (MIP-RMIP), being able to prove optimality for all but P9 (out of memory termination). In contrast, R-AM requires significantly more constraints and although able to find a better solution for P9 (197.3), failed to find the optimal solutions for P5 and P13 and to prove optimality in five other cases. The other aspect worth highlighting is that the optimal solutions for the relaxed one robot model are at least 10% worse than those for unlimited robot availability (given in Table 2). Thus, the plant can benefit from multiple robots. More on this to follow.

One robot

For the one robot scenario, we now test three different models. Besides the new ORM model, we consider the also

Table 2. Computational Performance of Model URM

	(|I|,|M|-1)	DV	SV	SE	RMIP	MIP	CPUs	Nodes
P1	(8,4)	64	121	125	73.6	83.8	0.38	389
P2	(10,4)	100	171	157	86.8	101	1.60	1331
P3	(12,4)	144	229	189	99.3	115.5	9.39	16739
P4	(15,4)	225	331	237	119.0	140.8	178	300377
P5	(18,4)	324	451	285	140.0	166.6	1391†	1628007
P6	(8,8)	64	169	233	102.2	118.2	0.61	521
P7	(10,8)	100	231	293	115.4	134.5	7.02	7597
P8	(12,8)	144	301	353	128.1	150	73.5	89587
P9	(15,8)	225	421	443	147.8	174.5	1357‡	893267
P10	(8,12)	64	217	341	133.6	156.5	0.94	498
P11	(10,12)	100	291	429	148.7	175.1	8.24	4540
P12	(12,12)	144	373	517	160.7	190.6	58.2	38265
P13	(15,12)	225	511	649	182.2	216.2	3375	2044556

DV = discrete variables. SV = single variables. SE = single equations. RMIP = solution of the relaxed linear problem. MIP = problem solution.
†Out of memory termination (OM), best possible solution (BPS) = 163.6.
‡OM, BPS = 167.3.

Table 3. Computational Performance for Relaxed One Robot Models

	R-ORM						R-AM					
	DV	SV	SE	RMIP	MIP	CPUs	CPUs	RMIP	MIP	DV	SV	SE
P1	64	121	125	85.4	95.1	0.39	2.40	54.3	95.1	28	102	296
P2	100	171	157	100.5	115.5	1.30	81	61.7	115.5	45	136	450
P3	144	229	189	116.5	133.7	7.72	3600*	69.2	133.7	66	175	636
P4	225	331	237	141.4	163.1	65.4	3600[†]	79.4	163.1	105	241	975
P5	324	451	285	168.1	194.2	2524	3600[‡]	90.11	*195.6*	153	316	1386
P6	64	169	233	113.9	130	0.63	3.34	84.06	130	28	165	584
P7	100	231	293	130.2	149.4	5.67	148	92.03	149.4	45	216	890
P8	144	301	353	146.8	169.1	62.8	3600[§]	99.77	169.1	66	271	1260
P9	225	421	443	172.1	*197.5*	1501[¶]	3600**	110.7	197.3	105	361	1935
P10	64	217	341	146.1	170.3	2.44	6.24	116.5	170.3	28	229	872
P11	100	291	429	164.1	192.2	12.7	259	125.6	192.2	45	296	1330
P12	144	373	517	180.4	210.7	38.1	3600[††]	133.2	210.7	66	367	1884
P13	225	511	649	207.0	241.4	2959	3600[‡‡]	145.2	*242.4*	105	481	2895

Suboptimal solutions in italic.
*BPS = 110.5; [†]BPS = 102.3; [‡]BPS = 108.4; [§]BPS = 141.3; [¶]OM, BPS = 190.4; **BPS = 134.4; [††]BPS = 177.4; [‡‡]BPS = 172.3.

Table 4. Solution and Computational Times for One Robot Models

	Solution			CPUs			RMIP		Best possible solution		
Model	ORM	BK	O-AM	ORM	BK	O-AM	ORM/BK	O-AM	ORM	BK	O-AM
P1	95.6	95.6	95.6	0.82	1.31	55.3	85.4	54.3	–	–	–
P2	115.6	115.6	115.6	4.95	3.17	3600	100.5	61.7	–	–	102.6
P3	134.1	134.1	135	46.1	13.3	3600	116.5	69.17	–	–	87.0
P4	163.6	163.6	167.2	1682	104	3600	141.4	79.42	–	–	94.1
P5	194.7	195.7	202.8	3600	706*	3600	168.1	90.11	188.5	191.9	102.4
P6	131.6	132.6	131.6	28.7	3600	350	113.9	84.06	–	127.0	–
P7	153.5	–	155.8	3600	3600	3600	130.2	92.03	144.0	142.6	115.6
P8	172.5	193.6	182.2	3600	3600	3600	146.8	99.77	156.9	152.5	118.8
P9	214.6	–	218.7	3600	3600	3600	172.1	110.7	184.2	176.8	124.4
P10	170.6	–	170.6	92.7	3600	2298	146.1	116.5	–	170.3	–
P11	199.1	–	206.6	3600	3600	3600	164.1	125.62	179.1	178.9	147.2
P12	–	–	–	3600	3600	3600	180.4	133.19	194.5	190.7	150.6
P13	782.1	–	–	3600	3600	3600	207.0	145.15	179.1	216.0	159.6

*Out of memory termination.

hybrid time slots/sequencing model by Bhushan and Karimi[28] (BK) and the simplification of the multiple robot model by Aguirre and Méndez[30] (O-AM). The implementation of BK features constraints 10a, 10b, 11, and 12 of their work[28] (the implementation of performance enhancing constraint 13 turned the model infeasible, and we were unable to debug it). The main results are given in Table 4, while information about the problem sizes is left for Table 5.

While for the relaxed one robot scenario there was a single problem that could not be solved to optimality, now that is mostly true, clearly reflecting the increase in complexity of bringing the robot sequencing constraints into the model.

Table 5. Model Statistics One Robot Models

	Discrete Variables			Single Variables			Single Equations		
Model	ORM	BK	O-AM	ORM	BK	O-AM	ORM	BK	O-AM
P1	141	167	588	198	240	661	762	357	1416
P2	201	235	945	272	326	1036	1158	461	2250
P3	269	311	1386	354	420	1495	1634	565	3276
P4	386	440	2205	492	576	2341	2498	721	5175
P5	521	587	3213	648	750	3376	3542	877	7506
P6	714	666	2044	819	803	2181	2899	1601	4616
P7	1004	926	3285	1135	1097	3456	4437	2157	7370
P8	1302	1194	4818	1459	1399	5023	6263	2713	10764
P9	1764	1611	7665	1960	1867	7921	9542	3547	17055
P10	1968	1618	4396	2121	1819	4597	6613	3863	9608
P11	2906	2335	7065	3097	2586	7316	10255	5433	15370
P12	3881	3081	10362	4110	3382	10663	14550	7045	22476
P13	5360	4218	16485	5646	4594	16861	22164	9469	35655

Table 6. Results for the Two-Stage Heuristic Approach

Model	SolnPoolGap	# Solutions	Best	DV	Makespan	TCPUs	R-ORM Optimal?	Gain (%)	vs. ORM (%)
P1	0.008	12	1	77	95.6	2.87	Yes	0	0
P2	0	18	1	101	116.0	6.27	Yes	0	−0.3
P3	0	26	3	125	134.2	25.9	Yes	0	−0.1
P4	0	23	1	161	164.7	125	Yes	0	−0.7
P5	0	23	4	197	194.8	3269	Yes	0	−0.1
P6	0.005	10	4	650	131.6	9.2	No	2.1	0
P7	0.005	17	3	904	152.6	27.7	No	2.6	0.6
P8	0.004	17	7	1158	173.7	216	No	1.7	−0.7
P9	0	1	1	1539	205.4	1482*	Yes	0.0	4.3
P10	0.008	14	3	1904	170.7	12.7	No	0.2	−0.1
P11	0.005	16	13	2806	195.7	152	No	0.8	1.7
P12	0.005	18	11	3737	215.6	4318	No	1.3	∞
P13	0	1	–	5135	–	7101	Yes	–	–

*URM problem stopped at 1450 CPUs.

In fact, no solution could be found for P12 and for P13 the solution is very poor.

Overall, the new proposed model is again the best overall performer in terms of computational time and quality of the solution returned. ORM successfully proves optimality under the hour mark for six problems (P1-P4, P6, and P10), and it always provides the best solution. In addition, the integrality gap at termination (last columns in Table 4) is lower except for P5 and P13 for which BK did a better job. The conceptually similar model of Bhushan and Karimi[28] can be considered the second best overall performer, being indeed the best for P2-P4. This tells us that BK handles better an increase in the number of lots, provided that the number of baths remains low (recall the problem features in Table 2). Already for 8 baths, the use of general precedence variables becomes a better approach, even without the use of the lot-slot assignment variables. Notice that despite the significantly worse relaxation (RMIP column), O-AM surpasses BK in performance, and it can still find good solutions for P7-P9 and P11, while being able to prove optimality for P6 and P10.

From the model statistics in Table 5, one can see that BK has the advantage of leading to smaller problem sizes. Nevertheless, the number of binary variables resulting from ORM and BK is roughly the same, which in turn are much lower than the number from O-AM. This is a direct consequence of the significant domain reduction that can be performed when relating different pairs of (slot,unit) instead of (lot,unit) pairs, as in AM. With the latter model, a similar approach can only be applied once the sequence of lots in the baths is known. The real advantage of restricting the robot sequencing variables to pairs of units with $m<m'$ in BK[28] (recall Eq. 45) instead of $m \neq m'$ (Eq. 29), is reflected more on the number of constraints, less than half those from ORM, than on the number of binaries. These must however be more complex overall, since BK handles worse an increase in problem size, having trouble even to find feasible solutions, as already discussed.

One robot with two-stage heuristic approach

After more or less identifying the size of problems that can be addressed by full-space models, we switch our attention to the heuristic approach that enables to widen the range of tractable problems. As described earlier, the method first solves the relaxed one robot model to find a few different sequences, which are then fixed before solving the sequence constrained one robot model several times. The drawback is that the optimal solution may be lost in the process. Indeed, this happens in 6 cases (P2-P5, P8, and P10, see Table 6) but the increase in makespan is lower than 0.7%. These were however not our primary target since only P5 and P8 were not solved to optimality by ORM.

The main accomplishment was for P12 (12 lots in 12 baths), for which none of the three full-space methods could find a feasible solution in 1 h. Within a couple of minutes, the algorithm could generate 18 different sequences within a 0.5 % optimality gap with respect to the relaxed one robot problem. Of these, sequence #11 led to the best schedule for the one robot model (shown in Figure 5), featuring a makespan equal to 215.6. The 4.3% improvement for P9 was also very significant, considering that R-ORM was stopped at 1450 CPUs before running out of memory and that, because of this, there was a single iteration. On the downside, we still could not find a good solution for P13.

From Table 6, one can see that CPLEX parameter *Populatelim* is just a rough estimate on the number of solutions generated for the solution pool, since there were cases (P3-P5) with more solutions than the specified value (20). It is

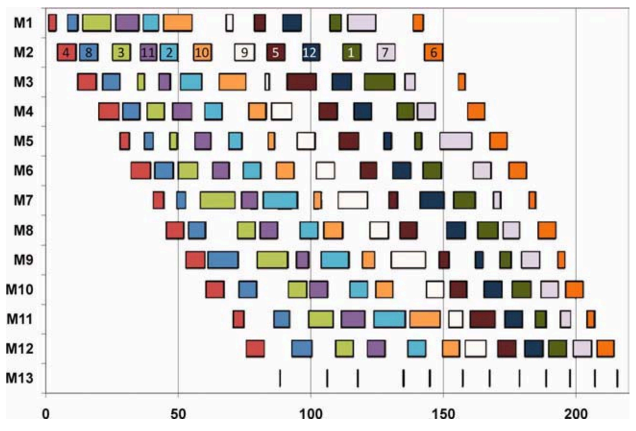

Figure 5. Best found solution for P12 considering a single robot.

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

Table 7. Results for the Multiple Robot Models Considering Two Robots

	MRM						AM					
	DV	SV	SE	RMIP	MIP	CPUs	CPUs	RMIP	MIP	DV	SV	SE
P1	704	761	2461	73.6	83.8	9.90	52.5	52.9	83.8	668	741	3024
P2	1100	1171	3879	86.8	101	32.1	3600*	59.9	101	1045	1136	4820
P3	1584	1669	5617	99.3	115.5	163	3600†	67.0	*116.7*	1506	1615	7032
P4	2475	2581	8824	119.0	140.8	3600‡	3600§	76.6	*145.5*	2355	2491	11130
P5	3564	3691	12751	140.0	169	3600¶	3600**	86.7	*177.4*	3393	3556	16164
P6	2224	2329	8481	102.2	118.2	107	321	81.3	118.2	2188	2325	9616
P7	3520	3651	13487	115.4	134.5	1320	3600††	88.4	*138.2*	3465	3636	15380
P8	5112	5269	19645	128.1	150.9	3600‡‡	3600§§	95.4	*158*	5034	5239	22488
P9	8055	8251	31042	147.8	181.9	3600¶¶	3600***	105.1	*198.2*	7935	8191	35670
P10	4640	4793	18085	133.6	156.5	185	3600†††	112.3	*156.6*	4604	4805	19792
P11	7380	7571	28855	148.7	175.6	3600‡‡‡	3600§§§	120.2	*183.9*	7325	7576	31700
P12	10752	10981	42121	160.7	199.1	3600¶¶¶	3600****	126.6	*204.5*	10674	10975	46392
P13	16995	17281	66700	182.2	465.3	3600††††	3600‡‡‡‡	136.8	–	16875	17251	73650

Suboptimal solutions in italic.
*Best possible solutions: 92.5;†83.6; ‡138.7; §88.7.¶154.2; **97.3; ††112.4; ‡‡145.3; §§109.7; ¶¶158.9; ***116.4; †††156.5; ‡‡‡172.8; §§§139.3.¶¶¶172.1; ****139.2; ††††190.9; ‡‡‡‡146.3.

also interesting to observe that in these problems, and also in P2, all sequences were optimal for the relaxed one robot case. In others, we had to relax the optimality gap in order to generate a reasonable number of sequences. More importantly, in P6-P8 and P10-P12, the sequences leading to the best schedule were not optimal for the R-ORM problem and we achieved gains up to 2.6% when considering more than just one optimal sequence, which was the approach followed by Bhushan and Karimi.[28] Furthermore, the additional sequences bring a small computational cost, since after the first iteration, the cutoff values make the constrained ORM problems easier to solve. Finally, while the number of variables and constraints remains the same when compared with the unconstrained ORM problem, there is only a small reduction in the number of binary variables, once more confirming that the AWS scheduling complexity lies within the robot.

Two robots

Increasing the number of robots in the system allows us to lower the makespan. Interestingly, in all problems that could be solved to optimality, the solutions from the multiple robot model MRM are equal to those of the unlimited robot model URM (compare Table 7 with Table 2). This tells us that two robots are enough to achieve maximum productivity and thus there is no point in increasing the computational studies for a higher number of robots. Again the new proposed hybrid formulation is better than the pure sequencing variables model AM, which is able to match the solution only for P1, P2, and P6.

Notice the significant increase in model size, particularly in the number of constraints, which reaches 66,700 for MRM instead of 22,164 for the one robot model ORM (compare Table 7 with Table 5). Interestingly, the same number of problems can be solved to optimality by the proposed formulation: 6, with MRM succeeding in P7 and failing in P4, an opposite behavior to that of ORM. Despite leading to higher computational times in the problems that can also be solved to optimality by ORM, the multiple robot model seems to respond better to an increase in problem size, since it can still find a good solution for P12 (199.1, 4.4% above the solution of URM, check Table 2), whereas no feasible solution was returned by ORM.

Conclusions

This article has addressed the scheduling of AWSs using multiple robots for transporting wafer lots between baths. A new hybrid formulation has been proposed that relies on the concept of time slots to implicitly determine lot sequencing in the chemical and water baths, and on general precedence sequencing variables for the transportation tasks. Model performance has been evaluated through the solution of 13 test problems taken from the literature and compared with a closely related model and to another relying solely on sequencing variables. The results have shown that the new model was able to solve more problems to optimality and did always find the best solution of the group in 1 h of computational time.

This article has also shown that both relaxed and constrained versions of the proposed model can be employed as the basis of an efficient heuristic procedure for the single robot problem. The relaxed version neglects robot constraints and has been used to generate a few very good sequences concerning lot processing in the baths. Once the lot-bath binary variables have been fixed, the problems become simpler and near optimal solutions can be generated. The solution improved for the majority of the problems that could not be solved to optimality by the full-space model, but, most importantly, a very good feasible solution was found in a case where all three full-space models failed. Unfortunately, and due to the rapidly growing number of sequencing variables, no solution was found for the 15 lots in 12 baths problem, so other approaches should be tried beyond this size.

Acknowledgments

The authors acknowledge financial support from Fundação para a Ciência e Tecnologia and Ministerio de Ciencia, Tecnología e Innovación Productiva, under the Scientific Bilateral Cooperation Agreement between Argentina and Portugal (2010–2011) and from AECID under Grant PCI-D/024726/09.

Literature Cited

1. Méndez CA, Cerdá J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput Chem Eng*. 2006;30:913–946.
2. Pinto JM, Grossmann IE. A logic-based approach to scheduling problems with resource constraints. *Comput Chem Eng*. 1997;21: 801–818.
3. Méndez CA, Cerdá J. *An MILP framework for short-term scheduling of single-stage batch plants with limited discrete resources*. In: Grievink J, van Schijndel J, editors. *Computer-aided Chemical Engineering*. 2002; Vol. 10, Elsevier, 721.
4. Janak SL, Lin X, Floudas CA. 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.
5. Maravelias CT, Grossmann IE. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2003;42:3056–3074.
6. Castro PM, Barbosa-Póvoa AP, Matos HA, Novais AQ. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Ind Eng Chem Res*. 2004;43:105–118.
7. Castro PM, Harjunkoski I, Grossmann IE. New continuous-time scheduling formulation for continuous plants under variable electricity cost. *Ind Eng Chem Res*. 2009;48:6701–6714.
8. Gupta S, Karimi IA. An improved MILP formulation for scheduling multiproduct, multistage batch plants. *Ind Eng Chem Res*. 2003;42: 2365–2380.
9. Méndez CA, Henning GP, Cerdá J. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Comput Chem Eng*. 2001;25:701–711.
10. Harjunkoski I, Grossmann I. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comput Chem Eng*. 2002;26:1533–1552.
11. Castro PM, Grossmann IE, Novais AQ. Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers. *Ind Eng Chem Res*. 2006;45:6210–6226.
12. Prasad P, Maravelias CT. Batch selection, assignment and sequencing in multi-stage, multi-product processes. *Comput Chem Eng*. 2008;32:1114–1127.
13. Sundaramoorthy A, Maravelias CT. Simultaneous batching and scheduling in multistage multiproduct processes. *Ind Eng Chem Res*. 2008;47:1546–1555.
14. Castro PM, Erdirik-Dogan M, Grossmann IE. Simultaneous batching and scheduling of single stage batch plants with parallel units. *AIChE J*. 2008;54:183–193.
15. Castro PM, Grossmann IE. New continuous-time MILP model for the short-term scheduling of multistage batch plants. *Ind Eng Chem Res*. 2005;44:9175–9190.
16. Castro PM, Novais AQ. Short-term scheduling of multistage batch plants with unlimited intermediate storage. *Ind Eng Chem Res*. 2008;47:6126–6139.
17. Castro PM, Novais AQ. Scheduling multistage batch plants with sequence-dependent changeovers. *AIChE J*. 2009;55:2122–2137.
18. Shaik MA, Floudas CA. Novel unified modeling approach for short-term scheduling. *Ind Eng Chem Res*. 2009;48:2947–2964.
19. Shaik MA, Floudas CA. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. *Comput Chem Eng*. 2008;32:260–274.
20. Li J, Floudas CA. Optimal event point determination for short-term scheduling of multipurpose batch plants via unit-specific event-based continuous-time approaches. *Ind Eng Chem Res*. 2010;49:7446–7469.
21. Zeballos LJ, Castro PM, Méndez CA. Integrated constraint programming scheduling approach for automated wet-etch stations in semiconductor manufacturing. *Ind Eng Chem Res*. 2011;50:1705–1715.
22. Subbiah S, Tometzki T, Panek S, Engell S. Multi-product batch scheduling with intermediate due dates using priced timed automata models. *Comput Chem Eng*. 2009;33:1661–1676.
23. He Y, Hui CW. A novel search framework for multi-stage process scheduling with tight due dates. *AIChE J*. 2010;56:2103–2121.
24. Sundaramoorthy A, Maravelias CT, Prasad P. Scheduling of multistage batch processes under utility constraints. *Ind Eng Chem Res*. 2009;48:6050–6058.
25. Uzsoy R, Lee CY, Marti-Vega LA. A review of production planning and scheduling models in the semiconductor industry. Part 1: System characteristics, performance evaluation and production planning. *IIE Trans*. 1992;24:47–60.
26. Geiger DC, Kempf KG, Uzsoy R. A tabu search approach to scheduling and automated wet etch station. *J Manuf Syst*. 1997; 16:102–116.
27. Bhushan S, Karimi IA. Heuristic algorithms for scheduling an automated wet-etch station. *Comput Chem Eng*. 2004;28:363–379.
28. Bhushan S, Karimi IA. An MILP approach to automated wet-etch station scheduling. *Ind Eng Chem. Res*. 2003;42:1391–1399.
29. Sundaramoorthy A, Maravelias CT. A general framework for process scheduling. *AIChE J*. 2011;57:695–710.
30. Aguirre AM, Méndez CA. *A novel optimization method to automated wet-etching station scheduling in semiconductor manufacturing systems*. In: Pierucci S, Buzzi Ferraris G, editors. *Computer Aided Chemical Engineering*. 2010; Vol. 28. Elsevier, 883–888.
31. *CPLEX 12*. In *GAMS—The Solver Manuals*. GAMS Development Corporation, Washington DC, 2010.

Manuscript received Oct. 20, 2010, and revision received Feb. 23, 2011.