Integrated Constraint Programming Scheduling Approach for Automated Wet-Etch Stations in Semiconductor Manufacturing

Luis J. Zeballos,*,†,‡ Pedro M. Castro,‡ and Carlos A. Méndez§

Facultad de Ingeniería Química, Universidad Nacional del Litoral, Santa Fe, Argentina, Unidade Modelação e Optimização de Sistemas Energéticos, Laboratório Nacional de Energia e Geologia, Lisboa, Portugal, and Universidad Nacional del Litoral—CONICET (INTEC), Santa Fe, Argentina

This article addresses a challenging resource constrained flow shop scheduling problem from automated wet-etch stations (AWSs) in wafer fabrication of semiconductor manufacturing facilities by means of a constraint programming (CP) methodology. Wet etching in wafer fabrication is a difficult process considering the material handling limitations and mixed intermediate policies. The proposed integrated approach consists of both a CP model and an efficient search strategy in order to handle the different features of the process. The domain-specific search strategy significantly improves the computational performance, a key aspect given the high combinatorial complexity of the problem. The search strategy objective is to avoid losing time due to early bad choices in the exploration and to produce good quality solutions with low computational effort. The applicability of the proposed integrated CP methodology is successfully tested with several examples taken from the literature, featuring a different number of jobs and baths.

1. Introduction

The scheduling problem is an important issue in process operations in order to improve production performance. Scheduling approaches can be classified according to the type of network structure they can handle, in terms of the function of equipment units. The classification can range from the simplest single machine problem, where a feasible or optimal order sequence must be obtained, to more complex multipurpose systems, where a given unit may be used by tasks belonging to different production stages. In between, there are the multistage, multiproduct plants, with products flowing through the stages. Furthermore, one may encounter several intermediate storage and transfer policies between consecutive stages.

Due to the industrial importance of multistage, multiproduct environments and the fact that their scheduling problems are challenging, significant research effort involving optimization approaches has been given to this problem over the past decades. Several excellent reviews can be found from Pekny and Reklaitis,[1] Grieco et al.,[2] Kallrath,[3] Floudas and Lin,[4] and Méndez et al.[5] Despite significant advances there are still several challenges that remain unaddressed. For example, there are many issues related to the specific capabilities of the methods for handling a large number of operational characteristics (e.g., storage and transfer policies), different types of constrained resources, and different objectives (e.g., makespan, earliness, or cost minimization). Finally, there are also questions on the limitations and strengths of the various optimization techniques used in the literature and the problem sizes that can practically be addressed.

An analysis of the reviews reveals that scheduling problems of multistage, multiproduct plants have been tackled by a variety of optimization approaches as well as other solution methods. Nevertheless, most of the approaches reported in the literature rely on mathematical programming (MP) approaches, which usually lead to mixed integer linear programming (MILP) models. Due to the inherent complexity of the problems, there are other solution methods, such as dispatching rules, metaheuristics, and artificial intelligence based methodologies. Moreover, hybrid approaches have also been applied. Constraint Programming[6,7] is a method based on artificial intelligence techniques which has been effectively used by the process system engineering community to deal with different classes of scheduling problems.[8−11]

Constraint programming (CP) is a paradigm for modeling and solving optimization problems that consists of two phases. The first one refers to the problem representation itself, using integer and Boolean variables with the possibility of variables being indexed by other variables. Furthermore, constraints can be linear, convex, and logic. The second phase uses tree search procedures (which enumerate assignments of values to variables) combined with domain reduction and constraint propagation algorithms to solve problems. In addition, it is worth remarking that CP is a rigorous approach that given enough computational time will give the optimal solution to optimization problems.

In the process system engineering community, CP has not quite been evaluated considering the two phases. In most cases, while formulations have been proposed for problem representation, only default search procedures, which do not exploit domain-specific knowledge, have been considered in the second phase. The exceptions in terms of the development of effective and efficient search strategies, which are a fundamental support for solving hard combinatorial optimization problems,[7] are the papers by Zeballos and Henning[12] and Zeballos and Méndez.[13]

CP has also been integrated with MILP approaches in order to take advantage of their complementary strengths.[8,14] Such hybrid methods decompose the original problem into two subproblems, one MILP and one CP. Models are separately solved, and information obtained after solving the subproblems is interchanged between them. Hybrid approaches have been shown to be better than stand-alone MILP and CP models,[8,9,15] particularly in single stage problems with parallel units.[16]

A few papers have shown that the full advantage of the CP paradigm is obtained when addressing highly constrained optimization problems considering the makespan as an objective function.[11,13,14,17] The good performance of CP for makespan

* To whom correspondence should be addressed. E-mail: luis.zeballos@lneg.pt.
† Universidad Nacional del Litoral.
‡ Laboratório Nacional de Energia e Geologia.
§ INTEC.

10.1021/ie1016199
Published on Web 12/28/2010

minimization is based on the optimization process that is performed, which is continuously tightening the bound on the objective function value. When the objective function is updated, the time horizon is stretch contracted, leading to a direct domain reduction on the model variables. Thus, the reason for the good performance is that the mentioned bounding procedure acts as an additional domain reduction process, which is executed together with the domain reduction and constraint propagation algorithms used by CP solvers to accelerate the search.

Much of the work in the field of production scheduling has been focused on traditional multistage, multiproduct plants,[17−20] which typically assumes unlimited intermediate storage and transfer policies. Nevertheless, some contributions have included this type of considerations,[21−23] which are critical in the challenging optimization problem arising from semiconductor manufacturing facilities addressed in this paper.

Semiconductor manufacturing is one of the fastest growing industries in the world because of the diverse market focusing on integrated circuits for networking, storage components, telecommunications/wireless, consumer, computer, and storage systems that have become necessary tools of today.[24] According to Hung and Wang,[25] the overall manufacturing process can be separated into six serial processes, which normally are performed separately in different fabrication areas: (1) material fabrication, (2) wafer fabrication, (3) wafer probe, (4) assembly, (5) raw testing, and (6) final testing. Uzsoy et al.[26,27] provide a comprehensive description of the semiconductor manufacturing process.

Typical operations in semiconductor manufacturing are cleaning, diffusion, oxidation, etching, ion implantation, electrical-beam writing in wafer fabrication, baking in wafer probing, and burn-in operation in testing. While semiconductor manufacturing involves several operations, most contributions address only diffusion in wafer fabrication and burn-in operation in device testing.[24] Thus, a few papers are directly related to the scheduling of electrical-beam writing, etching, and baking in wafer probing. One good classification and meta analysis on scheduling in semiconductor manufacturing can be found in Mathirajan and Sivakumar.[24]

In wafer fabrication, lots of wafers of raw silicon or gallium arsenide are processed to arrange the paths of metal and wafer material to produce the required circuits. This operation involves many batch/semicontinuous physical and chemical steps with the key process being wet etching. The scheduling of automatic wet-etch stations (AWSs) is considered as a highly complex problem,[19] mainly because of the process features and the critical resources (the material handling limitations and mixed intermediate policies). The problem subject of this paper can be described as the scheduling of serial flow shop multiproduct stations with ZW/NIS (zero wait/no intermediate storage) and LS/NIS (local storage/no intermediate storage) policies as well as a shared material handling system with finite carrying capacity (one lot at a time).[28]

Regarding schedule performance measures, various objective functions can be considered. Nevertheless, in industrial practice, AWSs of wafer fabrication use makespan minimization. The selection is based on management's desire to improve throughput of the AWSs because they can become a constraint on factory output.[29] Furthermore, the selection of this objective is supported by considering that (a) managers wish to minimize the number of AWSs in the plant because they take up a great deal of clean room floor space[29] and (b) decreasing the total time required for the production of all wafers leads to lower inventory and contamination and results in higher profits.[28]

The inherent complexity of the scheduling problem of AWSs has been addressed by several authors, mainly coming from the process system engineering community.[28−31] Proposals have presented exact (e.g., mathematical approaches), nonexact, and nonoptimal algorithms (e.g., heuristic algorithm based on tabu search) with different types of assumptions in order to make the problem tractable (e.g., the material handling system as a noncritical resource and a single robot for moving the wafers). It is important to remark that the trend for this type of scheduling is toward the development of mechanisms that yield good solutions, since practical problems cannot be solved to optimality due to resource limitations.

Geiger et al.[29] presented a heuristic algorithm based on tabu search (TS) for the scheduling problem considering a single robot and makespan minimization. These authors first obtained a sequence of lots on baths and then introduced the ordered set of lots in an algorithm for scheduling transfers and processing. Thus, they presented a nonexact algorithm where transfer times and processing times are separately addressed.

Bhushan and Karimi[28] developed a mixed integer linear programming (MILP) approach for minimizing the total time required to process a given set of wafers. Several reformulations and constraints were numerically evaluated to identify the best formulation when considering the scheduling problem with only one robot. In addition, since MILP solutions become prohibitive for moderately sized problems, a nonoptimal two-step strategy based on such a mathematical formulation was developed. The two-step procedure uses the MILP model without the robot constraints (i.e., unlimited number of robots is considered) for determining the job sequence in baths and then imposes the single-robot restrictions on that sequence to obtain a feasible schedule. Later, Bhushan and Karimi[30] introduced other heuristic procedures in order to address large instances of the scheduling problem of AWSs with a single robot. The complete procedures are based on several techniques, such as simulated annealing, TS, and heuristic algorithms for determining initial sequencing and timing of jobs. The authors concluded that the TS sequencing procedure combined with two specific job-scheduling algorithms exhibited a better performance than the one reported by Geiger et al.[29]

More recently, Aguirre and Méndez[31] developed a MILP formulation that, in contrast to previous approaches, considers more than one robot as part of the material handling system. Similarly to Bhushan and Karimi,[19] these authors proposed a two-step strategy based on their MILP model to solve the whole problem in a sequential manner. The two-step strategy makes moderately sized case studies tractable.

Bixby et al.[15] used MIP and CP formulations as main components of a special purpose decomposition algorithm, called STARTS for space−time allocation for real-time scheduling. This method iterates alternately over the space and time dimensions. The algorithm uses separated, dedicated operational models for each fabrication process area in order to make the overall problem tractable. Although the iterative algorithm is a general approach, only considerations about the diffusion operation were given in the work. Model implementations for the different fabrication areas were not reported.

The goal of this article is to introduce an effective and efficient constraint programming methodology for the scheduling problem of AWSs in an integrated way, which generates a detailed schedule of production activities and transfer operations that complies with stringent intermediate storage policies. Since transport-related issues are tackled, it is assumed that material handling devices are critical resources. In addition, input and

output buffers are not taken into account due to the fact that their capacities are not regarded as limiting. It is important to note that, since the addressed problem is highly constrained and makespan minimization is the objective, we have favorable conditions to use a constraint programming technique. In this respect, the proposed formulation takes advantage of the simplicity and the declarative power of the paradigm and is oriented to short-term scheduling. Industrial-scale problems consisting of hundreds of lots, dozens of stages, and long scheduling horizons can be addressed using concepts introduced for MILP formulations.[32−34]

The rest of the paper is structured as follows. Section 2 describes the scheduling problem under consideration. Section 3 introduces the CP model for the AWS scheduling problem at hand. One domain-specific search strategy is presented in section 4. Results are presented in section 5. Finally, conclusions are pointed out in section 6.

2. Problem Description

Semiconductor manufacturing involves many batch/semicontinuous physical and chemical operations. The main stage in semiconductor manufacturing facilities is wafer fabrication, which is a procedure composed of several repeated sequential steps to produce complete electric circuits. At the beginning of the production process, the wafer is covered with a thin uniform layer of SiO_2. Next, selected portions of the wafer are marked in order to form a circuit configuration. This step is called photolithography or photomasking. The etching process is used immediately after the photolithography to etch the unwanted material from the wafer. Since this operation is not selective, the circuit configuration must be traced over the wafer. It is important to note that etching is a key step in the wafer fabrication process that involves transfers of expensive wafer lots between baths and severe constraints on bath times. In semiconductor manufacturing facilities, the wet-etching process is carried out by one or more highly automated stations (AWSs). Thus, these units are responsible for removing the unnecessary film of SiO_2 formed on the surface of the wafer, in a series of chemical and deionizing baths.

The AWS deals with group of wafers of the same type, hereafter named wafer lots. Lots in the input buffer come from the upstream process. After processing in the last bath, lots enter the output buffer. In this type of technology, wafer lots are moved across a chemical or water bath to another by means of automated material handling devices, for example, robots. One or more transportation units move lots from bath to bath for processing. Thus, baths are linked by a transportation system that picks up lots in order to execute the material movements. The resulting problem is the scheduling of serial flow shop multiproduct stations with ZW/NIS and LS/NIS policies as well as the restricted material handling system. Figure 1 shows a schematic representation of a typical AWS.

The scheduling problem for AWSs to be tackled in this work has the following features:

(a) A set of N lots of wafers ($i = 1, 2, ..., N$), hereafter named jobs, must be manufactured following predefined recipes that show the sequence of baths to be visited. It is assumed that wafer lots follow the same sequence of baths across the system, considering the input and output buffers ($j = 0, 1, ..., M + 1$, where M is the total number of baths). Since baths alternate in the AWS, odd baths have chemicals ($Jc = \{1, 3, 5, ..., M - 1\}$) and even baths have water ($Jw = \{2, 4, 6, ..., M\}$).

(b) While chemical baths are followed by a zero wait storage policy (ZW), water baths can be used as local storage.

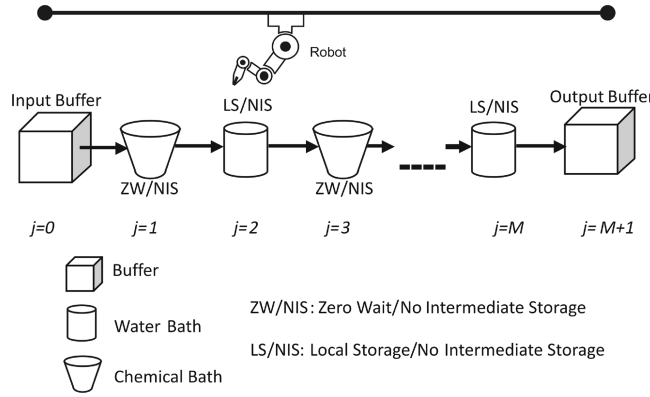

Figure 1. Schematic representation of a typical AWS.

(c) The processing time for a given lot depends on the lot and the bath (t_{ij}).

(d) A bath can process one lot at a time.

(e) Chemical and water baths are linked by robots which transport a wafer's lots from a bath to another. Since the transport system is composed of a limited number of robots, it becomes a critical resource.

(f) Robots cannot move multiple lots at a time and cannot be used as temporary buffer.

(g) Robots are free of collisions since they have different paths to execute their movements.

(h) Transfer times between consecutive baths for lots depend on a single route involving the origin and destination. The time for transferring lots between successive baths $j - 1$ and j is represented by π_j.

(i) Baths and robots do not need any setups.

(j) There are no breakdowns associated with baths or robots.

Given all the above features, the scheduling problem consists of determining (i) the wafer's lot sequence at each bath, (ii) the allocation and sequence of transport activities to be carried out by each robot, and (iii) the timings of the processing and transport activities. The goal of this problem is makespan minimization.

3. Constraint Programming Model

In this section, the CP model for the scheduling problem of an AWS is presented. The approach has been implemented in the ILOG OPL constraint programming language,[7] embedded in the OPL Studio 3.7 Package.[35] It resorts to some specific scheduling constructs, which are available in the ILOG Scheduler[36] software package.

3.1. Nomenclature.

Subscripts

i = lot
j = bath or buffer
r = robot

Sets

I = set of lots or jobs
J = set of baths and buffers
Jw = set of water baths
Jc = set of chemical baths
R = set of alternative robots

Parameters

t_{ij} = residence time of job i in bath j
π_j = transfer time between $j - 1$ and j
TPT_i = total processing time for lot i

Special Parameters and Variables. The model handles the following parameters connected with resources:

$bath_j$ = models bath j (chemical or water) where the lots can reside during a given time. Baths are defined as unary resources, which are a special type of renewable resources of capacity 1. Thus, unary resources can execute just one operation on a part at any time and become reusable once the operation being done is finished.

$robot_r$ = represents the robot r. This device is another unary resource that can transport only one wafer's lot at a time.

Model Variables. In this work two types of activities are employed: *PrTask* and *TrTask*. Each activity is described by means of duration, start time, and end time variables (i.e., *Task.duration*, *Task.start*, and *Task.end*) that verify the condition *Task.start* + *Task.duration* = *Task.end*. While the first type represents the processing of lots in baths, the second one represents movements executed by the robot transporting lots between consecutive baths.

$PrTask_{ij}$ = corresponds to the processing of lot i in bath j

$TrTask_{ijj+1}$ = models robot movements transporting lot i from bath j to $j + 1$

Mk = makespan, corresponds to the total time required to complete all the jobs in the wet-etch station

3.2. Model Constraints. The CP approach employs some specific scheduling constructs available in the modeling language ILOG OPL Studio.[35] One of them is *precedes*, which imposes a proper sequence of nonoverlapping activities. Another construct is *requires*, which prescribes the assignment of renewable resources demanded by activities. Finally, one of the most important constructs is *activityHasSelectedResource*. It acts like a predicate that assumes a value equal to one when an activity has been assigned to a specific resource belonging to a given set of alternative resources.

Baths Assignment Constraints.

$$PrTask_{ij} \text{ requires } bath_j \quad \forall i \in I, \quad \forall j \in J \tag{1}$$

$$PrTask_{ij} \text{ precedes } PrTask_{ij+1} \quad \forall i \in I, \quad \forall j \in J,\ j \neq last(J) \tag{2}$$

Constraint 1 prescribes that lot i must be assigned to bath j belonging to the set J. Since baths have been declared as unary resources, all the activities that are assigned to them will be automatically sequenced without requiring additional constraints. Constraint 2 enforces a proper sequencing of tasks corresponding to any pair of consecutive processing operations at baths j and $j + 1$ to be executed on lot i by resorting to the special construct *precedes*. Therefore, the activity located at the right-hand side cannot be started until the activity on the left-hand side is finished.

Baths Timing Constraints.

$$PrTask_{ij}.\text{duration} = t_{ij} \quad \forall i \in I, \quad \forall j \in Jc \tag{3}$$

$$PrTask_{ij}.\text{duration} \geq t_{ij} \quad \forall i \in I, \quad \forall j \in Jw \tag{4}$$

Constraint 3 fixes the duration of the $PrTask_{ij}$ activity according to the job and the assigned chemical bath. The task duration must be equal to the residence time since the intermediate storage policies "zero wait" and "no intermediate storage" must be followed in the chemical bath j. In turn, constraint 4 places a lower bound on the duration of the $PrTask_{ij}$ activity according to the job and the assigned water bath. Processing activities can be delayed after their predefined minimum residence times since the intermediate storage policies "local storage" and "no intermediate storage" must be followed in the water bath j.

Robots Allocation and Timing Constraints.

$$TrTask_{ij-1j} \text{ requires } R \quad \forall i \in I, \quad \forall j \in J,\ j \neq first(J) \tag{5}$$

$$TrTask_{ij-1j}.\text{duration} = \pi_j \quad \forall i \in I, \quad \forall j \in J,\ j \neq first(J) \tag{6}$$

$$TrTask_{ij-1j}.\text{start} = PrTask_{ij-1}.\text{end} \quad \forall i \in I, \quad \forall j \in J,\ j \neq first(J) \tag{7}$$

$$TrTask_{ij-1j}.\text{end} = PrTask_{ij}.\text{start} \quad \forall i \in I, \quad \forall j \in J,\ j \neq first(J) \tag{8}$$

Constraint 5 prescribes that the transportation task $TrTask_{ij-1j}$ must be assigned to just one robot of the set of alternative material handling resources. Constraint 6 fixes the length of the transportation activity $TrTask_{ij-1j}$ of lot i between baths $j - 1$ and j. The duration of this task is strictly equal to the time required by a given robot to go from bath $j - 1$ to j since this resource cannot be used as temporary storage. Constraints 7 and 8 specify the start and end times of activities that take place in robots. Expression 7 enforces the start of task $TrTask_{ij-1j}$ of job i to coincide with the end of the processing activity on lot i in bath $j - 1$. Constraint 8 sets the end of activity $TrTask_{ij-1j}$ of job i to be equal to the start of the processing task on lot i in bath j.

$$\begin{gathered} activityHasSelectedResource(TrTask_{ijj+1}, R, robot_r) \wedge \\ activityHasSelectedResource(TrTask_{i'j-1j}, R, robot_r) \wedge \\ TrTask_{ijj+1} \text{ precedes } TrTask_{i'j-1j} \Rightarrow \\ PrTask_{ij}.\text{end} + \pi_{j+1} + \pi_j \leq PrTask_{i'j}.\text{start} \\ \forall i, i' \in I, i \neq i', \quad \forall j \in J, \quad \forall r \in R, j \neq last(J),\ j \neq first(J) \end{gathered} \tag{9}$$

Given that one robot is not able to deliver a lot in a bath and immediately pick up another one from the same bath, these activities must be executed with a certain temporal separation. In order to do that, the temporal relationship between the processing activities performed in both lots is fixed. Constraint 9 prescribes the situation when the activity corresponding to lot i is performed before the one representing lot i'; the robot must first transport lot i from bath j to $j + 1$ (π_{j+1}) and later move lot i' from bath $j - 1$ to j (π_j).

$$\begin{gathered} PrTask_{ij}.\text{end} + \pi_{j+1} + \pi_j \leq PrTask_{i'j}.\text{start} \vee PrTask_{i'j}.\text{end} + \\ \pi_{j+1} + \pi_j \leq PrTask_{ij}.\text{start} \quad \forall i, i' \in I, i \neq i', \\ \forall j \in J,\ j \neq last(J) \end{gathered} \tag{10}$$

It is important to note that when a single robot is available to carry out transportation activities, constraint 9 can be simplified. Thus, constraint 10 is needed in the model in order to prescribe the temporal relationships between the processing activities performed in the lots.

Objective Function. If makespan is chosen as the objective function, expressions 11 and 12 allow its definition. This goal

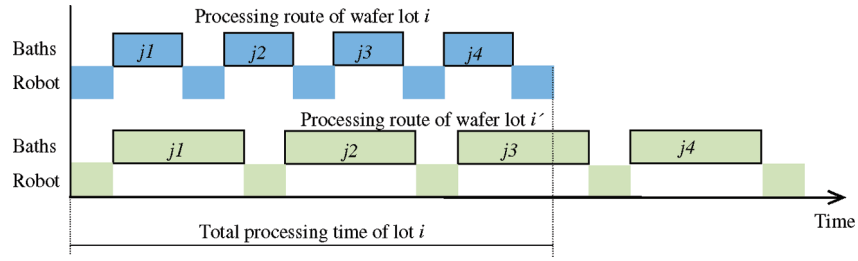

Figure 2. Conceptual representation of the execution of two wafer lots with different total processing times.

```
search {                                                                                    1
   forall(j in J:  j>first(J) ordered by decreasing ord(j))  {                              2
      forall( i in I ordered by increasing TPT[i]){                                         3
         tryall(n in 1..card(R))                                                            4
            activityHasSelectedResource(TrTask[i,j-1,j],R,robot[nextc(first(R),n-1+ord(i))]);  5
         };                                                                                 6
      };                                                                                    7
   forall(j in J:  j<last(J) ordered by decreasing ord(j))  {                               8
      forall( i in I ordered by increasing TPT[i]){                                         9
         try                                                                                10
            PrTask[i,j].start = dmin(PrTask[i,j].start)                                     11
            |                                                                               12
            PrTask[i,j].start > dmin(PrTask[i,j].start)                                     13
         endtry;                                                                            14
         };                                                                                 15
      };                                                                                    16
};                                                                                          17
```

Figure 3. Search strategy that guides the variable domain reduction procedure (GVDR).

used in the proposed CP approach aims at minimizing the total time required to complete all the lots in the wet-etch stations.

$$PrTask_{ij}.\text{end} \leq Mk \quad \forall i \in I, \quad \forall j \in J,\ j = last(J) \tag{11}$$

$$\min Mk \tag{12}$$

4. Search Strategies

CP systems allow users to choose one search strategy from a set of default ones or define one which can be tailored to a given problem type. This last feature makes the CP paradigm very attractive for solving real-life or large-size optimization problems. Some of the most common default strategies are depth-first search (DFS), slice-based search (SBS), and depth-bounded discrepancy search (DBDS). Since they do not take advantage of problem information, their computational performances can be significantly improved in general. In this work, we introduce a domain-specific search procedure that attempts to avoid losing time due to early bad choices in the search and tries to produce good quality solutions with reduced computational effort. This objective is sought out by guiding the exploration process, and not by restricting the search space.

The search strategy presented in this section attempts to improve the computational performance by resorting to a balanced assignment of transportation activities to robots and a variable domain reduction approach associated with processing tasks that guide the movement through the search space. For both procedures, activities are arranged in identical form. Each method starts with activities associated with lots in the last bath, then, it continues with tasks corresponding to the lot connected to the previous bath and so on, until reaching tasks associated with the first bath. Another characteristic of the arrangement is that it focuses on tasks corresponding to lots that seem to be more demanding in a given time period, in terms of transportation resources, i.e., the ones that demand more times the robots. This finding can be noticed by analyzing Figure 2 corresponding to two wafer lots, with different total processing times, in a system with four baths.

As can be seen, since the total processing time of lot i is smaller than the one of lot i', the robot is used more times by lot i than by lot i' considering the total processing time of lot i. Thus, for each lot, the ordering procedure calculates its associated total processing time (TPT_i) just by adding the processing times required in all baths. The processing times in the bath with the smallest average processing time are affected by a weighing factor (wf, greater than 1) in order to emphasize the bath relevance in the complete processing sequence. The average processing time of each bath is computed adding the processing times of all lots requiring the bath and then dividing the result by the number of lots. Afterward, tasks corresponding to lots are organized in an increasing arrangement of total processing times in order to try to address first those lots that seem to be more demanding in terms of transport resources.

The proposed strategy is referred to as guided variable domain reduction (GVDR). The strategy GVDR is depicted in Figure 3. Statements in lines 2 and 3 order the assignment of transportation activities describing the movements of wafer lots between consecutive baths according to the arrangement of activities pointed out before. Therefore, the strategy starts with transportation tasks corresponding to the lot movements to the last bath (j), then it continues with tasks corresponding to the lot passages to the previous bath ($j - 1$) and so on, until reaching tasks associated with the second bath. For each bath, tasks connected with lots are addressed in an increasing order of total processing times. Considering that arrangement of transportation tasks, the "tryall" instruction (line 4 in Figure 3) tries to allocate the transportation activity corresponding to the lot with the smallest total processing time in the last bath to robots, then it attempts with the task associated with the next lot in the last bath and so on, until reaching the transportation activity corresponding to the lot with the biggest total processing time in the last bath (j). Then, the procedure continues in the same fashion with the lot movements associated with the previous

bath ($j - 1$) by taking them in the order that was previously found. Thus, it proceeds chronologically to build an initial allocation of activities to robots in a consistent fashion.

The assignment step starts with the task corresponding to the lot with the smallest total processing time in the last bath, $TrTask_{ij}$, one in turn, trying to allocate this to one of the devices that belongs to the set of alternative robots R. If an assignment is found to be infeasible, the system will backtrack and try another one. When all robots belonging to set R have been tried and have also failed, the backtracking step is more profound and affects the previously allocated task. Therefore, if necessary, all the assignments can be revised and the backtracking process can achieve the allocation of the activity corresponding to the first lot in the last bath. It is worth noting that the third argument of the *activityHasSelectedResource* predicate (line 5 in Figure 3) is a function that attempts each task assignment by iterating over the corresponding set of alternative robots in a circular way ("nextc"). Consequently, if an activity $TrTask_{ij}$ has been initially assigned to robot r, the system tries to allocate the next activity $TrTask_{i'j}$ to unit r', which is the next one in set R. If the number of wafer lots is greater than the number of robots, once the last unit of set R has been assigned an activity, the next task is allocated to the first unit of such set. Therefore, the procedure begins a second step of assignments.

If all transportation tasks have successfully been allocated, the variable domain reduction procedure begins. It attempts to find the timing to the processing activities operating on the domains of their start time variables. Considering the same arrangement of baths and lots used in the assignment step (lines 8 and 9 in Figure 3), the domain pruning, which corresponds to statements in lines 8−14, can start. This recursive procedure operates on the domains of the start time variables associated with lots. The domains of such variables are characterized by two extreme points: the earliest start and the latest start times (EST and LST). The proposed domain reduction approach performs the pruning by adopting lower values for the ESTs (line 11). The domain pruning procedure begins by assigning the least possible value to the start time of the lot with the smallest total processing time in the last bath. If such variable instantiation succeeds, it proceeds likewise with the next lot in the last bath. It continues in a similar way until all the domains of the lots associated with the last bath have been tried. Then, the procedure continues in the same fashion with the lots in the previous bath by taking them in the order that was previously found. If an infeasible solution is found, backtracking takes place by relaxing the domain of the last addressed variable and then attempting the allocation of a value greater than the previous one (line 13). Eventually, the domain reduction procedure succeeds in finding a feasible solution to the problem. When the domain pruning procedure cannot find a set of start times for all processing tasks, the backtracking will be even deeper and will reach the assignment of transport activities. The strategy ends when the entire search tree has been explored.

To show the effect of using search procedures, the model presented in section 3 was solved with three default methods (DFS, SBS, and DBDS) as well as with another search procedure that takes advantage of domain knowledge. The SGVDR strategy (simplified guided variable domain reduction) is a slightly modified version of the GVDR strategy where lots are addressed in the specific order shown in the set of lots and not organized in an increasing order of total processing times. Thus, no clever order is given for lots in SGVDR. In addition, it is worth noting that in cases where only one robot is considered, lines 2−5 (Figure 3) are not needed and therefore GVDR strategy is only composed of lines 8−16.

Table 1. Problem IDs and Main Characteristics

	[$M \times N$]	
problem	baths	jobs
P1	6	5
P2	6	15
P3	6	25
P4	12	5
P5	12	15
P6	12	25
P7	4	8
P8	4	8
P9	12	10

5. Computational Results

In the literature, there are a huge number of case studies connected with the scheduling problem in semiconductor manufacturing. Nevertheless, most of them are related not to the wet-etching process, but to other typical fabrication areas in the semiconductor industry. In addition, some problems are simplified to cope with the assumptions considered in formulations. For example, Ham et al.[37] developed a MILP approach without taking into account transfer and storage limitations, and assuming identical processing times for all combinations of jobs, steps, and machines (equal to 1). Then, the mathematical formulation was tested with problems involving 15 operations, 5 flexible machines, and 60 jobs.

In this section, nine test problems are solved to illustrate the capabilities of the CP approach (model + search strategy). Examples were taken from the specific and more relevant literature related to the wet-etching operation involved in semiconductor manufacturing. Table 1 summarizes the characteristics of the various case studies that are going to be considered. Data for case studies P1−P6 and P9 were taken from Bhushan and Karimi[30] and are given in Table 6 in the Appendix. Examples P1−P3 correspond to the first 6 consecutive baths and the first 5, 15, and 25 wafer lots, respectively. Case studies P4, P5, P6, and P9 comprise the 12 consecutive baths and the first 5, 15, 25, and 10 wafer lots, respectively. Problem P6 corresponds to the [12 × 25] case study introduced by Bhushan and Karimi.[30] Processing and transfer times for case study P7, introduced by Bhushan and Karimi,[28] are given in Table 7 in the Appendix. Example P7 corresponds to a [4 × 8] scheduling problem, with case study P8 being a slightly modified version, previously studied by Aguirre and Méndez.[31] In problem P8 transfer times are 10 times larger than in P7, while the processing times are the same. Only one robot is considered in problems P1−P9.

Case studies were solved to a maximum time limit of 3600 CPU seconds by the commercial software ILOG OPL Studio 3.7,[35] based on the ILOG Solver[38] and Scheduler[36] packages. In all cases, a computer consisting of an AMD Athlon 64 X2 Dual Core 2.2 GHz processor with 1 GB of RAM was used.

5.1. Comparison of Search Strategies. The goal of this section is to evaluate the quality of the CP model and the search strategies introduced. It is worth noting that the CP model will be tested considering the default search strategies DFS, SBS, and DBDS as well as the new proposed SGVDR and GVDR search strategies. In all cases where GVDR was used, the weighing factor took the value $wf = 1.2$, which was experimentally proved to be appropriate. It is important to note that the search strategy was adjusted to obtain the best performance considering the case studies found in the literature. Nevertheless,

Table 2. Computational Results for Problems P1−P6[a]

problem	variables	constraints	first solution		best solution/optimal		search strategy
			makespan	CPU time	makespan	CPU time	
P1 [6 × 5]	226	380	NS	−	NS	3600[b]	DFS
			97.4	398.5	82.6	403.3	SBS
			91.6	35.47	82.6	40.81	DBDS
			93.7	0.05	82.6	3.84	SGVDR
			92.6	0.01	82.6	2.84	GVDR
P2 [6 × 15]	676	1665	NS	−	NS	3600[b]	DFS
			NS	−	NS	3600[b]	SBS
			NS	−	NS	3600[b]	DBDS
			208.6	0.13	189.2	2301[b]	SGVDR
			205.4	0.14	185.0	350[b]	GVDR
P3 [6 × 25]	1126	3650	NS	−	NS	3600[b]	DFS
			NS	−	NS	3600[b]	SBS
			NS	−	NS	3600[b]	DBDS
			336.0	0.45	312.1	2717[b]	SGVDR
			325.1	0.53	297.3	1346[b]	GVDR
P4 [12 × 5]	406	695	NS	−	NS	3600[b]	DFS
			NS	−	NS	3600[b]	SBS
			NS	−	NS	3600[b]	DBDS
			153.2	0.04	144.1	0.22[b]	SGVDR
			161.5	0.06	144.1	0.39[b]	GVDR
P5 [12 × 15]	1216	3060	NS	−	NS	3600[b]	DFS
			NS	−	NS	3600[b]	SBS
			NS	−	NS	3600[b]	DBDS
			308.8	1.10	281.5	1260[b]	SGVDR
			294.0	0.76	273.2	949[b]	GVDR
P6 [12 × 25]	2026	7675	NS	−	NS	3600[b]	DFS
			NS	−	NS	3600[b]	SBS
			NS	−	NS	3600[b]	DBDS
			465.6	9.92	451.8	818[b]	SGVDR
			497.5	17.29	443.4	493[b]	GVDR

[a] NS, no solution was found in 3600 s. [b] Unable to prove optimality within the imposed time limit (3600 s).

if instance characteristics change, the search mechanism may require a readaptation to achieve the best performance.

Computational results for examples P1−P6 are presented in Table 2. It is clear that the model performs poorly with default search strategies (DFS, SBS, and DBDS). Thus, there is an obvious need for search strategies exploiting domain-specific knowledge, such as the structure of the manufacturing plants.

The GVDR strategy showed the best performance of the five search strategies because it allowed finding better solutions for problems P2−P6. As can be seen, the best solution reached for problem P6 when adopting the GVDR strategy was 1.86% better than the one obtained when taking into account the SGVDR strategy. Furthermore, the best solution was achieved in short CPU time. The GVDR strategy led to the attainment of an optimal solution for problem P1 slightly faster than SGVDR and considerably quicker than SBS and DBDS. While SGVDR and GVDR achieved solutions for all problems, default strategies were not able to find a solution for cases P2−P6 within the predefined time limit. In addition, both SGVDR and GVDR strategies allowed improving the quality of the initial solutions.

Since Table 2 illustrates the first feasible solutions that were found and the differences among these and the optimal or suboptimal ones reached within 3600 s, another analysis of the CP performance can be done from this perspective. In all cases, the incorporation of the SGVDR and GVDR strategies originated solutions in which the quality of these was quickly improved. In addition, it can be seen that in most cases, with the exceptions of P4 and P6, the GVDR strategy originated first solutions of better quality than the SGVDR strategy. Finally, Table 2 also shows the problem sizes, to evaluate the impact of an increase in the number of wafer lots and baths on the problem size. It follows that when the problem size increases, it becomes more difficult to find a feasible solution and prove optimality. In fact, only P1 can be solved to optimality, meaning that there is still room for improvement.

5.2. Comparison of the CP Approach with Others Taken from the Literature. The scheduling problem of AWSs has been considered by different contributions, such as Bhushan and Karimi[30] as well as Aguirre and Méndez.[31] These authors solved different case studies, among them P6−P8. It is worth noting that, since SGVDR, DFS, SBS, and DBDS search mechanisms tested in this study rendered a lower computational performance than the GVDR strategy in most of examples shown in Table 2, this section only shows computational results corresponding to the CP model with the GVDR strategy. Results for the CP approach are compared with those obtained using our implementation of the MILP approach given by Aguirre and Méndez.[31] In order to make the direct comparison of CPU times, the mathematical formulation was implemented in ILOG OPL Studio 3.7[35] and solved with CPLEX 9.1.[39]

Table 3 shows a comparison between the results of the CP approach and the MILP model considering problems P1−P9. For case studies P4, P7, and P8, it should be noted that the proposed CP formulation requires lower CPU times to obtain the best solutions. For problems P4, P7, and P8, this contribution achieved the optimal solution in 0.39, 1.32, and 1.40 s, respectively, in comparison with 7.36, 12.58, and 72.34 s required by the mathematical model. Nevertheless, it is worth remarking that while the CP approach was unable to guarantee global optimality for the best solutions generated within the maximum CPU time for examples P4, P7, and P8, the MILP formulation did that in 14.49, 33.16, and 152.68 s, respectively. For the moderate and large case studies (P3, P5, P6, and P9), the CP approach achieved feasible solutions, while the mathematical method found no solution within the given time limit for examples P3, P5, and P6. Moreover, the solution reached with the CP approach for problem P9 is 3.5% better than the one obtained with the MILP model. Another important remark is that the computational performance of the MILP formulation decreased faster than the one of the CP approach.

Table 3. Comparison of Results Corresponding to Problems P1−P9

			first solution		best solution/optimal		
problem	variables	constraints	makespan	CPU time	makespan	CPU time	approach
P1 [6 × 5]	511	1050	218.1	0.01	82.6	0.94	MILP
	226	380	92.6	0.01	82.6	2.84	CP + GVDR
P2 [6 × 15]	4756	10500	196.1	1687	195.2	3600[a]	MILP
	676	1665	205.4	0.14	185.0	350[a]	CP + GVDR
P3 [6 × 25]	13301	29750	NS	−	NS	3600[a]	MILP
	1126	3650	325.1	0.53	297.3	1346[a]	CP + GVDR
P4 [12 × 5]	1711	3510	154.4	2.38	144.1	(7.36) 14.49	MILP
	406	695	161.5	0.06	144.1	0.39[a]	CP + GVDR
P5 [12 × 15]	16906	35880	NS	−	NS	3600[a]	MILP
	1216	3060	294.0	0.76	273.2	949	CP + GVDR
P6 [12 × 25]	47777	102051	NS	−	NS	3600[a]	MILP
	2026	7675	497.5	17.29	443.4	493.37[a]	CP + GVDR
P7 [4 × 8]	661	1512	87.825	0.17	84.37	(12.58) 33.16	MILP
	265	632	96.585	0.02	84.37	1.32[a]	CP + GVDR
P8 [4 × 8]	661	1512	139.01	3.45	120.47	(72.34) 152	MILP
	265	632	128.20	0.05	120.47	1.40[a]	CP + GVDR
P9 [12 × 10]	7316	9768	206.30	3452	206.30	3452[a]	MILP
	811	1715	232.8	0.38	199.00	3440[a]	CP + GVDR

[a] Unable to prove optimality within the imposed time limit (3600 s).

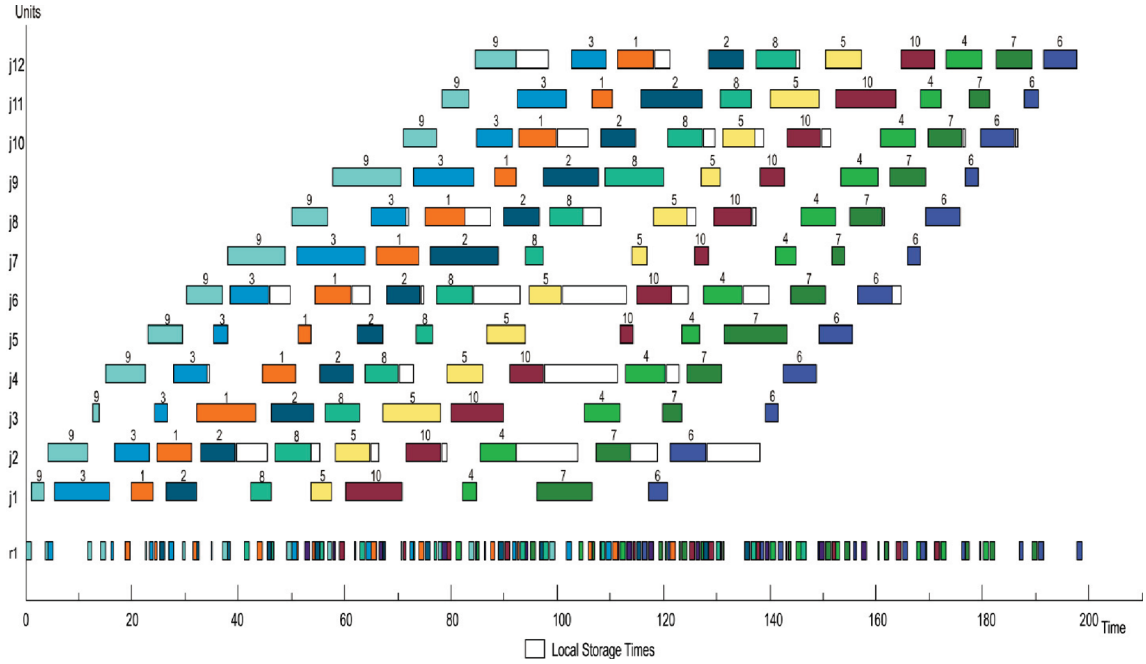

Figure 4. Gantt diagram depicting the best solution for example P9.

Table 4. Comparison of Results Corresponding to Problem P6 [12 × 25][a]

		first solution		best solution		
variables	constraints	makespan	CPU time	makespan	CPU time	approach
47777	102051	NS	−	NS	3600[b]	MILP model
−	−	−	−	478.6	−	A2′ heuristic
2026	6725	497.8	17.29	443.4	493.97[b]	CP model + GVDR

[a] NS, no solution was found in 3600 s; −, not reported. [b] Unable to prove optimality within the imposed time limit (3600 s).

The best schedule generated for problem P9 is shown in Figure 4. This figure depicts the bath's workload and the requirements of robot and moreover shows the high coordination that must exist between the production system and the material handling device.

Bhushan and Karimi[30] solved different examples, one of them being P6, which involves 12 consecutive baths and 25 wafer lots and is thus larger than P7−P9. Table 4 presents the computational statistics for their[30] heuristic approach, for the MILP model by Aguirre and Méndez[31] and our CP integrated approach. Table 4 lists the results reported in ref 21 for the A2′ heuristic algorithm, which had the best performance among all algorithms tested. A2′ is a heuristic algorithm composed of three parts: (1) one sequencing technique based on TS, (2) one heuristic for initial sequence called nNEH, and (3) one scheduling algorithm based on iterative improvements referred to as II.

An analysis of the results in Table 4 shows that the CP formulation presented the best performance. While the MILP model could not obtain a feasible solution within the time limit of 3600 s, the solution reached by the heuristic algorithm was worse than the one obtained using the CP formulation. As can be seen, the best solution reached by the proposed approach is 7.35% better than the one obtained when taking into account

Table 5. Comparison of Results Corresponding to Problems P2 and P3 with Two and Three Robots[a]

			first solution		best solution		
problem	variables	constraints	makespan	CPU time	makespan	CPU time	approach
P2, two robots	4996	22380	181.3	410.85	170.9	3589[b]	MILP model
	676	3450	179.4	0.48	169.7	4.46[b]	CP model + GVDR
P2, three robots	5116	32670	165.6	546.34	164.2	2371[b]	MILP model
	676	4710	187.5	0.08	164.0	2009[b]	CP model + GVDR
P3, two robots	13701	63550	NS	−	NS	3600[b]	MILP model
	1126	8750	291.7	2.48	274.8	44.16[b]	CP model + GVDR
P3, three robots	13901	92950	NS	−	NS	3600[b]	MILP model
	1126	12350	289.4	3.2	269.6	56.38[b]	CP model + GVDR

[a] NS, no solution was found in 3600 s. [b] Unable to prove optimality within the imposed time limit (3600 s).

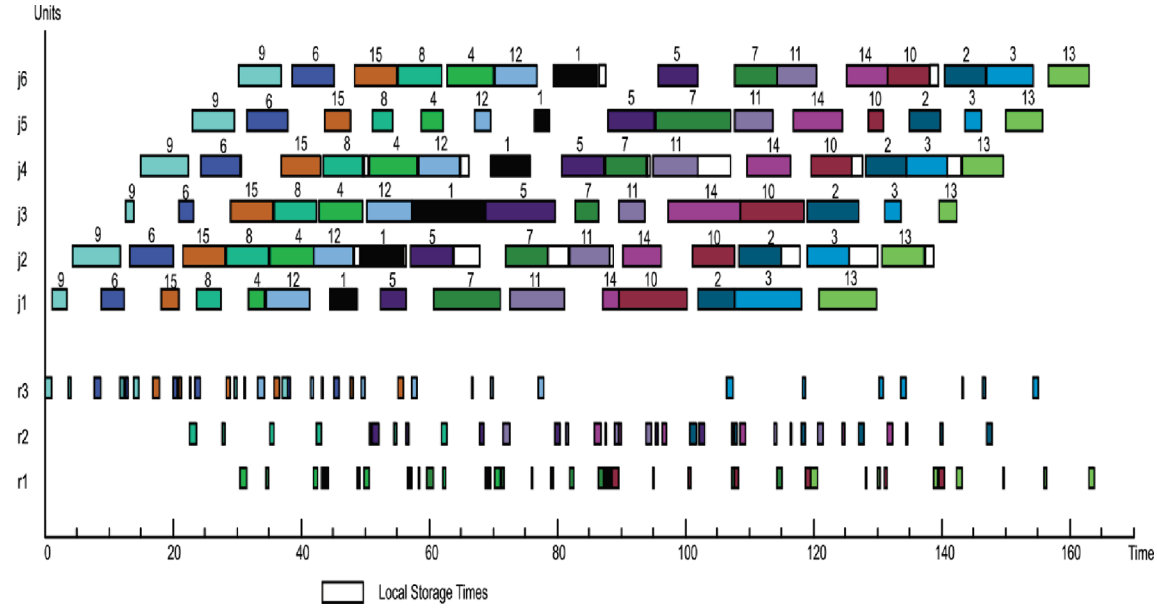

Figure 5. Gantt diagram depicting the best solution for problem P2 with three robots.

the A2′ heuristic. In addition, the CP approach achieved a good quality solution (443.4) in short CPU time (8.23 min).

5.3. Performance with Multiple Robots. In order to extend the comparison between the CP approach and the MILP model for problems involving multiple robots, Table 5 lists the model statistics and computational results, for case studies P2 and P3 with two and three robots. It can again be seen that, in all instances, the CP approach originated first feasible solutions in very short CPU times and quickly improved the quality of solutions. It is important to remark that while the CP approach was able to obtain feasible solutions for all cases within the maximum CPU time enforced, the MILP formulation achieved feasible solutions only for P2. Furthermore, for these cases, the CP formulation achieved better quality solutions in less time. While the best solution for the instance with two robots was achieved in 4.46 s by the CP approach, the MILP formulation obtained a lower quality solution in 3589 s. Nevertheless, the solution quality between MILP and CP is roughly the same for problem P2 for the imposed time limit.

Figure 5 depicts the Gantt chart describing the best solution reached for problem P2 with three robots when the GVDR strategy is employed. It can be noticed that transport tasks were allocated to robots trying to balance the load of these.

6. Conclusions

An integrated CP approach for the scheduling problem of AWSs involving multiple robots in the material handling system and strict intermediate storage policies has been presented. The formulation is composed of two parts: a new model and an efficient search strategy that is shown to have a major impact on the computational performance when compared to default search strategies. The proposed search strategy (GVDR) speeds up the solution process by guiding the assignment of transport activities to material handling devices and the domain variable reduction procedure on the processing activities. Overall, it puts emphasis on the idea that an underlying search strategy that is tailored to specific problems is fundamental for solving complex optimization problems.

In the 13 case studies considered, the integrated approach had a good computational performance. Despite not reaching optimal solutions for moderate and large-size case studies within the 3600 s time limit, solutions of very good quality were achieved within a few minutes of CPU time. The proposed CP approach was shown to compare favorably with the recent MILP formulation reported by Aguirre and Méndez[31] as well as with the heuristic approach of Bhushan and Karimi.[30] In particular, it is important to remark that, when problem sizes increase, the computational performance of the CP approach decreases more slowly than the one of the MILP formulation. In addition, in contrast to rigorous MILP models, the CP formulation is able to guide the search in order to find a good solution in a reasonable time, even for medium-scale problems. In view of the above, the proposed integrated approach has the potential to be used by industry as a powerful decision-making tool.

Table 6. Processing Times of Jobs on Baths and Transfer Times of Jobs for Examples P1−P6 and P9

	bath											
job	1	2	3	4	5	6	7	8	9	10	11	12
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
19	9.7	6.7	7.6	6.4	10.9	6.9	2.6	6.7	4.6	6.6	10.1	6.3
20	2.6	6.7	2.9	6.5	10.4	6.9	2.6	6.7	11.5	6.6	3.7	6.2
21	4.7	6.6	4.9	6.9	2.6	6.8	12.7	6.2	2.6	6.7	6.9	6.4
22	2.5	6.3	2.6	6.6	7.9	6.8	12.5	6.8	2.6	6.5	7.8	6.4
23	11.4	6.4	8.9	6.6	2.7	6.4	11.4	7.4	11.3	6.8	2.9	6.9
24	6.8	6.5	2.8	7.5	3.9	7.2	9.8	6.5	8.6	6.3	11.8	6.2
25	8.8	6.9	8.8	6.8	11.3	6.8	11.3	6.1	6.7	6.5	2.6	6.4
$\pi_1 = 1.2$	$\pi_2 = 0.6$	$\pi_3 = 0.8$	$\pi_4 = 1.0$	$\pi_5 = 0.4$	$\pi_6 = 0.6$	$\pi_7 = 1.0$	$\pi_8 = 1.0$	$\pi_9 = 0.8$	$\pi_{10} = 0.4$	$\pi_{11} = 0.8$	$\pi_{12} = 1.0$	$\pi_{13} = 1.2$

Table 7. Processing Times of Jobs on Baths and Transfer Times of Jobs for Example P7

	bath			
job	1	2	3	4
1	11.10	6.68	5.24	6.92
2	8.47	6.35	10.10	7.02
3	9.19	6.35	4.60	6.71
4	10.80	7.12	10.20	6.83
5	7.40	7.05	4.07	6.58
6	10.80	6.76	1.01	6.37
7	3.48	6.67	1.41	6.46
8	2.51	6.23	8.00	6.23
$\pi_1 = 0.1$	$\pi_2 = 0.2$	$\pi_3 = 0.15$	$\pi_4 = 0.175$	$\pi_5 = 0.25$

Acknowledgment

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

Appendix

Processing times of jobs on baths and transfer times of jobs for examples P1−P6 and P9 (Table 6) and for example P7 (Table 7) are included.

Literature Cited

(1) Pekny, J. F.; Reklaitis, G. V. Towards the convergence of theory and practice: A technology guide for scheduling/planning methodology. Pekny, J. F., Blau, G. E., Eds.; AIChE Symposium Series, Vol. 94; 1998; pp 91−111.

(2) Grieco, A.; Semeraro, Q.; Tolio, T. A Review of Different Approaches to the FMS Loading Problem. *Int. J. Flexible Manuf. Syst.* **2001**, *13*, 361–384.

(3) Kallrath, J. Planning and scheduling in the process industry. *OR Spectrum* **2002**, *24*, 219–250.

(4) 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.

(5) Méndez, C. A.; Cerdá, 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.

(6) Marriot, K.; Stuckey, P. *Programming with Constraints. An Introduction*; The MIT Press: Cambridge, MA, 1999.

(7) Van Hentenryck, P. *The OPL optimization programming language*; The MIT Press: Cambridge, MA, 1999.

(8) Harjunkoski, I.; Grossmann, I. E. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comput. Chem. Eng.* **2002**, *26* (11), 1533–1552.

(9) Maravelias, C. T.; Grossmann, I. E. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Comput. Chem. Eng.* **2004**, *28* (10), 1921–1949.

(10) Roe, B.; Papageorgiou, L. G.; Shah, N. A hybrid MILP/CLP algorithm for multipurpose batch process scheduling. *Comput. Chem. Eng.* **2005**, *29* (6), 1277–1291.

(11) Huang, W.; Chung, P. W. H. Integrating routing and scheduling for pipeless plants in different layouts. *Comput. Chem. Eng.* **2005**, *29* (5), 1069–1081.

(12) Zeballos, L. J.; Henning, G. P. CP method for the scheduling of multiproduct continuous plants with resource constrains. *Comput.-Aided Chem. Eng.* **2006**, *21*, 1961–1966.

(13) Zeballos, L. J.; Méndez, C. A. An integrated CP-based approach for scheduling of processing and transport units in pipeless plants. *Ind. Eng. Chem. Res.* **2010**, *49*, 1799–1811.

(14) Hooker, J. N. Logic, optimization and constraint programming. *INFORMS J. Comput.* **2002**, *14*, 295–321.

(15) Bixby, R.; Burda, R.; Miller, D. Short-Interval Detailed Production Scheduling in 300mm Semiconductor Manufacturing using Mixed Integer and Constraint Programming. In Proceedings of the 17th Annual SEMI/IEEE, Advanced Semiconductor Manufacturing Conference. IEEE Computer Society Press: Boston, MA, 2006; pp 148–154.

(16) Jain, V.; Grossmann, I. E. Algorithms for hybrid MILP/CP model for a class of optimization problems. *INFORMS J. Comput.* **2001**, *13*, 258–276.

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

(18) Castro, P. M.; Grossmann, I. E. New Continuous-Time MILP Model for the Short-Term Scheduling of Multistage Batch Plants. *Ind. Eng. Chem. Res.* **2005**, *44*, 9175–9190.

(19) Hui, C. W.; Gupta, A. A novel MILP formulation for short term scheduling of multistage multi-product batch plants. *Comput. Chem. Eng.* **2000**, *24*, 1611–1617.

(20) Méndez, C. A.; Cerdá, J. Short-term scheduling of multistage batch processes subject to limited finite resources. *Comput.-Aided Chem. Eng.* **2004**, *15B*, 984–989.

(21) Liu, J.; MacCarthy, B. L. A global MILP Model for FMS scheduling. *Eur. J. Oper. Res.* **1997**, *100*, 441–453.

(22) Mendez, C. A.; Cerda, J. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optim. Eng.* **2003**, *4*, 7–22.

(23) Sundaramoorthy, A.; Maravelias, C. T. Modeling of Storage Constraints in Batching and Scheduling of Multi-stage Processes. *Ind. Eng. Chem. Res.* **2008**, *47* (17), 6648–6660.

(24) Mathirajan, M; Sivakumar, A. I. A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *Int. J. Adv. Manuf. Technol.* **2006**, *29*, 990–1001.

(25) Hung, Y. F.; Wang, Q. Z. A new formulation technique for alternative material planning an approach for semiconductor bin allocation planning. *Comput. Ind. Eng.* **1997**, *32*, 281–297.

(26) Uzsoy, R.; Lee, C. Y.; Martin-Vega, L. A. A review of production planning and scheduling models in the semiconductor industry, part I: system characteristics, performance evaluation and prod planning. *IIE Trans.* **1992**, *24*, 47–60.

(27) Uzsoy, R.; Lee, C. Y.; Martin-Vega, L. A. A review of prod planning and scheduling models in the semiconductor industry, part II: shop floor control. *IIE Trans. Sched. Logistics* **1994**, *26*, 44–55.

(28) Bhushan, S.; Karimi, I. A. An MILP approach to automated wet-etch station scheduling. *Ind. Eng. Chem. Res.* **2003**, *42* (7), 1391–1399.

(29) Geiger, C. D.; Kempf, K. G.; Uzsoy, R. A tabu search approach to scheduling an automated wet etch station. *J. Manuf. Syst.* **1997**, *16* (2), 102–116.

(30) Bhushan, S.; Karimi, I. A. Heuristic algorithms for scheduling an automated wet-etch station. *Comput. Chem. Eng.* **2004**, *28* (3), 363–379.

(31) Aguirre, A. M.; Méndez, C. A. A novel optimization method to automated wet-etch station scheduling in semiconductor manufacturing systems. *Comput.-Aided Chem. Eng.* **2010**, *28*, 883–888.

(32) Roslof, J.; Harjunkoski, I.; Bjorkqvist, J.; Karlsson, S.; Westerlund, T. An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. *Comput. Chem. Eng.* **2001**, *25*, 821–828.

(33) Janak, S. L.; Floudas, C. A.; Kallrath, J.; Vormbrock, N. Production Scheduling of a Large-Scale Industrial Batch Plant. II. Reactive Scheduling. *Ind. Eng. Chem. Res.* **2006**, *45*, 8253–8269.

(34) Castro, P. M.; Harjunkoski, I.; Grossmann, I. E. Optimal Short-Term Scheduling of Large-Scale Multistage Batch Plants. *Ind. Eng. Chem. Res.* **2009**, *48*, 11002–11016.

(35) ILOG. *OPL Studio 3.7, User's Manual*; 2005.

(36) ILOG. *Scheduler 6.0, User's Manual*; 2005.

(37) Ham, M.; Lee, Y. H.; Fowler, J. W. Integer Programming-based Real-time Scheduler in Semiconductor Manufacturing. In Proceedings of the 2009 Winter Simulation Conference. Rossetti, M. D.; Hill, R. R., Johansson, B., Dunkin, A., Ingalls, R. G., Eds.; Austin, TX, 2009; pp 1657−1666.

(38) ILOG. *Solver 5.0, User's Manual*; 2005.

(39) ILOG. *CPLEX 9.1, User's Manual*; 2005.

Received for review July 29, 2010
Revised manuscript received October 15, 2010
Accepted November 30, 2010

IE1016199