A CP-based Approach for Scheduling of Automated Wet-Etch Stations

Luis J. Zeballos1,2, Pedro M. Castro2 and Carlos A. Méndez3

1Universidad Nacional del Litoral - Facultad de Ingeniería Química, Santa Fe, Argentina, luis.zeballos@ineti.pt
2 Unidade Modelação e Optimização de Sistemas Energéticos, Laboratório Nacional de Energia e Geologia, Lisboa, Portugal, pedro.castro@ineti.pt
3 INTEC (Universidad Nacional del Litoral - CONICET), Santa Fe, Argentina, cmendez@intec.unl.edu.ar

Abstract
This contribution addresses the resource-constrained flowshop scheduling problem of Automated Wet-etch Stations (AWSs) in wafer fabrication of semiconductor manufacturing facilities by means of a Constraint Programming (CP) methodology. The proposed integrated approach consists of both a CP model and an efficient search strategy. The former handles different features found in the complex automated wet-etching process, such as material handling and mixed intermediate storage policies. The domain-specific search strategy, significantly improves the computational performance, a key aspect given the high combinatorial complexity of the problem. The applicability of the proposed integrated CP methodology is successfully tested with four examples taken from the literature, featuring a different number of jobs and stages, with the outcome being better solutions with fewer computational resources.

Keywords: Scheduling, Constraint Programming, Wet-Etch Stations.

1. Introduction
Semiconductor manufacturing involves many batch/semicontinuous physic and chemical operations. The wafer fabrication, which is the main stage in semiconductor manufacturing facilities, 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 SiO2. 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. Thus, these units are responsible for removing the unnecessary film of SiO2 formed on the surface of the wafer, in a series of chemical and deionizing baths.

The inherent complexity of the scheduling problem of automated wet-etch stations has been addressed in several works. However, the reported approaches still have weaknesses. Proposals have presented non-exact and non-optimal algorithms (e.g. heuristic algorithm based on tabu search) with severe assumptions in order to make the problem tractable, such as considering transfer times jointly with processing times. Geiger et al.1 presented a heuristic scheduling algorithm based on Tabu search for makespan minimization. These authors first obtain a sequence of lots on baths and then introduce the ordered set of lots in an algorithm for scheduling transfers and processing. Thus, they presented no-exact algorithm where transfer times and processing times are separately addressed. In turn, Bhushan and Karimi2 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. In addition, a non-optimal two-step strategy based on the pointed out mathematical formulation was developed for solving moderately sized problems. Later, Bhushan and Karimi3 introduced other heuristic algorithms for larger problems, which exhibited a better performance than the one by Geiger et al.1. More recently, Aguirre and Méndez4 developed a MILP continuous-time formulation for the scheduling of AWS processes in the semiconductor industry. Similar to Bhushan and Karimi2, 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.

The goal of this contribution is to overcome some of the limitations pointed out before. This paper presents a constraint programming formulation that addresses the scheduling problem of AWSs in an integrated way, generating 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 act as 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. The approach has been implemented in the ILOG OPL constraint programming language\textsuperscript{5}, embedded in the OPL Studio Package\textsuperscript{6}. It resorts to some specific scheduling constructs, which are available in the ILOG Scheduler\textsuperscript{7} software package.

2. Problem description

The AWS deals with group of wafers of the same type, hereafter named wafer lots. Lots in the input buffer came from the upstream process. One or more robots move lots from bath to bath for processing. 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. Thus, baths are linked by a transportation system that picks up lots in order to execute the material movements. The problem addressed in this work can be described as the scheduling of serial flowshop 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). Figure 1 shows a schematic representation of a typical AWS.

![Figure 1: Schematic representation of a typical Automated Wet-etch Station](image)

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

a) A set of $N$ lots of wafers ($i = 1, 2, \ldots, N$), 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 ($j = 0, 1, \ldots, M+1$, where $M$ is the total number of baths). Since baths alternate in the AWS, odd baths have chemicals ($Jc = \{1, 3, 5, \ldots, M-1\}$) and even baths have water ($Jw = \{2, 4, 5, \ldots, M\}$).

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

c) The processing time for a given lot depends on the lot and the bath.

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

e) Chemical and water baths are linked by robots which transports 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 temporal buffer.

g) Transfer times between consecutive baths for lots depend on the origin and destination only.

h) Baths and robots do not need any setups.

i) 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 allocating and sequencing of transport activities to be carried out by the robot as well as (iii) the timings of the processing and transport activities. The goal of this problem is makespan minimization.
3. Constraint Programming Paradigm

Constraint Programming is a paradigm for modeling and solving optimization problems. This technique consists of two phases. The first one refers to the problem representation itself, using integer and boolean variables and additionally, variables can be 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. Nowadays, this technique is successfully being applied in many domains, specially scheduling. It is important to note that CP is better than other techniques due to its enormous expressiveness.

The constraint programming paradigm has been used by the process system engineering community (Harjunkoski and Grossmann; Maravelias and Grossmann, Roe et al.). An analysis of the most relevant contributions reveals that in a few cases, it has been tested in its pure form. In addition, a few papers have addressed the development of effective and efficient search strategies (Zeballos and Henning, Zeballos and Méndez).

4. Constraint Programming Model

In this section, the CP model for the scheduling problem of a AWS is presented.

Nomenclature

Subscripts
- \( i \): Lot.
- \( j \): Bath or buffer.
- \( r \): Robot.

Sets
- \( I \): Set of lots or jobs.
- \( J \): Set of baths and buffers.
- \( J_w \): Set of water baths.
- \( J_c \): Set of chemical baths.
- \( R \): Set of alternative Robots.

Parameters
- \( t_{ij} \): Residence time of job \( i \) in bath \( j \)
- \( \pi_{j} \): 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**: models any type of bath (chemical or water) where the lots can reside during a given time. Baths are unary resources, which are resources that can process only one activity at a time.

- **robot**: represents a robot. 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, \( PrTask \) and \( TrTask \) are employed. Each activity is described by means of duration, start and end time variables (i.e. \( Task.duration \), \( Task.start \) and \( Task.end \)), that are related among themselves. 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 \( j \).
- \( TrTask_{ij-1} \): models robot movements transporting lot \( i \) from bath \( j \) to \( j+1 \).
- \( Mk \): corresponds to the total time required to complete all the jobs in the wet-etch station.
Model Constraints

The CP approach employs some specific scheduling constructs available in the modeling language ILOG OPL Studio (ILOG, 2003a). One of them is \textit{precedes}, which imposes a proper sequence of non-overlapping activities. Another construct is \textit{requires}, which prescribes the assignment of renewable resources demanded by activities. Finally, one of the most important constructs is \textit{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.

\textbf{Baths assignment constraints}

\begin{align*}
Pr_{\text{Task}}_{ij} & \text{ requires bath}_j \quad \forall i \in I, \forall j \in J \tag{1} \\
Pr_{\text{Task}}_{ij} & \text{ precedes } Pr_{\text{Task}}_{ij'} \quad \forall i \in I, \forall j, j' \in J, j \neq \text{last}(J) \tag{2}
\end{align*}

Constraint (1) is an assignment relation prescribing that lot \(i\) must be assigned to the 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 \textit{precedes}. Therefore, the activity located at the right hand side cannot be started until the activity on the left hand side is finished.

\textbf{Baths timing constraints}

\begin{align*}
Pr_{\text{Task}}_{ij} & \text{ duration } = t_j \quad \forall i \in I, \forall j \in J_c \tag{3} \\
Pr_{\text{Task}}_{ij} & \text{ duration } \geq t_j \quad \forall i \in I, \forall j \in J_w \tag{4}
\end{align*}

Constraint (3) fixes the duration of the \(Pr_{\text{Task}}_{ij}\) activity according to the job and the assigned chemical bath. The task duration must be equal to 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 \(Pr_{\text{Task}}_{ij}\) activity according to the job and the assigned water bath. Processing activities can be delayed more than the predefined minimum residence time since the intermediate storage policies “Local Storage” and “No Intermediate Storage” must be followed in the water bath \(j\).

\textbf{Robots allocation and timing constraints}

\begin{align*}
Tr_{\text{Task}}_{ij-1j} & \text{ requires } R \quad \forall i \in I, \forall j \in J, j \neq \text{first}(J) \tag{5} \\
Tr_{\text{Task}}_{ij-1j} & \text{ duration } = \pi_j \quad \forall i \in I, \forall j \in J, j \neq \text{first}(J) \tag{6} \\
Tr_{\text{Task}}_{ij-1j} & \text{ start } = Pr_{\text{Task}}_{ij-1j} \text{ end } \quad \forall i \in I, \forall j \in J, j \neq \text{first}(J) \tag{7} \\
Tr_{\text{Task}}_{ij-1j} & \text{ end } = Pr_{\text{Task}}_{ij-1j} \text{ start } \quad \forall i \in I, \forall j \in J, j \neq \text{first}(J) \tag{8}
\end{align*}

Constraint (5) prescribes that the transportation task \(Tr_{\text{Task}}_{ij-1j}\) must be assigned to just one robot of the set of alternative material handling resources. Constraint (6) fixes the length of the transportation activity \(Tr_{\text{Task}}_{ij-1j}\) 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 \(j'\) to \(j\) since this resource cannot be used as temporal 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 \(Tr_{\text{Task}}_{ij-1j}\) 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 \(Tr_{\text{Task}}_{ij-1j}\) of job \(i\) to be equal to the start of the processing task on lot \(i\) in bath \(j\).
When a robot carries out two transportation activities associated with different lots processed in the same bath, the temporal relationships between the processing activities performed in the lots must be prescribed. Thus, the timing is determined in constraints (9). This 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 \( j+1 \) and later move lot \( i' \) from bath \( j-1 \) to \( j \).

\[
PrTask_{i, \text{end}} + \pi_{j+1} + \pi_j \leq PrTask_{i', \text{start}} \quad \forall i, j \in I, i \neq i', \forall j \in J, j \neq \text{last}(J)
\]

It is important to note that when a single robot is only 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 objective function, expressions (11) and (12) allow its definition. This goal used in the proposed CP approach aims at minimizing the total time required to complete all the lots in the wet-etch stations.

\[
PrTask_{i, \text{end}} \leq Mk \quad \forall i \in I, j \in J, j = \text{last}(J)
\]

\[
\text{Min } Mk
\]

**5. Search Strategy**

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. 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, in general, their computational performances are poor. 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 feasible solutions with reduced computational effort. The proposed strategy is referred as GVDR (such acronym stands for Guided Variable Domain Reduction).

Strategy GVDR is depicted in Fig. 2. Statements in lines 2 and 3 arrange the processing activities in order to carry out a domain reduction procedure over the task start times. The strategy will start with tasks corresponding to lots in the last bath (\( j \)), then, it will continue with tasks corresponding to lots in the previous bath (\( j-1 \)) and so on, until reaching tasks associated with the first bath. One of the characteristics of the proposed search scheme is that it focuses on lots that seem to be more demanding in a given time period, in terms of transport resources; i.e., the ones that demand more times the robots. For each lot, the ordering procedure calculates its associated total processing time (TPT) just by adding the processing times required by the lot in all baths. The processing times in the bath with the smallest average processing time is affected by a weighing factor (WF, greater than 1) in order to emphasize the bath relevance in the complete processing sequence. The underlying rationale is that this bath involves more intensive use of the critical transport system. Afterwards, lots are organized in an increasing order of total processing times.

Having performed such arrangement of baths and lots, the domain pruning, which corresponds to statements in lines 4 to 8, can start. This recursive procedure operates on the domains of the start time variables associated to 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. 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
relaxing the domain of the last addressed variable and then attempting the allocation of a value greater than the previous one. Eventually, the domain reduction procedure succeeds in finding a feasible solution to the problem. The strategy ends when the entire search tree has been explored.

```
search {
    forall(j in J: j<last(J) ordered by decreasing ord(j)) { 
        forall( i in I ordered by increasing TPT[i]){
            try
                PrTask[i,j].start = dmin(PrTask[i,j].start)
            |
                PrTask[i,j].start > dmin(PrTask[i,j].start)
            endtry;
        }
    };
}
```

Figure 2: Search strategy that guides the Variable Domain Reduction procedure (GVDR)

6. Computational results

In this section, four test problems taken from the literature are solved to illustrate the capabilities of the CP approach (model + search strategy). It is worth noting that several search strategies were tested before accepting the one that was described in the previous section. The adopted one presented the best computational time-solution quality relationship. In all cases where GVDR was used, the weighing factor took the following value: \( wf = 1.2 \), which has experimentally proved to be appropriate. The computational performance of the approach was analyzed according to one of the most employed performance measure, the makespan.

Example 1 corresponds to the AWS scheduling problem introduced by Bhushan et al.\(^2\) considering four consecutive baths and eight wafer lots (\(N\times M = [4x8]\)). Case study 2 is a slightly modified version of the previous problem, which was studied by Aguirre and Méndez\(^4\). Thus, while in problem 2 transfer times are 10 times larger than the original case study, processing times are equal. Case study 3 involves twelve consecutive baths and ten wafer lots ([12x10]). Data for this example comprise the twelve consecutive baths and the first ten wafer lots of the example [12x25] introduced by Bushan and Karimi\(^3\). Finally, Problem 4 corresponds to the [12x25] case study introduced by Bushan and Karimi\(^3\). Only one robot is considered in Problems 1 to 4.

Case studies were solved to a maximum time limit of 3600 seconds of CPU by the commercial software ILOG OPL Studio 3.7, based on the ILOG Solver (ILOG, 2003) and Scheduler (ILOG, 2003) packages. The results of the CP model using GVDR strategy are compared with those obtained using our implementation of the ORM (One Robot Model) MILP approach given in Aguirre and Méndez (2010). In order to make the direct comparison of CPU times, the mathematical formulation was implemented in ILOG and solved with Cplex 9.1. In all cases a computer consisting of an AMD Athlon 64 X2 Dual Core 2.2GHz processor with 1 GB of RAM was used.

Table 1: Comparison of results corresponding to problems 1 and 2.

<table>
<thead>
<tr>
<th>MxN</th>
<th>Statistics</th>
<th>Problem 1</th>
<th>Problem 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORM MILP Model</td>
<td>ORM CP Approach</td>
<td>ORM MILP Model</td>
</tr>
<tr>
<td>4x8</td>
<td>Binary Variables</td>
<td>588</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td>Cont. Variables</td>
<td>73</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Constraints</td>
<td>1512</td>
<td>632</td>
</tr>
<tr>
<td></td>
<td>Makespan</td>
<td>84.37</td>
<td>84.37</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>33.16</td>
<td>1.32(^a)</td>
</tr>
</tbody>
</table>

\(^a\) Reaches the imposed time limit (3600 s).

Table 1 shows a comparison between the results of the CP approach and the MILP model considering problems 1 and 2. For both case studies, it should be noted that the proposed CP formulation requires fewer CPU times to obtain the best solutions. For problem 2, this contribution achieved the optimal solution in 1.4 s, in comparison to almost 2.5 min required by the mathematical model. Nevertheless, the CP approach was not able to guarantee the
global optimality of the best solutions generated within the maximum CPU time enforced for these examples. Table 2 presents a comparison for a moderately sized case study (Problem 3). As can be seen, the solution reached with the CP approach is better than the one obtained with the MILP model, since the makespan decreased its value from 206.30 to 199.00.

Table 2: Comparison of results corresponding to problem 3.

<table>
<thead>
<tr>
<th>MxN</th>
<th>Statistics</th>
<th>ORM MILP Model</th>
<th>ORM CP Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>12x10</td>
<td>Variables</td>
<td>7316</td>
<td>811</td>
</tr>
<tr>
<td></td>
<td>Constraints</td>
<td>9768</td>
<td>2300</td>
</tr>
<tr>
<td></td>
<td>Makespan</td>
<td>206.30</td>
<td>199.00</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>3600(^a)</td>
<td>3440(^a)</td>
</tr>
</tbody>
</table>

\(^a\) Reaches the imposed time limit (3600 s).

The best schedule generated for problem 3 is shown in Figure 3. 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.

The scheduling problem of the Automated Wet-etch Stations that manufactures wafer lots has been considered by different contributions, like Bhushan and Karimi\(^3\). These authors solved different case studies, one of them is Problem 4, which involves twelve consecutive baths and twenty five wafer lots NxM=[12x25]. As can be seen, this problem is bigger than case studies 1 to 3. Table 3 presents the computational statistics for case study 4 considering the heuristic approach reported by Bhushan and Karimi\(^3\), the ORM MILP model and the proposed CP formulation. Table 3 shows the results obtained with the A2’ heuristic algorithm, which has the best performance among all algorithms tested by the previous pointed out authors. A2’ is a heuristic algorithm composed of three parts: 1) one

![Figure 3: Gantt diagram depicting the best solution for Example 3](image-url)
sequencing technique based on Tabu Search, 2) one heuristic for initial sequence called nNEH and 3) one scheduling algorithm based on iterative improvements referred as II. An analysis of the results in Table 3 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 seconds, the solution reached by the heuristic algorithm was worse than the one obtained using the CP formulation. In addition, the CP approach achieved a good quality solution (443.4) in short CPU times (8.22 min). Given the obtained results, it is important to remark that the computational performance of the MILP formulations decreased faster than the one of the CP approach.

Table 3: Computational results corresponding to the problem 4 reported by Bhushan and Karimi (2004)

<table>
<thead>
<tr>
<th>MxN</th>
<th>Statistics</th>
<th>A2' Heuristic Bhushan &amp; Karimi³</th>
<th>ORM MILP Model</th>
<th>ORM CP Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>12x25</td>
<td>Variables</td>
<td>No reported</td>
<td>47777</td>
<td>2026</td>
</tr>
<tr>
<td></td>
<td>Constraints</td>
<td>No reported</td>
<td>102051</td>
<td>7675</td>
</tr>
<tr>
<td></td>
<td>Makespan</td>
<td>478.6</td>
<td>-</td>
<td>443.4</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>No reported</td>
<td>-</td>
<td>493.37³</td>
</tr>
</tbody>
</table>

a Reaches the imposed time limit (3600 s).

7. Conclusions
A CP approach for the scheduling problem of Automated Wet-etch Stations that manufactures wafer lots has been presented. The formulation is composed of two parts, a model and a search strategy. The model addresses the scheduling of lots considering the material handling system and strict intermediate storage policies. Therefore, the contributions of this work are a) the development of an integrated model considering at the same time the production and material handling scheduling problems for AWSs and b) the incorporation of an efficient search methodology, which impacts on the computational performance of the CP approach. It is important to note that, in all case studies, the approach had a good computational performance. Despite it could not reach optimal solutions for moderate and large-size case studies within the 3600 seconds of time limit, solutions of good quality were achieved for these. In addition, the proposed CP approach compares favorably with the one of Aguirre and Méndez⁴ as well as Bhushan and Karimi³.

Acknowledgements
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.

References

