AN IMPROVEMENT-BASED MILP OPTIMIZATION APPROACH TO COMPLEX AWS SCHEDULING

[1]Adrián M. Aguirre, [1]Carlos A. Méndez*, [2]Gloria Gutierrez and [2]Cesar De Prada
[1]INTEC (UNL-CONICET), Güemes 3450, 3000 Santa Fe, Argentina
[2]Universidad de Valladolid, c/ Real de Burgos s/n, 47011 Valladolid, Spain

Abstract

The Automated Wet-etch Station (AWS) is one of the most critical stages of a modern semiconductor manufacturing system (SMS), which has to simultaneously deal with many complex constraints and limited resources. Due to its inherent complexity, real-world automated wet-etch station scheduling problems are very difficult to solve using traditional mathematical formulations. Thus, heuristic, meta-heuristics and simulation-based methods have been reported in literature to provide feasible solutions with reasonable CPU times.

This work presents a novel hybrid MILP-based decomposition strategy that combines the benefits of a rigorous MILP (Mixed Integer Linear Programming) continuous-time formulation with the flexibility of dynamic heuristic procedures. The schedule generated provides near-optimal dynamic solutions to challenging industrial-sized automated wet-etch station scheduling problems with moderate computational cost. Also, this methodology provides more than a 10% of improvement in comparison with the best results reported in literature for the most complex problem instances analyzed.

Keywords

Hybrid decomposition approach, MILP-based strategies, Large-scale scheduling problems, Semiconductor Manufacturing System (SMS), Wafer fabrication, Modeling and Optimization.

Introduction

The solution of real-world scheduling problems has greatly attracted the attention of the research and industrial community for many years. In particular, the permutation flowshop sequencing problem (PFSP) is one of the most widely known production scheduling problems in literature, in which a set of jobs *(i=1,2,...,N)* has to be processed on every machine j, following a predefined sequence, as illustrated in Figure 1.

In this kind of problems, each job is performed through a sequence of units *j=1,2,3,...,M*, during a fixed processing time, where every machine *j* can only perform one job at a time, i.e. it is a unary resource where job preemptions are not allowed. Permutation flowshop sequencing problems are usually focused on finding the best processing job sequence that minimizes the completion time of the last job in the schedule, which is widely known as the makespan *(MK)* criterion.

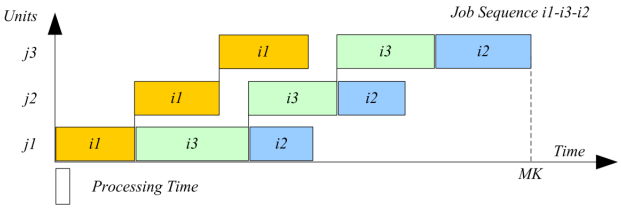

Figure 1. Permutation Flowshop Sequencing Problem Scheme (PFSP)

*email address: cmendez@intec.unl.edu.ar

This work is focusing in a critical process in the semiconductor manufacturing industry, commonly known as Automated Wet-Etch Station (AWS). Flowshop process operations in the AWS are characterized by a major increase in complexity compared to the original PFSP. The automated wet-etch station is a complex operating system in which many process constraints and limited resources must be simultaneously considered. This station involves a series of successive stages of chemical and water baths and shared automated lot transfer devices, which have to be perfectly coordinated in order to ensure that mixed intermediate storage (MIS) policies, such as zero-wait (ZW) and non-intermediate storage (NIS), are strictly satisfied in every bath (Ku and Karimi, 1990). Thus, the efficient and coordinated operation of this stage will reduce the total processing time required to complete all the jobs in the semiconductor production system, providing, at the same time, a better utilization of critical limited resources.

Due to the nature of the binary sequencing decisions needed to achieve the optimal solution, this kind of flowshop sequencing problems becomes an inherently NP-hard problem. Moreover, industrial-scale AWS scheduling problems have been difficult to solve up to optimality in a reasonable computational time using traditional solvers and methods. Consequently, the application of exact methods, such as Mixed Integer Linear Programming (MILP) approaches, has become prohibitive for these kinds of problems, and its applicability has been restricted to relatively small-sized cases (see Ruiz and Maroto, 2005 and Méndez et al., 2006).

Over the past years, different contributions have been reported to achieve suitable solutions to challenging AWS scheduling problems with an acceptable computational effort. Most of these approaches were based on heuristic and meta-heuristic methodologies (Ruiz and Maroto, 2005), avoiding the utilization of mixed integer mathematical formulations in large-sized cases. Geiger et al. (1997) were the first ones in developing a heuristic algorithm based on Tabu Search (TS), with the main goal of finding a near-optimal production schedule to the AWS. Subsequently, Bhushan and Karimi (2003) introduced a slot-based MILP mathematical model to solve short-term scheduling problems, minimizing the makespan in an AWS station. Later, Bhushan and Karimi (2004) evaluated the proper combination of different algorithms and meta-heuristics such as Simulated Annealing (SA) and Geiger's Tabu Search procedures to identify the best strategy for optimizing operations in the AWS. Most recent approaches to address the resource-constrained flowshop scheduling problem of the AWS were developed based on a Constraint Programming (CP) formulation (Zeballos, Castro and Méndez, 2011) and a MILP continuous-time mathematical model (Aguirre, Méndez and Castro, 2011). The former contribution combines a CP model that handles the complex production restrictions imposed by AWS station with an efficient domain-search strategy to reduce the high combinatorial complexity of the model. The latter approach relies on a robust MILP model for the scheduling of AWS operations (Aguirre, Méndez and Castro, 2011). This method provides an optimal solution for job sequence and timing and a detailed activity program for the robot with the aim to minimize the makespan criterion.

In order to exploit the inherent benefits of these techniques and the robustness of exact methods, a novel hybrid two-stage MILP-based decomposition strategy is proposed here to provide an efficient and near-optimal solution to complex industrial-scale flowshop scheduling problems with modest computational effort. The solution strategy combines heuristic techniques with a rigorous mathematical formulation, making it possible to generate and gradually improve a solution, in an iterative way. This methodology joins the advantages of a rigorous MILP continuous-time formulation with the flexibility of heuristic procedures, such as decomposition-aggregation and improvement-based techniques. Additional information of these methods are presented in Méndez et al. (2006) and in Castro et al. (2009).

Problem Statement

The automated wet-etch operation is carried out in very complex station in the wafer fabrication process of semiconductor manufacturing industries. It performs the cleaning of semiconductor wafer lots by using a sequence of immersion baths. The automated wet-etch station is composed by several types of successive immersion baths *"j"*, chemical (j_{odd}) and water baths (j_{even}) arranged in an alternate way. In chemical baths $(j_{odd}=1,3,5,\ldots,M-1)$, wafers are etched during a predefined period of time to then being purged in the de-ionized water in rinse baths $(j_{even}=2,4,6,\ldots,M)$. The exposure time on chemical baths must be fulfilled strictly, because wafers may be easily damaged by the overexposure to reagents. In water baths, a minimum residence time is only enforced. This kind of operational constraints are commonly known as Zero-Wait (ZW) and Local Storage (LS) policies (Ku and Karimi, 1990; Méndez et al., 2006). Thus, AWS provides a complex manufacturing environment in which Mixed Intermediate Storage policies (MIS) must be ensured (Bhushan and Karimi, 2003).

This station may have different lineal configurations of immersion baths. The structure of the common AWS is composed by an Input Buffer $(j=0)$ and an Output Buffer $(j=M+1)$ in both extreme points and a series of immersion baths are arranged between these points. These baths are typically structured in a row in which the first bath is chemical and the last one is a water bath (see Figure 2).

In addition, automated material-handling devices, like robots, are used as shared resources for transferring wafer lots between consecutive baths (Aguirre, Méndez and Castro, 2011). The scheduling problem of these robots in the AWS has adopted increasing importance in real-life operations in recent years. Strict conditions must be enforced to provide a feasible activity program of each

robot in the system: a) every robot should be used as a unary shared resource for transferring lots between baths; b) each transfer task has a fixed transportation time and the robots cannot delay the delivery task more than this known time; c) robots only can transfer one wafer lot at a time, to guarantee the non-intermediate storage policy (NIS) followed in the system and; d) robots collisions and deadlocks are undesirable conditions that have to be avoided in the system.

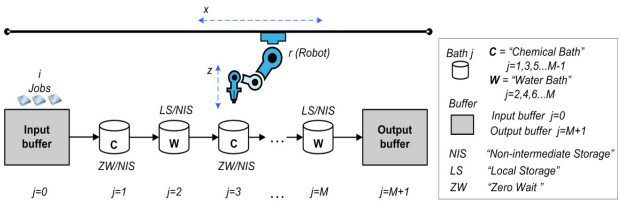

Figure 2. Automated Wet-etch Station (AWS) process scheme

The problem to be faced in this work corresponds to the scheduling of a set of wafer lots or jobs, from $i=1,2,...,N$, in M stages or baths, in a serial flowshop multiproduct batch process with strict storage policies $(SP_j = NIS,LS,ZW)$ in every stage and a single shared resource (r = Robot) with finite capacity for the material movement, as depicted in the Gantt Chart shown in Fig. 3.

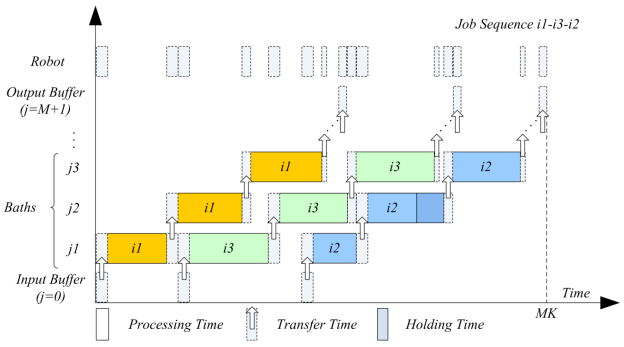

Figure 3. Flowshop AWS scheduling problem for N jobs in M baths in a single robot

Next section describes the development and the application of a hybrid MILP-based decomposition strategy to the AWS scheduling problem in industrial applications. In turn, the best job sequence and timing for the processing operations as well as the detailed pick-up and delivery activity program for the robot are determined with the main objective of minimizing the total time required to perform all the wafer lots in the system *(MK)*.

The MILP mathematical Model

In this section, a modified version of a previous continuous-time formulation developed by Aguirre, Méndez and Castro (2011), which relies on the general precedence concept is introduced. The model used here incorporates additional constraints that permit to decompose the whole problem into different independents sub-problems that can be solved in a sequential manner. Each sub-problem corresponds to a simplified scheduling problem, where a proper MILP formulation is developed for decision-making on timing (Eqs. 1-4), job sequencing (Eqs. 5-6) and transfer operations (Eqs. 7-8). Specific binary variables $(X_{(i,i')}$ and $Y_{(i,j,i',j')})$ for job's sequencing and transfer sequencing decisions and continuous variables $(Ts_{(i,j)}, Tf_{(i,j)})$ for the start time and the completion time of a job i in a bath j, are used in the model to achieve a correct synchronization of processing and transportation activities. Timing information for these activities are provided by parameters $t_{(i,j)}$ and π_j, which represent the processing time of job i in a bath j and the transfer time of every job defined between consecutive baths j-1 and j. Also, big-M constraints are represented with a large parameter M_T in equations (5) to (8).

Job's and transfer's sequencing-timing constraints for the full-space AWS scheduling problem (Aguirre et al., 2011)

$$Tf_{(i,j)} = Ts_{(i,j)} + t_{(i,j)} \quad \forall i = 1...N, \forall j \in J_{odd} \;, SP_j = ZW \tag{1}$$

$$Tf_{(i,j)} \geq Ts_{(i,j)} + t_{(i,j)} \quad \forall i = 1...N, \forall j \in J_{even} \;, SP_j = LS \tag{2}$$

$$Ts_{(i,j)} = Tf_{(i,j-1)} + \pi_j \quad \forall i = 1...N, \forall j \in J : (j > 1) \;, SP_j = NIS \tag{3}$$

$$Ts_{(i,j)} \geq \pi_j \qquad \forall i = 1...N, j = 1 \tag{4}$$

$$Ts_{(i,j)} \geq Ts_{(i',j+1)} + \pi_j - M_T(1 - X_{(i,i')}) \quad \forall i,i' \in I : (i > i'), \forall j \in J \tag{5}$$

$$Ts_{(i',j)} \geq Ts_{(i,j+1)} + \pi_j - M_T X_{(i,i')} \quad \forall i,i' \in I : (i > i'), \forall j \in J \tag{6}$$

$$Ts_{(i',j')} \leq Ts_{(i,j)} - \pi_j + M_T(1 - Y_{(i,j,i',j')}) \quad \forall i,i' \in I : (i > i'), \forall j,j' \in J : (j'-1 > j \vee j > j'+1) \tag{7}$$

$$Ts_{(i,j)} \leq Ts_{(i',j')} - \pi_{j'} + M_T Y_{(i,j,i',j')} \quad \forall i,i' \in I : (i > i'), \forall j,j' \in J : (j'-1 > j \vee j > j'+1) \tag{8}$$

Additional timing constraints for partial predefined processing sequence.

Let's suppose that the position in the processing sequence of a given job i is known beforehand and assume that *Position(i)* defines the order of a job i in the current processing sequence. Then, we can predefine and fix the value of the sequencing variables X related to every pair of jobs that have been already sequenced in the system. The

corresponding value of the sequencing variable $X_{(i,i')}$ can be easily determined as indicated in equation (9).

$$X_{(i,i')} = \begin{cases} 1 & if\ [Position\ (i') < Position\ (i)] \\ 0 & if\ [Position\ (i') > Position\ (i)] \\ 0-1 & otherwise \end{cases} \quad (9)$$

Therefore, if jobs *i* and *i'* are already scheduled and *Position(i') < Position(i),* job *i'* will be processed before job *i* in the production sequence, i.e. $X_{(i,i')}=1$ in equations (5) and (6). On the other hand, if *Position(i') > Position(i)*, then $X_{(i,i')}=0$ and job *i'* will be executed after job i in the processing sequence. Anyway, if we do not know a prior information about the position of jobs *i* and *i'* in the job's sequence then variable $X_{(i,i')}$ will adopt values 0-1 depending on the sequencing and timing decisions of the model provided in equations (5) and (6).

Additional sequencing transfers constraints for the partial reduction of the problem size.

It is easy to demonstrate that not all pair of transfers needs to be explicitly sequenced in the system. For example, for pair of transfers *(i,j)-(i',j')* and *(i,j)-(i'',j'')* that satisfies the condition stating that *Position(i') > Position(i)* in the recipe order and $j' \geq j+1$ or $j-[Position(i')-Position(i)] \leq j''$, then we can assure that transfer *(i,j)* will be always executed before than both transfers *(i',j')* and *(i',j'')* respectively. In contrast, for transfers *(i,j)-(i'',j')* and *(i,j)-(i'',j''),* if *Position(i'') < Position(i)* in the recipe order and $j+[Position(i)-Position(i'')] \geq j'$ or $j'' \leq j-1$, then transfers *(i,j)* will be executed always after transfer *(i'',j')* and *(i'',j'')* in the transfer sequence (see Fig. 4). However, if any of these cases are applied then the value of the sequencing transfer variable $Y_{(i,j,i',j')}$ will be determined by the model in equations (7) and (8) by adopting values 0-1 respectively. In consequence, the behavior of the variable $Y_{(i,j,i',j')}$ for any pair of transfer *(i,j)* and *(i',j'),* can be easily predetermined as stated by equation (10).

$$Y_{(i,j,i',j')} = \begin{cases} 1 & if\ [Position\,(i') < Position\,(i)] \wedge [(j + (Position(i) - Position(i')) \geq j') \vee (j' \leq j-1)] \\ 0 & if\ [Position\,(i') > Position\,(i)] \wedge [(j - (Position(i') - Position(i)) \leq j') \vee (j' \geq j+1)] \\ 0-1 & otherwise \end{cases} \quad (10)$$

To incorporate the knowledge about transfer sequencing variables in the model it is only necessary to add the equation expressed below (Eq. 11) in the domain of equations (7) and (8) related to every pair of transfers satisfying the conditions stated above (see Eqs. 12-13).

$$V_{(i,j,i',j')} = \begin{cases} 0 & [Conditions\ in\ Eq.(10)] \\ 1 & otherwise \end{cases} \quad (11)$$

$$Ts_{(i',j')} \leq Ts_{(i,j)} - \pi_j + M_T(1 - Y_{(i,j,i',j')}) \quad \forall i,i' \in I : (i > i'), \forall j,j' \in J, V_{(i,j,i',j')} = 1 \quad (12)$$

$$Ts_{(i,j)} \leq Ts_{(i',j')} - \pi_{j'} + M_T Y_{(i,j,i',j')} \quad \forall i,i' \in I : (i > i'), \forall j,j' \in J, V_{(i,j,i',j')} = 1 \quad (13)$$

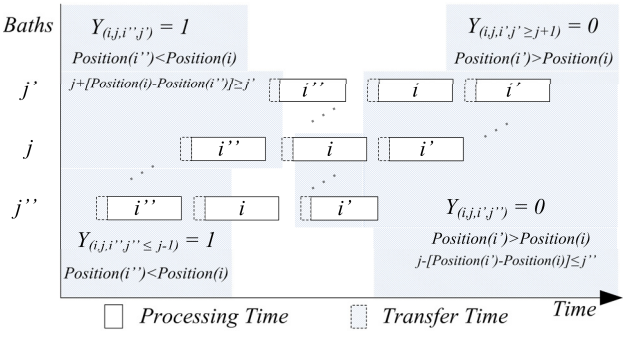

Figure 4. Pre-sequenced transfers decisions

Also, another way to reduce much more the search space of the problem without losing good quality solutions is trying to fix certain transfer decisions, of the already inserted jobs in the system, by applying the following expression (see Eq. 14). As it is presented before this condition can be incorporated into the model in equations (12) and (13) by the parameter $V_{(i,j,i',j')}$ expressed in equation (15). Thus, $tts_{(i,j)}$ provides an additional information to the model representing the initial time of the already inserted transfer *(i,j)* in the system.

$$Y_{(i,j,i',j')} = \begin{cases} 1 & if\ [Position\ (i') < Position\ (i)] \wedge [(j + (Position\,(i) - Position\,(i')) < j') \wedge (tts_{(i,j)} > tts_{(i',j')})] \\ 0 & if\ [Position\ (i') > Position\ (i)] \wedge [(j - (Position\,(i') - Position\,(i)) > j') \wedge (tts_{(i,j)} < tts_{(i',j')})] \\ 0-1 & otherwise \end{cases} \quad (14)$$

$$V_{(i,j,i',j')} = \begin{cases} 0 & [Conditions\ in\ Eq.(14)] \\ 1 & otherwise \end{cases} \quad (15)$$

Last equations demonstrate that it is possible to reduce the combinatorial complexity of the whole system, eliminating unnecessary model decisions that can be defined beforehand. But, we can also improve the model convergence by exploiting the particular problem structure without any prior knowledge about the order of jobs in the system. This is done by combining the sequencing variable $X_{(i,i')}$ for jobs with the sequencing variable $Y_{(i,j,i',j')}$ for transfers in one expression (see Eqs. 16-17).

$$Y_{(i,j,i',j')} \leq X_{(i,i')} \quad \forall i,i' \in I : (i > i'), \forall j,j' \in J : (j' > j+1) \quad (16)$$

$$Y_{(i,j,i',j')} \geq X_{(i,i')} \quad \forall i,i' \in I : (i > i'), \forall j,j' \in J : (j' < j-1) \quad (17)$$

Objective Function.

The objective function aims at generating a schedule with the minimum total time required to complete all the jobs in the system. This criterion is defined by a continuous variable called makespan *(MK)* (see Eq. 18).

$$MK \geq Ts_{(i,j)} \qquad \forall i \in I, j = M+1 \quad (18)$$

Hybrid MILP-based decomposition strategy

The procedure introduced in this work relies on the MILP model developed by Aguirre, Méndez and Castro (2011), the dynamic and reactive scheduling methods and the heuristic reduction techniques, such as decomposition-aggregation techniques and improvement-based techniques, summarized in a review paper presented by Méndez et al. (2006) and in Castro et al. (2009). Also, some concepts presented in Kopanos et al. (2010) regarding a two-stage decomposition solution strategy for solving real-world scheduling problems in multi-product multi-stage batch process, are also considered in our work.

The proposed hybrid algorithm aims towards achieving the whole solution of the original AWS scheduling problem in a sequential manner. In this procedure, a heuristic-based two-stage decomposition strategy divides the whole problem into two sub-problems that must be iteratively solved. In the first stage, the best permutation job sequence p is determined, ignoring the limited transportation resources, whereas in the second stage the job's transfers schedule is provided in order to generate the detailed activity program for the robot, based on the previous predefined production sequence (Fig. 5).

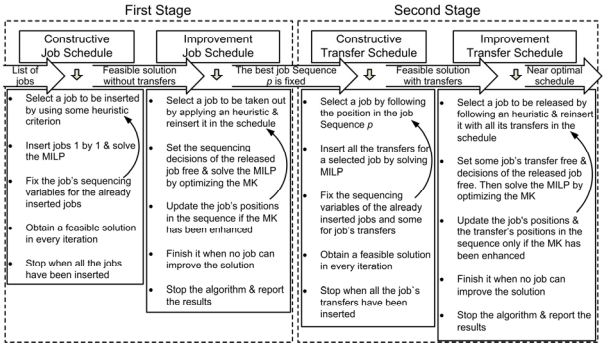

Figure 5. Hybrid decomposition strategy

Thus, a reduced size problem is faced in every stage by using a multi-stage optimization-based decomposition strategy. Therefore, the proposed solution strategy consists on a constructive step followed by a local improvement step, in where a proper continuous-time MILP formulation is developed to reach an optimal schedule solution in each stage.

The constructive step method is a decomposition-aggregation technique, in which an initial solution is generated in an iterative way by inserting, one-by-one, the jobs from the set of non-sequenced jobs. In the constructive step, a feasible production schedule p is generated in an iterative form, by selecting one job at a time by following a heuristic criterion. The insertion of the selected job is driven by the *MK* criterion; taking into account the already inserted jobs. For this, a reduced MILP model is solved by fixing the sequencing decisions for the already inserted jobs in the system (see Fig. 5 and Table 1). After the constructive step, a feasible solution of the production schedule of the system is provided.

In the improvement step, the feasible solution previously obtained is progressively enhanced by selecting a job to be taken out from the schedule, to be then reinserted in the schedule of the remaining non-released jobs. A suitable MILP is solved by choosing the job to be rescheduled in each iteration (see Fig. 5 and Table 1).

Table 1. MILP-based models for every step of the solution strategy proposed

STEP	MILP-based Model
Constructive Job Schedule	Equations (1-6, 9 & 18)
Improvement Job Schedule	Equations (1-6, 9 & 18)
Constructive Transfer Schedule	Equations (1-15 & 18)
Improvement Transfer Schedule	Equations (1-18)

Results

In this section, some different examples are proposed to illustrate the behaviour of the hybrid solution strategy proposed above. These examples are derived from data provided by Bhushan and Karimi (2004), also appeared in Aguirre, Méndez and Castro (2011), regarding processing and transfer times. The problems tested are from different *MxN* configuration of the first *M* baths and *N* jobs in the AWS station. These settings determine the problem size, which is closely linked with the model structure, according to all decisions and restrictions involved (see Table 2).

Table 2. Comparison of different models statistics in real-life problem instances

Baths x Jobs	ORM-MILP				ORM-CP			BK	RCURM-MILP				ORM-Hybrid	
MxN	BV	CV	MK	CPUs	CV	MK	CPUs	MK	BV	CV	MK	CPUs	MK	CPUs
12x8	4396	201	170.6[a]	180	-	-	-	-	4368	229	171.3[a]	1.4	170.6[a]	13.61
12x10	7056	251	202.2[a]	3600[d]	811	199[b]	3440[d]	-	7020	296	197.2[a]	7.8	198.2[a]	24.42
12x12	10362	301	222[a]	3600[d]	-	-	-	-	10296	367	215.8[a]	2655	222.6[a]	100.8
12x15	16485	376	NFS[a]	3600[d]	1216	273.2[b]	949[d]	-	16380	481	NFS[a]	3600[d]	268.2[a]	866.8
12x20	29830	501	NFS[a]	3600[d]	-	-	-	-	29640	691	NFS[a]	3600[d]	330.6[a]	3600[d]
12x25	47100	626	NFS[a]	3600[d]	2026	443.4[b]	493.37[d]	478.6[c]	46800	926	NFS[a]	3600[d]	396.3[a]	3600[d]

BV = Binary Variables. CV = Continuous Variables. MK = Makespan. NFS = No feasible solution found. CPUs = Computational time in seconds. Results obtained by using: (a) GAMS with Gurobi 6.0 in an Intel PC Core 2 Quad parallel processing in 4 threads, (b) ILOG with CPLEX 9.1 in an AMD Athlon 64 X2 Dual Core 2.2GHz processor with 1GB of RAM. (c) No time reported. (d) Termination criterion (3600sec.).

Table 2 describes the principal results obtained by different solution approaches for many real-life scheduling problems presented above. The comparison comprises an full-space mathematical formulation (MILP) for the entire problem ORM (One Robot Model), a two-stage MILP-based approach called RCURM (Resource Constrained Unlimited Robot Model) and the ORM model by using both Constraint Programming procedure (CP) from Zeballos et al., (2011) and our Hybrid MILP-based solution strategy. The RCURM approach was introduced before by Bhushan and Karimi (2003). This two-stage procedure is based on: i) solve the first model, called URM (Unlimited Robot Model), for the original problem without robot restrictions and then ii) solve the second model, which was named ORM (One Robot Model), by introducing robot restrictions in a single robot. The solutions reported in Table 2 demonstrate that our proposed approach provides better results than full-space ORM (MILP/CP) models and RCURM-MILP procedure with shortest CPU time.

For industrial-sized cases, the exact methods turned out to be unmanageable due to the high number of binary decisions. So, for this kind of problems, both the heuristic-based approaches and the hybrid approaches are needed.

In the last case, the solution of the heuristic approach developed by Bhushan and Karimi (2004), here named BK, has been improved by the CP approach (Zeballos et al., 2011) and also by the hybrid solution strategy proposed in this work. The solution found by the CP approach is better than the best solution in 7.4%. Finally, our solution approach can even improve the CP solution of 443.4 units, obtaining an important reduction in the MK (>10%) with an acceptable computational effort (3600sec.). As a conclusion, our method is comparable with the best approaches existing in literature, providing even better results for large-size problems with a reasonable computational cost.

Conclusions

A novel hybrid decomposition strategy based on a MILP continuous-time formulation has been presented to solve large-scale scheduling problems arising in AWS of the semiconductor industry. In contrast to typical scheduling solution techniques, this strategy lies on an exact method to sequentially generate and improve a detailed schedule of production activities and transfer operations, assuring the stringent intermediate storage policies of the system.

Furthermore, it has been demonstrated that the proposed model can effectively solve the whole problem by using the proposed MILP-based decomposition technique, in which the entire problem is divided into two sub-problems and the results of the first one are used to solve the second one in a sequential manner. In this case, the schedule of production activities is first generated to then incorporate the detailed schedule of transfer operations, fixing the previously defined production sequence. Also, it has been clearly illustrated that each sub-problem can be solved with a two-stage decomposition strategy, where the solution is generated, and then gradually improved, in an iterative way, by using a simplified MILP-based algorithm in each stage.

The main contribution of this work is to provide a robust and, at the same time, very flexible strategy for the solution of large-scale problems in a sequential way, by using a proper combination of heuristic procedures and exact mathematical formulations. This methodology will be able to face complex operative decisions on multi-product multi-stage industrial processes with shared resources and complex production constraints.

Acknowledgments

Financial support received from AECID under Grant PCI-D/030927/10 and from UNL under Grant PI-66-337 is fully appreciated.

References

Aguirre, A. M., Méndez, C. A., and Castro, P. M. (2011). A Novel Optimization Method to Automated Wet-Etch Station Scheduling in Semiconductor Manufacturing Systems. *Computer Chemical Engineering*, *In Press.*

Bhushan, S., and Karimi, I.A. (2003). An MILP approach to automated wet-etch station scheduling. *Industrial and Engineering Chemistry Research*, 42(7), 1391-1399.

Bhushan, S., and Karimi, I.A. (2004). Heuristic algorithms for scheduling an automated wet-etch station. *Computers and Chemical Engineering,* 28(3), 363–379.

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

Geiger, C.D., Kempf, K.G., Uzsoy, R. (1997). A tabu search approach to scheduling an automated wet etch station. *Journal of Manufacturing Systems*, 16(2), 102-116.

Kopanos, G.M., Méndez, C.A., Puigjaner, L. (2010). MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. *European Journal of Oper. Res.,*207,644-655

Ku, H., and Karimi, I.A. (1990). Completion time algorithm for serial multiproduct batch processes with shared storage. *Computers and Chemical Engineering*, 14, 49-69.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering*, 30, 913.

Ruiz, R. and Maroto C.A. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Res.*, 165, 479-494.

Zeballos, L. J., Castro, P. M., Méndez, C. A. (2011). An integrated constraint programming scheduling approach for automated wet-etch stations in semiconductor manufacturing. *Ind. and Eng. Chem. Res.*, 50, 1705-1715.