Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

ANALYZING DIFFERENT DISPATCHING POLICIES FOR PROBABILITY ESTIMATION IN TIME CONSTRAINT TUNNELS IN SEMICONDUCTOR MANUFACTURING

Alexandre Lima, Valeria Borodin,
Stéphane Dauzère-Pérès

Department of Manufacturing Sciences and Logistics
CMP, Ecole des Mines de Saint-Etienne,
CNRS UMR 6158 LIMOS,
F-13541 Gardanne, FRANCE

Philippe Vialletelle

Advanced Engineering Methods Department
STMicroelectronics
F-38926 Crolles, FRANCE

ABSTRACT

In semiconductor manufacturing, new technologies impose more and more time constraints in product routes, i.e. a maximum time between two (often non-consecutive) operations. The management of Time Constraint Tunnels (TCTs, combining multiple time constraints) in high-mix facilities is becoming more and more challenging. This paper first recalls an approach for estimating the probability that a lot at the entrance of a TCT will leave the TCT on time. This approach relies on a list scheduling algorithm using a dispatching policy with random components. Three dispatching policies are presented. Computational experiments on industrial data comparing these policies are discussed. Perspectives are drawn to extend the approach and support decision making.

1 INTRODUCTION

In semiconductor manufacturing, the last twenty years have been governed by Moore's law. New challenges arise that make decreasing chips in size less and less cost-effective. Maintaining a consistent yield is one of the main challenges. The yield of a wafer corresponds to the percentage of functional chips. To maintain high yield rates, Time Constraints (TCs) are defined between various process steps. A TC has a start step, an end step and a time limit that should not be exceeded in order for the wafers to retain expected physical and chemical properties. With the increase of the density of integration in products, the number of time constraints per product tends to increase exponentially. TCs often follow each other in close succession and overlap to the point where they start forming Time Constraint Tunnels (TCTs) (cf. Figure 1). In the complex context of semiconductor manufacturing, managing TCTs turns out to be very difficult.

In the wafer fabrication facility under study, whilst lots are handled automatically once they enter a TC or a TCT, their release within tunnels is managed by humans. The entrance of each tunnel is regulated like a toll i.e. opened or closed depending on whether there is enough capacity to process lots in time within the tunnel. Defining what *enough capacity* is, however, a hard question. As of now, it is mostly done by hand with simplistic throughput estimates, and is overall time consuming and very stressful for the operators managing TCTs.

The content of this paper is structured as follows. The next section summarizes the state-of-the-art related to TCT management. Section 3 details the problem statement and recalls the existing probability estimation approach. Three dispatching policies are presented in Section 4. Extensive computational experiments are conducted and discussed in Section 5. Finally, the paper ends with some concluding remarks including an outline of topics for future research.

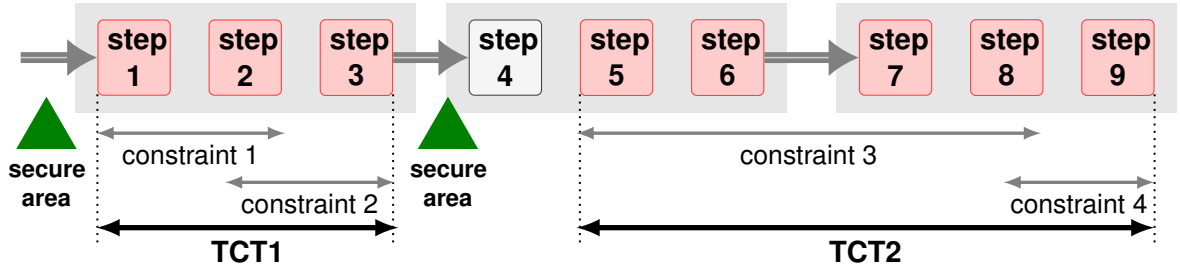

Figure 1: Schematic representation of successive TCTs.

2 LITERATURE REVIEW

A simple industrial solution to the problem would be to set up regulatory approaches based on Work-In-Process levels allowed in a tunnel i.e. to define a *tunnel capacity*. This might even give decent results in "linear" facilities running few product routes with medium to large chip pattern sizes. However, the overall cycle time variability often encountered in semiconductor plants, associated in particular to re-entrant flows and a High-Mix Low Volume context, makes this kind of approaches difficult to implement if not unfeasible.

The problem of TCT management in semiconductor manufacturing can most notably be modeled as a flexible job-shop scheduling problem with multiple overlapping time windows. While the industrial problems raised by queue times or time constraints have gathered increasing interest over the last few years, for topics ranging from predictive maintenance planning (Kalir and Tirkel 2016) to reentrant flow modeling and robust dispatching rule determination (Arima, Kobayashi, Wang, Sakurai, and Monma 2015), real-time oriented scheduling solutions to the problem have to the best of our knowledge seldom been covered in the literature without at least some modifications. As a matter of fact, we have not restricted our search to this specific problem, but broadened it to include flexible flow-shop scheduling problems as well, such as (Attar, Mohammadi, Tavakkoli-Moghaddam, and Yaghoubi 2014, Attar, Mohammadi, and Tavakkoli-Moghaddam 2013, Chen and Yang 2006, Fondrevelle, Oulamara, and Portmann 2008). Time constraint modeling varies greatly between publications but the most frequent hypothesis is that a time constraint cannot span over more than two consecutive steps and cannot overlap, as in (Attar, Mohammadi, Tavakkoli-Moghaddam, and Yaghoubi 2014, Chen and Yang 2006, Chien and Chen 2007, Fondrevelle, Oulamara, and Portmann 2008, Sun, Choung, Lee, and Jang 2005). This restriction is not present in (Attar, Mohammadi, and Tavakkoli-Moghaddam 2013, Kalir and Tirkel 2016), but overlapping is not taken into account either. Only (Sadeghi, Dauzère-Pérès, Yugma, and Lepelletier 2015), (Klemmt and Mönch 2012), (Arima, Kobayashi, Wang, Sakurai, and Monma 2015) and (Zhou and Wu 2017) propose models where time constraint can both span more than two consecutive steps and can overlap. This modeling better suits the way time constraints are defined in the semiconductor industry.

Most of the publications are focused on solving the problem for rather small instances, usually up to two or three processing tools and small subsets of operations, especially compared with the reality of industrial instances (Attar, Mohammadi, Tavakkoli-Moghaddam, and Yaghoubi 2014, Attar, Mohammadi, and Tavakkoli-Moghaddam 2013, Fondrevelle, Oulamara, and Portmann 2008, Sun, Choung, Lee, and Jang 2005). It is also noteworthy that, in these publications, executions times are most often too long to be put in practice in a production context. There has been a specific coverage of the diffusion area for this problem as in (Chien and Chen 2007, Yugma, Dauzere-Péres, Artigues, Derreumaux, and Sibille 2012, Jung, Pabst, Ham, Stehli, and Rothe 2014, Zhou and Wu 2017, Kalir and Tirkel 2016). This is most likely explained by the fact that the processes involved in this work area are among the most sensitive to physical phenomena. They also involve a wide variety of complex batching tools with very long process times that further complicate the management of time constraints.

From a methodological point of view, the most common solution approaches can be found in the literature, such as integer programming in (Chen and Yang 2006, Chien and Chen 2007, Jung, Pabst, Ham,

Stehli, and Rothe 2014, Kalir and Tirkel 2016). While population based meta-heuristic are also frequently found in the form of genetic algorithms (Chien and Chen 2007), particle swarm (Attar, Mohammadi, Tavakkoli-Moghaddam, and Yaghoubi 2014), geography based optimization (Attar, Mohammadi, and Tavakkoli-Moghaddam 2013) or cross entropy genetic algorithms (Kalir and Tirkel 2016) and tend to provide high quality results, they also offer some of the worst execution times. These traits are also shared by exact resolution methods such as branch and bound (Fondrevelle, Oulamara, and Portmann 2008) whose computational times are prohibitive and not applicable to large industrial instances. However, simpler list based scheduling heuristic (Sadeghi, Dauzère-Pérès, Yugma, and Lepelletier 2015, Klemmt and Mönch 2012) offer an interesting compromise between solution quality and scalability and do not limit the way time constraints are modeled. Note that more recent heuristic approaches proposed in (Knopp, Dauzère-Pérès, and Yugma 2017) and (Zhou and Wu 2017) provide excellent results in rather short execution times for the specific case of batching in the diffusion area. In the light of the above background, the aim of this paper is to contribute to the state of the art in this direction.

3 PROBLEM DESCRIPTION

In order to prevent wafer scraps or reworking, we propose to control lots before they enter a TCT by an approach that estimates the probability that a lot will leave the TCT on time. As proposed in (Sadeghi, Dauzère-Pérès, Yugma, and Lepelletier 2015), a disjunctive graph representation is used to determine the completion time of a specific lot, by considering all relevant information for a given TCT, at a given point in time, such as Work In Process (WIP), status and qualification of tools, process times, etc. The probability of respecting each time constraint is then estimated by replicating a fast list scheduling algorithm.

For a given TCT at a given point in time, let $\mathscr{L} = \{1,\ldots,L\}$ be the set of lots of wafers to be processed. Each lot $l \in \mathscr{L}$ has a sequence of N^l operations $\mathscr{J}^l = (j^l_1,\ldots,j^l_{N^l})$ (route of the job) to complete. Operations in $\mathscr{J}^l$ have to be performed in succession, i.e. there is a precedence constraint between operations j^l_k and j^l_{k+1}, $\forall k \in \{1,\ldots,N^l-1\}$. Each operation j^l_k can be performed on a subset of tools $\mathscr{M}^l_k \subseteq \mathscr{M}$, where $\mathscr{M}$ is the set of all tools, and has associated machine dependent process times p^l_{km}, $\forall m \in \mathscr{M}^l_k$.

Let us introduce a fictitious lot l^F that needs to enter the TCT, and thus has a set $\mathscr{T}$ of time constraints to respect. Time Constraint $t_{k1,k2} \in \mathscr{T}$, where $k1 < k2$, starts at the beginning of operation j^F_{k1} and ends when operation j^F_{k2} is completed. TCs follow each other in close succession and can overlap, i.e. $\exists(k1,k2,k3,k4)$ such that $t_{k1,k2} \in \mathscr{T}$, $t_{k3,k4} \in \mathscr{T}$, $k1 < k3$ and $k3 < k2$.

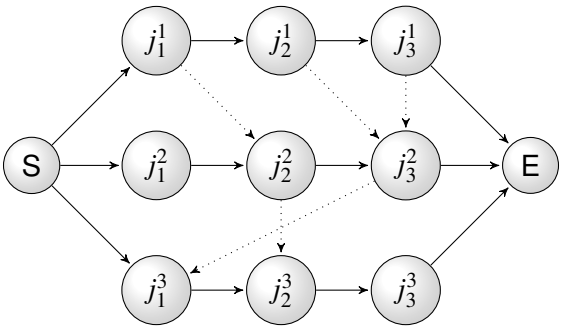

Figure 2: Conjunctive graph representation. Conjunctive arcs are plain for the routes, and dotted for the sequence on machines.

Let us model the sequence of lots via a classical disjunctive graph $G = (\mathscr{N},\mathscr{C},\mathscr{D})$. The set of nodes $\mathscr{N}$ includes the set of all operations $\mathscr{J}$ of the lots $\mathscr{L} \cup \{l^F\}$, and two fictitious start and end nodes S and E. $\mathscr{C}$ is the set of conjunctive arcs between the nodes, which ensure the precedence constraints between every two consecutive operations. $\mathscr{D}$ is the set of disjunctive arcs modeling the disjunctive constraints between operations that can be processed on the same tool.

Once a sequence is defined (cf. Figure 2), each arc can be labeled with the processing time of the start node. Recall that processing times are machine dependent. The minimum time between two nodes is equal to the longest path between these nodes. If there is a time constraint between two nodes, computing the minimum time between the start and end nodes of the TC allows us to check if the corresponding lot has exceeded the TC. In order to evaluate time constraints, a large number N of conjunctive graphs are randomly generated from a single disjunctive graph using a list scheduling algorithm with a randomized dispatching policy. The number of generations is determined by a convergence based criterion, used as a stopping rule in the proposed approach. Once the generations are completed, the number of times that the fictitious lot l^F verifies its time constraints is calculated. Since calculations are only performed *a posteriori*, this approach emphasizes the dispatching policy. In the next sections, three randomized dispatching policies are introduced and discussed.

4 DISPATCHING POLICIES

4.1 Pure random policy

Based on both disjunctive and conjunctive graphs, the list scheduling algorithm 1 introduced in (Sadeghi, Dauzère-Pérès, Yugma, and Lepelletier 2015) provides a realistic and feasible schedule. It originally relies on a pure random dispatching policy, used to generate a great number of schedules. In order to estimate the probability that a lot l^F satisfies each time constraint of a given TCT, the above procedure is ran N times. Let C^F_{k1} and C^F_{k2} be the completion times of operations j^F_{k1} and j^F_{k2}. Let us denote by $C^F_{k2} - c^F_{k1}$ the cycle time of l^F between its $k1^{th}$ and $k2^{th}$ operations. After each run, the value of the following logic expression is calculated:

$$C_{k2} - C_{k1} \leq t_{k1,k2}, \quad \forall t_{k1,k2} \in \mathscr{T} \tag{1}$$

Let $count_{k1,k2}$ be the number of times that the logic expression (1) is true. Following (Sadeghi, Dauzère-Pérès, Yugma, and Lepelletier 2015), the probability of satisfying TC $t_{k1,k2} \in \mathscr{T}$ is determined by:

$$\hat{P}(C_{k2} - C_{k1} \leq t_{k1,k2}) = \frac{count_{k1,k2}}{N}, \quad \forall t_{k1,k2} \in \mathscr{T} \tag{2}$$

This approach heavily relies on the law of large number and thus requires a large number of iterations N to be effective. From an industrial standpoint, it has however some notable drawbacks that we aimed to improve with new dispatching policies.

4.2 Extension to a random priority based dispatching policy

Pure randomness is not a faithful representation of the wafer fabrication reality. Lots are not randomly assigned to a tool, but actually have to respect a large set of rules, known as *dispatching rules*. In practice, these rules mostly revolve around a priority system. A base priority is assigned to each lot at the beginning of its product route. This base priority depends on different factors such as due date, importance of the original order, whether or not the lot is a prototype lot or an engineering lot, etc. While this base priority cannot change, the total priority is actually influenced by several factors including whether or not the lot has entered a TCT, and how long it has been waiting at an operation. This total priority can also be manually changed, albeit temporarily, in order to catch up with delays or release the necessary capacity of some tools, etc. Lots are sorted by their total priority so that the one with the highest priority will always be queued up first in front of the machines on which it can be processed. Machine selection also has its own rules, but for the sake of simplicity, let us assume an arbitrary selection rule. Besides, the priority gained by lots waiting at an operation also depends on the corresponding work area. Since there are many heterogeneous factors to take into account, it is not possible to easily consider all of them in an algorithm without trading off heavily in terms of execution times. This is why we decided to settle for the dispatching rule that accounts for about 75% of the situation in the fabrication facility. We also assume that increase

ReadyList: Shuffled list of nodes ready to process
ProcessingList: List of nodes already processing sorted by their completion times
CurrentTime: Current time in the algorithm
$Idle_m \in \{0,1\}, m \in \mathscr{M}$: state of machine m, i.e. $Idle_m = 0$ if m is busy, and $Idle_m = 1$ otherwise

Step **0** :
$CurrentTime = 0$;
ReadyList is initialized with all nodes ready to process;
Shuffle *ReadyList*;
ProcessingList is initialized with all nodes being processed by tools;
Sort *ProcessingList*;
Initialize $Idle_m$, $\forall m \in \mathscr{M}$, to its current state;
while *all nodes have not been scheduled* **do**
  Step **1** :
  for *all nodes* $j_k^l \in ReadyList$ **do**
    Step **1.1** :
    for *all machines* $m \in \mathscr{M}_k^l$ **do**
      if $idle_m = 0$ *and m is the first such machine* **then**
        $C_k^l = CurrentTime + p_{km}^l$;
        $idle_m = 1$;
        Add and sort j_k^l in *ProcessingList*;
        Remove j_k^l from *ReadyList*;
      end
    end
  end
  Step **2** :
  Select first node j_k^l from *ProcessingList*;
  $CurrentTime = C_k^l$;
  Step **3** :
  for *all nodes* $j_k^l \in ProcessingList$ **do**
    if $C_k^l \leq CurrentTime$ **then**
      Remove j_k^l from *ProcessingList*;
      Randomly insert j_{k+1}^l in *Readylist*;
      Retrieve m, that was processing j_k^l;
      $idle_m = 0$
    end
  end
end

Algorithm 1: List scheduling algorithm.

of priority when a lot enters a time constraint is fixed and included in the base priority, as only lots within time constraint tunnels are studied in this paper.

The chosen rule, called *standard dispatching rule* is frequently encountered in semiconductor manufacturing as it is simple, but effective. It faithfully models the total priority as a function of a base priority factor and the waiting time at each operation. Let π_{base}^l be the base priority of lot $l \in \mathscr{L}$ and π_{total}^l the total

priority, then:

$$\pi^l_{total}(\pi^l_{base}, DAS^l) = \pi^l_{base}(1 + DAS^l),$$

where DAS^l stands for Day at Step and corresponds to the time spent by the lot l waiting to be processed in its given operation. This rule aims at minimizing the total tardiness of lots per unit of base priority.

Based on this dispatching rule, let us take Algorithm 1 one step further, by uniformly drawing the lots available to be scheduled in accordance with their total priority at the current time, instead of pure random drawing. More precisely, the probability of selection π^l_{select} of a lot $l \in \mathscr{L}$ is computed as follows:

$$P^l_{select} = \frac{\pi^l_{total}(\pi^l_{base}, DAS^l)}{\sum_{k \in \mathscr{L}_{available}} \pi^k_{total}(\pi^k_{base}, DAS^l)}.$$

This policy can be implemented by modifying Step **0** and **1** of Algorithm 1:

Step **0**:
Compute $\pi_{Total} = \sum_{j^l_k \in ReadyList} \pi^l_{total}$;
Step **1**:
while *all nodes in* ReadyList *have not been checked* **do**
  Initialize $LocalSum = 0$;
  Draw a random integer *Rand* between 0 and π_{Total};
  for *all unchecked nodes* j^l_k *in ReadyList* **do**
    Compute $LocalSum = LocalSum + \pi^l_{total}$;
    if $Rand \leq LocalSum$ **then**
      Do Step **1.1** of Algorithm 1;
      mark j^l_k as checked;
      $\pi_{Total} = \pi_{Total} - \pi^l_{total}$;
      Break out of loop
    end
  end
end

Algorithm 2: List scheduling algorithm based on priority.

Algorithm 2 takes into account the wafer fab priority system without neglecting the overall variability. The most critical lots will have a higher probability of being selected. This is particularly true towards the end of an iteration when fewer lots remain. One drawback of this approach is that it requires to constantly update the priority of lots, which consequently increases the execution time compared to Algorithm 1.

5 EXPERIMENTAL RESULTS

5.1 Instances description

Industrial instances have been selected over an horizon of four consecutive months, with a gap of one month between the second and third selected months. In each month, two instances were selected spread at least one week apart. Five TCTs have been selected, that are representative of the wafer fab under study, since they cover all the areas that are the most affected by time constraints. Each TCT covers a different number of operations, and can include up to five time constraints. As shown in Table 1 ("N/A" means that no data were available to build the instance), the sizes of the instances are very large with an average of 400 lots per instance and 20 000 operations to schedule, and between 30 and 200 tools.

Instance	TCT 1		TCT 2		TCT 3		TCT 4		TCT 5	
	Oper.	Lots	Oper.	Lots	Oper.	Lots	Oper.	Lots	Oper.	Lots
3	N/A	N/A	22 009	427	21 214	474	26 208	336	17 308	348
4	4 531	224	N/A	N/A	15 687	446	N/A	N/A	19 768	424
8	6 467	392	26 744	514	22 208	550	26 477	373	19 305	402
9	N/A	N/A	N/A	N/A	23 464	668	35 109	490	15 528	456
10	11 010	568	32 226	621	31 020	671	38 474	523	22 992	522
11	N/A	N/A	31 003	598	11 963	269	5 628	501	22 437	469
14	9 638	485	N/A	N/A	N/A	N/A	9 814	445	N/A	N/A
15	N/A	N/A	N/A	N/A	N/A	N/A	8 322	380	N/A	N/A

Table 1: Description of instances.

5.2 Comparative analysis

In addition to the pure random dispatching policy (RAND) and the priority based dispatching policy (PIT) presented earlier, we also investigated a variant of the latter, called WAIT, where only the priority induced by waiting times is taken into account. This was achieved by assigning the same base priority to all lots.

The computational results are first analyzed globally in order to detect general trends and compare the performance of each dispatching policy. Then, the results of each TCT are discussed to draw more specific conclusions.

PIT is the policy that provides the highest success rates. While this is not always the case (e.g. instance 9 of TCT 4), it is consistent with the fact that the fictitious lot has a medium to high base priority. Such a lot is thus most often scheduled before other lots (with PIT), which leads to higher success rates in general. However, PIT provides some of the worst success rates in certain cases (e.g. instances 9 and 10 of TCT 4 instances and instances 10 and 11 of TCT 5), which means that the current WIP includes many lots constrained by the TCT that have been waiting for a while. These lots have the same or higher base priority than the fictitious lot.

On the other hand, WAIT persistently provides worse success rates than the other two policies. It may seem counter-intuitive, since TCTs are all located at critical points of the product flow. More importantly, because of how TCTs are managed, more lots tend to accumulate in front of a TCT when it is closed, and then to be quickly flushed through the TCT when it is open. Since the fictitious lot is always the last one to arrive at the TCT by construction, it is heavily penalized by only taking waiting times into consideration.

In most cases, when a TCT has a low (e.g. instance 4 of TC 32) or very high (e.g. most instances of TCs 34 and 53) success rate, this is verified by all three dispatching policies. This means that, from a pure decision support standpoint, it does not really matter which policy is adopted since they all provide the same information for these cases.

In a number of instances, the three dispatching policies (in particular, WAIT and RAND) can have very similar results (e.g. instances 10 and 11 of TC 21). This means that probabilities and process times are mostly evenly distributed across the lots and that there are no particular critical points in the product flow.

These trends reveal a key point: PIT provides a better modeling of the reality than RAND. It highlights the significant impact that priority settings have on the product flow and that they are not negligible as was previously assumed. This is more acutely reflected by the discrepancies found between PIT, RAND, and WAIT for instance 11 of TC 23 and instance 9 of TC 41.

Additionally, when considering each individual TCT, further elements of interest emerge:

- TCT 1: TC 11 proves to have the lowest success rates for all instances except instance 4. Instance 4 appears to be a classical case of product flow slowdown due to tool unavailability. Where all the other instances of TC 12 have high success rates, instance 4 has low success rates. Conversely, it has near 100% success rate for TC 11. This situation occurs when the tunnel is blocked and manually regulated at the time. Then TC 11 would always be verified since the lots are being

INST	METHOD	TCT 1		
		TC 11	TC 12	TC 13
4	RAND	1.00	0.55	1.00
	WAIT	1.00	0.56	1.00
	PIT	1.00	0.89	1.00
8	RAND	0.65	1.00	1.00
	WAIT	0.69	1.00	1.00
	PIT	0.97	1.00	1.00
10	RAND	0.87	0.97	1.00
	WAIT	0.98	1.00	1.00
	PIT	1.00	1.00	1.00
14	RAND	0.67	0.97	1.00
	WAIT	0.57	1.00	1.00
	PIT	0.98	1.00	1.00

Table 2: Results: TCT 1.

INST	METHOD	TCT 2			
		TC 21	TC 22	TC 23	TC 24
3	RAND	1.00	0.94	0.59	0.46
	WAIT	1.00	0.98	0.34	0.27
	PIT	1.00	1.00	0.75	0.44
8	RAND	1.00	0.96	0.57	0.45
	WAIT	1.00	1.00	0.38	0.15
	PIT	1.00	1.00	0.59	0.17
10	RAND	0.80	0.81	0.59	0.53
	WAIT	0.80	0.71	0.43	0.27
	PIT	0.75	0.90	0.72	0.57
11	RAND	0.83	0.82	0.56	0.49
	WAIT	0.80	0.87	0.38	0.19
	PIT	0.88	0.98	0.93	0.62

Table 3: Results: TCT 2.

INST	METHOD	TCT 3			
		TC 31	TC 32	TC 33	TC 34
3	RAND	0.96	1.00	0.73	0.39
	WAIT	0.98	1.00	0.86	0.21
	PIT	1.00	1.00	1.00	0.92
4	RAND	0.40	0.04	1.00	1.00
	WAIT	0.23	0.02	1.00	1.00
	PIT	0.57	0.15	1.00	1.00
8	RAND	0.96	0.92	0.58	0.98
	WAIT	1.00	0.97	0.81	0.93
	PIT	1.00	1.00	1.00	1.00
9	RAND	0.97	0.80	0.17	1.00
	WAIT	0.95	0.93	0.23	1.00
	PIT	0.96	1.00	0.52	1.00
10	RAND	0.79	0.72	0.32	1.00
	WAIT	0.58	0.80	0.54	1.00
	PIT	0.83	0.98	0.94	1.00
11	RAND	0.50	0.78	0.24	1.00
	WAIT	0.42	0.90	0.18	1.00
	PIT	0.59	1.00	0.58	1.00

Table 5: Results: TCT 3.

INST	METHOD	TCT 4
		TC 41
3	RAND	0.77
	WAIT	0.78
	PIT	0.85
8	RAND	0.87
	WAIT	0.93
	PIT	0.90
9	RAND	0.32
	WAIT	0.05
	PIT	0.15
10	RAND	0.56
	WAIT	0.31
	PIT	0.15
11	RAND	0.63
	WAIT	0.43
	PIT	0.42
14	RAND	0.22
	WAIT	0.08
	PIT	0.17
15	RAND	0.43
	WAIT	0.25
	PIT	0.75

Table 4: Results: TCT 4.

INST	METHOD	TCT 5				
		TC 51	TC 52	TC 53	TC 54	TC 55
3	RAND	1.00	1.00	1.00	0.70	0.56
	WAIT	1.00	1.00	1.00	0.73	0.15
	PIT	1.00	1.00	1.00	1.00	0.55
4	RAND	1.00	0.33	1.00	0.97	1.00
	WAIT	1.00	0.15	1.00	1.00	1.00
	PIT	1.00	0.52	1.00	1.00	1.00
8	RAND	1.00	1.00	1.00	0.42	0.21
	WAIT	1.00	1.00	1.00	0.48	0.05
	PIT	1.00	1.00	1.00	0.89	0.27
9	RAND	1.00	0.72	1.00	1.00	0.97
	WAIT	1.00	0.69	1.00	1.00	0.98
	PIT	1.00	0.98	1.00	1.00	1.00
10	RAND	1.00	0.95	1.00	0.56	0.50
	WAIT	1.00	0.98	1.00	0.63	0.20
	PIT	1.00	1.00	1.00	0.88	0.23
11	RAND	1.00	0.95	1.00	0.55	0.48
	WAIT	1.00	0.98	1.00	0.45	0.27
	PIT	1.00	1.00	1.00	0.92	0.32

Table 6: Results: TCT 5.

introduced in order to respect it and TC 12 would cross the location of a tool breakdown and thus be much harder to respect than when all possible tools are available.

- TCT 2: A funnel effect occurs in TCT 2. The success rates seem to decrease the further along the lot goes through the tunnel. This is due to the fact that this tunnel concerns a specific geometry that requires unique single tools, most of which are located towards the end of the tunnel.
- TCT 3: All the constraints of this tunnel are notoriously short, but so are process times. This induces high variability in the tunnel. Unlike other instances, there does not seem to be a pattern where a particular TC is harder to complete in time. Throughout the instances, it clearly appears that each TC has at some point low success rates (instance 11 of TC 31, instance 4 of TC 32, instance 9 of TC 33, instance 3 of TC 34), and has otherwise very high success rates. It can be shown that each drop in success rates is related to a specific tool breakdown in different parts of the tunnel. As a matter of fact, this tunnel was considered very difficult to complete due to high variability, and then modified. This is also why no data are available for instances 14 and 15.
- TCT 4: This single TC highlights most of the effects described above, such as lower success rates for PIT than RAND (instances 11 and 14 of TC 41) or the other way around (instances 3 and 15 of TC 41).
- TCT 5: Low volumes passing through this tunnel mostly explain the high success rates for all TCs, and drops in success rates are directly related to machine breakdowns as for TCT 3.

5.3 Convergence analysis

Both Algorithms 1 and 2 rely on repeated random sampling and converge by the law of large numbers. In the light of this fact, the convergence criterion has been chosen to design the stopping rule in the numerical experiments. The output of both algorithms is a vector of discrete probability distributions for each time constraint. Each discrete probability distribution D_k ($k \in \mathscr{T}$) has two events:

- Event 1: The lot successfully crosses the TC on time with probability $\hat{P}_k$.
- Event 0: The lot exceeds the TC with probability $1-\hat{P}_k$.

Let $V^N = (\hat{P}_1^N, \ldots, \hat{P}_T^N)$ be the output vector of the N^{th} iteration. The distance between two probability distributions of each TC obtained in two successive iterations is measured in terms of the unitary euclidean distance d_2^u:

$$d_2^u(D_k^{N-1}, D_k^N) = \|(\hat{P}_k^{N-1} - \hat{P}_k^N), ((1-\hat{P}_k^{N-1}) - (1-\hat{P}_k^N))\|_2^u = \\ = \frac{\sqrt{(\hat{P}_k^{N-1} - \hat{P}_k^N)^2 + [(1-\hat{P}_k^{N-1}) - (1-\hat{P}_k^N)]^2}}{\sqrt{2}} = \sqrt{(\hat{P}_k^{N-1} - \hat{P}_k^N)^2} = |\hat{P}_k^{N-1} - \hat{P}_k^N| \quad (3)$$

Furthermore, a lot is considered to not satisfy a TCT if and only if it does not satisfy at least one of the TCs in the TCT. Thus, it makes sense to measure the distance d between two successive output vectors V^N and V^{N-1} by taking the infinite norm as metrics:

$$d(V^N, V^{N-1}) = \|V^N - V^{N-1}\|_\infty = max_{k \in \mathscr{T}} \left\{ d_2^u(D_k^{N-1}, D_k^N) \right\} \quad (4)$$

An initial replication count $RC^{initial}$ and a minimum increment MI are defined so that, for the N^{th} iteration of the algorithm, the replication count $RC_N = RC_{N-1} + \frac{d(D^N, D^N - 1)}{\varepsilon} MI$, $RC_0 = RC^{initial}$ and $RC_1 = RC^{initial} + MI$, where ε is the algorithm precision. The algorithms stop when $d(V^N, V^{N-1}) \leq \varepsilon$. Hence, the further away we are from convergence according to our criterion and precision, the faster the replication count is increased. For the purpose of this study and based on the industrial requirements, $RC^{initial} = MI = 30$ and $\varepsilon = 0.05$.

Table 7 underlines some important facts. First of all, the convergence is overall achieved after a smaller number of iterations with both WAIT and PIT for most instances and TCTs than with RAND.

Instance		TCT 1			TCT 2			TCT 3			TCT 4			TCT 5		
		RAND	PIT	WAIT	RAND	PIT	WAIT	RAND	PIT	WAIT	RAND	PIT	WAIT	RAND	PIT	WAIT
3	IT Count	N/A	N/A	N/A	164	250	190	157	174	139	60	100	110	291	60	60
	EXE(s)	N/A	N/A	N/A	15	111	88	17	94	78	5	39	43	23	22	22
4	IT Count	161	120	109	N/A	N/A	N/A	200	100	263	N/A	N/A	N/A	60	212	231
	EXE(s)	1	5	4	N/A	N/A	N/A	16	37	98	N/A	N/A	N/A	6	96	11
8	IT Count	329	110	60	259	60	289	166	226	60	60	60	99	149	60	181
	EXE(s)	9	14	9	38	42	212	26	154	43	6	28	48	18	30	92
9	IT Count	N/A	N/A	N/A	N/A	N/A	N/A	278	60	149	222	60	60	60	120	60
	EXE(s)	N/A	N/A	N/A	N/A	N/A	N/A	60	66	163	45	54	56	8	67	37
10	IT Count	60	60	60	247	388	327	187	248	110	299	119	60	221	60	159
	EXE(s)	6	28	24	60	446	400	53	325	154	73	132	69	44	49	136
11	IT Count	N/A	N/A	N/A	209	109	301	171	60	100	180	60	149	255	60	277
	EXE(s)	N/A	N/A	N/A	42	118	338	5	8	13	32	57	144	35	40	185
14	IT Count	60	60	60	N/A	N/A	N/A	N/A	N/A	N/A	160	60	110	N/A	N/A	N/A
	EXE(s)	4	15	16	N/A	N/A	N/A	N/A	N/A	N/A	7	14	26	N/A	N/A	N/A
15	IT Count	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	60	60	60	N/A	N/A	N/A
	EXE(s)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3	10	10	N/A	N/A	N/A

Table 7: Iteration count and execution time for all TCTs and instances.

This is not surprising since less variability is expected with random dispatching policies that take priorities into account. Note also that execution times remain below the 5-minute limit that is set as an industrial requirement. A second important fact is that both PIT and WAIT are considerably more computationally intensive than RAND. Even though PIT better fits the industrial reality, a trade-off might be required in terms of computational times. Hence, when RAND is able to provide the same information than PIT, it is not necessary to use PIT to save time, in particular when the success rate is either 0% or 100%.

Finally, the table also shows that, in several cases, the number of iterations is 60 which is by design the minimum number of iterations before the stopping criterion can be met. When crossing this information with the one of the previous tables, it is interesting to note that most of these cases translate into a success rate of 100% or 0%. In these cases, the TCs are so exceedingly violated or easily respected that only the analysis of cycle time convergence can provide accurate and sound insights. However, from a pure decision support point of view, these extreme cases are irrelevant because no additional information is necessary to take decisions.

6 CONCLUSION

This paper deals with Time Constraint Tunnels (TCTs) between consecutive process steps in wafer fabrication, the management of which becomes more and more intricate. Being written in the continuity of (Sadeghi, Dauzère-Pérès, Yugma, and Lepelletier 2015), this paper investigates three dispatching policies for estimating the probability that a lot successfully exits a TCT on time. By better representing the industrial reality, the priority based dispatching policies provide more accurate results and better reflect TCTs intrinsic structure than the original proposed policy, at the expense of an increase of computational times. However, from a decision standpoint, the pure random policy remains viable for unequivocal cases, i.e. when TCs are either always satisfied or not satisfied at all.

Future work will be dedicated to improving and expanding the decision support potential of the proposed approach, by applying more thorough statistical analysis tools, in order to refine the convergence criterion, and to more closely assess each of the parameters set a given in this paper, such as the value of epsilon, the choice of norms and the iterative method to achieve convergence. The ultimate goal being to support decisions for the management of TCTs, we intend to enhance and extend the approach to derive valuable managerial insights, such as **(i)** cycle time improvement, **(ii)** detection of root causes of TC violations, etc.

REFERENCES

Arima, S., A. Kobayashi, Y.-F. Wang, K. Sakurai, and Y. Monma. 2015. "Optimization of Re-Entrant Hybrid Flows With Multiple Queue Time Constraints in Batch Processes of Semiconductor Manufacturing". *IEEE Transactions on Semiconductor Manufacturing* 28 (4): 528–544.

Attar, S., M. Mohammadi, and R. Tavakkoli-Moghaddam. 2013. "Hybrid Flexible Flowshop Scheduling Problem with Unrelated Parallel Machines and Limited Waiting Times". *The International Journal of Advanced Manufacturing Technology* 68 (5-8): 1583–1599.

Attar, S., M. Mohammadi, R. Tavakkoli-Moghaddam, and S. Yaghoubi. 2014. "Solving a New Multi-Objective Hybrid Flexible Flowshop Problem With Limited Waiting Times and Machine-Sequence-Dependent Set-Up Time Constraints". *International Journal of Computer Integrated Manufacturing* 27 (5): 450–469.

Chen, J.-S., and J.-S. Yang. 2006. "Model Formulations for the Machine Scheduling Problem with Limited Waiting Time Constraints". *Journal of Information and Optimization Sciences* 27 (1): 225–240.

Chien, C.-F., and C.-H. Chen. 2007. "A Novel Timetabling Algorithm for a Furnace Process for Semiconductor Fabrication with Constrained Waiting and Frequency-Based Setups". *OR Spectrum* 29 (3): 391–419.

Fondrevelle, J., A. Oulamara, and M.-C. Portmann. 2008. "Permutation Flowshop Scheduling Problems with Time Lags to Minimize the Weighted Sum of Machine Completion Times". *International Journal of Production Economics* 112 (1): 168–176.

Jung, C., D. Pabst, M. Ham, M. Stehli, and M. Rothe. 2014. "An Effective Problem Decomposition Method for Scheduling of Diffusion Processes Based on Mixed Integer Linear Programming". *IEEE Transactions on Semiconductor Manufacturing* 27 (3): 357–363.

Kalir, and Tirkel. 2016. "Scheduling Preventive Maintenance Within a Queue Time for Maximum Throughput in Semiconductor Manufacturing". In *Proceedings of the 2016 Winter Simulation Conference (WSC)*, edited by P. Frazier, T. Roeder, R. Szechtman, and E. Zhou, 2750–2761: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Klemmt, A., and L. Mönch. 2012. "Scheduling Jobs with Time Constraints between Consecutive Process Steps in Semiconductor Manufacturing". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, and R. Pasupathy, 194. Winter Simulation Conference: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Knopp, S., S. Dauzère-Pérès, and C. Yugma. 2017. "A Batch-Oblivious Approach for Complex Job-Shop Scheduling Problems". *European Journal of Operational Research* 263 (1): 50–61.

Sadeghi, R., S. Dauzère-Pérès, C. Yugma, and G. Lepelletier. 2015. "Production Control in Semiconductor Manufacturing with Time Constraints". In *Advanced Semiconductor Manufacturing Conference (ASMC), 2015 26th Annual SEMI*, 29–33: IEEE.

Sun, D.-S., Y.-I. Choung, Y.-J. Lee, and Y.-C. Jang. 2005. "Scheduling and Control for Time-Constrained Processes in Semiconductor Manufacturing". In *Semiconductor Manufacturing, 2005. ISSM 2005, IEEE International Symposium on*, 295–298: IEEE.

Yugma, C., S. Dauzere-Péres, C. Artigues, A. Derreumaux, and O. Sibille. 2012. "A Batching and Scheduling Algorithm for the Diffusion Area in Semiconductor Manufacturing". *International Journal of Production Research* 50 (8): 2118–2132.

Zhou, Y., and K. Wu. 2017. "Heuristic Simulated Annealing Approach for Diffusion Scheduling in a Semiconductor Fab". In *Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on*, 785–789: IEEE.

AUTHOR BIOGRAPHIES

ALEXANDRE LIMA is a Ph.D. student in Industrial Engineering at Ecole des Mines de Saint-Etienne (EMSE),Gardanne, France. He works as an Engineer at STMicroelectronics Crolles. He received the Engineering degree and the Master's degree in Industrial Engineering from Ecole Nationale des Mines de Saint-Etienne, France, in 2015. He also received the Industrial Engineering master degree from Ecole Nationale des Mines de Saint-Etienne,France in 2014. His research interest are in the fields of discrete event simulation, scheduling and operations research applied to semiconductor manufacturing. His email is alexandre.lima@st.com.

VALERIA BORODIN is an Associate Professor at the CMP of Ecole Nationale Supérieure des Mines de Saint-ÉEtienne since October 2015. She received in 2014 her Ph.D degree in Optimization and Systems Safety from the University of Technology of Troyes, France. She works in the areas of quantitative operations management at the different levels of decision making. Her email address is valeria.borodin@emse.fr.

STEPHANE DAUZERE-PERES is Professor at the Center of Microelectronics in Provence (CMP) of the EMSE. He received the Ph.D. degree from the Paul Sabatier University in Toulouse, France, in 1992; and the H.D.R. from the Pierre and Marie Curie University, Paris, France, in 1998. He was a Postdoctoral Fellow at the Massachusetts Institute of Technology, U.S.A., in 1992 and 1993, and Research Scientist at Erasmus University Rotterdam, The Netherlands, in 1994. He has been Associate Professor and Professor from 1994 to 2004 at the Ecole des Mines de Nantes in France. He was invited Professor at the Norwegian School of Economics and Business Administration, Bergen, Norway, in 1999. Since March 2004, he is Professor at the Ecole des Mines de Saint-Etienne. His research interests broadly include modeling and optimization of operations at various decision levels (from real-time to strategic) in manufacturing and logistics, with a special emphasis on semiconductor manufacturing. He has published more than 60 papers in international journals and contributed to more than 120 communications in conferences. Stéphane Dauzère-Pérès has coordinated multiple academic and industrial research projects, and also five conferences. His email address is dauzere-peres@emse.fr.

PHILIPPE VIALLETELLE is a Senior Member of Technical Staff at STMicroelectronics Crolles, France. After receiving an Engineering degree in Physics from the Institut National des Sciences Appliquees in 1989, he entered the semiconductor industry by working on ESD and physical characterization. His next experience was in Metrology before enlarging his scope of activities to Process Control. He finally integrated the Factory Integration world, through Industrial Engineering and is now responsible for the definition of advanced methodologies and tools for the Crolles 300mm production line. Expert in Manufacturing Sciences for Process and Production Control items, he is now in charge of driving collaborative projects at European level. His email address is philippe.vialletelle@st.com.