Alexandre Lima, Valeria Borodin, Stéphane Dauzère-Pérès & Philippe Vialletelle (2020) A sampling-based approach for managing lot release in time constraint tunnels in semiconductor manufacturing, International Journal of Production Research, DOI: 10.1080/00207543.2020.1711984

A Sampling-Based Approach for Managing Lot Release in Time Constraint Tunnels in Semiconductor Manufacturing

Alexandre Lima[a,b], Valeria Borodin[a], Stéphane Dauzère-Pérès[a,c] and Philippe Vialletelle[b]

[a]Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS,
CMP, Department of Manufacturing Sciences and Logistics, F-13541 Gardanne, France
E-mail: {borodin, dauzere-peres}@emse.fr
[b] Advanced Engineering Methods Department, ST Microelectronics, F-38926 Crolles, France
Email: {alexandre.lima, philippe.vialletelle}@st.com
[c] Department of Accounting, Auditing and Business Analytics, BI Norwegian Business School, 0484 Oslo, Norway

ABSTRACT
For the sake of product yield and quality considerations, Time Constraints (TCs) are imposed between process operations in various multi-product manufacturing systems. Often spanning a number of operations, time constraints tend to follow each other in close succession and overlap, forming thus Time Constraint Tunnels (TCTs). The regulation problem of releasing lots in these time constraint tunnels is particularly challenging in semiconductor manufacturing systems, because of re-entrant flows, machine heterogeneity and High Mix Low Volume (HM-LV) production configurations, which are typical in many wafer fabrication facilities. In such an evolving and time-varying context, this paper proposes a sequential sampling-based approach to estimate the probability that, prior to its release, a lot leaves a given time constraint tunnel on time. The proposed approach proves to be competitive in various respects by: (i) taking into account industry specific features, (ii) being industrially tractable, and (iii) being sensitive and responsive to the current manufacturing system. Based on real-life instances, numerical experiments highlight the computational effectiveness and the industrial soundness of the proposed problem modeling together with the solution approach.

KEYWORDS
Time constraints; Probability estimation; Sequential sampling-based approach; Semiconductor manufacturing.

1. Introduction

Semiconductor manufacturing is characterized by long product cycle times (2-3 months) and a very fast product turnover with new highly differentiated products being released every 4-6 months. This makes the fabrication environment, already constrained by numerous manufacturing requirements, difficult for the management of capacity and production planning. In particular, small semiconductor plants (called fabs for short) operate in High Mix and Low Volume (HM-LV) production modes, often using heterogeneous sets of machines, purchased over time to complete numerous manufacturing process operations. These factors combined with a high level of automation and the complexity of product routes, often including several hundred operations and re-entrant flows, make semiconductor manufacturing the most complex industrial fabrication process.

Wafers have become extremely sensitive to external conditions, even in controlled environments, with topological dimensions being only a few atom layers thick. By letting a lot *out in the open* in the clean room for a certain period of time between process operations,

natural phenomena can occur and affect the wafer chemical properties or degrade its topology. Such phenomena include, but are not limited to, oxidation, crystal formation and ion migration. The activation of these physical processes might be prohibitive for the subsequent wafer processing. If that is the case, wafer are scrapped or reprocessed, if possible. In order to prevent the chemical and physical degrading reactions, time limits are imposed between key operations. Hence, let us define a *Time Constraint* (TC) by a start operation and an end operation, jointly with a time limit.

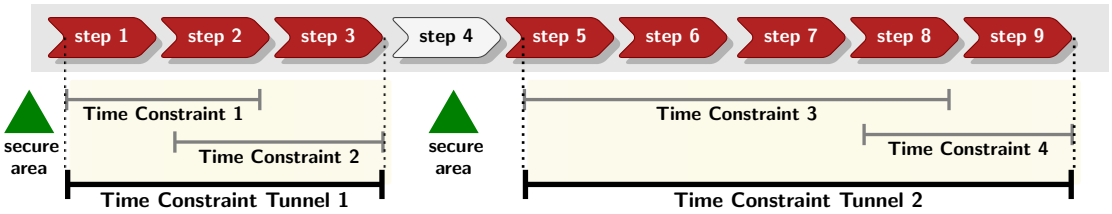

Figure 1.: An illustrative example of two successive TCTs (Lima et al. 2019)

Due to the size shrinkage of transistors, the increased sensitivity leads to a growing number of time constraints, especially for the smallest geometries. This multitude of time constraints overlap and follow each other in close succession, as depicted in Figure 1. A succession of Time Constraints has been named *Time Constraint Tunnel* (TCT) in the related literature (Sadeghi et al. 2015; Lima et al. 2017, 2019).

TCTs are characterized by three elements: (i) a start operation, (ii) an end operation, and (iii) a set of time constraints. The entrance of each tunnel is called a *secure area*, since it is the only place where a lot (i.e. a set of wafers) can wait without risking product degradation. The analogy with tunnels is notable, as once a lot enters a TCT, no turning back is possible like in a real-life tunnel. No significant slowing down or stopping, it can only push forward until the exit. By the same token, TCTs are managed on the shop floor following this analogy. Like at a toll at the entrance of a tunnel, lots are held before a TCT in a secure area, until *the way is clear*, i.e. the current capacity utilization of the tunnel makes it safe and able to suitable perform all of their operations so that it comes out on time.

The management of TCTs at the operational level consists in regulating the release of lot in TCTs. This task is usually performed manually, through human supervision with a central manager having the discretion of determining what *the way is clear* means. Note that the management setup is limited as *tunnel regulators*, by relying on acquired knowledge and empirical evidence to identify tunnels and to know when and how many lots to release. This task is extremely stressful and time consuming for managers, given the large number of tunnels to be handled (6,000 tunnels divided in 270 classes in the fab of STMicroelectronics Crolles).

This paper addresses the problem of TCT management in an industrial HM-LV context and aims at supporting the dispatching regulation of a lot in a given TCT. Carried out in the continuity of the research stream initiated in (Sadeghi et al. 2015; Lima et al. 2017, 2019), the contribution of this paper is multiple. First, it improves the accuracy of the problem modeling proposed in (Lima et al. 2017), by explicitly integrating a priority-based dispatching policy widely adopted in semiconductor fabs. Second, an accelerated version of the sampling-based approach presented in (Lima et al. 2017) is developed. This new version is empirically shown to be able to handle the curse of problem dimensionality in highly time-varying industrial settings. Extensive numerical experiments conducted on industrial data show the applicability of the proposed approach.

The content of this paper is structured as follows. Section 2 gives a review of the state of the art related to time constraint management. Section 3 contextualizes and states the problem un-

der study. The underlying graph-based modeling and its evaluation are described in Section 4. Section 5 provides an accelerated version of the sampling-based approach proposed in (Lima et al. 2019). Extensive numerical experiments are conducted on real industrial instances in Section 6, and the obtained results are reported and analyzed. The practical relevance of the proposed sampling-based approach is discussed in Section 7. Some concluding remarks are presented in Section 8, including an outline of topics for future research.

2. Literature review

The related literature on time constraint management heavily leans towards semiconductor manufacturing oriented topics. However, this finding does not circumscribe the scope of TC applications. Several other production systems are also subject to these scheduling constraints, such as the automotive (Lee et al. 2018), the steel and metal industries (Feng, Chu, and Che 2018; Wu et al. 2016), and the health-care industry (Barz and Rajaram 2015).

The notion of *time constraint* strictly speaking is not conventional and can be found in the literature under different denominations, namely *queue time* or *process queue time* (Wu, Lin, and Chien 2012; Kobayashi, Kuno, and Arima 2013; Wu et al. 2016; Lee, Chen, and Wu 2005; Wu, Lin, and Chien 2010; Wu et al. 2012; Yurtsever, Kutanoglu, and Johns 2009), *time window* (Feng, Chu, and Che 2018; Wang et al. 2018; Cho et al. 2014; Jung et al. 2014), *time lag* or *time link* (Wang, Huang, and Li 2018; Maleck and Eckert 2017; Knopp, Dauzère-Pérès, and Yugma 2017). To the best of our knowledge, the term *Time Constraint Tunnel* has itself been introduced for the first time by Sadeghi et al. (2015) and extended in our previous work (Lima et al. 2017, 2019). Note that simpler representations of successions of TCs have been the subject of several publications, which either deal with only two consecutive time constraints (Wang, Huang, and Li 2018; Feng, Chu, and Che 2018; Wu, Lin, and Chien 2012; Maleck et al. 2017; Maleck and Eckert 2017; Pappert et al. 2016; Kitamura, Mori, and Ono 2006; Robinson 1998; Lee, Chen, and Wu 2005; Wu, Lin, and Chien 2010; Wu et al. 2012; Yurtsever, Kutanoglu, and Johns 2009; Jung et al. 2014), or focus on a single critical TC under multiple process queue time constraints (Wu et al. 2016).

The problem of TCT management is mainly handled in the framework of the following well-known classes of problems:

- *Classical scheduling problem*, where the goal is to assign tasks to resources while optimizing a set of criteria, such as the makespan (Kim and Lee 2017), maximum tardiness (Yugma et al. 2012), or the overall TC violation (Knopp, Dauzère-Pérès, and Yugma 2017),
- *Capacity planning problem*, where the goal is to determine the maximum capacity to be allowed in a tunnel without any TC violation (Pappert et al. 2016; Kitamura, Mori, and Ono 2006; Johnzén, Dauzère-Pérès, and Vialletelle 2011; Robinson 1998),
- *Production control problem*, which deals with the production rate, while satisfying TCs (Wu, Lin, and Chien 2012; Maleck and Eckert 2017; Wu et al. 2016; Lee, Chen, and Wu 2005; Wu, Lin, and Chien 2010; Wu et al. 2012; Cho et al. 2014).

Within the framework of current industrial contexts, the previously mentioned capacity planning approaches seem to be phasing out for managing TCTs. This is due in part to the static nature of the problem statement, that mostly seeks to minimize the violation of TCTs on average, and are more suitable for the tactical decision level rather than to support operational decisions. Capacity planning approaches can be completely helpless in a very time-varying context. It is worth noting that, the high variability of the product flow and the high rate of both planned and unplanned machine downtimes can entirely change the capacity landscape

of the fabrication plant in a relatively short time.

As far as the solution methods are concerned, exact resolution methods quickly prove to be computationally prohibitive, due to the curse of dimensionality and the large size of industrial instances (Wu, Lin, and Chien 2012; Wu et al. 2016). As such, exact Mixed Integer Linear Programming (MILP) based methods (Maleck et al. 2017; Jung et al. 2014; Cho et al. 2014) and branch-and-bound schemes (Kim and Lee 2017) do not provide convenient results without a drastic trade-off in terms of execution time and solution quality (Cho et al. 2014), being viable only for unrealistically small instances (Kim and Lee 2017). However, hybridized MILP approaches combined with the Lagrangian relaxation (Kaihara, Kurose, and Fujii 2012) or constructive heuristics (Wang, Huang, and Li 2018; Johnzén, Dauzère-Pérès, and Vialletelle 2011; Maleck and Eckert 2017) can improve the decision support potential of exact solution methods. Dynamic programming and queuing-based paradigms also received much attention, despite the limitations they posed relative to their reduced scalability and basic problem assumptions, e.g. arrival rates are assumed to be known or predictable (Barz and Rajaram 2015; Krämer, Hafsi, and Bai 1997; Feng, Chu, and Che 2018; Wu, Lin, and Chien 2012, 2010; Wu et al. 2012, 2016; Kitamura, Mori, and Ono 2006).

To better deal with the intractability of TC management and react in a time-varying TC environment, constructive heuristics (Wang, Huang, and Li 2018; Kaihara, Kurose, and Fujii 2012; Wang et al. 2018; Klemmt and Mönch 2012; Knopp, Dauzère-Pérès, and Yugma 2017; Lee, Chen, and Wu 2005; Yurtsever, Kutanoglu, and Johns 2009; Yugma et al. 2012) and dispatching policies (Maleck and Eckert 2017; Kobayashi, Kuno, and Arima 2013) show strong growth in popularity. In order to facilitate a deeper exploration of the solution space, greedy constructive heuristics have been combined with local search or lightweight metaheuristics (Yugma et al. 2012; Knopp, Dauzère-Pérès, and Yugma 2017). Let us emphasize the industrial soundness of list scheduling approaches, which appear to be a suitable alternative to cope with large instances specific to complex manufacturing systems in acceptable execution times (Klemmt and Mönch 2012; Wang et al. 2018; Kaihara, Kurose, and Fujii 2012; Yurtsever, Kutanoglu, and Johns 2009).

As previously noted in (Pappert et al. 2016), the problem of managing TCs has evolved and become central, in line with the pursuit of higher yield and quality specifications and thus much smaller critical dimensions, and has nowadays the following requirements.

- Be industrially responsive, scalable and tractable: Over the last few years, a wealth of research efforts have been focused on addressing real-life industrial applications (Kobayashi, Kuno, and Arima 2013; Kitamura, Mori, and Ono 2006; Klemmt and Mönch 2012; Knopp, Dauzère-Pérès, and Yugma 2017; Lee, Chen, and Wu 2005; Yugma et al. 2012; Yurtsever, Kutanoglu, and Johns 2009; Jung et al. 2014; Li et al. 2016).
- Provide decision support or automate TC management: In this respect, note that handling TC management via a production control problem is not appropriate as a result of its objective function, which aims at smoothly managing the production flow and control the throughput rate via a sequence of operations, while minimizing TC violations.
- Take into account industry specific features, e.g. re-entrant flows (Wang, Huang, and Li 2018; Feng, Chu, and Che 2018; Klemmt and Mönch 2012; Knopp, Dauzère-Pérès, and Yugma 2017; Yurtsever, Kutanoglu, and Johns 2009), HM-LV volume product settings (Feng, Chu, and Che 2018; Wang et al. 2018).

The constant evolution in technologies and the increased competitiveness in the semiconductor industry push manufacturers towards more stringent quality requirements and yield control. Hence, the problem of TC management has come into acute focus in the last few years. This fact is reflected by an upward trend observed in the rate of studies carried out on

this topic.

Following the research stream initiated in (Sadeghi et al. 2015) and improved in (Lima et al. 2017, 2019), this paper provides a sampling-based probability estimation approach dedicated to managing TCTs in an industrial context. The proposed decision support tool is drawn on an efficient state-of-the-art constructive heuristic, able to face the complexity and intractability of very large instances in complex time-varying environments. Being conceived and tailored for a generic case study arisen in semiconductor manufacturing, we believe that the proposed approach can be applied in other similarly complex manufacturing systems.

3. Problem statement

In industrial practice, time constraint tunnels are managed by making gate-keeping decisions at the point of their entrance (Sadeghi et al. 2015; Lima et al. 2017, 2019; Wang et al. 2018). Mirroring the industrial reality, consider the problem from a similar perspective by adopting the point of view of a single lot.

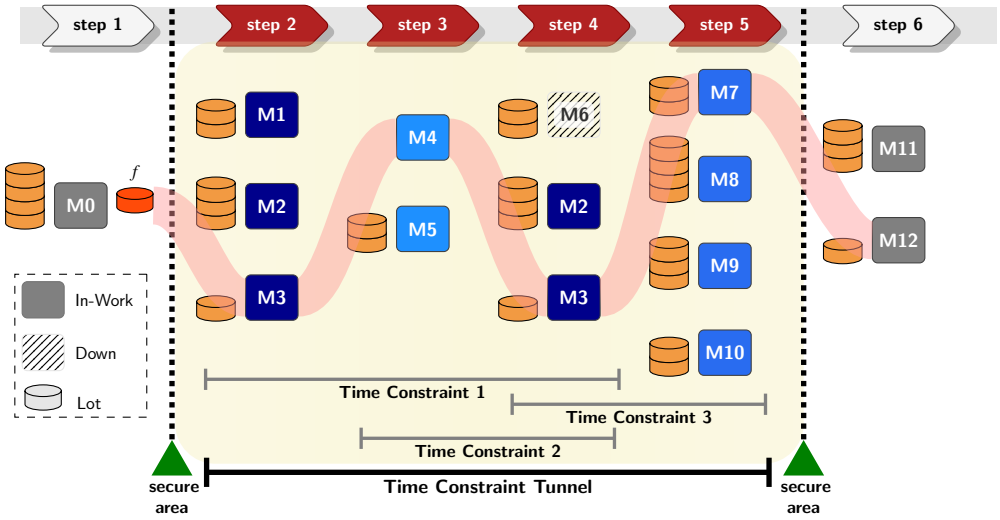

Figure 2.: A lot f about to enter a TCT including 3 TCs

The complexity associated with the regulation of time constraint tunnels is illustrated in Figure 2. Let us describe a TCT through the prism of a single lot f having to be processed through a sequence of operations: (i) the lot can share the same machines with other lots in the Work-In-Process (WIP), i.e. with lots already queued up for processing on the set of TCT machines or which may arrive for processing while the lot of interest f traverses the TCT (see stacked lots in front of machines). (ii) the sequence of process operations highlighted by the shaded path M0-M3-M4-M3-M7-M12 is not unique. (iii) re-entrant flows, such as the repetition of machines M2 and M3, are common. (iv) some machines can be down, see e.g. machine M6, and (v) machines are often heterogeneous, with different throughput rates, allowing or not the lot batching, etc.

At any point in time, the state of all machines is known for the whole fab and the current position of lots in the WIP can be recovered. To support the decision making in real time related to the release of a single lot in a TCT, we take advantage of this available evolving data to develop a context-sensitive and responsive sampling-based approach. The goal is to estimate the probability that a lot ready to be released in a TCT at a given point in time, will satisfy all of its TCs.

Before proceeding to the problem modeling, let us clarify the scope of the studied problem and the assumptions made in this paper:

- Time constraints are not considered as problem constraints and belong to problem data. Time constraints are only interpreted as a time limit to be verified *a posteriori*, and have no direct impact on how the problem is modeled.
- Only a single lot about to enter a given tunnel is considered and, respectively, the satisfaction of only its TCs is evaluated. It does not mean that other lots are not subject to time constraints. In accordance with the industrial perspective of fully automated fab control including the release of lots at the entrance of a tunnel, the TCT satisfaction is evaluated individually for each lot.
- Given that the current state of all machines is known and fixed, the problem data are deterministic. Breakdowns, that may occur while the lot of interest goes through the tunnel, are not considered. Since most TCTs have an average cycle time smaller than 48 hours, this assumption is reasonable for short periods of times.
- Batching processes are not explicitly modeled. A throughput average per lot is calculated based on the average batch size and throughput per batch.
- Setup times are not explicitly taken into account. Machines are assumed to be running at a near full capacity and setup times are included in processing times. This assumption is consistent with the fact that handling and transportation are entirely automated and done by robots in masked time in our industrial context. Machines are therefore mostly fully saturated within tunnels.
- The machine selection to assign lots is done in an arbitrary manner. As the problem at hand is a flexible job-shop scheduling problem, several machines with different properties are often available to process a job. When this is the case, machines are selected accordingly to a *first found, first served* policy.
- The Work-In-Process lots are collected via a specific algorithm (see Algorithm 1).

For the reader's convenience, the notations used in this paper are summarized in Table A1.

4. Problem modeling

Consider a lot f about to enter a sequence of operations defined by its associated process route and having to respect an uninterrupted set of time constraints, i.e a time constraint tunnel denoted by `TCT`. Let $\mathscr{T}$ be the set of time constraints in `TCT`. Each time constraint $t_{se} \in \mathscr{T}$ begins with operation j_s^f and ends after operation j_e^f, where $j_s^f, j_e^f \in \mathscr{J}$ and $s < e$. The time constraints in `TCT` follow each other and can overlap, i.e. there are at least two time constraints $t_{s_1e_1}, t_{s_2e_2} \in T$ such that $s_1 < e_2$ and $s_2 < e_1$.

The notion of tunnel can be closely interrelated with both: (i) the operations, that have to be processed in order, and (ii) the different, non necessarily disjoint, sets of machines required to process each operation.

Let $\mathscr{L}$ be the set of current WIP lots in `TCT`, retrieved via Algorithm 1. Each lot $\ell \in \mathscr{L} \cup \{f\}$ has a precedence-constrained sequence (route) of n^ℓ operations (jobs) to perform, denoted by $\mathscr{J}^\ell = (j_1^\ell, j_2^\ell, \ldots, j_{n^\ell}^\ell)$. Let $\mathscr{J} = \bigcup_{\ell \in \mathscr{L} \cup \{f\}} \mathscr{J}^\ell$ be the set of all operations.

A lot can be processed on several different machines, which may in turn have different throughput rates for each operation. As illustrated in Figure 2, nothing prevents a machine from belonging to the processing groups of various operations. A subset of machines $\mathscr{M}_k^\ell \subset \mathscr{M}$ is qualified to perform an operation j_k^ℓ of lot $\ell \in \mathscr{L}$. The related processing times p_{km}^ℓ are machine-dependent, $\forall m \in \mathscr{M}_k^\ell$.

4.1. *Work-In-Process retrieval*

Owing to the intricacy of the production routes and because of the re-entrant flows in the flexible job-shop configuration of the fabrication facility, many lots may impact lot f, directly or not, namely:

- Lots having a first degree impact, i.e. those that directly compete for the machines in the production route of lot f (see stacked lots in front of machines in Figure 2),
- Lots that do not intersect lot f in their processing, but may share the same machines as lots having a first degree impact.

Algorithm 1 : WIP retrieval algorithm ($\mathscr{J}^f$)

Set $\lambda \geq 1$, a smoothing parameter to refine the machine-dependent processing times
Set $\omega \geq 1$, a calibration coefficient to refine the depth of lot retrieval
$\texttt{cycleTimeEstimate} = 0$
$\mathscr{L} = \emptyset$
for each operation $j_k^f \in \mathscr{J}^f$ **do**
  for each machine $m \in \mathscr{M}_k^f$ **do**
    $\texttt{cycleTimeEstimate} = \texttt{cycleTimeEstimate} + p_{km}^f \cdot \lambda$
    Let S be the set of operations sharing machine m
    for each operation $s \in \mathscr{S}$ **do**
      $\texttt{cycleTime}[s] = 0$
      while $\texttt{cycleTime}[s] \leq \texttt{cycleTimeEstimate}$ **do**
        Retrieve $\mathscr{L}_s$, i.e. the set of lots located at operation s
        $\mathscr{L} = \mathscr{L} \cup \mathscr{L}_s$
        Retrieve $s_{\texttt{pred}}$, the predecessor of operation s in the routing of lots in $\mathscr{L}_s$
        $\texttt{cycleTime}[s] = \texttt{cycleTime}[s] + \omega \cdot \max_{h \in M_{\texttt{current}}, \ell \in \mathscr{L}_{\texttt{set}}} \{p_{s,h}^{\ell}\}$, where $\mathscr{M}_{\texttt{current}}$ is the set of qualified machines for processing operation s
        $s = s_{pred}$
      end while
    end for
  end for
end for
return $\mathscr{L}$

Note that considering all the lots running in the fab leads to significant unjustified data processing and computing time increases. Hence, let us only take into account the lots that might have a significant impact on the traversal of lot f and give us a good approximation of the local context surrounding lot f, i.e.:

- Only consider the lots having a first degree impact on lot f, i.e. only those that might directly cross paths with lot f. Due to the repetitive nature of process operations in a production route and the presence of re-entrant flows, this filter is sufficiently revealing and not necessarily as restrictive as it seems.
- To give a projection into the future, we take into account the lots that might affect lot f, while it is going through the tunnel. The identification of the respective lots requires an estimation of the cycle times of lot f at each process operation beforehand. In this sense, for an accurate cycle time estimation, the machine-dependent processing times related to lot f are refined by a smoothing parameter $\lambda \geq 1$. Specifically, to coincide with the assumption of fully saturated machines, λ is equal to 1. Meanwhile, the maximum process time of all the qualified machines is kept as an upper approximation for the operations of lots to consider in the WIP.

Lot f has a sequence of operations $\mathscr{J}^f$, which can be performed by a set of qualified machines $\mathscr{M}_k^f$. For each machine $m \in \mathscr{M}_k^f$ in this set, Algorithm 1 considers all operations $s \in \mathscr{S}$ sharing machine m and other similarly qualified operations on m. The set $\mathscr{L}_s$ of lots, located at these operations or prior to these operations on their routes, is retrieved. The depth of lot retrieval is estimated by factoring the sum of process times with a calibrated add-on coefficient $\omega \geq 1$, which in this paper is fixed to 2. Akin to λ, this coefficient helps us to provide a more accurate estimation of the overall cycle time of the lots in the fab.

Algorithm 1 generates the WIP having a first level of interference with lot f. By virtue of the highly cyclical nature of the fabrication process, and given that lots are not hampered by other lots requiring processing by disjoint machines at a given point in time, the first degree of interference is a representative WIP approximation.

4.2. *Disjunctive graph modeling and evaluation*

Many state-of-the-art solution methods for complex job-shop scheduling problems are based on the disjunctive graph representation (Roy and Sussmann 1964; Ovacik and Uzsoy 1997; Adams, Balas, and Zawack 1988; Dauzère-Pérès and Paulli 1997). Given its applicability, let us adopt the disjunctive graph representation to model the problem when a lot is about to enter a time constraint tunnel. This representation captures all the relevant data, namely: (i) the set of machines required for processing and their qualifications, (ii) the WIP that might have a significant impact on lot f traversing TCT, and (iii) the process flow and routing information of all the lots.

To do this, let $G(\mathscr{J} \cup \{S,E\}, \mathscr{C} \cup \mathscr{D})$ be a disjunctive graph. The set of all operations $\mathscr{J}$ defines the set of nodes together with the dummy start node S and end node E. Let $\mathscr{C}$ denote the set of oriented conjunctive arcs, which model the precedence constraints between the nodes. The set of disjunctive arcs $\mathscr{D}$ models all the possible configurations of operations on the processing machines. Arcs are weighted with processing times, which are both machine- and operation- dependent. The length of the longest path between any two nodes belonging to a route of a given lot is to the cycle time of this lot between the corresponding operations.

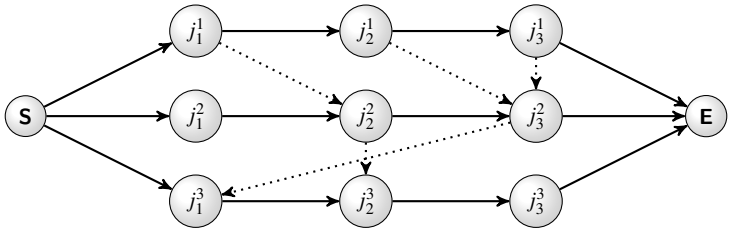

Figure 3.: Conjunctive graph representation. Conjunctive arcs are plain for the routes, and dotted for the sequence on machines (Lima et al. 2017).

Given the number of operations and machines to consider (see the description of the industrial instances in Section 6.1), the time-varying cardinality of the admissible domain of lot schedules suffers of the curse of dimensionality. The selection and orientation of the disjunctive arcs (see Figure 3) can be efficiently performed by using the list scheduling greedy heuristic proposed in (Sadeghi et al. 2015) and improved in (Lima et al. 2017). A new priority-oriented version of this method is formalized in Algorithm 2.

Algorithm 2 relies on a random selection of nodes from a list of nodes available to be scheduled `ReadyList`, according to a given dispatching policy investigated in Section 5. This allows feasible schedules to be always determined as long as the required machine capacity is available by: (i) distinguishing the subset of unscheduled operations (set `ReadyList`) from

Algorithm 2 : Priority-based list scheduling algorithm ($\texttt{TCT}$, $\mathscr{J}$, $\mathscr{J}^f$)

$\texttt{currentTime} = 0$
Initialize the state of machines $m \in \mathscr{M}$ to their current state $\texttt{idle}_m$ to 0 if busy and 1, otherwise
Initialize and sort by increasing values of completion time c_k^ℓ an ordered set $\texttt{ProcessingList}$ with all j_k^ℓ already being processed by the set of machines $\mathscr{M}$ at run time, $\ell \in \mathscr{L}$
Initialize an ordered set $\texttt{ReadyList} = (\mathscr{J} \cup \mathscr{J}^f) \setminus \texttt{ProcessingList}$
while $\texttt{ReadyList} \neq \emptyset$ **do**
  for $j_k^\ell \in \texttt{ReadyList}$ **do**
    Update $\texttt{DAS}^\ell$
    Update $\pi_{\texttt{total}}^\ell$
  end for
  $\pi_{\texttt{Total}} = \sum_{j_k^\ell \in \texttt{ReadyList}} \pi_{\texttt{total}}^\ell$
  while all operations $j_k^\ell \in \texttt{ReadyList}$ have not been checked **do**
    $\texttt{localSum} = 0$
    Let $\texttt{rand}$ to be a random integer between 0 and $\pi_{\texttt{Total}}$
    for all $j_k^\ell \in \texttt{ReadyList}$ **do**
      if j_k^ℓ is not checked **then**
        $\texttt{localSum} = \texttt{localSum} + \pi_{\texttt{total}}^\ell$
        if $\texttt{rand} \leq \texttt{localSum}$ **then**
          Let $m \in \mathscr{M}_k^\ell$ such that $\texttt{idle}_m = 0$ and m is the first such machine
          $c_k^\ell = \texttt{currentTime} + p_{km}^\ell$
          $\texttt{idle}_m = 1$
          $\texttt{ProcessedList} = \texttt{ProcessingList} \cup \{j_k^\ell\}$
          Sort $\texttt{ProcessingList}$
          $\texttt{ReadyList} = \texttt{ReadyList} \setminus \{j_k^\ell\}$
          Mark j_k^ℓ as checked
          $\pi_{\texttt{Total}} = \pi_{\texttt{Total}} - \pi_{\texttt{total}}^\ell$
        end if
      end if
    end for
  end while
  $j_k^\ell = \texttt{ProcessingList}[0]$
  $\texttt{currentTime} = c_k^\ell$
  while $c_k^\ell \leq \texttt{currentTime}$ and $|\texttt{ProcessingList}| \geq 1$ **do**
    $\texttt{ProcessingList} = \texttt{ProcessingList} \setminus \{j_k^\ell\}$
    Randomly insert j_k^ℓ in $\texttt{ReadyList}$
    $\texttt{idle}_m = 0$, where m is the machine that was processing j_k^ℓ
    $j_k^\ell = \texttt{ProcessingList}[0]$
  end while
end while

the operations being processed (set $\texttt{ProcessingList}$) in the set of all operations $\mathscr{J} \cup \mathscr{J}^f$ under evaluation, (ii) updating the priority of lots in $\mathscr{L} \cup \{f\}$, and (iii) assigning operations with the highest priority to an available machine. Being extremely efficient at generating schedules, Algorithm 2 is used to quickly obtain a fairly large number of samples for the sampling-based approach provided in Section 5.

5. Accelerated sampling approach based on a triggering threshold

Given the complex structure and time-varying states of TCTs, deriving an analytical relationship to evaluate the satisfaction of a TCT by a given lot seems impossible. Following our

previous work (Lima et al. 2017), let us statistically infer the successful releasing of a lot in a given TCT via a sampling-based probability estimation approach based-on a triggering threshold. Consistent with the TCT structure and its state at a given point in time, a sequence of binary response experiments is performed by using Algorithm 3.

Let $\xi_{se} \sim Be(p)$ be a Bernoulli random variable representing that a lot satisfies the time constraint $t_{se} \in \mathscr{T}$ with the probability of success $\mathbb{P}\{\xi_{se} = 1\}$. Denote by $\{X_i\}_{se}^{i \geq 1}$ a series of independent Bernoulli trials performed by Algorithm 3. The sample space of each individual trial is $X_{se}^i \in \{0,1\}$.

To determine the empirical probability $\hat{p}_{se}$ of $\mathbb{P}\{\xi_{se} = 1\}$, the cycle time between two operations j_s^f and j_e^f of lot f is computed for each experiment:

$$c_e^f - c_s^f,$$

where c_s^f and c_e^f are the completion times of operations $j_s^f, j_e^f \in \mathscr{J}$. Under the independent Bernoulli trial assumption, the estimated probability of satisfying $t_{se} \in \mathscr{T}$ is calculated as follows:

$$\hat{p}_{se} = \hat{\mathbb{P}}\left\{c_e^f - c_s^f \leq t_{se}\right\} = \frac{\sum_{i=1}^{n} X_{se}^i}{n}, \forall t_{se} \in \mathscr{T}$$

where n is the sampling size and $X_{se}^i = 1$ if the following inequality holds, $i = \overline{1,n}$:

$$c_e^f - c_s^f \leq t_{se}, \quad \forall t_{se} \in \mathscr{T}$$

Sampling-based methods are conventionally categorized into two main classes (Crombecq, Laermans, and Dhaene 2011; Liu, Ong, and Cai 2018; Mendo and Hernando 2006).

- *One-shot sampling:* For a fixed sampling size, all the observations are gathered at once. The following exploitation is then based on this dataset without evaluating any additional samples later. Without prior knowledge on the estimators of interest, it remains difficult, if not impossible, to predetermine an appropriate sample size. Focused on the problem under study, Sadeghi et al. (2015) highlighted, via several computational experiments, the difficulty to determine the right number of runs, and the drawbacks related to the unreliability of the obtained results by a one-shot sampling-based approach.
- *Sequential sampling:* Samples are generated in an iterative manner, i.e. they are analyzed from previous iterations to draw new representative samples. To limit the computational time expenses required by sampling-based approaches, Lima et al. (2017) improved the work of Sadeghi et al. (2015) by transforming the one-shot design into an iterative process, which is stopped when a sampling convergence condition is satisfied. To go further in meeting industrial requirements, we propose a new version of the sequential sampling-based method of Lima et al. (2017) by introducing a triggering threshold.

Note that in the problem under study, nor the probability $\mathbb{P}\{\xi_{se} = 1\}$, neither other distribution parameters of ξ_{se} are available analytically. Without any knowledge on the safety of a given TCT, the probability that a lot leaves the TCT on time is determined based on a specially generated sequential sampling set. More precisely, the exploration strategies and sampling efficiency are discussed in Section 5.1. For the sake of computational time reduction, Section 5.2 addresses the sampling convergence. The integration of a triggering threshold is discussed in Section 5.3.

5.1. *Exploration and sampling efficiency*

Let us discuss two dispatching policies investigated empirically in (Lima et al. 2017):

- **Pure random dispatching policy.** Lots are consecutively assigned to machines with respect to a uniform distribution on the set of unscheduled operations `ReadyList`. While the condition of the stopping rule is not met, feasible schedules are generated by a list scheduling algorithm.
- **Priority-based random dispatching policy.** In industrial practice, each lot is usually endowed with both a *base* priority at the beginning of its product route and a *total priority*. The base priority is fixed and depends on extrinsic factors (e.g. the importance of the original order, the type of lot, the date), while the total priority can vary and is used to smooth the production flow (i.e. to absorb delays). In accordance with these priorities, dispatching rules are defined, with respect to which lots are consecutively assigned to machines.

  One of the most frequently used dispatching rules in semiconductor manufacturing, is the so-called *standard dispatching rule*. This rule expresses the total priority as a function of a base priority factor and the waiting time at each operation. Let $\pi^{\ell}_{\texttt{base}}$ be the base priority of lot $\ell \in L$ and $\pi^{\ell}_{\texttt{total}}$ its total priority, then:

$$\pi^{\ell}_{\texttt{total}}(\pi^{\ell}_{\texttt{base}}, \texttt{DAS}^{\ell}) = \pi^{\ell}_{\texttt{base}}(1 + \texttt{DAS}^{\ell})$$

  where $\texttt{DAS}^{\ell}$ (the short form of Day At Step) is the time spent by lot ℓ waiting to be processed in a given operation. Note that this rule aims at minimizing the total tardiness of lots per unit of base priority.

  Based on the standard dispatching rule, let us tailor the algorithm proposed in (Lima et al. 2017), in order to select the available lots to be scheduled in compliance with their total priority at the current time, instead of a uniformly random drawing scheme. The probability of selection $\mathbb{P}^{\ell}_{\texttt{select}}$ of a lot $\ell \in L$ is thus calculated as follows:

$$\mathbb{P}^{\ell}_{\texttt{select}} = \frac{\pi^{\ell}_{\texttt{total}}(\pi^{\ell}_{\texttt{base}}, \texttt{DAS}^{\ell})}{\sum_{k \in L} \pi^{k}_{\texttt{total}}(\pi^{k}_{\texttt{base}}, \texttt{DAS}^{k})}$$

  The standard dispatching rule is thus explicitly integrated in Algorithm 2. Without neglecting the overall variability, Algorithm 2 lies on the priority system adopted by a semiconductor manufacturing facility. Consequently, the most critical lots have a higher probability of being selected.

The sampling representativeness is essentially based on two pillars, namely *space-fillingness* and *non-collapsingness* (Simpson, Lin, and Chen 2001; Crombecq, Laermans, and Dhaene 2011). With regard to the non-collapsing property, the probability to extract the same schedule is negligible for both dispatching policies: (i) by virtue of a random space-filling based design, and (ii) given that the composition of `ReadyList` and `ProcessingList` are highly variable and their sizes are significant.

The pure random dispatching policy tends to uniformly sample the entire admissible schedule domain, ensuring that every feasible solution in the domain has an equiprobable chance to be selected. The priority-based dispatching policy pushes forward not only feasible solutions from the admissible set, but also realistic ones from an industrial standpoint.

For particular production settings and compared to the priority-based dispatching policy, the sampling approach operating with the pure random policy needs to generate much

more samples in order to be able to exhibit realistic results. One drawback, particular to the priority-based dispatching policy, lies in the time effort required to constantly update lot priorities. However, this computational expense remains negligible with respect to the calculations needed to be performed based on the pure random policy for constructing an industrially representative sampling set.

5.2. *Sampling convergence*

Based on $\{X_i\}_{se}^{i\geq 1}$, we are primarily interested in the point estimator $\hat{p}_{se}$, for which the following features hold:

- The parameter $\hat{p}_{se}$ represents both the mean of the sampling distribution and the probability that a lot satisfies the time constraint $t_{se} \in T$. That is, $\{X_i\}_{se}^{i\geq 1}$ is a consistent estimator for the unknown $\mathbb{P}\{\xi_{se} = 1\}$.
- If n is sufficiently large, the sampling distribution of $\hat{p}_{se}$ is approximately normal, and the confidence interval for $\mathbb{P}\{\xi_{se} = 1\}$ is:

$$\hat{p}_{se} - Z_{\alpha/2}\sqrt{\frac{\hat{p}_{se}(1-\hat{p}_{se})}{n}} \leq p \leq \hat{p}_{se} + Z_{\alpha/2}\sqrt{\frac{\hat{p}_{se}(1-\hat{p}_{se})}{n}}$$

  where α is the probability of type I error and $Z_{\alpha/2}$ is the percentage point of the normal distribution $N(0,1)$ such that $\mathbb{P}\{z \geq Z_{\alpha/2}\} = \frac{\alpha}{2}$.

Generally speaking, the reliability of a point estimation via sampling-based approaches is expressed in terms of either the related confidence level with a given error interval, or accuracy/precision measures (e.g. mean-square error, root mean square deviation). For a desired accuracy level, the required number of trials depends on the statistical properties about the estimate of interest. Nevertheless, as explained in Section 5.1, in many studies of complex systems via simulation, the sampling size is either fixed in advance, or a sequential procedure is applied, in the framework of which the sampling size is iteratively determined by the simulation output, while using knowledge acquired from the simulation itself.

The point estimator $\hat{p}_{se}$ is closer to the true value as the sampling size n increases. However, to confirm the successful releasing of a lot in a given TCT, the sampling size increases as the true probability $\mathbb{P}\{\xi_{se} = 1\}$ decreases, $\forall t_{se} \in \mathscr{T}$. To provide accurate responses in a competitive computational time, we can afford to be less rigorous with respect to the confidence interval, since small probabilities that a lot exits a TCT on time are out of practical interest.

Denote by $\hat{D}_{se}^{i} = \{\hat{p}_{se}^{i}, 1-\hat{p}_{se}^{i}\}$ the empirical probability distribution associated to each $t_{se} \in T$ and $\boldsymbol{D}^{i} = \left\{\mathbb{D}_{se}^{i}\right\}_{t_{se}\in\mathscr{T}}$ the output vector after the i^{th} iteration in Algorithm 3. Let us measure the distance between two probability distributions of each TC obtained at two successive iterations in terms of the Euclidean norm d_2:

$$\begin{aligned} d_2(\hat{D}_{se}^{i-1}, \hat{D}_{se}^{i}) &= \|(\hat{p}_{se}^{i-1} - \hat{p}_{se}^{i}), \big((1-\hat{p}_{se}^{i-1}) - (1-\hat{p}_{se}^{i})\big)\|_2 = \\ &= \sqrt{(\hat{p}_{se}^{i-1} - \hat{p}_{se}^{i})^2 + [(1-\hat{p}_{se}^{i-1}) - (1-\hat{p}_{se}^{i})]^2} = \sqrt{2(\hat{p}_{se}^{i-1} - \hat{p}_{se}^{i})^2} = \sqrt{2}|\hat{p}_{se}^{i-1} - \hat{p}_{se}^{i}| \quad (1) \end{aligned}$$

Lot f is considered to exceed a TCT if at least one time constraint $t_{se} \in \mathscr{T}$ is not respected. Accordingly, the distance d between two successive output vectors $\boldsymbol{D}^{i} = \{\hat{D}_{se}^{i}\}_{t_{se}\in T}$ and $\boldsymbol{D}^{i-1} = \{\hat{D}_{se}^{i-1}\}_{t_{se}\in\mathscr{T}}$ is calculated via the infinite norm $\forall i > 0$:

$$d_\infty(\boldsymbol{D}^{i-1}, \boldsymbol{D}^{i}) = \|\boldsymbol{D}^{i} - \boldsymbol{D}^{i-1}\|_\infty = \max_{t_{se}\in T}\left\{d_2(\hat{D}_{se}^{i-1}, \hat{D}_{se}^{i})\right\}$$

For simplicity, let us discard the constant coefficient in expression (1), since it does not affect the convergence evaluation. Let ε be the desired precision. Following a preliminary study, let us assume a minimal increment MI predetermined, and consider an initial replication count IC so that, for iteration $i > 0$ of Algorithm 2 in Algorithm 3, the replication count is $\mathtt{RC}^i = \mathtt{RC}^{i-1} + \mathtt{MI}$. The generation of new samples stops when the following inequality is verified:

$$d_\infty(\boldsymbol{D}^i, \boldsymbol{D}^{i-1}) \leq \varepsilon \tag{2}$$

5.3. *Sampling triggering threshold*

Prior to this work, Lima et al. (2017) proposed a sampling-based approach drawn on the following considerations: (i) The sampling set is entirely generated at each iteration, i.e. it does not inherit the samples generated during previous iterations, (ii) A weak convergence criterion is defined as a stopping condition (see Inequality (2)). In what follows, let us improve this sampling procedure by using the following ingredients, that rely on the fact that the collection $\{X_i\}_{se}^{i\geq 1}$ is independent and identically distributed, and thus that incremental sampling techniques are applicable:

- The initial iteration count IC is sufficiently representative and can be estimated in the framework of a preliminary observation phase (Lima et al. 2017).
- In compliance with the decision maker specifications, the point estimators $\hat{p}_{se}, \forall t_{se} \in \mathscr{T}$, are evaluated using:
  - θ: A nominal value for each point estimator $\hat{p}_{se}, \forall t_{se} \in \mathscr{T}$,
  - $\theta - \beta$: A triggering threshold, where $\beta \in [0, 1-\theta]$ is a tolerance margin.

  Let us express the distance between a point estimator $\hat{p}_{se}$ and the nominal value θ by means of a triggering threshold $\theta - \beta$. A point estimator $\hat{p}_{se}$ is considered far from the nominal value θ if $\hat{p}_{se} < \theta - \beta$, and has an insignificant probability to reach the nominal value θ. The successful release of a lot in a TCT implies that $\hat{p}_{se} \geq \theta, \forall t_{se} \in \mathscr{T}$. Consequently, the existence of at least one point estimator $\hat{p}_{se}$ far from the nominal value θ, has a prohibitive effect on the successful release of a lot in a TCT.

  Given that the initial iteration count IC is sufficiently representative to evaluate the distance between all point estimators and nominal value θ, it is not relevant from a practical point of view to continue the sampling process if $\exists t_{se} \in \mathscr{T}$ such that $\hat{p}_{se}^i < \theta - \beta$.
- Apart from the convergence-based stopping condition (2), let us translate the triggering threshold into algorithm stopping conditions. The algorithm execution stops when at least one of the following inequalities holds:

$$\left[\begin{array}{ll} d_\infty(\boldsymbol{D}^{i-1}, \boldsymbol{D}^i) \leq \varepsilon & \\ \hat{p}_{se}^i \geq \theta, & \forall t_{se} \in \mathscr{T} \\ \hat{p}_{se}^i < \theta - \beta, & \exists t_{se} \in \mathscr{T} \end{array}\right. \tag{3}$$

  The set of inequalities (3) involves relevant implications to the industrial decision support in both cases: (i) If we are confident that, according to the current iteration, there is a high probability level that the lot crosses all TCs with at least a confidence level of θ, or (ii) One can already provide the corresponding negative output when there exists at least one TC with a probability below $\theta - \beta$.
- Define the sampling procedure as follows:
  - Let the initial sampling set contain RC = IC iterations,

- Add newly generated MI samples to the previous sampling set at each iteration,
- Stop when at least one inequality in (3) is verified.

Algorithm 3 : List scheduling algorithm with triggering threshold ($\mathscr{J}^f$, TCT, IC, MI, $\varepsilon, \theta, \beta$)

$\mathscr{J}$ = Algorithm 1 ($\mathscr{J}^f$)
RC = IC
$\boldsymbol{D}^1 = \emptyset$ is the array containing the current resulting values of $\{\hat{p}_{se}\}_{t_{se} \in \mathscr{T}}$
upperStoppingCondition = false
lowerStoppingCondition = false
$i = 1$
repeat
  $\boldsymbol{D}^{i-1} = \boldsymbol{D}^i$
  $\boldsymbol{D}^i$ = Algorithm 2 (TCT, $\mathscr{J}$, $\mathscr{J}^f$) iterated RC times
  Compute $d_\infty(\boldsymbol{D}^i, \boldsymbol{D}^{i-1})$
  if $\hat{p}^n_{se} > \theta, \forall \hat{p}^i_{se} \in \hat{D}^i$ **then**
    upperStoppingCondition = true
  end if
  if $\exists t_{se} \in \mathscr{T}$ such that $\hat{p}^i_{se} < \theta - \beta$ **then**
    lowerStoppingCondition = true
  end if
  $i = i + 1$
  RC = RC+MI
until upperStoppingCondition = true or lowerStoppingCondition = true or $d_\infty(\boldsymbol{D}^i, \boldsymbol{D}^{i-1}) \leq \varepsilon$

The incremental sampling approach based on a triggering threshold is formalized in Algorithm 3. By virtue of its design, the proposed accelerated probability estimation approach has several advantages. The strengthened stopping condition via a triggering threshold $\theta - \beta$ can significantly decrease the execution time of the sampling procedure. The decision maker has thus the possibility to explicitly indicate the desired decision-support accuracy, besides the convergence precision ε.

6. Numerical experiments

Highly heterogeneous industrial instances have been used to perform extensive computational experiments for a twofold purpose: (i) to evaluate the competitiveness of Algorithm 3 with respect to the sampling-based approach of Lima et al. (2017) in terms of computational time and solution quality (see Section 6.3), and (ii) to empirically examine the resilience of Algorithm 3 against different possible use cases in the fab, by observing the behavior and the magnitude of the output fluctuations for different state of the TCTs (see Section 7).

6.1. *Instance description*

Industrial instances $I \in \{1, 2, \ldots, 10\}$ have been constructed by taking a full fab snapshot of a concatenation of different models and reports, some of which are directly linked to the Manufacturing Execution System of the fab. They 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 representative TCTs have been thus selected: `TCT1, TCT2, TCT3, TCT4, TCT5`. Note that these TCTs cover all the critical fab workshop areas, that are the most affected by time constraints. Each TCT includes various numbers of operations, and can contain up to four time constraints. As shown in Table 1, the instances sizes are very large with an average of 600 lots per instance, 7,000 operations to schedule, and between 30 and 200 machines. Note that the instances are ordered chronologically.

Table 1.: Description of industrial instances

I	TCT1		TCT2		TCT3		TCT4		TCT5	
	# Nodes	# Lots	# Nodes	# Lots	# Nodes	# Lots	# Nodes	# Lots	# Nodes	# Lots
1	-	-	3,729	545	796	256	3,944	685	5,161	707
2	64,231	380	-	-	734	268	-	-	5,356	673
3	56,388	404	4,322	639	1,044	346	2,058	577	4,743	666
4	-	-	-	-	1,308	437	4,277	864	3,455	640
5	-	-	4,918	774	1,262	492	4,130	859	3,650	689
6	-	-	4722	741	2,213	554	2,430	407	5,821	846
7	-	-	5,550	807	2,391	558	-	-	6,191	866
8	-	-	5,107	745	1,979	476	2,423	428	5,962	872
9	-	-	-	-	1,529	466	-	-	-	-
10	-	-	-	-	1,589	467	-	-	-	-

Due to machines breakdowns or recipe qualifications issues when instances were recorded, a small disparity can be observed relative to the available instances per TCT. The selected instances can be sensitive to machine breakdowns or even planned maintenance activities, since it is not possible to exactly know the state of the machines before it changes. As far as instances $I \in \{9,10\}$ are concerned, the lack of corresponding data for `TCT4` is due to a modification of its modeling in the fab related to its stringent impact on production. This however sheds an interesting light on the results obtained for `TCT4`.

6.2. *Parameter selection*

Let us exploit the findings obtained during a preliminary observation phase discussed in (Lima et al. 2017) and assume that $\varepsilon = 0.05$, $\texttt{IC} = 30$ and $\texttt{MI} = 20$. The value of the convergence precision parameter ε is context-dependent and, in our case, particularly suited to the industrial problem at hand. The preliminary observation phase showed that 30 is a sufficient starting initial iteration count, as the convergence is usually achieved around 60 replications for the priority-based scheduling policy. Thus, a reasonable half of the representative solution space is initially explored. The parameter choice for ε, `IC` and `MI` are reinforced by the numerical experiments in Section 6.3.

To study the sensitivity of the triggering threshold, different values of θ and β have been selected. Note that when $\theta + \beta > 1$, the upper bound $\min\{1, \theta + \beta\}$ is taken. From a purely industrial point of view, these parameters would be set according to the requirements of decision makers, in particular regarding the nominal value θ.

- $\theta \in \{0.8, 0.9\}$: These values are specially high to better comply with the problem industrial dimension and a low risk approach, which are relevant when dealing with TCTs.
- $\beta \in \{0.05, 0.1, 0.2\}$: These values allow for a good assessment of the tolerance margin impact both on the decision support provided by Algorithm 3 and its overall computational time.

Computational experiments have been carried out on a workstation with 8-core Intel(R) Xeon(R) CPU E3-1240 v5 clocked @ 3.50 giga-hertz.

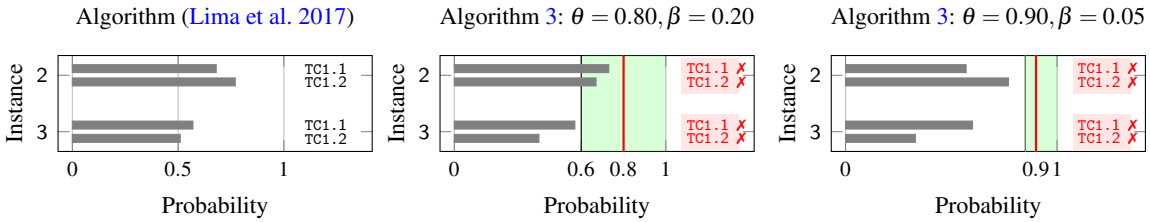

Figure 4.: Computational analysis: TCT1

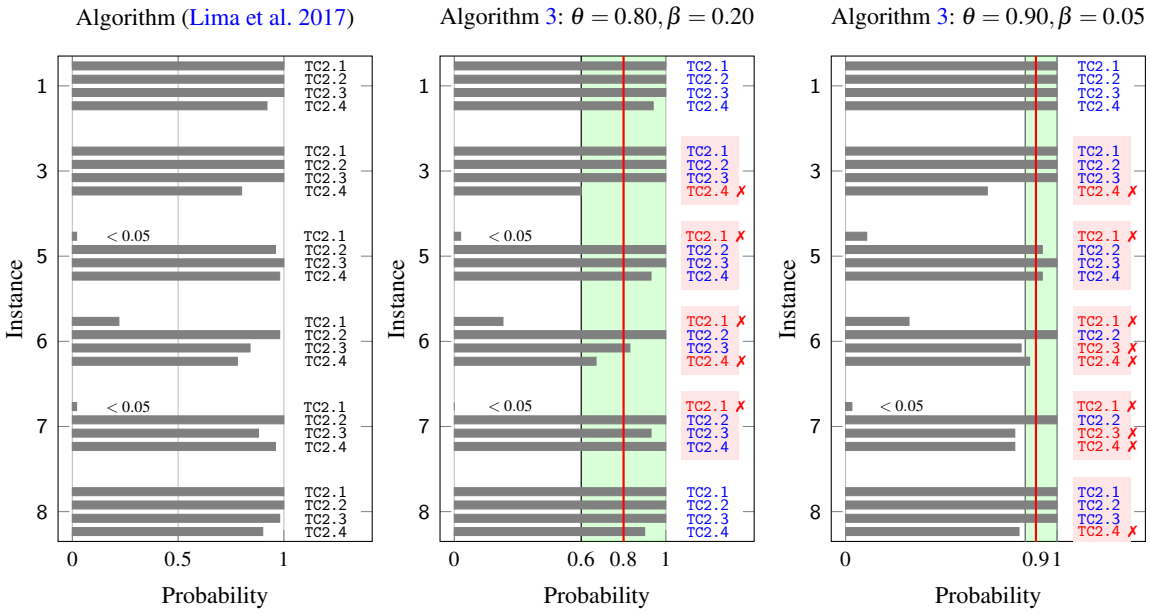

Figure 5.: Computational analysis: TCT2

6.3. *Numerical analysis*

Sampling-based approach in (Lima et al. 2017) versus Algorithm 3. For each TCT and each instance, the savings of Algorithm 3 compared to Algorithm (Lima et al. 2017) are given in the last two columns of Tables B1-B5, and are calculated as follows:

$$\overline{\Delta n} = \frac{1}{6} \sum_{\substack{\theta \in \{0.8, 0.9\}, \\ \beta = \{0.05, 0.1, 0.15\}}} \Big[n\big(\text{Algorithm (Lima et al. 2017)}\big) - n\big(\text{Algorithm 3}(\theta, \beta)\big) \Big]$$

$$\overline{\Delta \text{CPU}} = \frac{1}{6} \sum_{\substack{\theta \in \{0.8, 0.9\}, \\ \beta = \{0.05, 0.1, 0.15\}}} \Big[\text{CPU}\big(\text{Algorithm (Lima et al. 2017)}\big) - \text{CPU}\big(\text{Algorithm 3}(\theta, \beta)\big) \Big]$$

Figures 4-8 and Tables B1-B5 show the impact of the stopping conditions based on a triggering threshold, in terms of the output quality and computational times. With negligible output quality losses, Algorithm 3 reduces the computational times by about 45%.

Let us consider in particular instances $I \in \{2, 3\}$ of TCT1, for which the number of iterations and execution time are significantly larger: 402 seconds for instance $I = 2$ and 378 seconds for instance $I = 3$ versus 125 seconds and 152 seconds (see Table B1). All of this is done for a fairly similar output quality (see Figures 4-8). Note that TCT1, which includes only a single

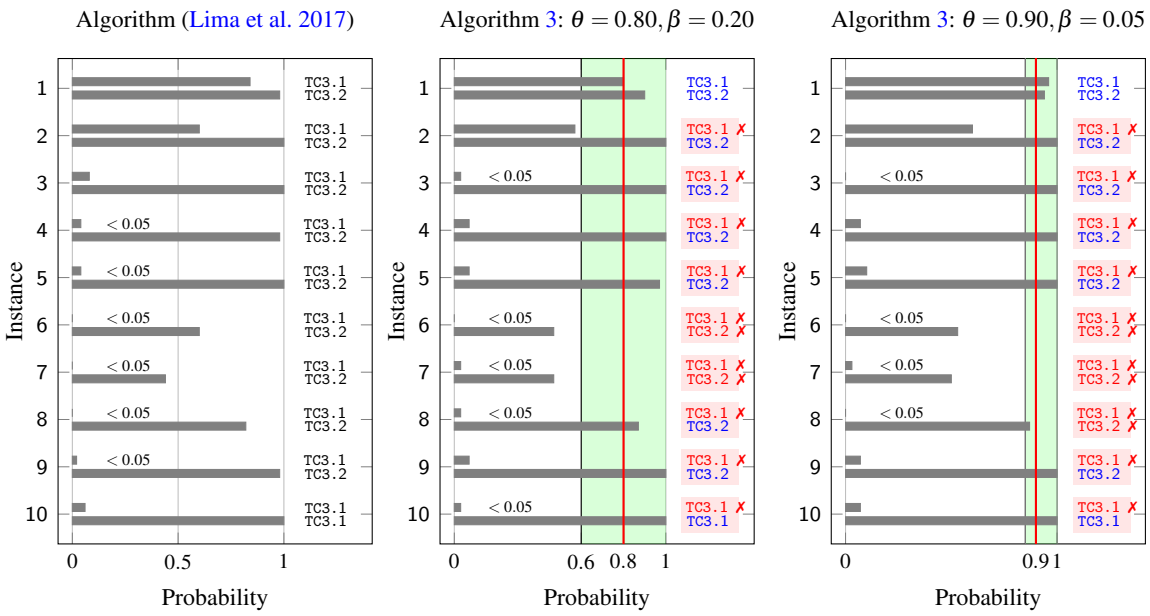

Figure 6.: Computational analysis: TCT3

TC located in a very loaded area of the fab counting more than 60,000 nodes, is a notable exception in terms of CPU time (see Table 1). In this respect, it is important to observe the linear scalability of Algorithm 3 depending on number of nodes in the graph.

The most notable improvement over Algorithm (Lima et al. 2017) comes from the incremental sampling procedure and the strengthening of the stopping conditions via the set of conditions (3). As exhibited in the numerical results, this upgrade leads to a drastic decrease of the computational times by up to 65%, lower than one minute for many instances (see Tables B1-B5).

Impact of the triggering threshold and role of stopping conditions (3). As expected and shown in Figures 4-8, the larger β, the more pessimistic the decision support, while the execution time trends upward (see Tables B1-B5).

Unsurprisingly, the threshold θ has no significant impact on the CPU time, since θ is context-dependent and expresses the level of risk tolerated by decision makers. On the contrary, the tolerance margin β affects the CPU time. As highlighted for example by TCT4 (instance $I = 1$), the iteration count required to achieve one of the stopping conditions is larger for $\beta = 0.20$ (see Table B4). This trend is verified across all instances except for extreme cases, for which the initial obtained probability is very low (see instances $I \in \{3,4\}$ of TCT3 in Table B3).

As reflected in Figure 6 for TCT3 (instances $I = 3$ and up), there is no interest in continuing to iterate when the tunnel is clearly not passing. Generally speaking, Inequalities (3) are easier to fulfill than Inequality (2) (see the penultimate column in Tables B1-B5). However, there are cases when Inequality (2) is clearly verified, e.g. instance $I = 1$ of TCT3.

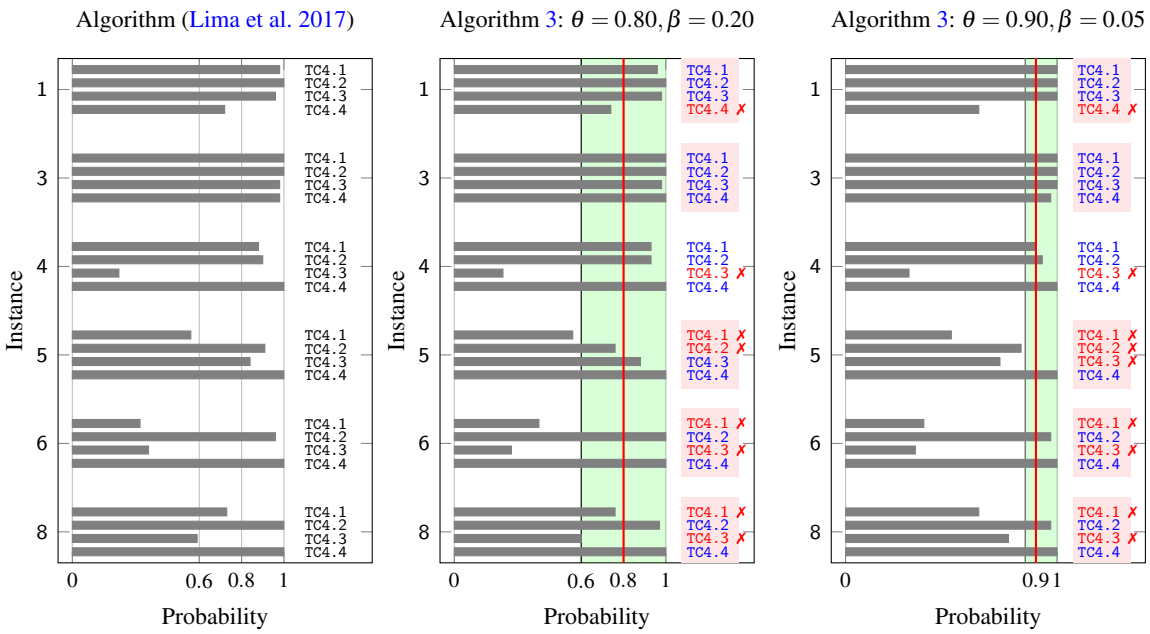

Figure 7.: Computational analysis: TCT4

7. Managerial implications

The management of time constraints is a challenging task, requiring to balance between product yield and cycle time, whose complexity is growing with the increasing number of TCTs induced by the evolution of product technologies. Nowadays, the decision of releasing a lot in a TCT is usually manual. This is a time consuming and stressful task, as decision makers have to make critical decisions in a short amount of time on potentially extremely expensive products all day long. Decision makers are under pressure to maximize the throughput of TCTs without violating time constraints to prevent lots to be scrapped or reworked. Hence, there is a strong need for reliable and relevant decision support.

The proposed sampling-based approach helps to support the transition from a human gatekeeping decision to whether or not to release a lot in a TCT, towards an automated lot release system. Such a system could be used without human intervention to release lots in TCTs, or jointly with a dispatching or scheduling system. In this line, the output of Algorithm 3 provides insights from different perspectives.

From a qualitative standpoint, the comparison between the estimated probability and the nominal value θ provides an immediate information regarding the possibility of releasing a lot in the context encased by the instance. For example, a lot has a high probability of not exceeding `TC5.1-TC5.3`, way above θ in most configurations for `TCT5`, as shown in Figure 8. On the other hand, it is readily apparent that `TC5.4` has very low probability values, to the point where it would be pointless to release a lot at that given point in time. In addition, the result analysis can also provide valuable insights on the position of blocking elements in TCTs. Since `TC5.4` is the last TC of `TCT5`, one might assume that a machine is down at the end of the tunnel or a WIP accumulation blocks the last few operations. Of course, these elements strictly speaking are not a sufficient proof to accurately identify the root of the problem, but in combination with the decision maker's expertise, they can contribute to, or hasten, the process of detecting issues to solve inside the tunnel.

`TCT5` is also an example of how it is possible to characterize whether the tunnel is in a

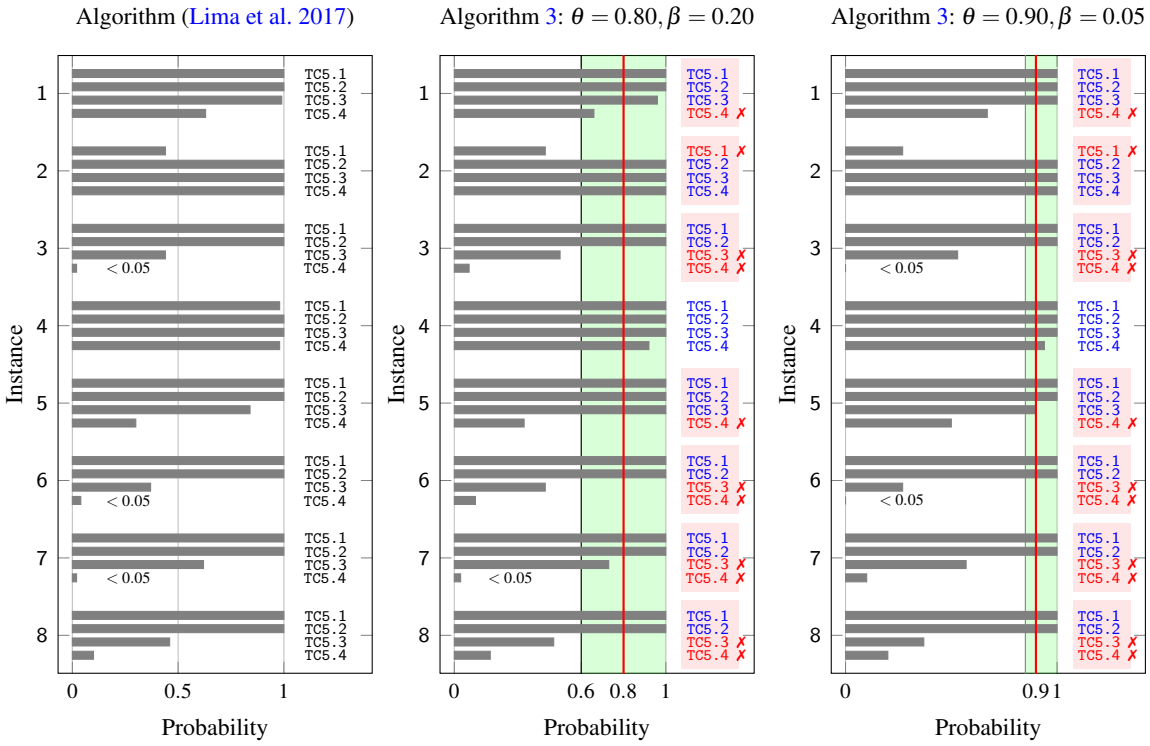

Figure 8.: Computational analysis: TCT5

passing or non-passing state. This is dependent on whether there are machine breakdowns, planned maintenance activities, or considerable WIP in the TCT. Based on the probability estimate and nominal value θ, it is possible to determine whether or not to release a lot. For example, TCT5 clearly presents a passing behavior for instance $I = 4$. On the contrary, TCT3 (a small TCT in terms of number of nodes) shows degrading results starting from instance $I = 2$ and onward, which reflects the load up of this specific area of the fab over the months. As shown in Figure 6, TCT3 becomes non passing after instance $I = 2$ even for relaxed triggering thresholds, which can be explained by the priorization of a specific technology that does not include the product type of the lot under study.

From a quantitative standpoint, even though the number of iterations can statistically speaking appear low, the output given by Algorithm 3 can be converted into useful information on the situation inside TCTs at a given point in time. For example, let us examine instance $I = 5$ of TCT4, and to a lesser extent instance $I = 8$ in Table B4. The results for TC4.1 are around 50%, while the results for TC4.2-TC4.4 are almost all above $\theta = 0.80$. Being the first TC in TCT4, the decision maker has the most leverage on TC4.1, in terms of lot speed up and manual release of other lots through the tunnel. Despite the low probability of successful traversal of TC4.1, the decision maker, aware of the favorable situation of TC4.2-TC4.4, might send the lot, while prioritizing it or speeding it up to guarantee the satisfaction of TC4.1. In this particular case, a 50% chance of exiting TC4.1 on time is a useful information, as the decision would not have been the same, in the situation when a lot has e.g. less that a 30% probability to pass through TC4.2-TC4.4.

The nominal value θ of the point estimator defines how much risk can be taken in the decision support provided by Algorithm 3. The large values used for θ are consistent with a very low risk approach when managing time constraints. Direct feedback from the decision makers, i.e. the time constraint regulators, are important to fine tune this parameter.

To summarize, the numerical experiments conducted on large-scale real-life instances show the industrial applicability of the proposed decision support approach. The results have been obtained in less than one minute in most cases, a computational time which is very close to near real time from a production point of view. Besides the computational competitiveness of the proposed approach, it appears to be: (i) *Flexible* via the introduced triggering threshold, (ii) *Resilient* against heterogeneous instances, and (iii) *Powerful* in terms of decision support.

8. Conclusion and perspectives

This paper focuses on the problem of managing time constraint tunnels, particularly relevant in semiconductor manufacturing. Together with an accurate problem representation, that is able to capture the industrial features and to benefit from its real-life practices, a sampling-based probability estimation approach is proposed. Being drawn on efficient state-of-the-art methods, this approach extends the previous research of Sadeghi et al. (2015) and Lima et al. (2017) preponderantly in two ways: (i) an accelerated version of the algorithm given in (Lima et al. 2017) is provided, and (ii) the appropriateness of the problem modeling jointly with the industrial soundness of the solution method is empirically highlighted via extensive computational experiments conducted on real-life instances. We believe that the proposed decision tool can be used in other production systems subject to time constraints.

In spite of the practical interest and decision support offered by the proposed problem modeling jointly with an efficient solution method, there are still several ways in which the decision tool can be improved:

- By introducing stochastic machine state changes into the list scheduling algorithm to account for unplanned machine breakdowns,
- By explicitly integrating batching machine processes,
- By extending the approach to the simultaneously release of multiple lots in a given tunnel.

The last research direction raises a number of open questions (Lima et al. 2019). What is the maximum number of lots that can be released in a given TCT, while satisfying all corresponding time constraints? If lots have different priorities, how to evaluate and construct the best successful lot mix? How to quantify and handle the correlation between lots sent in the same TCT?

Acknowledgments

This work has been partially financed by the ANRT (Association Nationale de la Recherche Technique) through the PhD number 2015/0899 with CIFRE funds and a cooperation contract between STMicroelectronics and Mines Saint-Etienne.

Appendix A. Notations

Table A1.: Notations

$\mathscr{L}$	Set of lots, indexed by $\ell \in \mathscr{L}$
$\mathscr{J}$	Set of operations, indexed by $j \in \mathscr{J}$
$\mathscr{M}$	Set of machines, indexed by $m \in \mathscr{M}$
$\mathscr{T}$	set of time constraints of a given TCT
f	Lot about to enter a TCT
$t_{se} \in \mathscr{T}$	Time constraint starting with $j_s^\bullet$ and ending after $j_e^\bullet$, $j_s^\bullet, j_e^\bullet \in \mathscr{J}$
c_e^f	Completion time of operation $j_e^f \in \mathscr{J}$
p_{km}^ℓ	Machine-dependent processing time
$\texttt{DAS}^\ell$	Time spent by lot ℓ waiting to be processed in a given operation
$G(\mathscr{J} \cup \{S,E\}, \mathscr{C} \cup \mathscr{D})$	Disjunctive graph defined by: (i) The set of nodes $\mathscr{J}$, a fictitious start node S and a fictitious end node E, and (ii) The union of the set of conjunctive arcs $\mathscr{C}$ and the set of disjunctive arcs $\mathscr{D}$
$\pi_{\texttt{base}}^\ell$	Base priority of lot $\ell \in \mathscr{L}$
$\pi_{\texttt{total}}^\ell$	Total priority of lot $\ell \in \mathscr{L}$
$\mathbb{P}_{\texttt{select}}^\ell$	Probability to select lot $\ell \in \mathscr{L}$
$\xi_{se} \sim Be(p)$	Bernoulli random variable
$\{X_i\}_{se}^{i \geq 1}$	Series of independent Bernoulli trials
$\hat{p}_{se}$	Empirical probability that a lot satisfies constraint $t_{se} \in \mathscr{T}$
$\hat{D}_{se}^i = \{\hat{p}_{se}^i, 1-\hat{p}_{se}^i\}$	Empirical probability distribution associated to $t_{se} \in \mathscr{T}$
n	Sampling size
θ	Nominal value for the point estimator
$\beta \in [0, 1-\theta]$	Tolerance margin
$\theta - \beta$	Triggering threshold
IC	Initial replication count
MI	Minimal replication increment
ε	Convergence precision

Appendix B. Computational results

Table B1.: [1,2]Computational results for TCT1

I	Algorithm (Lima et al. 2017)				Algorithm 3						$\overline{\Delta n}$	$\overline{\Delta \text{CPU}}$
	TC 1.1	**TC 1.2**	**n**	**CPU**	θ	β	**TC 1.1**	**TC 1.2**	**n**	**CPU**		
2	0.68	0.77	90	402	0.80	0.05	0.77	0.80	30	125	60	277 (-69%)
						0.10	0.57	0.73	30	125		
						0.20	0.73	0.67	30	125		
					0.90	0.05	0.57	0.77	30	125		
						0.10	0.73	0.73	30	124		
						0.20	0.77	0.77	30	124		
3	0.57	0.51	70	378	0.80	0.05	0.67	0.33	30	152	40	226 (-60%)
						0.10	0.50	0.53	30	152		
						0.20	0.57	0.40	30	153		
					0.90	0.05	0.60	0.33	30	153		
						0.10	0.73	0.53	30	152		
						0.20	0.50	0.43	30	151		
										Mean	50	252 (-65%)

1. The relatives gaps are given in parenthesis.
2. The running times (CPU) are given in seconds.

Table B2.: [1,2]Computational results for TCT2

I	Algorithm (Lima et al. 2017)						Algorithm 3								$\overline{\Delta n}$	$\overline{\Delta \text{CPU}}$
	TC 2.1	**TC 2.2**	**TC 2.3**	**TC 2.4**	**n**	**CPU**	θ	β	**TC 2.1**	**TC 2.2**	**TC 2.3**	**TC 2.4**	**n**	**CPU**		
1	1.00	1.00	1.00	0.92	50	26	0.80	0.05	1.00	1.00	1.00	0.87	30	15	10	6 (-23%)
								0.10	1.00	1.00	1.00	0.93	30	15		
								0.20	1.00	1.00	1.00	0.94	50	26		
							0.90	0.05	1.00	1.00	1.00	1.00	30	15		
								0.10	1.00	1.00	1.00	0.92	50	26		
								0.20	1.00	1.00	1.00	0.92	50	26		
3	1.00	1.00	1.00	0.80	50	57	0.80	0.05	1.00	1.00	1.00	0.80	50	56	13	17 (-30%)
								0.10	1.00	1.00	0.97	0.70	30	33		
								0.20	1.00	1.00	1.00	0.60	30	32		
							0.90	0.05	1.00	1.00	1.00	0.67	30	33		
								0.10	1.00	1.00	1.00	0.70	30	32		
								0.20	1.00	0.98	1.00	0.88	50	56		
5	0.02	0.96	1.00	0.98	50	76	0.80	0.05	0.03	0.97	0.97	1.00	30	44	20	31 (-41%)
								0.10	0.10	1.00	1.00	0.97	30	45		
								0.20	0.03	1.00	1.00	0.93	30	45		
							0.90	0.05	0.10	0.93	1.00	0.93	30	44		
								0.10	0.03	0.97	1.00	0.97	30	45		
								0.20	0.13	0.90	1.00	1.00	30	45		
6	0.22	0.98	0.84	0.78	50	70	0.80	0.05	0.20	1.00	0.80	0.87	30	40	20	30 (-43%)
								0.10	0.20	1.00	0.87	0.93	30	41		
								0.20	0.23	1.00	0.83	0.67	30	40		
							0.90	0.05	0.30	1.00	0.83	0.87	30	40		
								0.10	0.20	1.00	0.87	0.80	30	41		
								0.20	0.17	0.97	0.90	0.70	30	41		
7	0.02	1.00	0.88	0.96	50	92	0.80	0.05	0.03	1.00	0.90	0.90	30	53	20	39 (-42%)
								0.10	0.03	1.00	0.87	0.93	30	53		
								0.20	0.00	1.00	0.93	1.00	30	52		
							0.90	0.05	0.03	1.00	0.80	0.80	30	53		
								0.10	0.03	1.00	0.93	0.97	30	53		
								0.20	0.00	0.97	0.93	0.93	30	53		
8	1.00	1.00	0.98	0.90	50	76	0.80	0.05	1.00	1.00	0.96	0.80	50	75	3	6 (-8%)
								0.10	1.00	0.97	1.00	0.93	30	44		
								0.20	1.00	1.00	1.00	0.90	50	75		
							0.90	0.05	1.00	1.00	1.00	0.82	50	75		
								0.10	1.00	1.00	0.98	0.92	50	75		
								0.20	0.98	1.00	1.00	0.84	50	75		
														Mean	14	22 (-31%)

1. The relatives gaps are given in parenthesis.
2. The running times (CPU) are given in seconds.

Table B3.: [1,2]Computational results for TCT3

I	Algorithm (Lima et al. 2017)				Algorithm 3						$\overline{\Delta n}$	$\overline{\Delta \text{CPU}}$ (s)
	TC 3.1	**TC 3.2**	**n**	**CPU** (s)	θ	β	**TC 3.1**	**TC 3.2**	**n**	**CPU** (s)		
1	0.84	0.98	50	2	0.80	0.05	0.78	1.00	50	2	-3	0 (0%)
						0.10	0.86	0.96	50	1		
						0.20	0.80	0.90	50	2		
					0.90	0.05	0.96	0.94	50	2		
						0.10	0.87	0.99	70	2		
						0.20	0.80	0.92	50	1		
2	0.60	1.00	70	2	0.80	0.05	0.53	1.00	30	1	33	1 (-50%)
						0.10	0.57	1.00	30	1		
						0.20	0.57	1.00	70	2		
					0.90	0.05	0.60	1.00	30	1		
						0.10	0.67	1.00	30	1		
						0.20	0.63	1.00	30	1		
3	0.08	1.00	50	5	0.80	0.05	0.03	1.00	30	3	20	2 (-40%)
						0.10	0.10	1.00	30	3		
						0.20	0.03	1.00	30	3		
					0.90	0.05	0.00	1.00	30	3		
						0.10	0.00	1.00	30	3		
						0.20	0.03	1.00	30	3		
4	0.04	0.98	50	9	0.80	0.05	0.07	1.00	30	5	20	4 (-44%)
						0.10	0.00	1.00	30	5		
						0.20	0.07	1.00	30	6		
					0.90	0.05	0.07	1.00	30	5		
						0.10	0.00	1.00	30	5		
						0.20	0.03	1.00	30	5		
5	0.04	1.00	70	15	0.80	0.05	0.13	1.00	30	6	40	9 (-60%)
						0.10	0.03	1.00	30	6		
						0.20	0.07	0.97	30	6		
					0.90	0.05	0.10	1.00	30	6		
						0.10	0.00	1.00	30	6		
						0.20	0.00	1.00	30	6		
6	0.00	0.60	50	16	0.80	0.05	0.00	0.40	30	10	20	7 (-44%)
						0.10	0.00	0.53	30	9		
						0.20	0.00	0.47	30	9		
					0.90	0.05	0.00	0.53	30	9		
						0.10	0.00	0.57	30	9		
						0.20	0.00	0.33	30	9		
7	0.00	0.44	70	25	0.80	0.05	0.00	0.60	30	10	40	15 (-60%)
						0.10	0.00	0.60	30	10		
						0.20	0.03	0.47	30	10		
					0.90	0.05	0.00	0.50	30	10		
						0.10	0.03	0.50	30	10		
						0.20	0.00	0.63	30	10		
8	0.00	0.82	50	10	0.80	0.05	0.03	0.80	30	6	20	4 (-40%)
						0.10	0.03	0.80	30	6		
						0.20	0.03	0.87	30	6		
					0.90	0.05	0.00	0.87	30	6		
						0.10	0.00	0.90	30	6		
						0.20	0.00	1.00	30	6		
9	0.02	0.98	50	9	0.80	0.05	0.03	1.00	30	5	20	4 (-44%)
						0.10	0.03	1.00	30	5		
						0.20	0.07	1.00	30	6		
					0.90	0.05	0.07	1.00	30	5		
						0.10	0.10	1.00	30	5		
						0.20	0.03	1.00	30	5		
10	0.06	1.00	50	9	0.80	0.05	0.03	1.00	30	5	20	4 (-44%)
						0.10	0.03	1.00	30	5		
						0.20	0.03	1.00	30	5		
					0.90	0.05	0.07	1.00	30	5		
						0.10	0.07	0.97	30	5		
						0.20	0.00	1.00	30	5		
										Mean	23	5 (-43%)

1. The relatives gaps are given in parenthesis.
2. The running times (CPU) are given in seconds.

Table B4.: [1,2]Computational results for TCT4

I	Algorithm (Lima et al. 2017)						Algorithm 3								$\overline{\Delta n}$	$\overline{\Delta \text{CPU}}$ (s)
	TC 4.1	**TC 4.2**	**TC 4.3**	**TC 4.4**	**n**	**CPU** (s)	θ	β	**TC 4.1**	**TC 4.2**	**TC 4.3**	**TC 4.4**	**n**	**CPU** (s)		
1	0.98	1.00	0.96	0.72	50	37	0.80	0.05	0.97	1.00	1.00	0.63	30	21	10	9 (-41%)
								0.10	1.00	1.00	0.98	0.70	50	36		
								0.20	0.96	1.00	0.98	0.74	50	36		
							0.90	0.05	1.00	1.00	1.00	0.63	30	20		
								0.10	1.00	1.00	0.97	0.67	30	21		
								0.20	1.00	1.00	0.96	0.64	50	36		
3	1.00	1.00	0.98	0.98	50	23	0.80	0.05	1.00	1.00	1.00	0.97	30	14	10	5 (-65%)
								0.10	1.00	1.00	1.00	1.00	30	13		
								0.20	1.00	1.00	0.98	1.00	50	23		
							0.90	0.05	1.00	1.00	1.00	0.97	30	13		
								0.10	1.00	1.00	1.00	1.00	50	23		
								0.20	1.00	1.00	1.00	0.98	50	23		
4	0.88	0.90	0.22	1.00	50	84	0.80	0.05	0.93	0.90	0.30	1.00	30	48	20	36 (-18%)
								0.10	0.90	0.90	0.33	1.00	30	48		
								0.20	0.93	0.93	0.23	1.00	30	48		
							0.90	0.05	0.90	0.93	0.30	1.00	30	48		
								0.10	0.80	0.93	0.30	1.00	30	48		
								0.20	0.93	0.97	0.23	1.00	30	48		
5	0.56	0.91	0.84	1.00	70	117	0.80	0.05	0.60	0.83	0.87	1.00	30	47	37	64 (-13%)
								0.10	0.47	0.83	0.77	1.00	30	47		
								0.20	0.56	0.76	0.88	1.00	50	81		
							0.90	0.05	0.50	0.83	0.73	1.00	30	47		
								0.10	0.30	0.80	0.83	1.00	30	47		
								0.20	0.43	0.93	0.90	1.00	30	47		
6	0.32	0.96	0.36	1.00	50	19	0.80	0.05	0.47	0.93	0.27	1.00	30	11	20	8 (-79%)
								0.10	0.50	0.97	0.27	1.00	30	11		
								0.20	0.40	1.00	0.27	1.00	30	11		
							0.90	0.05	0.37	0.97	0.33	1.00	30	11		
								0.10	0.43	0.93	0.30	1.00	30	11		
								0.20	0.23	0.97	0.20	1.00	30	11		
8	0.73	1.00	0.59	1.00	70	25	0.80	0.05	0.57	1.00	0.63	1.00	30	10	33	13 (-60%)
								0.10	0.73	1.00	0.67	1.00	30	10		
								0.20	0.76	0.97	0.60	1.00	70	25		
							0.90	0.05	0.63	0.97	0.77	1.00	30	10		
								0.10	0.73	1.00	0.83	1.00	30	10		
								0.20	0.70	1.00	0.83	1.00	30	10		
														Mean	22	22 (-46%)

1. The relatives gaps are given in parenthesis.
2. The running times (CPU) are given in seconds.

Table B5.: [1,2]Computational results for TCT5

I	Algorithm (Lima et al. 2017)						Algorithm 3								$\overline{\Delta n}$	$\overline{\Delta \text{CPU}}$ (s)
	TC 5.1	**TC 5.2**	**TC 5.3**	**TC 5.4**	**n**	**CPU** (s)	θ	β	**TC 5.1**	**TC 5.2**	**TC 5.3**	**TC 5.4**	**n**	**CPU** (s)		
1	1.00	1.00	0.99	0.63	70	69	0.80	0.05	1.00	1.00	0.98	0.76	50	44	27	30 (-44%)
								0.10	1.00	1.00	0.97	0.67	30	26		
								0.20	1.00	1.00	0.96	0.66	50	48		
							0.90	0.05	1.00	1.00	1.00	0.67	30	26		
								0.10	1.00	1.00	1.00	0.67	30	26		
								0.20	1.00	1.00	0.99	0.79	70	63		
2	0.44	1.00	1.00	1.00	50	43	0.80	0.05	0.33	1.00	1.00	1.00	30	24	20	19 (-44%)
								0.10	0.20	1.00	1.00	1.00	30	24		
								0.20	0.43	1.00	1.00	1.00	30	24		
							0.90	0.05	0.27	1.00	1.00	1.00	30	24		
								0.10	0.43	1.00	1.00	1.00	30	24		
								0.20	0.23	1.00	1.00	1.00	30	24		
3	1.00	1.00	0.44	0.02	50	59	0.80	0.05	1.00	1.00	0.63	0.03	30	34	20	26 (-44%)
								0.10	1.00	1.00	0.57	0.07	30	33		
								0.20	1.00	1.00	0.50	0.07	30	33		
							0.90	0.05	1.00	1.00	0.53	0.00	30	34		
								0.10	1.00	1.00	0.50	0.00	30	33		
								0.20	1.00	1.00	0.67	0.10	30	33		
4	0.98	1.00	1.00	0.98	50	46	0.80	0.05	1.00	1.00	1.00	0.97	30	26	7	7 (-15%)
								0.10	1.00	1.00	1.00	1.00	30	26		
								0.20	1.00	1.00	1.00	0.92	50	46		
							0.90	0.05	1.00	1.00	1.00	0.94	50	45		
								0.10	0.96	1.00	1.00	0.92	50	46		
								0.20	0.94	1.00	1.00	0.96	50	45		
5	1.00	1.00	0.84	0.30	50	57	0.80	0.05	1.00	1.00	0.90	0.37	30	31	20	25 (-44%)
								0.10	1.00	1.00	0.80	0.40	30	32		
								0.20	1.00	1.00	1.00	0.33	30	32		
							0.90	0.05	1.00	1.00	0.90	0.50	30	31		
								0.10	1.00	1.00	0.90	0.23	30	32		
								0.20	1.00	1.00	0.90	0.43	30	32		
6	1.00	1.00	0.37	0.04	70	153	0.80	0.05	1.00	1.00	0.23	0.00	30	62	40	91 (-59%)
								0.10	1.00	1.00	0.40	0.03	30	62		
								0.20	1.00	1.00	0.43	0.10	30	62		
							0.90	0.05	1.00	1.00	0.27	0.00	30	62		
								0.10	1.00	1.00	0.37	0.07	30	62		
								0.20	1.00	1.00	0.30	0.03	30	62		
7	1.00	1.00	0.62	0.02	50	108	0.80	0.05	1.00	1.00	0.63	0.07	30	62	20	45 (-42%)
								0.10	1.00	1.00	0.77	0.27	30	64		
								0.20	1.00	1.00	0.73	0.03	30	63		
							0.90	0.05	1.00	1.00	0.57	0.10	30	63		
								0.10	1.00	1.00	0.67	0.10	30	62		
								0.20	1.00	1.00	0.50	0.00	30	62		
8	1.00	1.00	0.46	0.10	50	106	0.80	0.05	1.00	1.00	0.70	0.03	30	61	20	45 (-42%)
								0.10	1.00	1.00	0.57	0.03	30	61		
								0.20	1.00	1.00	0.47	0.17	30	61		
							0.90	0.05	1.00	1.00	0.37	0.20	30	61		
								0.10	1.00	1.00	0.57	0.07	30	61		
								0.20	1.00	1.00	0.40	0.13	30	61		
														Mean	22	36 (-42%)

1. The relatives gaps are given in parenthesis.
2. The running times (CPU) are given in seconds.

References

Adams, Joseph, Egon Balas, and Daniel Zawack. 1988. "The shifting bottleneck procedure for job shop scheduling." *Management science* 34 (3): 391–401.

Barz, Christiane, and Kumar Rajaram. 2015. "Elective patient admission and scheduling under multiple resource constraints." *Production and Operations Management* 24 (12): 1907–1930.

Cho, Lucky, Hangmi Michelle Park, Jennifer K Ryan, Thomas C Sharkey, Chihyun Jung, and Detlef Pabst. 2014. "Production scheduling with queue-time constraints: Alternative formulations." In *IIE Annual Conference. Proceedings*, 282. Institute of Industrial and Systems Engineers (IISE).

Crombecq, Karel, Eric Laermans, and Tom Dhaene. 2011. "Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling." *European Journal of Operational Research* 214 (3): 683–696.

Dauzère-Pérès, Stéphane, and Jan Paulli. 1997. "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search." *Annals of Operations Research* 70: 281–306.

Feng, Jianguang, Chengbin Chu, and Ada Che. 2018. "Cyclic jobshop hoist scheduling with multi-capacity reentrant tanks and time-window constraints." *Computers & Industrial Engineering* 120: 382–391.

Johnzén, Carl, Stéphane Dauzère-Pérès, and Philippe Vialletelle. 2011. "Flexibility measures for qualification management in wafer fabs." *Production Planning and Control* 22 (1): 81–90.

Jung, Chihyun, Detlef Pabst, Myoungsoo Ham, Marcel Stehli, and Marcel 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.

Kaihara, Toshiya, Shinji Kurose, and Nobutada Fujii. 2012. "A proposal on optimized scheduling methodology and its application to an actual-scale semiconductor manufacturing problem." *CIRP Annals-Manufacturing Technology* 61 (1): 467–470.

Kim, Hyun-Jung, and Jun-Ho Lee. 2017. "A Branch and Bound Algorithm for Three-Machine Flow Shop with Overlapping Waiting Time Constraints." *IFAC-PapersOnLine* 50 (1): 1101–1105.

Kitamura, Shoichi, Kazuyuki Mori, and Akira Ono. 2006. "Capacity planning method for semiconductor fab with time constraints between operations." In *SICE-ICASE, 2006. International Joint Conference*, 1100–1103. IEEE.

Klemmt, Andreas, and Lars Mönch. 2012. "Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing." In *Proceedings of the Winter Simulation Conference*, 194. Winter Simulation Conference.

Knopp, Sebastian, Stéphane Dauzère-Pérès, and Claude Yugma. 2017. "A batch-oblivious approach for Complex Job-Shop scheduling problems." *European Journal of Operational Research* 263 (1): 50–61.

Kobayashi, Akihiro, Takahiro Kuno, and Sumika Arima. 2013. "Re-entrant flow control in Q-time constraints processes for actual applications." In *e-Manufacturing & Design Collaboration Symposium (eMDC), 2013*, 1–4. IEEE.

Krämer, Franz-Josef, Mohsen El Hafsi, and Sherman X Bai. 1997. "Production flow control for a manufacturing system with flexible routings." *Production and Operations Management* 6 (1): 37–56.

Lee, Jun-Ho, Cong Zhao, Jingshan Li, and Chrissoleon T Papadopoulos. 2018. "Analysis, design, and control of Bernoulli production lines with waiting time constraints." *Journal of Manufacturing Systems* 46: 208–220.

Lee, Yih-Yi, CT Chen, and Chester Wu. 2005. "Reaction chain of process queue time quality control." In *Semiconductor Manufacturing, 2005. ISSM 2005, IEEE International Symposium on*, 47–50. IEEE.

Li, Minqi, Feng Yang, Reha Uzsoy, and Jie Xu. 2016. "A metamodel-based Monte Carlo simulation approach for responsive production planning of manufacturing systems." *Journal of Manufacturing Systems* 38: 114 – 133.

Lima, Alexandre, Valeria Borodin, Stéphane Dauzère-Pérès, and Philippe Vialletelle. 2017. "Analyzing different dispatching policies for probability estimation in time constraint tunnels in semiconductor

manufacturing." In *2017 Winter Simulation Conference (WSC)*, 3543–3554.

Lima, Alexandre, Valeria Borodin, Stéphane Dauzère-Pérès, and Philippe Vialletelle. 2019. "Sampling-based release control of multiple lots in time constraint tunnels." *Computers in Industry* 110: 3 – 11.

Liu, Haitao, Yew-Soon Ong, and Jianfei Cai. 2018. "A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design." *Structural and Multidisciplinary Optimization* 57 (1): 393–416.

Maleck, Christian, and Tobias Eckert. 2017. "A comparison of control methods for production areas with time constraints and tool interruptions in semiconductor manufacturing." In *Electronics Technology (ISSE), 2017 40th International Spring Seminar on*, 1–6. IEEE.

Maleck, Christian, Gerald Weigert, Detlef Pabst, and Marcel Stehli. 2017. "Robustness analysis of an MIP for production areas with time constraints and tool interruptions in semiconductor manufacturing." In *Simulation Conference (WSC), 2017 Winter*, 3714–3725. IEEE.

Mendo, Luis, and José M Hernando. 2006. "A simple sequential stopping rule for Monte Carlo simulation." *IEEE Transactions on Communications* 54 (2): 231–241.

Ovacik, I.M., and R. Uzsoy. 1997. *Decomposition methods for complex factory scheduling problems*. Kluwer Academic Publishers.

Pappert, Falk Stefan, Tao Zhang, Oliver Rose, Fabian Suhrke, Jonas Mager, and Thomas Frey. 2016. "Impact of time bound constraints and batching on metallization in an opto-semiconductor fab." In *Winter Simulation Conference (WSC), 2016*, 2947–2957. IEEE.

Robinson, Jennifer K. 1998. "Capacity planning in a semiconductor wafer fabrication facility with time constraints between process steps." PhD diss., Citeseer.

Roy, B, and B Sussmann. 1964. "Les Problèmes d'Ordonnancement avec Contraintes Disjonctives." *Note DS n.9 bis, SEMA* .

Sadeghi, Rezvan, Stéphane Dauzère-Pérès, Claude Yugma, and Guillaume Lepelletier. 2015. "Production control in semiconductor manufacturing with time constraints." In *Advanced Semiconductor Manufacturing Conference (ASMC), 2015 26th Annual SEMI*, 29–33.

Simpson, Timothy W, Dennis KJ Lin, and Wei Chen. 2001. "Sampling strategies for computer experiments: design and analysis." *International Journal of Reliability and Applications* 2 (3): 209–240.

Wang, Bailin, Kai Huang, and Tieke Li. 2018. "Permutation flowshop scheduling with time lag constraints and makespan criterion." *Computers & Industrial Engineering* 120: 1–14.

Wang, Mengchang, Sandeep Srivathsan, Edward Huang, and Kan Wu. 2018. "Job Dispatch Control for Production Lines With Overlapped Time Window Constraints." *IEEE Transactions on Semiconductor Manufacturing* 31 (2): 206–214.

Wu, Cheng-Hung, Yu-Ching Cheng, Ping-Ju Tang, and Jiun-Yu Yu. 2012. "Optimal batch process admission control in tandem queueing systems with queue time constraint considerations." In *Proceedings of the Winter Simulation Conference*, 204. Winter Simulation Conference.

Wu, Cheng-Hung, Wen-Chi Chien, Ya-Tang Chuang, and Yu-Ching Cheng. 2016. "Multiple product admission control in semiconductor manufacturing systems with process queue time (PQT) constraints." *Computers & Industrial Engineering* 99: 347–363.

Wu, Cheng-Hung, James T Lin, and Wen-Chi Chien. 2010. "Dynamic production control in a serial line with process queue time constraint." *International Journal of Production Research* 48 (13): 3823–3843.

Wu, Cheng-Hung, James T Lin, and Wen-Chi Chien. 2012. "Dynamic production control in parallel processing systems under process queue time constraints." *Computers & Industrial Engineering* 63 (1): 192–203.

Yugma, Claude, Stéphane Dauzère-Pérès, Christian Artigues, Alexandre Derreumaux, and Olivier Sibille. 2012. "A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing." *International Journal of Production Research* 50 (8): 2118–2132.

Yurtsever, Tanju, Erhan Kutanoglu, and Jennifer Johns. 2009. "Heuristic based scheduling system for diffusion in semiconductor manufacturing." In *Winter Simulation Conference*, 1677–1685.