*Proceedings of the 2014 Industrial and Systems Engineering Research Conference*
*Y. Guan and H. Liao, eds.*

# Production Scheduling with Queue-time Constraints: Alternative Formulations

**Lucky Cho, Hangmi Michelle Park, Jennifer K. Ryan and Thomas C. Sharkey**
**Rensselaer Polytechnic Institute**
**110 Eighth Street, Troy, NY 12180, USA**

**Chihyun Jung and Detlef Pabst**
**GLOBALFOUNDRIES**
**400 Stone Break Extension, Malta, NY 12020, USA**

## Abstract

In manufacturing, queue-time constraints occur when a set of consecutive process steps must be completed within a fixed time window. Such constraints are common in semiconductor manufacturing, particularly in the diffusion area. Violations of queue-time constraints can lead to rework, scrap and longer cycle times. These constraints add significant complexity to the already difficult problem of scheduling in a semiconductor fab. In this paper, we consider the gate-keeping decision. Given a job arriving at a queue-time zone, the decision to be made is whether and when the job should be allowed to enter the zone. To make this decision, we propose a mixed integer linear programming model of the local scheduling problem within the queue-time zone. Because the gate-keeping decision is made at a higher level than detailed scheduling, our local scheduling model is simplified, i.e., it does not incorporate issues such as batching. This simplification allows us to consider a larger set of arriving jobs when making the gate-keeping decision. We propose two alternative model formulations, time- and slot-based, for the local scheduling problem. We compare the performance of these two formulations using a set of numerical experiments generated based on process flow information from an actual semiconductor fab.

**Keywords**
Job shop scheduling; mixed integer programming; time window constraints; wafer fabrication.

## 1. Introduction

Wafer fabrication consists of numerous sequential processes. In addition, each lot may have its own sequence of steps to be processed. Thus, wafer fabrication is an important large-scale application of the job shop scheduling problem. In addition to the complexity inherent in typical job shop scheduling problems, wafer fabrication has additional complexity arising from queue-time constraints, in which a set of consecutive process steps must be completed within an exogeneously-specified time window. Violations of these queue-time constraints can significantly affect the quality and yield of the final product. However, ensuring that these constraints are satisfied can add significant complexity to the already difficult problem of scheduling the wafer fab. While there has been some research considering queue-time constraints within a specific area of a semiconductor fab (e.g., in the diffusion area; see [1]), the existing literature has not considered the development of more generalized queue-time control methods that can be applied to the wide variety of areas within the fab that are subject to queue-time constraints.

In this paper, we present a method for making the gate-keeping decisions, i.e., the decisions regarding whether or not a given lot should be allowed to start a specific queue-time constrained set of process steps (referred to as a queue-time zone). In practice, real-time dispatching (RTD) rules are commonly employed within semiconductor fabs for making the gate-keeping decisions. These rules generally rely on priority-based dispatching and thus can be run very quickly. However, relying entirely on RTD can result in many lots experiencing queue-time violations. Thus, as an alternative, in this paper we propose the a mixed integer linear programming (MILP) model with the following considerations:

- The proposed model is appropriate for a rolling horizon planning approach in which the gate-keping decisions must be made within 5 minutes, where the relevant data is updated every 10 minutes.

- Since our goal is to develop a generalized queue-time control method, our model is simplified, excluding issues such as batching and tool-dependent process times.
- Our model allows for both queue-time-constrained lots and non-queue-time-constrained lots.
- In our model, the gate-keeping decision is made based on the estimated time at which each admitted lot will leave the queue-time zone. This estimation is performed using an MILP model of the local scheduling problem within the queue-time zone. We propose two alternative formulations for the local scheduling problem: time- and slot-based. One goal of this research is to compare the peformance of these two formulations.

## 2. Literature

Although queue-time constraints are critical in semiconductor manufacturing operations in practice, they can significantly complicate the job shop scheduling problem. Unfortunately, there has not been much published work directly related to wafer fab scheduling with queue-time constraints. Thus, we take a broader view and consider general production systems with queue-time constraints. In general production systems with process queue-time constraints, we can categorize the literature according to the approach used to handle the constraints. A subset of the literature uses steady-state queuing analysis to estimate the probability of queue-time violations. For example, [2] uses a $GI/G/m$ queueing model to determine the optimal number of batch tools in semiconductor wafer fabrication in the presence of queue-time constraints. In addition, [3] study the problem of wasted of capacity caused by processing wafers that have already violated their queue-time constraints, given that wafer yields can be very low if the queue-time limit is violated. [3] presents a capacity planning model to determine the optimal number of tools using a $GI/G/m$ queueing system with two sequential queue-time constraints.

Other papers have considered the use of dispatching rules in deterministic scheduling systems with queue-time constraints. [4] studies the impact of queue-time constraints between the wet etch and furnace operations using Kanban-based approaches which limit the number of jobs in the queue-time zone. They evaluate their proposed approaches using discrete event simulation. [5] proposes a control mechanism called reaction chains, which controls the queue time processes by periodically reviewing the remaining workload in the system. [6] studies several dispatching methods under storage-time constraints in the food industry. They focus on a two stage process with a batch process in the first stage and multiple parallel packing processes in the second stage. In between the two stages is the intermediate storage, which is time-constrained. There are sequence dependent set up times at the first stage. The authors apply, and study the performance of, several well-known heuristic sequencing approaches.

Finally, more recent literature has sought to capture the dynamic nature of production systems under queue-time constraints. In other words, the literature in this category develops methods to incorporate real-time information into the model and decision-making. For example, [7] consider a short-interval local scheduling problem with queue time constraints using mixed integer and constraint programming. The authors argue that such an optimization-based approach is favorable to the more traditional approach of rule-based dispatching. The authors provide an overview of the STARTS (space-time allocation for real-time scheduling) algorithm developed by ILOG, including a discussion of the solution approach, objectives and data requirements. The authors also discuss the application of their approach to the diffusion process. [8] propose three heuristic algorithms to schedule lots in the diffusion area using basic ideas from list scheduling algorithms. In their work, no specific queue-time constraint is imposed. Instead, each lot has its own due date and the objective is to minimize total tardiness. [9] and [10] consider one of the main causes of queue-time violations in general production systems: unreliable tools and breakdowns. The authors argue that static capacity decision models are not effective when there is a chance of tool breakdown. Both [9] and [10] present methods that take into account the real-time tool status for optimal production control. While [9] considers a two-stage model with a single tool at each stage, [10] allows parallel processing with multiple non-identical tools. Both papers formulate the problem as a Markov decision process. Finally, [1] develops a batching and scheduling algorithm for the diffusion area. The authors formulate the scheduling problem using a disjunctive graph approach in which the decision variables are the start times of each operation. A two stage solution approach is proposed. The first stage is a constructive heuristic based on successive job insertion (jobs are sorted according to maximum time lag, release date and job priority). The second stage is a local search heuristic to improve the initial solution.

## 3. Model

In this section, we first describe the scheduling problem with queue-time constraints. We then provide two MILP formulations (slot- and time-based) for this problem. We conclude with a mathematical comparison of the formulations.

## 3.1 Problem Description

We first describe the local scheduling problem with queue-time constraints and gate-keeping decisions. There are a set of lots $\mathcal{C} = \{c|1,2,\ldots,C\}$. Each lot $c$ has a series of process steps $\mathcal{P}_c = \{p_c|1,2,\ldots,P_c\}$ to be completed. A job $j \in \mathcal{J} = \{j|1,2,\ldots,J\}$ is defined to be a lot-process step combination $(c, p_c)$. There is a set of available tools $\mathcal{M} = \{t|1,2,\ldots,M\}$ and each job $j$ can be performed by one of a predetermined subset of tools $\mathcal{M}_j \subseteq \mathcal{M}$. The processing time $p_{tcp}$ of each job $j = (c, p)$ also depends on the selected tool, $t$ . Each tool $t \in \mathcal{M}$ is available at earliest time $z_{t0}$ and each lot $c \in \mathcal{C}$ is ready to start its first process step at time $y_{c0}$. The local schedule and gate-keeping decisions are made based on a rolling horizon with a planning horizon of length $T$.

Queue-time constraints require that a given series of process steps be completed within a fixed time period. In other words, if there is a queue-time constrained lot $c$, its set of process steps $\mathcal{P}_c$ includes a subset of process steps, from $p$ to $p'$, which we call the queue-time zone. Queue-time constraints require that the time used from the completion time of job $(c, p)$, the first job in queue-time zone, to the start of job $(c, p')$, the last job of the queue-time zone, should not exceed the given queue-time limit, $Q_{(c,p),(c,p')}$. Since the queue-time starts immediately after the first job $(c, p)$ is completed, we define the gating operation as the process step just before the first job of the queue-time zone. The gate-keeping decision controls the admission of the queue-time constrained lots into the queue-time zones. Let $G$ denote the set of lots which have the first process step and the last process step of the queue-time zone in $\mathcal{P}_c$. For $c \in G$, we must determine whether or not $c$ passes the gate, i.e., starts the process step $p$, during this planning horizon. In this paper, for simplicity of presentation, we assume there is a single queue-time zone. However, the model that we present can easily be extended to allow for multiple queue-time zones.

The scheduling model we develop will be solved using a rolling-horizon approach. Thus, when the model is solved, i.e., at time 0, there may be some lots already located within a queue-time zone. Note that these lots are not included in the set $G$, i.e., the set $G$ includes only those lots for which a queue-time zone admission decision must be made. If the gate-keeping decision from the previous planning horizon admitted the lot into the queue-time zone, but did not schedule all of the steps in the queue-time zone, those lots will still be in the queue-time zone when the planning horizon starts. Let $W$ denote this set of lots. The set of process steps to be completed for lots in $W$ does not include the entire series of process steps from $p$ to $p'$, but rather includes only part of the queue-time zone, including the last step $p'$. For $c \in W$, we define the remaining queue-time to be $Q_{(c,p)}$. Notice that, due to previous scheduling decisions, in some cases $Q_{(c,p)}$ may not be sufficient time to finish the remaining steps in the queue-time zone. Therefore, for lots $c \in W$, we do not enforce the queue-time constraint. However, for these lots, we will penalize for violations of the queue-time constraint.

We next present our MILP formulation of the local scheduling problem. We seek to minimize the objective function, which consists of the following five components:

- The first component is a sum of the weighted completion (cycle) time of each job. We use the normalized weight $w_{cp} \in (0, 1]$ which is computed based on the given priority level of each lot $c$.
- The second component is a sum of the queue-time penalties/rewards. Lots $c \in G$, must satisfy their queue-time constraints. In addition. these lots may complete the queue-time zone in less than the allotted time, which is preferred for quality purposes. Thus, any remaining queue-time, referred to as slack, is assigned a reward, which reduces the objective value. On the other hand, as noted above, a lot which is already in the queue-time zone at time zero, $c \in W$, may violate the queue-time constraint. In this case, we compute the additional time required, beyond $Q_{(c,p)}$, until the lot starts the last job of the queue-time zone. We assign a penalty to this additional time. This penalty increases the objective value.
- The third component is a sum of the penalties for not admitting a queue-time constrained lot into the queue-time zone. If a lot $c \in G$ is not admitted to the queue-time zone, all the process steps after the gate will be delayed until at least the end of this planning horizon, $T$. Hence, we penalize this delay by $\lambda_c \in (0, 1]$, which is a normalized value of the penalty for a non-admitted lot $c$. We compute this penalty as the lot priority level $\times$ the number of process steps delayed by not entering the queue-time zone $\times$ length of the planning horizon, $T$.
- The fourth component is the weighted sum of the tool idle times. We include the total idle time of each tool $t$, where each tool is assigned a weight, denoted by $w_t$.
- The last component of the objective function is the weighted throughput. While satisfying the queue-time constraints is crucial for the quality of the final product, it is also important to complete as many as process steps as possible. If there is a process step which is not scheduled in this planning horizon, it cannot be processed

until after the end of this planning horizon. Thus, we subtract the weighted sum of the scheduled jobs multiplied by the length of the planning horizon, $T$, from the objective value.

In formulating the local scheduling model, we consider two different approaches: slot-based and time-based. The slot-based formulation determines which job $(c,p)$ is completed in the $s$-th slot of tool $t$, while the time-based formulation determines which job $(c,p)$ is completed by tool $t$ at time $\tau$. We assume that each tool has at least $C$ slots so that the problem is feasible if every lot must visit a given tool once. For the time-based formulation, we discretize the continuous time $T$ by the length $\varepsilon$, which is equal to the time between $\tau$ and $\tau+1$. The next two subsections describe the two formulation approaches. For convenience, the relevant notation is summarized in Table 1.

Table 1: Summary of notation

| Notation | Definition |
|---|---|
| $g_c$ | gate-keeping decision variable, binary |
| $x_{cpts}$ | slot-based job scheduling decision variable, binary |
| $x_{cpt\tau}$ | time-based job scheduling decision variable, binary |
| $y_{cp}$ | finish time of job $(c,p)$ |
| $z_{ts}$ | start time of the $s$-th slot on a tool $t$, in slot-based formulation |
| $z_{cp}$ | start time of job $(c,p)$, in time-based formulation |
| $v_c$ | queue-time slack/violation for lot $c$ |
| $c,p,P_c$ | index for lots, index for process steps, set of process steps to be completed for lot $c$ |
| $j=(c,p)$ | a job |
| $t,s$ | index for the tools, index for slots on a tool |
| $p_{tcp}$ | the processing time of $(c,p)$ on a tool $t$ |
| $\tau$ | index for time period, in time-based formulation |
| $G$ | the set of queue-time constrained lots, for which a gate-keeping decision must be made |
| $W$ | the set of lots located in the queue-time zone at time 0 |
| $Q_{(c,p),(c,p')}$ | queue-time limit between the end of $(c,p)$ and the start of $(c,p')$, for $c\in G$ |
| $Q_{(c,p)}$ | remaining queue-time for the lots in the queue-time zone at time 0, for $c\in W$ |
| $w_{cp},w_t$ | objective function weights for job $(c,p)$ and for tool $t$, respectively |
| $z_{t0},y_{c0}$ | ready time of tool $t$, lot $c$ for the initial job |
| $\lambda_c$ | penalty for not letting lot $c\in G$ into the queue-time zone |
| $T$ | planning horizon, i.e., the last time at which a job can be scheduled to start a process step |

**3.2 Slot-Based MILP Formulation**

The slot-based MILP formulation has five types of decision variable. The first is the gate-keeping decision, $g_c$, for lots $c\in G$, which is binary. If a lot $c$ is admitted to the queue-time zone, then $g_c=1$; otherwise, $g_c=0$. The second is the job scheduling decision, $x_{cpts}$, which is binary. If job $(c,p)$ is scheduled in the $s$-th slot of tool $t$, then $x_{cpts}=1$; otherwise $x_{cpts}=0$. The third decision is the finish time for job $(c,p)$, which is denoted by $y_{cp}$. The fourth decision is the start time of each slot $s$ of a tool $t$, which is denoted by $z_{ts}$. The final decision variable is the queue-time slack/violation. The remaining queue-time (or queue-time used beyond the queue-time limit) of lot $c$, denoted $v_c$, is:

$$v_c = \left(y_{cp'} - \sum_{t,s} p_{tcp'}x_{cp'ts} - y_{cp}\right) - Q_{(c,p)(c,p')}g_c \quad \text{for } c\in G \tag{1}$$

$$v_c = \left(y_{cp} - \sum_{t,s} p_{tcp}x_{cpts}\right) - Q_{(c,p)} \quad \text{for } c\ \in W. \tag{2}$$

Notice that $v_c$ will be negative if there is no queue-time violation (representing queue-time slack), while $v_c$ will be positive if the queue-time constraint is violated. Since we formulate the problem as a minimization, a negative $v_c$ represents a reward, while a positive $v_c$ represents a penalty. Constraint (20) will ensure that $v_c=0$ if $g_c=0$ for $c\in G$.

We are now ready to present the complete model. The objective function can be written as follows:

$$\min \quad \sum_{(c,p)} w_{cp}y_{cp} - w_q\sum_c v_c + \sum_{c\in G}\lambda_c(1-g_c) + \sum_{t=1}^{m} w_t\left(Y_t - z_{t0} - \sum_{s=1}^{C}\sum_{(c,p)} p_{tcp}x_{cpts}\right) - T\sum_{(c,p)} w_{cp}\sum_{(t,s)} x_{cpts}. \tag{3}$$

We next describe each of the constraints. First, if lot $c\in G$ enters the queue-time zone, all process steps $p\in\mathcal{P}_c$ before the gate should be scheduled. In other words, all jobs $j=(c,p)$ with the process step before the gate should be assigned to a certain slot $s$ of a certain tool $t \in \mathcal{M}_j$. Therefore, we have the following constraint:

$$\sum_{t \in \mathcal{M}_j} \sum_s x_{cpts} \geq g_c \qquad \forall j = (c,p) \text{ before the gate.} \tag{4}$$

If lot $c \in G$ enters the queue-time zone, all jobs in the queue-time zone should be scheduled. Thus, we have the following constraint:

$$\sum_{t \in \mathcal{M}_j} \sum_s x_{cpts} = g_c \qquad \forall j = (c,p) \text{ within the queue-time zone.} \tag{5}$$

If lot $c \in G$ enters the queue-time zone, the remaining of jobs after the queue-time zone may or may not be scheduled.

$$\sum_{t \in \mathcal{M}_j} \sum_s x_{cpts} \leq g_c \qquad \forall j = (c,p) \text{ after the last job of the queue-time zone.} \tag{6}$$

We let $Y_t$ represent the completion time of the last job on a tool $t$. Thus, $Y_t$ must be greater than or equal to the completion time of all the jobs processed on $t$. We therefore have the following constraint:

$$Y_t \geq z_{ts} + \sum_c \sum_{p \in P_c} p_{tcp} x_{cpts} \qquad \forall t,s. \tag{7}$$

The job allocation to any slot of any tool cannot be earlier than the tool ready time. Thus, we have the following:

$$z_{ts} \geq z_{t0} \qquad \forall t,s. \tag{8}$$

The completion time of any job $(c,p)$ cannot be earlier than the lot ready time of lot $c$. Thus, we have the following:

$$y_{cp} \geq y_{c0} \qquad \forall c,p. \tag{9}$$

Any slot $s$ of any tool $t$ cannot process more than one job $(c,p)$. Thus, we have the following constraint:

$$\sum_c \sum_{p \in P_c} x_{cpts} \leq 1 \qquad \forall t,s. \tag{10}$$

Any job $(c,p)$ can be scheduled at most once during the planning horizon $T$. Thus, we have the following:

$$\sum_t \sum_s x_{cpts} \leq 1 \qquad \forall c,p. \tag{11}$$

For each tool $t$, the scheduled jobs should fill the smallest available slot first, i.e., no job can be assigned to the $s$-th slot as long as $(s-1)$-st slot is still available to be assigned. Thus, we have the following constraint:

$$\sum_c \sum_p x_{cpts} \leq \sum_c \sum_p x_{cpt(s-1)} \qquad \forall t,s. \tag{12}$$

The $s$-th slot of tool $t$ is available only after tool $t$ processes the job assigned to the $(s-1)$-st slot. Thus, we have

$$z_{t(s-1)} + \sum_{(c,p)} p_{tcp} x_{cpt(s-1)} \leq z_{ts} \qquad \forall t,s. \tag{13}$$

The end of the horizon is the last time at which a job can be scheduled. Jobs do not need to be completed by the end of the planning horizon. However, all scheduled jobs must start work by $T$. Thus, the largest possible completion time of a job is the longest processing time after the planning horizon, $T$. This leads to the following constraint:

$$y_{cp} \leq T + \max_{t,c,p} p_{tcp} \qquad \forall c,p,t. \tag{14}$$

The completion times of all non-scheduled jobs are set equal to be the latest possible completion time, defined in (14):

$$y_{cp} \geq (1 - \sum_{(t,s)} x_{cpts})(T + \max_{t,c,p} p_{tcp}) \qquad \forall c,p. \tag{15}$$

The finish times and start times must be linked, as shown in the following constraints:

$$z_{ts} + p_{tcp} - y_{cp} + Bx_{cpts} \leq B \qquad \forall c,p,t,s \tag{16}$$

$$z_{ts} + p_{tcp} - y_{cp} - Bx_{cpts} \geq -B \qquad \forall c,p,t,s \tag{17}$$

Here $B$ is an arbitrary large number. We choose the smallest feasible $B$. Note that (16) and (17) imply that $z_{ts} + p_{tcp} = y_{cp}$ for $x_{cpts} = 1$, for any choice of $B$. However, $z_{ts} + p_{tcp} - y_{cp}$ can take any value when $x_{cpts} = 0$. For an unscheduled job with $x_{cpts} = 0$, (15) requires the finish time of the job is equal to $T + \max_{t,c,p} p_{tcp}$. Combining this with (16) and (17), it is required that $z_{ts} + p_{tcp} - B \leq T + \max_{t,c,p} p_{tcp} \leq z_{ts} + p_{tcp} + B$. Since $z_{ts} + p_{tcp}$ is the finish time of job $(c,p)$, which can be as early as zero and as late as $T + \max_{t,c,p} p_{tcp}$, the smallest feasible value for $B$ is $T + \max_{t,c,p} p_{tcp}$.

For each lot $c$, job $(c,p+1)$ cannot start until job $(c,p)$ is completed. Thus, we have the following constraints:

$$y_{c,p+1} - \sum_{(t,s)} p_{tc,p+1} x_{c,p+1,ts} - y_{cp} \geq 0 \qquad \forall c,p \tag{18}$$

$$y_{c1} - \sum_{(t,s)} p_{tc1} x_{c1ts} - y_{c0} \geq 0 \qquad \forall c. \tag{19}$$

For queue-time constrained lots, i.e., for $c \in G$, if the lot is admitted to the queue-time zone, the queue-time limit $Q_{(c,p),(c,p')}$ cannot be violated. This leads to the following constraint:

$$0 \le y_{cp'} - \sum_{(t,s)} p_{tcp'} x_{cp'ts} - y_{cp} \le Q_{(c,p)(c,p')}\, g_c \qquad \forall c \in G. \tag{20}$$

## 3.3 Time-Based MILP Formulation

The time-based MILP formulation also has five types of decision variable, similar to those defined for the slot-based MILP formulation. The gate-keeping decisions, $g_c$, and completion time of job $(c,p)$, $y_{cp}$ are defined as in the slot-based formulation. The job scheduling decision variable is $x_{cpt\tau}$, which is binary. If job $(c,p)$ is scheduled to be completed on the tool $t$ at time $\tau$, then $x_{cpt\tau} = 1$; otherwise, $x_{cpt\tau} = 0$. The next decision variable is the start time of each job, which is denoted by $z_{cp}$. The final decision variable is the queue-time slack/violation. The remaining queue-time (or queue-time used beyond the queue-time limit) of lot $c$ is measured as in (1) and (2):

$$v_c = (z_{cp'} - y_{cp}) - Q_{(c,p)(c,p')} g_c \qquad \forall c \in G \tag{21}$$

$$v_c = \sum_{\tau}\sum_{t} (\tau - p_{tcp}) x_{cpt\tau} - Q_{cp} \qquad \forall c \in W. \tag{22}$$

The objective function and constraints are similar to those in the slot-based formation. The objective function is:

$$\min \sum_{(c,p)} w_{cp} y_{cp} - \sum_{v_c} v_c + \sum_{c \in G} \lambda_c (1 - g_c) + \sum_{t=1}^{m} w_t \left( Y_t - z_{t0} - \sum_{\tau}^{T+\max_{t,c,p} p_{tcp}} \sum_{(c,p)} p_{tcp} x_{cpt\tau} \right) - T \sum_{(c,p)} w_{cp} \sum_{(t,\tau)} x_{cpt\tau}. \tag{23}$$

If lot $c \in G$ enters the queue-time zone, all process steps $p \in \mathcal{P}_c$ prior to the gate must be scheduled, i.e., must be allocated to some $(t,\tau)$. This results in the following constraint:

$$\sum_{t \in \mathcal{M}_j} \sum_{\tau} x_{cpt\tau} \ge g_c \qquad \forall j = (c,p) \text{ before the gate.} \tag{24}$$

If lot $c \in G$ enters the queue-time zone, all jobs in the queue-time zone must be scheduled. Thus, we have the following:

$$\sum_{t \in \mathcal{M}_j} \sum_{\tau} x_{cpt\tau} = g_c \qquad \forall j = (c,p) \text{ within the queue-time zone.} \tag{25}$$

If lot $c \in G$ enters the queue-time zone, the remaining jobs after the queue-time zone may or may not be scheduled. This results in the following constraint:

$$\sum_{t \in \mathcal{M}_j} \sum_{\tau} x_{cpt\tau} \le g_c \qquad \forall j = (c,p) \text{ after the last job of the queue-time zone.} \tag{26}$$

Let $Y_t$ represent the completion time of the last job on tool $t$. Then $Y_t$ must satisfy the following constraint:

$$Y_t \ge \tau \sum_{c} \sum_{p \in \mathcal{P}_c} x_{cpt\tau} \qquad \forall t, \tau. \tag{27}$$

The job allocation to any tool should not be earlier than the tool ready time. This results in the following:

$$\sum_{c} \sum_{p \in \mathcal{P}_c} x_{cpt\tau} (\tau - p_{tcp}) \ge z_{t0} \qquad \forall t, \tau. \tag{28}$$

The completion time of any job $(c,p)$ cannot be earlier than the lot ready time of $c$. This results in the following:

$$y_{cp} \ge y_{c0} \qquad \forall c, p. \tag{29}$$

Any tool $t$ cannot process more than one job $(c,p)$ at a time. This results in the following constraint:

$$\sum_{(c,p)} \sum_{\tau'=\tau}^{\tau + p_{tcp} - \varepsilon} x_{cpt\tau'} \le 1 \qquad \forall t, \tau. \tag{30}$$

Any job $j = (c,p)$ can be scheduled at most once during the planning horizon $T$. This results in the following:

$$\sum_{t \in \mathcal{M}_j} \sum_{\tau=0}^{T} x_{cpt\tau} \le 1 \qquad \forall j = (c,p). \tag{31}$$

The last moment at which a job can be scheduled is the end of planning horizon, $T$. This results in the following:

$$y_{cp} - \max_{t,c,p} p_{tcp} \le T \qquad \forall c, p. \tag{32}$$

The completion time of unscheduled jobs should be set equal to the latest possible completion time, as defined in (32). This results in the following:

$$y_{cp} \ge \sum_{\tau=0}^{T} \sum_{t} x_{cpt\tau} + (T + \max_{t,c,p} p_{tcp})(1 - \sum_{t \in \mathcal{M}_j} \sum_{\tau=0}^{T} x_{cpt\tau}) \qquad \forall j = (c,p). \tag{33}$$

The finish time and start time must be linked, as follows:

$$y_{cp} - z_{cp} = \sum_{t \in \mathcal{M}_j} \sum_{\tau} p_{tcp} x_{cpt\tau} \qquad \forall j = (c,p). \tag{34}$$

For each lot $c$, job $(c, p+1)$ cannot start until job $(c, p)$ is finished. This results in the following constraint:

$$z_{c,p+1} \geq y_{cp} \qquad \forall c, p. \tag{35}$$

For queue-time constrained lots, i.e., for lot $c \in G$, if the lot is admitted to the queue-time zone, the queue-time limit $Q_{(c,p)(c,p')}$ cannot be violated. This results in the following:

$$0 \leq z_{cp'} - y_{cp} \leq Q_{(c,p)(c,p')} g_c \qquad \forall c \in G. \tag{36}$$

## 3.4 Mathematical Comparison between Slot-Based and Time-Based MILP Formulation

One goal of the research described in this paper is to compare the different MILP formulations and to investigate under what conditions each formulation performs better in terms of computational complexity. In the slot-based formulation, constraints (16) and (17) link the start and the finish time of each job $(c, p)$. Since we require (16) and (17) for every possible combination of $c, p, t$ and $s$, we have approximately $2 * C * P * M * C = 2C^2PM$ constraints, where each constraint contains a big-$M$ type constant (denoted by $B$ in our formulation). The number of decision variables, $x_{cpts}$, is $C^2PM$ in the slot-based formulation. In the time-based formulation, constraints (32) - (34) link the start and the finish time of each job $(c, p)$. Since we require these constraints for each job $(c, p)$, we have approximately $3CP$ constraints, without any big-$M$ type constants. The number of decision variables, $x_{cpt\tau}$, is $CPM(T/\varepsilon)$ in the time-based formulation, where $T/\varepsilon$ is the number of time periods. Thus, the number of decision variables depends on the length of the planning horizon, $T$, and the length of a time period, $\varepsilon$.

As we can see, the slot-based formulation contains a larger set of linking constraints: $2C^2PM$ constraints in the slot-based formulation compared to $2CP$ in the time-based formulation. On the other hand, the time-based formulation can potentially contain a much larger number of scheduling decision variables, if the planning horizon is long: $CPM(T/\varepsilon)$ decision variables in the time-based formulation compared to $CPMC$ in the slot-based formulation. Hence, it is not immediately clear which formulation will perform best and the answer is likely to depend on a variety of factors.

## 4. Computational Results

To compare the performance of the time-based and slot-based formulations, we conducted a series of numerical experiments. In this section, we first describe how we generated our test data sets and how our numerical experiments were designed. We then present some preliminary results comparing the performance of the two formulations.

### 4.1 Data Generation and Experimental Design

We developed our preliminary experiments using a data set provided by our industrial partner. This data contained a partial process route (consisting of several hundred steps) for a single product type. For each step in the route, a (scaled) process time was provided, as well as a list of tools that could perform that step. These processing times were not tool-dependent. We were also provided with the starting and ending steps for each of the queue-time zones that occurred within this partial process route. In total there were several dozen queue-time zones, including some nested zones. However, some zones occurred multiple times along the route. Finally, lots could be assigned to one of three priority levels. We were provided with the percentage of lots within each level, as well as the appropriate weight to assign to each priority level. Based on this information, we generated a data set that could be used to test our MILP formulations. We did so by first choosing a subset of tools to be scheduled and then randomly generating a set of lots to be scheduled on those tools. We next describe the procedure we used to generate this data set.

We chose a single queue-time zone to schedule and included all of the tools in that zone in our subset of tools to schedule. We then looked to see what other equipment types (and associated tools) were "connected," i.e., frequently preceded or followed in the process route, the tools in that subset. We added only the most connected tools to the subset, to end up with a subset tools to be included in the scheduling model. Note that while we built the set of tools to schedule around a single queue-time zone, our final data set actually contains multiple queue-time zones. This is because any given tool may arise multiple times in the process route and may be part of multiple queue-time zones.

We next generated a set of lots and arrival times at the tools in our subset. We first randomly generated (according to a uniform distribution) a set of lots (WIP) distributed across the entire process route. For each lot, we recorded where along the route, i.e., at what process step, it was located. This gave us each lot's location at time 0. We then predicted how each lot would progress through the remainder of its route (using the raw processing times, appropriately inflated, as described below). We used these predictions to estimate arrival times of each unit of WIP at the tools in our subset. By generating the data in this way, different lots at a given tool are at different stages of the process route. Thus, the data look like data generated within a job shop setting, with different products having different process routes.

One objective of this research is to perform a sensitivity analysis in order to understand the conditions under which each formulation (time-based vs. slot-based) performs best. Thus, in generating our preliminary data sets, we considered a number of cases in which we varied the number of lots to be scheduled (the WIP level), the number of tools available, and the tightness of the queue-time constraints. Specifically, we varied the following problem parameters:

- WIP factor: The amount of WIP is set by assuming 90% utilization of bottleneck and a cycle time equal to a multiple of the raw processing time. We call the multiplier the WIP factor, denoted by $\omega$. By varying $\omega$, we change the "load" on the equipment being scheduled. We consider $\omega \in \{2,3,4\}$.
- Queue-time (QT) factor: We calculate the queue-time limit, $Q_{(c,p),(c,p')}$, as a multiple, $\nu \geq 1$, of the total raw processing time for steps within the queue-time zone. We refer to $\nu$ as the queue-time factor, where $\nu$ is a measure of the degree of tightness of the queue-time constraint. In practice, the degree of queue-time tightness can vary significantly across different queue-time zones. Thus, we consider $\nu \in \{1.5,5.5,13\}$.
- Tool factor: We also vary the number of tools. We consider a setting in which some tools may be down for repair, but nothing else in the fab is changed. We set the number of tools equal to a multiplier, $\phi$, multiplied by the base number of tools from the original data set. We refer to $\phi$ as the tool factor. We consider $\phi \in \{1,0.85,0.7\}$.

Using this procedure, we constructed a set of preliminary experiments. We first selected a subset of 40 tools to be scheduled. We next generated the WIP to be scheduled. For each value of the WIP factor, $\omega \in \{2,3,4\}$, we created five randomly generated sets of WIP. The average WIP levels across these five replications were 21 ($\omega = 2$), 30 ($\omega = 3$) and 40 ($\omega = 4$) lots. On average, 19.0% of the WIP had an associated gate-keeping decision. Another 30.2% of the WIP were already within a queue-time zone at time 0. We solved 135 total problems, consisting of 5 randomly-generated replications of each of the $27 = 3 \times 3 \times 3$ cases, based on the WIP factor, queue-time factor and tool factor.

We next describe how we set the planning horizon for each case. The planning horizon, $T$, must be set to ensure that a lot that is admitted to the queue-time zone at time 0 can complete processing in the queue-time zone before the end of the horizon. Otherwise, no lots will be admitted to the queue-time zone. Thus, to compute $T$, we first compute the sum of the raw processing times for all steps within the queue-time zone, including the gate operation and the final operation. Then, we multiply this sum by the queue-time factor and add a constant. The addition of the constant allows lots to be admitted for some period of time after time 0.

## 4.2 Experimental Results

In this section, we will present our preliminary experimental results. For conciseness, we will focus on the impact of the WIP factor and queue-time (QT) factor, with the tool factor fixed at $\phi = 1$. Our preliminary analysis indicates that the tool factor has limited impact on the overall results. Table 2 presents the performance gaps for the nine cases created by varying the WIP factor and QT factor. The numbers reported in columns 5 through 8 are the average performance gaps across the five replications in each case. To compute the performance gaps, we first ran the MILP in CPLEX 12.4 (on an Intel core i7-3740 QM processor, using optimality emphasis) for two hours to find a lower bound (LB) on the optimal solution. We next ran the MILP in CPLEX (using feasibility emphasis) for 5 and 10 minutes and recorded the best solution found, which serves as our upper bound (UB) on the optimal solution. The performance gap is calculated as the percentage difference between the UB and LB. Note that when computing the UB, we consider 5 and 10 minute runs due to the implementation restrictions faced by our industrial partner, i.e., as noted above, in practice the MILP model will be solved as part of a rolling horizon approach. The third and fourth columns of Table 2 provide some information on the size of the problem, independent of the formulation, for each of the nine cases. Recall that $(c, p)$ denotes a job, i.e., a combination of lot and process step, and that $t$ denotes a tool. Thus, the third and fourth columns report the average number of jobs and job-tool combinations, respectively, across the five replications. Both measures are increasing in the WIP factor and in the queue-time factor. The latter result is due to the fact that the planning horizon increases with the queue-time limit.

Table 2 indicates that, when considering the performance gap, which formulation performs better depends on the problem parameters. The slot-based formulation does better in four of the nine cases, including all cases in which the queue-time factor is small (equal to 1.5). The time-based formulation does better in five of the nine cases, including all but one of the cases in which the queue-time factor is large (equal to 5.5 or 13). Recall that, as the queue-time factor increases, the planning horizon also increases. As the planning horizon increases, the number of lots to be scheduled increases. In addition, the number of time periods (slots) increases in the time-based (slot-based) formulation. Overall, both problem formulations are increasing in size as the queue-time factor increases. If we investigate the size of the formulations in more detail, we find that the queue-time factor has a bigger impact on the size of the formulation than does the WIP factor, where size is measured in terms of the number of scheduling decision variables and the number of linking constraints, as discussed in Section 3.4. For the slot-based formulation, the number of scheduling decision variables and linking constraints increases dramatically with the queue-time factor. For the time-based formulation, the number of scheduling decision variables increases dramatically with the queue-time factor, while the number of constraints increases more slowly. This is a direct result of the analysis in Section 3.4, along with the fact that the number of slots and the number of time periods both increase as the queue-time factor increases.

To further understand the results shown in Table 2, it is important to consider the impact of increasing the queue-time factor. As this factor increases, two things happen. First, the queue-time limit is increased, giving lots more time to complete the queue-time zone. This should make the problem easier to solve. On the other hand, when the queue-time limit is increased, the planning horizon lengthened. This should make the problem more challenging to solve. From Table 2, we see that, when the WIP factor is small (equal to 2), the performance gaps are increasing in the queue-time factor. Thus, the second effect dominates when the amount of WIP is low. Note that when the amount if WIP is low, scheduling the jobs through the queue-time zone within the queue-time limit is easier due to less congestion, and thus the first effect is less important. When the WIP factor increases, the trade-off between the two effects kicks in. Thus, when the WIP factor is equal to three or four, we see inconsistent results, i.e., the performance gaps may increase and then decrease as the queue-time factor increases. For example, when the WIP factor is high, a tighter queue-time limit poses a significant challenge when making the gate-keeping decisions. Thus, in some cases, a substantial increase in the queue-time factor (e.g., from 5.5 to 13) can lead to a significant reduction in the performance gap.

Table 2: Performance gaps (in %) for time-based and slot-based formulations

| WIP factor | QT factor | Average $(c,p)$ | Average $(c,p,t)$ | Time-based (10 min) | Time-based (5 min) | Slot-based (10 min) | Slot-based (5 min) |
|---|---|---|---|---|---|---|---|
| 2 | 1.5 | 37 | 139.6 | 7.2 | 7.2 | 0.8 | 0.8 |
| 2 | 5.5 | 69.4 | 253.6 | 100.6 | 101.2 | 192.9 | 210.0 |
| 2 | 13 | 121 | 452.4 | 353.2 | 370.8 | 2156.0 | 3509.6 |
| 3 | 1.5 | 51 | 190 | 13.8 | 13.8 | 2.7 | 2.8 |
| 3 | 5.5 | 83.4 | 298.6 | 38.7 | 38.7 | 87.4 | 97.9 |
| 3 | 13 | 123.2 | 426.2 | 23.2 | 23.6 | 78.4 | 114.7 |
| 4 | 1.5 | 64.8 | 219.6 | 18.1 | 18.1 | 6.3 | 6.5 |
| 4 | 5.5 | 82.2 | 314.8 | 369.1 | 378.0 | 186.6 | 195.8 |
| 4 | 13 | 138.6 | 477 | 43.2 | 45.7 | 157.9 | 199.8 |

Table 3 compares the key performance measures for the slot-based and the time-based formulations, for 10 minute runs, where the percentage differences are computed as (slot-time)/time. These percentage differences are averaged across the five replications in each case. The objective function consists of a number of components, weighted according to priorities provided by our industrial partner. Table 3 presents average percentage differences for cycle time, idle time, throughput and the queue-time zone admission rate. A smaller cycle time and idle time are preferred, while a larger admission rate is preferred. On the other hand, given that the MILP is formulated as a minimization, our throughput measure is negative, with a more negative value being preferred. Thus, for the percent differences shown in Table 3, a negative value for cycle time and idle time (admission rate and throughput) implies that the slot-based (time-based) formulation performs better. Table 3 indicates that the time-based formulation generally performs better than the slot-based formulation, except in a few instances when the queue-time factor is 1.5. These results are consistent with those for the performance gap. In the table, the $*$ denotes a case in which one of the replications for the time-based formulation had idle time equal to zero. Thus, the percentage difference is undefined. In these cases, the entry in the table is the average percentage difference over the four remaining replications. Finally, the table indicates that the time-based formulation always does as least as well as the slot-based formulation on the admission rate.

## 5. Conclusions

In this paper, we considered the problem of scheduling a semiconductor wafer fab in the presence of queue-time constraints. We formulated a generalized MILP model of the local scheduling problem in the presence of queue-time

Table 3: Comparison of objective function components for slot-based vs. time-based (average % difference)

| WIP factor | QT factor | Cycle Time | Idle Time | Throughput | Admission Rate |
|---|---|---|---|---|---|
| 2 | 1.5 | 1.2 | -28.2* | 0.2 | -23.3 |
| 2 | 5.5 | 3.6 | 18.8 | -1.3 | 0.0 |
| 2 | 13 | 9.7 | 39.9 | -5.0 | -14.3 |
| 3 | 1.5 | -2.6 | 38.1* | 0.7 | -0.1 |
| 3 | 5.5 | 2.6 | 8.7 | -1.3 | -0.6 |
| 3 | 13 | 12.7 | 84.6 | -1.8 | -10.0 |
| 4 | 1.5 | -1.6 | 52.2* | -23.3 | -0.1 |
| 4 | 5.5 | 1.2 | 28.1 | -0.1 | 0.0 |
| 4 | 13 | 16.2 | 31.7 | -6.0 | -38.0 |

constraints. This model included decision variables associated with the gate-keeping decision, i.e., the decision regarding whether or not to admit an arriving job to the queue-time zone. We developed two alternative model formulations and compared the performance of these formulations using a set of numerical experiments generated based on process flow information provided by our industrial partner. One major goal of this research is to identify conditions under which each formulation performs best. The preliminary results indicate that which formulation is preferred depends critically on the problem parameters. Thus, extensive additional computational experiments are required in order to provide clear guidance regarding when each formulation should be applied. This is a topic for our future research.

## Acknowledgements

The authors gratefully acknowledge the funding provided for this research by GLOBALFOUNDRIES.

## References

[1] Yugma, C., Dauzere-Peres, S., and Artigues, C., 2012, “A Batching and Scheduling Algorithm for the Diffusion Area in Semiconductor Manufacturing,” International Journal of Production Research, 50(8), 2118-2132.

[2] Tu, Y.-M., and L, C.-S., 2006, “Capacity Determination Model with Time Constraints and Batch Processing in Semiconductor Wafer Fabrication,” Journal of the Chinese Institute of Industrial Engineers, 23(3), 192-199.

[3] Tu, Y.-M., and C, H.-N., 2009, “Capacity Planning with Sequential Two-Level Time Constraints in the Back-End Process of Wafer Fabrication,” International Journal of Production Research, 47(24), 6967-6979.

[4] Scholl, W., and Domaschke, J., 2000, “Implementation of Modeling and Simulation in Semiconductor Wafer Fabrication with Time Constraints Between Wet Etch and Furnace,” IEEE Transactions on Semiconductor Manufacturing, 13(3), 273-277.

[5] Lee, Y.-Y., Chen,C., and Wu, C., 2005, “Reaction Chain of Process Queue Time Quality Control,” ISSM, IEEE International Symposium, 47-50.

[6] Akkerman, R., Van Donk, D., and Gaalman, G., 2007, “Influence of Capacity- and Time Constrained Intermediate Storage in Two-Stage Food Production Systems,” International Journal of Production Research, 45(13), 2955-2973.

[7] Bixby, R., Burda, R., and Miller, D., 2006, “Short-Interval Detailed Production Scheduling in 300mm Semiconductor Manufacturing using Mixed Integer and Constraint Programming,” Proceedings of the Advanced Semiconductor Manufacturing Conference, 148-154.

[8] Kim, Y.-D., Joo, B.-J., and Choi, S.-Y., 2010, “Scheduling Wafer Lots on Diffusion Machines in a Semiconductor Wafer Fabrication Facility,” IEEE Transactions on Semiconductor Manufacturing, 23(2), 246-254.

[9] Wu, C.-H., and Lin, J.T., and Chien, W.,C-., 2010, “Dynamic Production Control in a Serial Line with Process Queue Time Constraint,” International Journal of Production Research, 48(13), 3823-3843.

[10] Wu, C.-H., Lin, J.T., and C, W.-C., 2012, “Dynamic Production Control in Parallel Processing Systems under Process Queue Time Constraints,” Computers & Industrial Engineering, 63, 192-203.