Heuristics for minimizing total weighted tardiness in complex job shops

S.J. MASON†*, J.W. FOWLER‡, W.M. CARLYLE§ and D.C. MONTGOMERY‡

†Department of Industrial Engineering, 4207 Bell Engineering Center, University of Arkansas, Fayetteville, Arkansas 72701
‡Department of Industrial Engineering, Arizona State University, P.O. Box 875906, Tempe, Arizona 85287-5906
§Operations Research Department, Naval Postgraduate School, Monterey, California 93943

(Received August 2004)

Semi-conductor manufacturing is arguably one of the most complex manufacturing processes in existence today. A semi-conductor wafer fabrication facility is comprised of batching machines, parallel machines, machines with sequence-dependent set-ups, and re-circulating product flow. The individual job release times and due dates combine with the other processing environment characteristics to form a ‘complex’ job shop scheduling problem. We first present a mixed-integer program (MIP) to minimize total weighted tardiness in a complex job shop. Since the problem is NP-hard, we compare a heuristic based on the MIP (MIP heuristic) with both a tuned version of a modified shifting bottleneck heuristic (SB heuristic) and three dispatching rules using random problem instances of a representative model from the literature. While the MIP heuristic typically produces superior schedules for problem instances with a small number of jobs, the SB heuristic consistently outperforms the MIP heuristic for larger problem instances. The SB heuristic’s superior performance as compared to additional dispatching rules is also demonstrated for a larger, ‘real world’ dataset from the literature.

Keywords: Scheduling; Integer programming; Shifting bottleneck; Heuristic

1. Introduction

Manufacturing integrated circuits on raw silicon wafers is arguably one of the most complex manufacturing processes in existence today. Individual wafers of a given product (job) type are required to complete approximately 200–400 process steps as they move through the wafer fabrication facility (wafer fab) in groups or lots of various size (for example, 25 wafers). Each lot of wafers competes with hundreds of other lots for processing time on the wafer fab’s 50–100 different tool groups. These tool groups, which have machines that can range in price from $100 000 to $14 000 000, typically contain a number of identical machines operating in parallel, some of which may be dedicated to the production of one or more specific product

*Corresponding author. Email: mason@uark.edu
(job) types. These machines process jobs in a number of ways, including single-wafer processing (such as photolithography), entire lot processing (such as wet sinks), and batch processing wherein multiple lots can be processed simultaneously (such as diffusion furnaces).

The capital-intensive nature of semi-conductor manufacturing necessitates the existence of a re-circulating or re-entrant processing environment in which a given job typically visits a given tool group a number of times during the process flow. In addition to re-entrant flow considerations, the parallel machine environment also exists due to machine reliability (unplanned downtime) issues. Some of the tool groups in the wafer fab, such as ion implanters, are subject to sequence-dependent set-ups, such that the amount of time required to properly configure a machine to process a subsequent job depends upon the machine’s current configuration as well as the selected job type. Finally, the existence of individual lot (job) release or ready times and due dates combine with the other processing environment characteristics mentioned above to present a ‘complex’ job shop scheduling problem (Mason et al. 2002). We consider here the problem of minimizing total weighted tardiness in a complex job shop, or

$$FJc[r_j, s_{jk}, B, recr] \sum w_j T_j$$

in the notation scheme of Lawler et al. (1982) and Pinedo (2002). The reader is referred to Uzsoy et al. (1992) for additional discussion of semi-conductor manufacturing.

A number of previous research efforts have focused on developing heuristics for the sub-problems that comprise this scheduling problem. In order to minimize the total weighted tardiness (TWT) of a single machine with sequence-dependent set-ups, Lee et al. (1991) extended Vepsalainen and Morton’s (1987) apparent tardiness cost priority rule in creating the apparent tardiness cost with set-ups (ATCS) index. Lee and Pinedo (1997) subsequently modified the ATCS index to create a heuristic for minimizing the TWT on parallel machines with sequence-dependent set-ups. However, re-entrant flow was not considered in any of this previous research.

Initial research efforts by Glassey and Weng (1991) and Weng and Leachman (1993) focused on developing batch formation strategies and real-time decision support methodologies for batch-processing machines. Fowler et al. (1992) investigated real-time control of single batch machines in a wafer fab, while Robinson et al. (1995) determined that knowledge of both lot (job) and machine statuses at upstream and downstream processing steps promoted better scheduling decisions on batch processing machines. Recently, Fowler et al. (2000) extended their initial real-time control work to multiple servers. While these researchers were cognizant of the re-entrant flow present in semi-conductor manufacturing, their research efforts were not focused on minimizing TWT on the batch processing machines.

Pinedo and Singer (1999) modified Adams et al.’s (1988) shifting bottleneck (SB) heuristic to evaluate the $Jm[r_j, \Sigma w_j T_j]$ problem. Although Pinedo and Singer (1999) demonstrate the efficacy of their approach over a number of problem instances, their results are not directly applicable to the $FJc[r_j, s_{jk}, B, recr] \Sigma w_j T_j$ problem, which is comprised of batching machines, parallel machines, machines with sequence-dependent set-ups, and re-entrant or re-circulating product flow. Mason et al. (2002) developed a modified SB heuristic for the $FJc[r_j, s_{jk}, B, recr] \Sigma w_j T_j$ problem that produced lower TWT values than four different dispatching rules.
In this paper, we present a mixed-integer programming heuristic (MIP heuristic) for minimizing TWT in a complex job shop. The performance of this MIP heuristic is compared with the modified shifting bottleneck heuristic of Mason et al. (2002) in terms of solution quality and execution speed for the mini-fab model of El Adl et al. (1996). For small problem instances, the MIP produces schedules with lower TWT than the modified SB heuristic, although it requires more execution time to arrive at these ‘better’ solutions. Therefore, using data from a larger, ‘real world’ complex job shop, the authors’ SB heuristic is compared with various dispatching rules to assess the ability of each approach to minimize the TWT in a complex job shop.

This paper is organized as follows. In section 2, we present our MIP heuristic for minimizing TWT in a complex job shop. The various components of the MIP are discussed, along with their respective roles in the overall scheduling problem. Section 3 describes the experimental design used to tune the SB heuristic’s solution performance, and then presents the comparison between the tuned SB heuristic and the MIP heuristic in terms of solution quality for various problem instances of the mini-fab model. Using a larger, more realistic wafer fab data model, the SB heuristic is compared with various dispatching rules in section 4 to assess the ability of each approach to maximize the wafer fab’s delivery performance. Finally, section 5 contains some concluding remarks, as well as potential directions for future research.

2. A MIP heuristic for minimizing TWT in complex job shops

Consider a set \( N \) of independent jobs, where job \( j \in N \) has a corresponding due date \( d_j \), ready time \( r_j \), priority or weight \( w_j \), and a predetermined process routing. Job \( j \)'s predetermined process routing describes the manufacturing operations or process steps required for the job to complete its processing. Let \( G \) denote the set of tool groups on which each job \( j \in N \) is processed. At a given process step in its process routing, job \( j \) is processed by one of the \( m_k \) identical (parallel) machines that comprise tool group \( k \in G \).

The process steps in job \( j \)'s process routing belong to one of two sets: single job or non-batching operations (\( O \)) and batching operations (\( S \)). The tool group required for processing a non-batching operation \( f \in O \) is mapped by \( o : O \rightarrow G \). In addition, the individual machine within tool group \( o(f) \) on which operation \( f \) is processed for \( p_f \) time units may be subject to a sequence-dependent set-up time of \( \psi_{gf} \) when changing the required machine’s configuration from operation \( g \) to operation \( f \).

Let \( R \) denote the set of batching recipes. Each batching step \( s \in S \) has an associated recipe \( \rho \in R \), which is mapped by \( \gamma : S \rightarrow R \). Jobs sharing the same recipe \( \rho \in R \) are processed (batched) together for \( \delta_{\rho} \) time units. The maximum number of jobs that can be batched together for recipe \( \rho \in R \) (the maximum batch size) is \( b_{\rho} \), while the minimum number \( b_{\rho} \) of batched jobs (the minimum batch size) must be at least one. If \( n \leq N \) jobs each require batch processing step \( s \in S \), then \( B_{\rho} = \{ \varphi \in \gamma^{-1}(\rho) \mid |\varphi| \leq b_{\rho} \} \) denotes the set of all potential batches for batching recipe \( \rho \in R \). Therefore, \( B_{\rho} \) contains \( \binom{n}{1} \) single-job batches, \( \binom{n}{2} \) two-job batches, and so on, with \( |B_{\rho}| = \sum_{i=1}^{\lceil \min(n,b_{\rho}) \rceil} \binom{n}{i} \).

If the tool group required for processing all steps with recipe \( \rho \in R \) is described by \( o : R \rightarrow G \), then the set of all potential batches for tool group \( k \in G \) is \( B = \bigcup_{\rho \in R} B_{\rho} \forall o(R) = k \). As is the case for non-batching operations, the individual
machine within tool group $\omega(h)$ on which operation $h \in B$ is processed may be subject to a sequence-dependent set-up time of $\psi_{ih}$ when changing the required machine’s configuration from potential batch $i$ to potential batch $h$. The objective is to find the schedule that minimizes $TWT (\Sigma_{j \in N} w_j T_j)$. As job $j$ may or may not visit each tool group $k \in G$ one or more times during its process routing, this problem is denoted by $FJc|r_j, s_{jk}, B, recr| \Sigma w_j T_j$.

2.1 Disjunctive graph for the complex job shop

The $FJc|r_j, s_{jk}, B, recr| \Sigma w_j T_j$ problem reduces to the $1|| \Sigma w_j T_j$ problem, which is known to be strongly $NP$-hard (Lenstra et al. 1977). Figure 1 depicts the disjunctive graph for the $FJc|r_j, s_{jk}, B, recr| \Sigma w_j T_j$ problem with $n = 3$ jobs and $m = 4$ tool groups. Node $U$ (operation 1) represents the artificial starting node that is common to all jobs. The set $V$ contains the artificial ending nodes for each job $j \in N$ (operations 6, 10, and 15), as the TWT objective function requires knowledge of each job’s completion time. The set $A$ is comprised of the conjunctive (solid) arcs in figure 1 and indicates the precedence constraints between operations of the same job, while the set of disjunctive (dashed) arcs $X$ denotes the precedence relationships between jobs that are processed on the same tool group $k \in G$. The cost or length of any arc $f \rightarrow g \in A$ equals $p_f$ for a non-batching operation and $\delta_f$ for a batching operation, the time required to process operation $f$.

Initially, the disjunctive graph does not indicate the existence of parallel machines. Before tool group $k$ is scheduled, the assignment of jobs to the $m_k$ machines in the tool group is unknown. Once this tool group’s schedule is determined, one arc in each of the disjunctive pairs associated with this tool group becomes conjunctive, establishing the precedence relationship between the pairs of operations performed by this tool group. The disjunctive graph of a feasible schedule

![Figure 1. Disjunctive graph for $FJc|r_j, s_{jk}, B, recr| \Sigma w_j T_j$ with $n = 3$ jobs and $m = 4$ tool groups.](image_url)
is acyclic. These newly added conjunctive arcs define the job-processing sequence on the $m_k$ individual machines in the tool group.

Tool groups 1 and 5 in figure 1 are batching tool groups with a maximum batch size of three and two jobs, respectively. The boxes in figure 1 list the potential batches that could be formed at the two tool groups. A batch process step’s processing time is not associated with the arcs leaving the corresponding node, but with the arcs leaving the node that represents the batch within which the associated job was processed. For example, figure 2 contains a feasible solution for the same problem displayed in figure 1. Note that while 0 time units are required to form a batch, the associated batch processing time $\delta_\beta$ is indicated on the arcs originating at each batch node. In addition, each pair of disjunctive arcs in figure 1 have been replaced by a single conjunctive arc for tool groups 1, 2, and 3, indicating the sequencing of the associated tool groups. Finally, the schedule depicted in figure 2 is feasible, as the graph is acyclic.

Figure 1 does not contain any reference to required sequence-dependent set-up times, as no tool groups have been scheduled. Assume that tool group $k$ is characterized by sequence-dependent set-ups. Once this tool group is scheduled, the cost or length of any arc $f \rightarrow g \in X$, where $X$ is the set of disjunctive scheduling arcs, equals $\Sigma_{e \in \partial f} \psi_{ef} \chi_{ef} + p_f$ for a non-batching operation and $\Sigma_{e \in B, e \neq f} \psi_{ef} \chi_{ef} + \delta_f$ for a batching operation. This cost equals the sum of the set-up time required to change a machine’s configuration from operation $e$ to operation $f$ and the time required to process operation $f$.

In the disjunctive graph of a job shop with no re-entrant or re-circulating flow, pairs of disjunctive arcs connect the set of operations performed on tool group $k$. Therefore, the set of disjunctive arcs corresponding to tool group $k$ forms a clique. However, when re-circulation is present in a job shop, as is the case in semiconductor manufacturing, a given job $j$ may contain multiple operations that require processing on tool group $k$. The operational precedence relationship for job $j$
requires that conjunctive arcs connect the nodes associated with this job. All pairs of operations in job \( j \) that require processing on tool group \( k \) are not connected with a disjunctive arc pair. Therefore, the existence of re-circulating flow through tool group \( k \) precludes the formation of a clique.

### 2.2 Mixed-integer program

Our objective is to minimize \( \Sigma_{j \in N} W_j T_j \), subject to various job, process routing, and tool group constraints. First, from the definition of tardiness, we have \( T_j - C_j \geq -d_j \) \( \forall j \in N \), with \( T_j \geq 0 \). Let \( y_f \) denote the start time of non-batch operation (node) \( f \), with \( y_f \geq 0 \) \( \forall f \in O \). Each non-batch node \( f \in O \) is associated with job \( j \in N \), which is mapped by \( \eta : O \rightarrow N \). Each job’s process routing, as defined by the set of conjunctive arcs \( A \), requires that \( y_f - y_f \geq p_f \) \( \forall (f, g) \in A; \eta(f) = \eta(g) \). In addition, \( y_f \geq r_j \) \( \forall (U, f) \in A \), as the first operation of job \( j \in N \) cannot begin its processing until the job is ready. Finally, it follows that

\[
C_j - y_f \geq p_f + \sum_{e \in O, e \neq f} \psi_{ef} x_{ef} \quad \forall (f, V_j) \in A, f \in O,
\]

as any sequence-dependent set-up time required for a job’s last operation must be included when determining the job’s completion time.

#### 2.2.1 Non-batching tool groups

If \( x_{fg} \in X \) is a \( \{0, 1\} \) variable indicating whether \( (x_{fg} = 1) \) or not \( (x_{fg} = 0) \) disjunctive arc \( f \rightarrow g \) becomes a conjunctive scheduling arc in node \( f \)'s associated non-batching tool group’s schedule for all \( f, g \in O \), then the job sequencing constraints for non-batching tool group \( k \in G \) can be stated as follows:

\[
y_g - y_f \geq p_f x_{fg} + \sum_{e \in O, e \neq f} \psi_{ef} x_{ef} - M(1 - x_{fg})
\]

(1)

\[
x_{gf} + x_{fg} \leq 1 \quad \forall f \neq g; \ f, g \in O; \ o(e) = o(f) = o(g) = k \in G
\]

(2)

\[
\sum_{f \in O} x_{fg} \leq 1 \quad \forall f \neq g; \ g \in O; \ o(f) = o(g) = k \in G
\]

(3)

\[
\sum_{g \in O} x_{fg} \leq 1 \quad \forall f \neq g; \ f \in O; \ o(f) = o(g) = k \in G
\]

(4)

\[
\sigma_g \geq 1 - \sum_{f \in O} x_{fg} \quad \forall f \neq g; \ g \in O; \ o(f) = o(g) = k \in G
\]

(5)

\[
\sum_{f \in O} \sigma_f \leq m_k \quad \forall o(f) = k \in G
\]

(6)

\[
x_{fg} + x_{gh} \leq 1 \quad \forall f, g, h \in O; \ f < g; \ g \neq h; \ \eta(f) = \eta(g); \ \eta(g) \neq \eta(h);
\]

\[
o(f) = o(g) = o(h) = k \in G
\]

(7)

Let \( \sigma_f \) be a \( \{0, 1\} \) indicator variable such that \( \sigma_f = 1 \) if operation \( f \in O \) is the first operation processed on a machine in non-batching tool group \( k \in G \) and
\( \sigma_f = 0 \) otherwise. Also, let \( M \) be a large positive number. Constraint (1) requires the start time of any operation \( g \in O \) that succeeds a given operation \( f \in O \) to be at least equal to

\[
y_f + p_f + \sum_{e \in O, e \neq f} \psi_{ef}x_{ef},
\]

thereby ensuring non-overlapping job processing at tool group \( k \). Note that (1) implies that no time is required to initially set-up a given machine before it begins processing jobs, as

\[
\sum_{e \in O, e \neq f} \psi_{ef}x_{ef} = 0 \quad \text{when } \sigma_f = 1.
\]

Constraint (2) permits at most one of the two disjunctive arcs connecting operations \( f \) and \( g \) to become a conjunctive scheduling arc. If operations \( f \) and \( g \) are sequenced on the same machine, equation (2) will be enforced with equality; otherwise, \( x_{gf} + x_{fg} = 0 \), as any two operations processed on different machines within the same tool group are not connected by a conjunctive scheduling arc. Constraints (3) and (4) maintain balance of flow at each operation (node) by requiring that each node can have at most one incoming and outgoing conjunctive scheduling arc, respectively. As the node corresponding to the first operation processed on a given machine will not have any incoming conjunctive scheduling arcs, equation (5) provides a means for identifying the first operation processed on a given machine in non-batching tool group \( k \in G \) (machine start node). Finally, equation (6) ensures that the number of machine start nodes is limited to the number of machines in the corresponding non-batching tool group, while equation (7) disallows cycles to be formed between nodes of different jobs that are processed on the same tool group.

### 2.2.2 Batching tool groups.

In order to describe the set of job sequencing constraints for batching tool groups, let \( x_{hi} \) be a \( \{0, 1\} \) variable indicating whether \( (x_{hi} = 1) \) or not \( (x_{hi} = 0) \) disjunctive arc \( h \rightarrow i \) becomes a conjunctive scheduling arc in node \( h \)'s associated batching tool group’s schedule for all \( h, i \in B \), the set of potentially formed batches. If \( \tau_h \) denotes the start time of batch node \( h \), with \( \tau_h \geq 0 \ \forall h \in B \), the job sequencing constraints for batching tool group \( k \in G \) can be stated as follows:

\[
\tau_h - y_f \geq p_f x_{fh} + \sum_{e \in O, e \neq f} \psi_{ef}x_{ef} - M(1 - \beta_h) \quad \forall h \in B, f \in h
\]

\[
\sum_{h \in B} x_{fh} = 1 \quad \forall f \in S
\]

\[
\sum_{f \in h} x_{fh} = \beta_h |h| \quad \forall h \in B
\]

\[
\sum_{g \in O \cup S} x_{hg} = \beta_h |h| \quad \forall h \in B
\]

\[
y_g - \tau_h \geq \delta_h x_{hg} + \sum_{e \in B, e \neq h} \psi_{eh}x_{eh} - M(1 - \beta_h) \quad \forall h \in B
\]
\[ \tau_i - \tau_h \geq \delta_h(x_{hi} - (1 - \beta_h)) + \sum_{e \in B, e \neq h} \psi_{eh} x_{eh} - M(1 - x_{hi}) \]

\[ \forall h, \ i \in B, \ h \cap i = \emptyset, \ \omega(e) = \omega(h) = \omega(i) = k \in G \] (13)

\[ x_{hi} + x_{hi} \leq 1 \ \forall h, \ i \in B, \ h \cap i = \emptyset, \ \omega(h) = \omega(i) = k \in G \] (14)

\[ \sum_{h \in B} x_{hi} \leq 1 \ \forall i \in B, \ h \cap i = \emptyset, \ \omega(h) = \omega(i) = k \in G \] (15)

\[ \sum_{i \in B} x_{hi} \leq 1 \ \forall h \in B, \ h \cap i = \emptyset, \ \omega(h) = \omega(i) = k \in G \] (16)

\[ \theta_i \geq 1 - \sum_{h \in B} x_{hi} - (1 - \beta_i) \ \forall i \in B, \ h \cap i = \emptyset, \ \omega(h) = \omega(i) = k \in G \] (17)

\[ \sum_{h \in B} \theta_h \leq m_k \ \forall \omega(h) = k \in G \] (18)

\[ \theta_h \leq \beta_h \ \forall h \in B \] (19)

Constraints (8)–(12) provide for the formation of batches, while (13)–(19) determine the sequencing of the formed batches. Let \( \beta_h \) be a binary indicator variable indicating whether \( (\beta_h = 1) \) or not \( (\beta_h = 0) \) potential batch \( h \in B \) is formed. Constraint (8) requires the start time of any batch operation \( h \in B \) that succeeds a given operation \( f \in h \) to be at least equal to \( y_f + p_f + \sum_{e \in O, e \neq f} \psi_{ef} x_{ef} \), thereby ensuring process routing continuity and obeying any existing sequence-dependent set-up requirements for operation \( f \). The restriction that each batch step can only join one batch is captured in equation (9). Constraints (10) and (11) enforce balance of flow at each batch node, ensuring the number of arcs entering and leaving batch node \( h \in B \) is equal to the size of the batch being formed. Finally, constraint (12) ensures the start time of any operation \( g \in O \) that succeeds batch operation \( h \in B \) is at least equal to \( \tau_h + \delta_h + \sum_{e \in B, e \neq h} \psi_{eh} x_{eh} \), again ensuring process routing continuity while observing sequence-dependent set-up requirements.

Let \( \theta_h \) be a \( \{0, 1\} \) indicator variable such that \( \theta_h = 1 \) if operation \( h \in B \) is the first operation processed on a machine in batching tool group \( k \in G \) and \( \theta_h = 0 \) otherwise. Constraint (13) enforces non-overlapping job processing at batching tool group \( k \) by requiring the start time of any batch operation \( i \in B \) that succeeds a given batch operation \( h \in B \) to be at least equal to \( \tau_h + \delta_h + \sum_{e \in B, e \neq h} \psi_{eh} x_{eh} \). As was the case for non-batching machines, constraints (8) and (13) imply that no time is required to initially set-up a given machine before it begins processing jobs, as \( \sum_{e \in B, e \neq h} \psi_{eh} x_{eh} = 0 \) when \( \theta_h = 1 \).

Constraint (14) permits at most one of the two disjunctive arcs connecting operations \( h \) and \( i \) to become a conjunctive scheduling arc. Balance of job sequencing flow is maintained at each batch node by (15) and (16), as they require that each batch node can have at most one incoming and outgoing conjunctive scheduling arc, respectively. Constraint (17) is the batching tool group version of equation (5), as it identifies the machine start node for batching tool group \( k \in G \). Finally, equation (18) ensures that the number of machine start nodes is limited to the number
of machines in the corresponding batching tool group, while equation (19) requires a batch to be formed ($\beta_h = 1$) before it can be a machine start node.

3. Comparison of tardiness minimization methodologies

3.1 The mini-fab model

El Adl et al. (1996) developed a mini-fab model to explore hierarchical modelling and control issues in re-entrant semi-conductor manufacturing facilities. While the mini-fab model contains only six processing steps performed by three different tool groups, it captures a number of the complexities inherent in a wafer fab (figure 3). Lots (jobs) flowing through the mini-fab visit each tool group twice, thus mimicking re-circulating flow. Tool groups 1 and 2 contain two machines operating in parallel, with tool group 1 containing batching machines that are capable of processing up to three jobs at a time. Jobs of any product type can be batched together at Step 1, while only jobs of the same product type may be batched together at Step 5. In order to replicate the sequence-dependent set-ups present in a wafer fab, the set-up time required for tool group 3 is assumed to be a function of product type and process step. When this tool group must be set-up to run a new product type, a 5-minute set-up is incurred. Similarly, when tool group 3 must be set-up to run a different process step, a 10-minute set-up is required. Finally, a 12-minute set-up is required to configure this tool group to run both a different process step and a new product type.

3.2 A MIP heuristic for complex job shops

Given sufficient time and computational resources, the MIP formulation of the complex job shop scheduling problem above will produce the optimal solution in terms of minimizing TWT. However, in practice, the amount of required solution time must be traded off with solution quality, as it makes no sense to wait an entire production shift (e.g. 12 hours in semi-conductor wafer fabs) to produce the optimal schedule for a set of jobs that have already completed their processing by the end of the shift. With this in mind, the MIP solver will be allowed 6 hours (i.e. one-half of a typical wafer fab production shift) to find its best solution for each problem instance. However, MIP heuristic solutions attained after 1 and 6 hours of wall clock time will be reported, as even 1 hour appears to be on the long side of acceptable in practice. Since the MIP is solved using a branch-and-bound approach, very often at least one
feasible solution has been produced by CPLEX prior to reaching the imposed solution time limit. The ‘best,’ sometimes optimal, feasible solution will be recorded during our experiments.

### 3.3 A shifting bottleneck heuristic for complex job shops

Decomposition methods for complex factory scheduling problems typically employ a ‘divide and conquer’ approach (Pinedo and Singer 1999). Adams et al.’s SB procedure (1988) decomposes the \( Jm|C_{\text{max}} \) problem into smaller, more tractable sub-problems, which are subsequently solved according to some specified sub-problem solution procedure (SSP). Once each sub-problem has been formulated and solved, it is evaluated in terms of a specified performance or machine criticality measure (Holtsclaw and Uzsoy 1996). The most critical machine is scheduled at each iteration of the procedure.

Mason et al.’s (2002) SB heuristic uses the batching apparent tardiness costs with set-ups (BATCS) rule to accommodate various types of tool groups that exist in a wafer fab (20). The BATCS rule was developed by extending Lee and Pinedo’s (1997) parallel machine apparent tardiness cost with set-ups rule to accommodate batch processing tool groups. BATCS is a composite dispatching rule (like its predecessor) that blends different heuristics that are effective when used in single-machine scheduling problems that consider job ready times and due dates with a factor that considers how full a batch is. The three heuristics are weighted shortest processing time, minimum slack, and set-up avoidance.

\[
I_{bj}(t, l) = \frac{w_{bj}}{p_{bj}} \exp\left( - \frac{(d_{bj} - p_{bj} + (r_{bj} - t))^+}{k_1\bar{p}} \right) \exp\left( - \frac{s_{bl,bj}}{k_2\bar{s}} \right) \left( \frac{|bj|}{b_i} \right) \tag{20}
\]

In (20), \( w_{bj} \) is the average weight of the jobs in batch \( bj \) and \( p_{bj} \) is the processing time of batch \( bj \), while \( d_{bj} \) and \( r_{bj} \) denote batch \( bj \)'s due date and ready time, respectively. Both \( k_1 \) (the look-ahead parameter) and \( k_2 \) (the set-up scaling factor) are as specified by Lee and Pinedo (1997). Further, \( \bar{p} \) and \( \bar{s} \) are calculated as the average processing time and set-up time of the potential batches that can be formed, while \( S_{bl,bj} \) is the sequence-dependent set-up time incurred when changing machine \( i \)'s configuration from batch \( bl \) to batch \( bj \). Finally, \( |bj| \) is the number of lots in batch \( bj \) (the size of batch \( bj \)) and \( b_i \) is the maximum number of jobs that can be batched together on tool group \( i \).

The BATCS index \( I_{bj}(t, l) \) is computed for each potential batch \( bj \) that can be formed from the set of unscheduled jobs (20). The batch with the largest \( I_{bj}(t, l) \) value is formed at time \( t \). The BATCS rule is flexible to accommodate all types of tool groups present in a wafer fab. For example, batching tool groups typically are not characterized by sequence-dependent set-up times (i.e. \( S_{bl,bj} = 0 \)). Therefore, the third term in (20) would drop out of the BATCS calculation, as it would equal 1 for all batches evaluated. Similarly, when a non-batching tool group must be scheduled, the batch size is one job (i.e. \( |bj| = b_i = 1 \)). This results in the fourth term in (20) dropping out of the BATCS calculation. The SB heuristic is capable of producing solutions to a 10-job instance of the mini-fab model in one second on a Pentium® IV 2.8 GHz machine with 1 GB of RAM; therefore, its computation time is negligible.
3.4 Tuning the heuristics

We use CPLEX as the solver for our MIP heuristic. While CPLEX provides the user with a default set of solver parameter settings, we constructed a mixed-level factorial experiment to identify the user-input settings that produced the most effective solutions for El Adl et al.’s (1996) mini-fab model problem instances, thereby ‘tuning’ the MIP heuristic’s performance for the \( F|j, s_{jk}, B, \text{recrc}\{\Sigma w_j T_j \} \) problem. The reader is referred to Mason et al. (2003) for further details.

A preliminary experiment was conducted using the mini-fab model to determine if the performance of the BATCS rule of Mason et al. (2002) could be improved by changing how a batch’s weight was calculated, how batches are formed, the length of the batching horizon, and when batching tool groups were scheduled. Two different methods for determining batch weight were investigated: computing the average weight of all jobs in a potential batch and assigning a batch weight equal to the maximum weight of all jobs in the batch. However, experimental results indicated no significant difference existed between the two proposed methods for calculating batch weight.

Two different batch formation approaches were also investigated. First, the heuristic would use the parallel-machine ATCS heuristic of Lee and Pinedo (1997) to find the single most critical job to schedule, then form the fullest batch possible with this job. The second approach was to simply employ the BATCS rule of Mason et al. (2002). As the number of jobs increase, the best batch formulation policy changes from ATCS to the BATCS rule, as the BATCS rule not only tries to form full batches, but also ensures the ‘best’ combination of jobs is selected for each batch. This is contrary to the ATCS policy, which identifies the most critical single job, then simply forms as full of a batch as possible with any other available jobs that match the selected job’s batching criteria.

Two different batching horizon policies were also studied: no batching horizon and an infinite batching horizon whereby all jobs are considered for batching, regardless of their ready times. The best batching horizon policy is to only consider jobs that are currently ready when there are either a small or large number of jobs in the fab. Experimental results suggest that when the shop is moderately loaded, considering all jobs when making batching decisions is the best batching horizon policy, regardless of the amount of time required for each job to become ready for batching. When the factory is lightly loaded, the amount of time the currently ready job(s) must wait for another job to arrive at the batching tool group can often exceed the amount of time required to process the ready job(s). Waiting for the arrival of the additional job to form a fuller batch would unnecessarily increase the completion time of all jobs in the batch. However, when the factory is heavily loaded, a sufficient number of jobs are typically ready at the batching tool group to form near or completely full batches, thereby alleviating the need for a non-zero batching horizon.

Finally, scheduling batching tool groups first typically resulted in the SB Heuristic producing lower TWT solutions for the mini-fab problem instances under consideration. This suggests the batching tool groups play a significant part in determining the TWT of each job in the complex job shop. Therefore, forming the ‘best’ batches, regardless of the resulting idle time that may be inserted into each tool group’s schedule, tends to produce schedules with low TWT.
3.5 Comparison of heuristic approaches

The tuned versions of the SB and MIP heuristics for the complex job shop problem are compared to three common dispatching rules using El Adl et al.'s (1996) mini-fab model. The dispatching rules used in the comparison tests were Lee and Pinedo's (1997) parallel-machine ATCS index with $k_1=k_2=0.1$ (ATCS), critical ratio, which favours jobs with small ratios of time remaining until the job is due (i.e., $d_j-t$) to the amount of processing time remaining to be completed for the job (CR), and earliest due date (EDD). All three of these dispatching rules considers job due date, with the ATCS approach also taking job weight into account.

We now consider 10 replicates (problem instances) of eight mini-fab test cases ranging in size from three to 10 jobs for three different maximum batch sizes: $b=2$, $b=3$, and $b=4$ jobs. In each of the 10 different problem instances created for each test case, job $j$'s weight $w_j$ has a 50% probability of being a discrete uniform random variable between one and 20 and a 50% probability of being a discrete uniform random variable between five and 15. These two cases represent different types of products that may exist in the complex job shop. Job $j$'s ready time $r_j$ will have a one-third probability of being equal to zero and a two-thirds probability of being uniformly distributed over the integers $[1, 250]$. Finally, job $j$'s due date $d_j$ is equal to $r_j + \zeta$, where $\zeta$ is distributed over the discrete uniform interval $[500, 750]$. As the theoretical processing time of a job is 625 minutes, this distribution of $\zeta$ will allow for both tight and loose job due dates.

Let $TWT(H, B, I)$ be the TWT value obtained by heuristic $H$ with batching horizon $B$ on problem instance $I$. When $H$ is the proposed MIP Heuristic or one of the three dispatching rule approaches, $B=\infty$. Further, let $BEST(I)=\min_{H,B}[TWT(H, B, I)]$. The results from the comparison of the two tuned solution methodologies and the three dispatching rules for the mini-fab instances of the complex job shop problem are given in tables 1 through 3 in terms of the ratio of $TWT(H, B, I)$ to $BEST(I)$ for the maximum batch size cases of $b=2$, $b=3$, and $b=4$, respectively. The values represent an average of the 10 replications performed for each number of jobs case. The ‘best’ scheduling heuristic is bolded for each test case, as a heuristic that produced the lowest TWT schedule in all 10 replications would have a ratio of $TWT(H, B, I)$ to $BEST(I)$ equal to one.

Table 1. Heuristic methodologies comparison for maximum batch size $b=2$ jobs.

<table>
<thead>
<tr>
<th>SB heuristic batching horizon (minutes)</th>
<th>No reoptimization</th>
<th>Reoptimization</th>
<th>Pure dispatching</th>
<th>MIP heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 60 120 180</td>
<td>0 60 120 180</td>
<td>ATCS CR EDD</td>
<td>After 1 hour</td>
</tr>
<tr>
<td>3</td>
<td>6.016 3.985 3.536 3.673</td>
<td>3.374 2.275 2.171 2.377</td>
<td>2.871 2.972 2.972</td>
<td>1.000</td>
</tr>
<tr>
<td>7</td>
<td>2.790 2.450 2.371 2.540</td>
<td>1.784 1.260 1.433 1.536</td>
<td>1.261 1.347 1.347</td>
<td>1.481</td>
</tr>
<tr>
<td>8</td>
<td>2.472 2.338 2.512 2.442</td>
<td>1.573 1.301 1.430 1.543</td>
<td>1.106 1.044 1.044</td>
<td>3.032</td>
</tr>
<tr>
<td>9</td>
<td>2.415 1.993 2.312 2.455</td>
<td>1.502 1.179 1.440 1.568</td>
<td>1.013 1.099 1.099</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>2.459 2.272 2.205 2.481</td>
<td>1.457 1.294 1.324 1.582</td>
<td>1.033 1.030 1.030</td>
<td>N/A</td>
</tr>
</tbody>
</table>

S.J. Mason et al.
In addition, when no feasible solutions were produced by a given heuristic approach for any of the 10 replications, N/A appears in the results tables.

The MIP heuristic outperforms the tuned SB heuristic with re-optimization for small problem instances with six jobs or less in one hour for all three maximum batch sizes investigated. However, the MIP heuristic’s superiority over the SB heuristic decreases as maximum batch size increases. Extending the MIP heuristic’s solution time to six hours allowed for superior TWT performance for the seven-job case, but only when $b = 2$. However, this amount of solution time is quite impractical for all practical purposes (in fact, one hour of MIP heuristic solution time could also be considered impractical in some cases). However, the probability of non-zero ready times dramatically improved the MIP heuristic’s ability to tighten its estimate of the problem’s lower bound, as infeasible, zero-length solutions were fathomed from the branch-and-bound tree in producing non-zero lower bounds on the TWT solution. The MIP heuristic found the optimal solution to all 10 three- and four-job mini-fab problem instances in 1 hour or less.

Table 2. Heuristic methodologies comparison for maximum batch size $b = 3$ jobs.

<table>
<thead>
<tr>
<th>SB heuristic batching horizon (minutes)</th>
<th>Pure dispatching</th>
<th>MIP heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>No reoptimization</td>
<td>Reoptimization</td>
<td>ATCS</td>
</tr>
<tr>
<td>Jobs 0 60 120 180</td>
<td>0 60 120 180</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2.880</td>
<td>2.354</td>
</tr>
<tr>
<td>7</td>
<td>2.710</td>
<td>2.086</td>
</tr>
<tr>
<td>8</td>
<td>2.247</td>
<td>2.267</td>
</tr>
<tr>
<td>9</td>
<td>2.071</td>
<td>2.078</td>
</tr>
<tr>
<td>10</td>
<td>2.373</td>
<td>2.091</td>
</tr>
</tbody>
</table>

Table 3. Heuristic methodologies comparison for maximum batch size $b = 4$ jobs.

<table>
<thead>
<tr>
<th>SB heuristic batching horizon (minutes)</th>
<th>Pure dispatching</th>
<th>MIP heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>No reoptimization</td>
<td>Reoptimization</td>
<td>ATCS</td>
</tr>
<tr>
<td>Jobs 0 60 120 180</td>
<td>0 60 120 180</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6.644</td>
<td>5.758</td>
</tr>
<tr>
<td>5</td>
<td>3.433</td>
<td>3.415</td>
</tr>
<tr>
<td>6</td>
<td>3.176</td>
<td>2.651</td>
</tr>
<tr>
<td>7</td>
<td>2.392</td>
<td>1.968</td>
</tr>
<tr>
<td>8</td>
<td>2.447</td>
<td>2.051</td>
</tr>
<tr>
<td>9</td>
<td>2.444</td>
<td>2.055</td>
</tr>
<tr>
<td>10</td>
<td>2.079</td>
<td>1.961</td>
</tr>
</tbody>
</table>
Although the tuned MIP heuristic was unable to produce a feasible solution for any of the nine- or 10-job problem instances in 1 hour, 6 hours of solution time resulted in the MIP heuristic producing solutions, albeit inferior ones, to six of the nine-job instances and six of the 10-job instances when \( b = 2 \). When maximum batch size is increased to \( b = 3 \) or \( b = 4 \), the MIP heuristic's performance degrades further, as it is able to produce a feasible solution to a single nine-job problem instance when \( b = 3 \). Finally, even though the MIP heuristic was able to produce better TWT solutions in a number of five- and six-job instances, the overall trend suggests the SB heuristic will outperform the tuned MIP heuristic in terms of solution quality as the problem size and/or maximum batch size increases.

The performance of the pure dispatching approaches investigated improves as the number of jobs increases. SB heuristic performance tends to surpass dispatching approaches when the maximum batch size is increased. However, the dispatching approaches, which require less than one second to process the 10 mini-fab jobs, could always be run as a sort of sanity check to verify the quality of any scheduling or optimization-based approach. In contrast, the SB heuristic requires an order of magnitude increase in computation time to schedule the same 10 mini-fab jobs. The incorporation of re-optimization into the SB heuristic appears to be worth the added computational effort (36% increase in computation time, on average), as re-optimization dramatically improves the solution quality produced by the SB heuristic. These preliminary experiments, while demonstrating that both the SB heuristic and the dispatching approaches are viable for producing schedules with low TWT, this initial experimentation is only concerned with 10 production jobs. In reality, the number of jobs to be scheduled in a complex job shop is often more than 10.

4. A larger wafer fab model

As Mason et al.'s (2002) SB heuristic consistently outperforms our MIP heuristic for the mini-fab model, subsequent experiments were run to test the heuristic’s performance in scheduling a larger, more complex wafer fab. The solution quality and execution speed of the heuristic were compared with seven well known dispatching rules in an attempt to measure each method’s effectiveness in minimizing TWT in a complex job shop. The dispatching rules considered are as follows:

- Earliest due date (EDD), which favours the job with the earliest due date.
- Priority-based earliest due date (PREDD), which selects the job with the highest priority (i.e., the job with the largest weight or the most important job). Within equal priorities, favour the job with the earliest due date.
- Priority-based first-in/first-out (PRFIFO), which selects the job with the highest priority. Within equal priorities, favour the job that first entered the tool group’s queue.
- Lee and Pinedo’s (1997) parallel machine apparent tardiness costs with set-ups rule (ATCS).
- Least slack (LSLACK), which schedules the job with the smallest slack, where slack is defined as the difference between the lot’s due date and remaining processing time.
- Priority-based critical ratio (PCR), which first selects the lots with highest priority, then schedules the one with the lowest critical ratio. If the job’s
due date is greater than the current time, the critical ratio is equal to 
\( (1 + \text{DueDate} - T_{\text{now}}) \times (1 + \text{LTF} \times \text{RPT})^{-1} \), where \( T_{\text{now}} \) is the current time, LTF is the lead-time factor and RPT is the job’s remaining process time. Otherwise, the critical ratio is computed as 
\( ((1 + T_{\text{now}} - \text{DueDate}) \times (1 + \text{LTF} \times \text{RPT}))^{-1} \) (Wright Williams & Kelly 2000).

- Shortest processing time (SPT), which favours the job with the shortest process time.

### 4.1 SEMATECH testbed datasets

In the early 1990s, a group of researchers at SEMATECH, then a United States’ semi-conductor research consortium, recognized that no representative, ‘real world’ wafer fab data existed in the public domain that could be used for testing new simulation packages, proposed heuristics and factory control policies, and other newly developed approaches to wafer fab/equipment scheduling. As part of SEMATECH’s Measurement and Improvement of Manufacturing Capacity project, Fowler et al. (1995) developed a structured set of file formats for specifying wafer fab data. While not containing any distributional information, the Testbed files provided a standardized way of capturing the process flow, rework, tool set, operator, and production volume requirements of a given wafer fab. Currently, there are seven factory datasets available at http://www.eas.asu.edu/~masmlab, along with a document detailing the format of each input file contained in the Testbed.

Testbed Dataset 1 was used to assess the effectiveness of the SB heuristic and the seven dispatching rules in minimizing TWT in a complex job shop. This dataset is composed of two different product flows and 83 different tool groups. Product 1 has 210 processing steps, while Product 2’s process routing contains 245 steps. Preliminary analysis of Dataset 1 was conducted using Wright Williams and Kelly’s (2000) Factory Explorer® v2.7 to identify the underlying factory’s bottleneck tools, as scheduling efforts are typically focused on these tools.

Dataset 1 represents a wafer fab whose corresponding tool set is sized for 16,000 wafer starts per month (320 lots or jobs per month); the maximum batch size for a majority of the batching tool groups is either four or six lots. To ensure some critical or bottleneck tools will exist in our investigation of a smaller subset of this production volume, the maximum batch size of each batching tool group was reduced from its specified value to two lots. In addition, the product mix in Dataset 1, which originally required two lots of Product 1 to be introduced into the factory for every lot of Product 2 started, was modified so that both products were released into the factory in equal amounts. This product mix modification should result in a greater probability of tool group set-ups and better batching efficiencies, as the fab’s batching criteria requires only like products be batched together on all batching tool groups except one. All tool group interruptions, wafer scrap and rework, and operators were removed from Dataset 1, as the SB heuristic does not take any of these factors into account during its analysis. Finally, in order to create a more appropriate set of data for our analysis, the sequence-independent set-up times associated with the medium- and high-current implanters in Dataset 1 were replaced with the sequence-dependent set-up times given in table 4.

These times are representative of the set-up required by ion implantation tools. Specifically, if there is any change in the tool’s required dopant species, 30 minutes
of set-up time is required. An additional 30 minutes is required to change from phosphorus or arsenic to boron or boron difluoride. Finally, changing the wafer’s tilt angle requires 15 minutes. Boron difluoride and arsenic steps have a tilt angle of 0 degrees, while boron and phosphorus steps required 7 degrees of tilt.

Once the factory’s product mix, batching tool group’s maximum batch size, and sequence-dependent set-ups were modified, Factory Explorer’s Capacity Analysis module was used to identify the top bottleneck tools of the modified dataset. The dataset’s top 10 bottleneck tools with purchase prices exceeding $100,000 were selected for the comparative analysis. If a tool were identified as a constraint in the fab that cost less than this amount, the fab director would surely be able to obtain the funds necessary to purchase one or more additional copies of this tool type. Wafer fabs contain a number of tools that cost in excess of $1,000,000—these more expensive tools should never consistently be starved of work because of the existence of a sub-$100,000 constraint tool.

The resulting bottleneck tool list from modified Dataset 1 is given in table 5 in descending order of constraint tool groups (30_DRIVE_OX is the modified dataset’s bottleneck tool). The columns in the table list each tool group’s name, tool type, the quantity of tools specified in the original Dataset 1, and the number of tools to be used in the comparative analysis. A serial tool is one that processes one job at a time, while a batch tool is capable of processing multiple jobs simultaneously. Finally, SDS refers to tools that are subject to sequence-dependent set-ups.

<table>
<thead>
<tr>
<th>Original quantity</th>
<th>Revised quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>30_DRIVE_OX</td>
<td>2</td>
</tr>
<tr>
<td>67_MATRIX</td>
<td>7</td>
</tr>
<tr>
<td>76_E_SINK</td>
<td>3</td>
</tr>
<tr>
<td>10_MED_CURRENT_IMP</td>
<td>4</td>
</tr>
<tr>
<td>11_HIGH_CURRENT_IMP</td>
<td>4</td>
</tr>
<tr>
<td>14_PEAK</td>
<td>2</td>
</tr>
<tr>
<td>6_NONCRIT_DEV</td>
<td>9</td>
</tr>
<tr>
<td>5_CRIT_DEV</td>
<td>12</td>
</tr>
<tr>
<td>31_OXIDE_1</td>
<td>6</td>
</tr>
<tr>
<td>37_POLY_DEP</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4. Sequence-dependent setup times for implanter tool groups.

<table>
<thead>
<tr>
<th>Proposed tool species</th>
<th>Phosphorus</th>
<th>Arsenic</th>
<th>Boron difluoride</th>
<th>Boron</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phosphorus</td>
<td>–</td>
<td>45</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>Arsenic</td>
<td>45</td>
<td>–</td>
<td>60</td>
<td>75</td>
</tr>
<tr>
<td>Boron difluoride</td>
<td>45</td>
<td>30</td>
<td>–</td>
<td>45</td>
</tr>
<tr>
<td>Boron</td>
<td>30</td>
<td>45</td>
<td>45</td>
<td>–</td>
</tr>
</tbody>
</table>
Only the top 10 bottleneck tool groups will be considered in our comparative analysis; all manufacturing steps that do not require processing on one of the tool groups listed in table 5 will be treated as simple processing delays. Therefore, the tool required for processing each non-critical step is replaced with a generic delay tool that has theoretically infinite capacity. This will shift the focus of the SB heuristic and the previously mentioned dispatching rules to the factory’s true bottleneck tool groups. Following this reduction procedure, Product 1’s 210 processing steps are reduced to 73 steps, while 97 processing steps will now be used to represent Product 2’s original 245 steps.

As only a percentage of Dataset 1’s production volume will be used in the comparative analysis, the tool set quantities need to be adjusted such that critical or bottleneck tools still existed in our experimental test cases. Preliminary experiments were conducted to determine the proper number of machines in each tool group for our analysis. The quantity of tools in tool group $k$ was determined using the daily going rate (DGR) as calculated by Factory Explorer of a single tool in $k$. A tool’s DGR is defined as the sum of product throughput rates divided by the tool’s capacity loading (Wright Williams & Kelly 2000). In other words, DGR expresses the number production units or jobs that a tool can process in a day. The quantity of tools in tool group $k$ was set equal to $\text{DGR}_{\text{max}} / \text{DGR}_k$, where DGR$_{\text{max}}$ is the maximum tool DGR across all ten-tool groups and DGR$_k$ is the DGR for a tool in group $k$. This specification of tool quantities was used to create a balanced tool set for use in the comparative analysis. Any fractional tool quantities were round up or down to the appropriate whole number of tools.

### 4.2 Experimental design

Ten problem instances are generated for two test cases of the modified Dataset 1 model: 20 jobs and 50 jobs. In each case, the jobs are divided evenly between Products 1 and 2. In each of the ten different problem instances created, job $j$ has its weight $w_j$ uniformly distributed in the interval $[1, 10]$, while its ready time $r_j$ has a 50% probability of being equal to zero and a 50% probability of being distributed uniformly on the interval $[1, 500]$. Finally, each job’s due date $d_j$ is uniformly distributed in the interval $[\text{RPT} - 0.25 \times \text{RPT}, \text{RPT} + 0.25 \times \text{RPT}]$, where RPT denotes each product’s raw processing time (18,959 minutes for Product 1 and 21,694 minutes for Product 2).

In testing the 20 and 50 job cases, every job has its own (potentially non-zero) ready time. Using this methodology, we emulate the existence of some jobs being available for scheduling/processing immediately (i.e. at $t = 0$), while also modelling future job arrivals to the system (i.e. jobs with $r_j > 0$). Each problem instance is scheduled using the seven dispatching rules discussed above and the SB heuristic using an infinite batching horizon. This infinite horizon enables the SB heuristic to determine if any benefit can be realized from delaying the formation and subsequent processing of a batch job in order to potentially form a fuller batch. The TWT of each of the eight schedules is measured to assess the ability of each methodology to minimize TWT in a complex job shop.
4.3 Experimental results

Let \( \text{TWT}(H, I) \) be the TWT value obtained by heuristic \( H \) on a problem instance \( I \). Further, let \( \text{BEST}(I) = \min_H [\text{TWT}(H, I)] \). Tables 6 and 7 present the results for the 20- and 50-job problem instances of the modified Dataset 1 experiments in terms of the ratio of \( \text{TWT}(H, I)/\text{BEST}(I) \). In these tables, MSB denotes the modified SB heuristic with an infinite batching horizon such that all potential jobs are considered at each batch process step regardless of their individual ready times. The average, variance, and median of the TWT\( (H, I) \) values are also displayed for each heuristic \( H \).

As table 6 indicates, the SB heuristic with an infinite batching horizon (MSB) produces schedules with the lowest TWT for seven of the ten 20-job test cases of modified Dataset 1 and thus has the lowest median value of TWT\( (H, I)/\text{BEST}(I) \). However, this method produced the worst schedule in test case 2, resulting in a slightly inflated value for both the average and variance of TWT\( (H, I)/\text{BEST}(I) \).

The case where a poor solution results from the SB heuristic (such as the 20-job Test Case 2), it would be easy to apply both the SB heuristic and one of the simpler dispatching rules (which can schedule many jobs in a few seconds), then select the best approach for the current problem instance. For the 50-job instances of modified Dataset 1, the SB heuristic produces the lowest TWT schedules in eight of the ten

<table>
<thead>
<tr>
<th>Test Case</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Avg</th>
<th>Var</th>
<th>Med</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRFIFO</td>
<td>1.09</td>
<td>1.03</td>
<td>1.46</td>
<td>1.24</td>
<td>1.25</td>
<td>1.15</td>
<td>1.25</td>
<td>1.07</td>
<td>1.22</td>
<td>1.13</td>
<td>1.19</td>
<td>0.02</td>
<td>1.19</td>
</tr>
<tr>
<td>ATCS</td>
<td>1.19</td>
<td>1.14</td>
<td>1.59</td>
<td>1.43</td>
<td>1.35</td>
<td>1.29</td>
<td>1.44</td>
<td>1.19</td>
<td>1.86</td>
<td>1.32</td>
<td>1.38</td>
<td>0.05</td>
<td>1.34</td>
</tr>
<tr>
<td>LSLACK</td>
<td>1.00</td>
<td>1.02</td>
<td>1.36</td>
<td>1.27</td>
<td>1.05</td>
<td>1.20</td>
<td>1.14</td>
<td>1.00</td>
<td>1.16</td>
<td>1.18</td>
<td>1.14</td>
<td>0.01</td>
<td>1.15</td>
</tr>
<tr>
<td>EDD</td>
<td>1.02</td>
<td>1.00</td>
<td>1.40</td>
<td>1.25</td>
<td>1.02</td>
<td>1.17</td>
<td>1.12</td>
<td>1.07</td>
<td>1.20</td>
<td>1.21</td>
<td>1.15</td>
<td>0.02</td>
<td>1.14</td>
</tr>
<tr>
<td>PCR</td>
<td>1.07</td>
<td>1.03</td>
<td>1.45</td>
<td>1.23</td>
<td>1.22</td>
<td>1.11</td>
<td>1.27</td>
<td>1.04</td>
<td>1.21</td>
<td>1.17</td>
<td>1.18</td>
<td>0.02</td>
<td>1.19</td>
</tr>
<tr>
<td>PREDD</td>
<td>1.07</td>
<td>1.03</td>
<td>1.45</td>
<td>1.24</td>
<td>1.25</td>
<td>1.11</td>
<td>1.27</td>
<td>1.03</td>
<td>1.21</td>
<td>1.17</td>
<td>1.18</td>
<td>0.02</td>
<td>1.19</td>
</tr>
<tr>
<td>SPT</td>
<td>1.07</td>
<td>1.21</td>
<td>1.62</td>
<td>1.45</td>
<td>1.33</td>
<td>1.43</td>
<td>1.74</td>
<td>1.21</td>
<td>2.18</td>
<td>1.36</td>
<td>1.46</td>
<td>0.10</td>
<td>1.39</td>
</tr>
<tr>
<td>MSB</td>
<td>1.06</td>
<td>2.90</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.15}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>1.21</td>
<td>0.36</td>
<td>1.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Avg</th>
<th>Var</th>
<th>Med</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRFIFO</td>
<td>1.00</td>
<td>1.02</td>
<td>1.11</td>
<td>1.27</td>
<td>1.01</td>
<td>1.12</td>
<td>1.29</td>
<td>1.14</td>
<td>1.15</td>
<td>1.24</td>
<td>1.14</td>
<td>0.01</td>
<td>1.13</td>
</tr>
<tr>
<td>ATCS</td>
<td>1.51</td>
<td>1.51</td>
<td>1.44</td>
<td>1.67</td>
<td>1.53</td>
<td>1.89</td>
<td>1.87</td>
<td>1.53</td>
<td>1.53</td>
<td>1.87</td>
<td>1.63</td>
<td>0.03</td>
<td>1.53</td>
</tr>
<tr>
<td>LSLACK</td>
<td>1.53</td>
<td>1.16</td>
<td>1.44</td>
<td>1.63</td>
<td>1.49</td>
<td>1.42</td>
<td>1.78</td>
<td>1.58</td>
<td>1.29</td>
<td>1.72</td>
<td>1.50</td>
<td>0.04</td>
<td>1.51</td>
</tr>
<tr>
<td>EDD</td>
<td>1.43</td>
<td>1.11</td>
<td>1.34</td>
<td>1.51</td>
<td>1.48</td>
<td>1.43</td>
<td>1.67</td>
<td>1.55</td>
<td>1.26</td>
<td>1.75</td>
<td>1.45</td>
<td>0.03</td>
<td>1.46</td>
</tr>
<tr>
<td>PCR</td>
<td>1.01</td>
<td>1.02</td>
<td>1.11</td>
<td>1.23</td>
<td>\textbf{1.00}</td>
<td>1.09</td>
<td>1.28</td>
<td>1.13</td>
<td>1.01</td>
<td>1.27</td>
<td>1.11</td>
<td>0.01</td>
<td>1.10</td>
</tr>
<tr>
<td>PREDD</td>
<td>1.00</td>
<td>\textbf{1.00}</td>
<td>1.12</td>
<td>1.22</td>
<td>1.00</td>
<td>1.10</td>
<td>1.26</td>
<td>1.19</td>
<td>1.15</td>
<td>1.25</td>
<td>1.13</td>
<td>0.01</td>
<td>1.14</td>
</tr>
<tr>
<td>SPT</td>
<td>1.66</td>
<td>1.59</td>
<td>1.64</td>
<td>1.81</td>
<td>1.81</td>
<td>1.82</td>
<td>2.02</td>
<td>1.82</td>
<td>1.62</td>
<td>2.12</td>
<td>1.79</td>
<td>0.03</td>
<td>1.81</td>
</tr>
<tr>
<td>MSB</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.08}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>1.20</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>\textbf{1.00}</td>
<td>1.03</td>
<td>0.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 6. Ratio of TWT\( (H, I)/\text{BEST}(I) \) for the 20-job instance of modified dataset 1. \( \text{BEST}(I) = \min_H [\text{TWT}(H, I)] \) is in bold.

Table 7. Ratio of TWT\( (H, I)/\text{BEST}(I) \) for the 50-job instance of modified dataset 1. \( \text{BEST}(I) = \min_H [\text{TWT}(H, I)] \) is in bold.
cases investigated (table 7). As was the case with the 20-job instance, the dispatching rules required less than five seconds each to completely schedule all 50 jobs. However, MSB requires an average of only 30 seconds to produce a schedule for the 50-job test cases on a Pentium® IV 2.8 GHz machine with 1 GB of RAM. Therefore, it is clear that as the number of jobs to be scheduled increases, the advantage of the MSB approach over pure dispatching continues to grow.

5. Conclusions and future research

A shifting bottleneck heuristic and a mixed-integer programming heuristic have been proposed for obtaining good solutions to the \( FJc|r_j, s_k, B, recr|\sum w_j T_j \) problem. The two solution approaches were compared using random problem instances of a representative model from the literature in terms of solution quality and execution speed. Experimentation using the best parameter settings indicates the MIP produces schedules with lower TWT than a modified shifting bottleneck heuristic from the literature for mini-fab model instances with a small number of jobs (four or less). However, the MIP requires additional execution time to arrive at these better solutions when compared to the time required by the modified SB heuristic to produce its solution. As the size of the mini-fab problem instance increases, the quality of the MIP’s TWT solution degrades significantly to the point that the MIP, for all problem instances larger than eight jobs, produces no solution.

The poor performance of the MIP, both in terms of solution quality and execution speed, is mostly due to the combinatorial number of batching combinations that often arises in the \( FJc|r_j, s_k, B, recr|\sum w_j T_j \) problem. The number of constraints required to describe the formation of batches and the sequencing of the corresponding batching nodes grows quite rapidly with problem instance size, causing the MIP to require more computing resources than the modified SB heuristic for a given problem. Future work involving the inclusion of additional cutting planes and the investigation of other polyhedral approaches, such as branch-and-cut, may help to promote faster, better TWT solutions. Applying various polyhedral techniques to the MIP presented here could result in a tightening of a given problem’s feasible region, thereby directly reducing the size of the corresponding branch and bound tree that must be analysed by the MIP solver.

The investigation into using the SB heuristic for scheduling larger wafer fabs suggests that while the SB heuristic consistently produces the best overall schedules in terms of TWT, a price must be paid in terms of solution speed. While the seven dispatching rules only required five seconds to produce a schedule, the SB heuristic required 30 seconds to schedule 50 jobs on a Pentium® IV 2.8 GHz machine with 1 GB of RAM. For even larger ‘real world’ problems with hundreds of jobs, the amount of time required to obtain a good solution may prove unacceptable. Clearly, the emergence of faster, parallel-processing computers will allow for reduced computation times, especially for the SB heuristic. These same machines, however, are still insufficient for optimally solving complex job shop scheduling problems with hundreds of jobs in any practical amount of computation time due to the combinatoric explosion in the number of batching possibilities that must be considered as \( n \) increases (see section 2’s discussion of \(|B_o|\) for more details).

Therefore, additional research needs to done to investigate methods for reducing the heuristic’s solution time while maintaining solution quality for even larger
problem instances. One potential area for future work that shows promise is to look at only those chains of nodes whose due date is affected by the newly scheduled tool group’s disjunctive arcs, rather than all nodes in the graph. This would reduce the number of nodes that must have their due dates recalculated at each iteration, directly resulting in a speed improvement. In addition, tool group-specific dispatching rules could be employed at the SSP level that may help to promote improved solution quality (Cigolini et al. 1999). Finally, investigation into potentially filtering initial batching combinations ‘smartly’ may also help to reduce the heuristic’s computation time.

Acknowledgements

This research was partially supported by the Semiconductor Research Corporation (97-FJ-492 and 2001-NJ-880) and the National Science Foundation (DMI-9713750). The authors would like to thank the two anonymous referees who helped to significantly improve the content and quality of this paper.

References