MINIMIZING FLOW TIME IN CYCLIC SCHEDULES FOR IDENTICAL JOBS WITH ACYCLIC PRECEDENCE: THE BOTTLENECK LOWERBOUND

by

W.T. Huh, P.L. Jackson, G. Janakiraman and N. Sawhney
Minimizing Flow Time in Cyclic Schedules for Identical Jobs with Acyclic Precedence: The Bottleneck Lower bound

Woonghee Tim Huh*, Ganesh Janakiraman, Peter L. Jackson
Operations Research and Industrial Engineering
Cornell University
Ithaca, NY 14853, USA

Nidhi Sawhney
Carmen Systems AB
Odinsgatan 9, Goteborg 411 03, Sweden.

Abstract

This paper studies minimizing flow time of a cyclic schedule for repeated identical jobs, where one job is started/completed in each cycle, subject to the schedule achieving maximum throughput. We propose a branch and bound method for a single machine problem, and use this method to derive an improved lower bound for the multiple machine problem.

Keywords: Cyclic Scheduling; Flow Time; Branch and Bound; Lower Bound.

*Corresponding author. Email: huh@orie.cornell.edu.
1 Cyclic Scheduling

Cyclic scheduling for manufacturing has been investigated as a paradigm for understanding the efficient operation of highly repetitive manufacturing systems. Hall [9] discusses qualitatively the benefits of cyclic scheduling in synchronizing manufacturing tasks. Whybark [17] describes the implementation of a periodic control system at a Finnish factory. Cyclic schedules are the basis for computing so called *takt* times in cellular manufacturing. Takt time computations are pervasive in modern discrete parts manufacturing. Such schedules are typically constructed manually in the form of operations routine sheets. [3]

The problem considered in this paper is a variant of the problem posed by Graves et al. [8] which is in the context of developing a scheduling system for integrated-circuit fabrication. Their problem is to schedule identical jobs cyclically in a re-entrant flow shop. Given a cycle length, their heuristic attempts to minimize the flow time of a cyclic schedule. Roundy [16] shows that this flow time minimization problem is NP-hard. He also develops a branch and bound search method for finding cyclic schedules which are not dominated by any other cyclic schedule in both cycle length and flow time.

Rao [15] uses a cyclic scheduling model for identical jobs to examine the tradeoff between the system performance measures of throughput and flow time. The precedence relationship among operations is either a serial line or an assembly line. He presents heuristics and analytic results for the problems of minimizing the cycle length, the flow time, or their weighted sum. The lower bound for flow time used for the purposes of evaluating these heuristics is based on the “longest path” to any operation, that is, the longest path of processing times in the precedence graph of the job. In this paper, we refer to this as the *Resource-Free Lower Bound* (RFLB) as it ignores contention for resource use by this or other jobs in the schedule.

Caggiano [4] considers cyclic schedules for a set of non-identical jobs. Precedence relations and delays between operations are considered. Given a fixed cycle length, her objective is to minimize a weighted sum of the flow times of the set of jobs. To evaluate her heuristics, she uses a weighted sum of the RFLBs for each job type.
Crama [6] presents basic combinatorial optimization problems, including cyclic scheduling problems, that are useful in production scheduling and discusses existing solution techniques and complexity results. Hanen and Munier [10] provide an extensive overview of structural results and algorithms for different cyclic scheduling problems. It is an excellent source for many references that explore different aspects of cyclic scheduling theory. Kamoun and Sriskandarajah [11] provide complexity results for three cyclic scheduling problems for flowshops, one of which is the problem of minimizing work-in-process inventory subject to achieving maximum throughput. Hall et al. [14] study complexity results for cyclic scheduling problems with cycle-time minimization as the objective. They also compare the complexity results with analogous results for makespan minimization problems. Middendorf and Timkovsky [12] show that many well-studied scheduling environments such as re-entrant flowshops, robotic flowshops, V-shops etc. are all special cases of an environment called a cycle-shop. They propose a classification of cycle-shops, survey existing complexity results and approximation results for different kinds of cycle-shops, and develop new results in this area. These results are also presented as tables that could serve as quick sources of reference.

We draw some insights from the relevant scheduling literature on the problem of minimizing the makespan of a set of non-repetitive jobs. Adams et al. [1] consider a makespan minimization problem for a jobshop. The main idea behind their Shifting Bottleneck procedure is to consecutively identify the "bottleneck" machine and sequence the operations on this machine using a method developed in Carlier [5]. In this paper, we focus on one bottleneck machine in developing a lower bound for the multiple machine problem. We also note that the problem addressed in our paper is a cyclic (or repetitive) version of the one considered by Balas et al. [2].

The problem considered in this paper is to find a cyclic schedule of minimum flow time for repeated identical jobs, where one job is started/finished in each cycle, subject to the schedule achieving maximum throughput. The job consists of multiple operations, and the precedence constraints between operations are specified by a general, directed, acyclic graph. Processing times on nodes and delays on arcs are specified. We let the bottleneck machine be the machine with the maximum load, where the load of a machine is defined
as the sum of processing times of the operations assigned to this machine. To ensure maximum throughput, the cycle length is set equal to the load of the bottleneck machine. In Section 2, we propose a branch and bound procedure to solve this problem when there is only one machine. In Section 3, this procedure is used to develop a lower bound on flow time for the multiple machine problem. This lower bound is superior to the RFLB. It is designed to yield improved benchmarks for heuristics. Some computational results are given in Section 4.

2 Single Machine Identical Job Problem

2.1 Problem Statement

Let \( V = \{1, 2, \ldots, n\} \) be the set of operations of the identical job, and let \( p : V \to \mathbb{Z}^+ \) be the processing times of each operation. Furthermore, let \( G = (V, A) \) be an acyclic, directed graph specifying the precedence relation of operations. Without loss of generality, we assume that \( G \) has dummy operations \( s \) and \( t \) with \( p(s) = p(t) = 0 \) such that \( s \) is the only source and \( t \) is the only sink in \( G \). We let \( q : A \to \mathbb{Z} \) represent the minimum delays between operations: for \( v \to w \in A \), operation \( w \) can commence no sooner than \( q(v \to w) \) time units after the completion of operation \( v \). All operations are performed on a single machine.

The cycle length \( C \) is given, and we assume that \( C \) satisfies \( \sum_{j \in V} p(j) = C \). This assumption ensures maximum throughput. It is essential in the subsequent branch and bound scheme. One job will be started every cycle and one job will be completed every cycle. The average number of jobs in process depends on the scheduled flow time of jobs.

We define the flow time of a cyclic schedule to be the length of time between the start and the end of a given job. The objective is to minimize the flow time using non-preemptive cyclic schedules.

Example 1 Figure 1 is a single machine problem with three operations. The cycle length is 5 units. The schedule given in Figure 2 is an optimal solution with flow time 16 units.
2.2 Computing a Lower Bound on Flow Time with Side Constraints

Suppose first that the starting epoch for each operation is restricted:

**Assumption 2** We are given \( a, b : V \rightarrow [0, C] \) such that the starting epoch for operation \( j \) is in the interval \([kC + a(j), (k + \delta(j))C + b(j)]\) for some integer \( k \), where \( \delta(j) = 1 \) if \( a(j) > b(j) \), and 0 otherwise.

Suppose we fix the starting epoch \( \gamma \in [0, C] \) for the dummy source operation \( s \). We recursively define the lower bound \( \phi(\gamma, n) \) on the starting epoch and the lower bound \( \eta(\gamma, n) \) on the ending epoch for each operation \( n \) given \( \gamma \) as the following:

1. \( \phi(\gamma, s) = \gamma = \eta(\gamma, s). \)

2. For operation \( j \neq s \), \( \phi(\gamma, j) \) is the smallest number such that it is greater than or equal to \( \eta(\gamma, j') + q(j' \rightarrow j) \) for each \( j' \in V \) with \( j' \rightarrow j \in A \), and that it is in some interval defined by \( a(j) \) and \( b(j) \) in Assumption 2:

\[
\phi(\gamma, j) = \min_{k \in \mathbb{N}} \min \{ \phi : \phi \geq \eta(\gamma, j') + q(j' \rightarrow j) \land j' \rightarrow j \in A; \phi \in [kC + a(j), (k + \delta(j))C + b(j)] \}.
\]

3. \( \eta(\gamma, j) = \phi(\gamma, j) + p(j). \)

It follows that \( \eta(\gamma, t) - \gamma \) is a lower bound on the flow time with given \( \gamma \) and Assumption 2. The above calculation can be performed in \( O(|A|) \) time: the calculation time is linear with respect to the size of the set of precedence relations.

We remark that if we set \( a(j) = 0 \) and \( b(j) = C \) for each \( j \in V \), then we obtain a lower bound \( \eta(\gamma, t) - \gamma |_{\gamma=0} = \eta(0, t) \), which corresponds to the length of a critical path. We call this the **Resource-Free Lower Bound** (RFLB).

Now, we want to find a lower bound on the flow time with side constraints for any value of \( \gamma \). For each arc \( j \rightarrow k \in A \), let its length be \( p(j) + q(j \rightarrow k) \). Let \( P_j \) denote a sequence of arcs in \( A \) describing an acyclic path from source \( s \) to node \( j \). We say an \( s \)-\( j \) path \( P_j \) is a **critical path** for operation \( j \), if it is a longest \( s \)-\( j \)
path. Let \( r_j \) be the length of a critical path \( P_j \). Then, in any feasible schedule, \( \gamma + r_j \leq \phi(\gamma, j) \). We say operation \( j \) is \textit{tight} with respect to \( \gamma \), \( a \) and \( b \) provided \( \gamma + r_j = \phi(\gamma, j) \). We denote by \( T(\gamma) \) the set of tight operations with respect to \( \gamma \), \( a \) and \( b \). It is easy to verify that if \( j \in T(\gamma) \) and an operation \( i \) is on \( P_j \), then \( i \) is also in \( T(\gamma) \). We also say operation \( j \) is \textit{right-justified} with respect to \( \gamma \) if \( \phi(\gamma, j) \mod C = b(j) \). Let \( R(\gamma) \) be the set of right-justified operations with respect to \( \gamma \). Let \( B = \text{argmin}_{\gamma \in [0, C]} \eta(\gamma, t) - \gamma \), the set of optimal flow-time minimizing values of \( \gamma \) under Assumption 2.

\textbf{Theorem 3} Under Assumption 2 (with the allowable start time restriction defined by \( a \) and \( b \)), there exists an optimal \( \gamma^* \in B \) such that \( T(\gamma^*) \cap R(\gamma^*) \neq \emptyset \).

\textbf{Proof:} Choose \( \gamma \in B \) such that \(|T(\gamma)| + |R(\gamma)|\) is as large as possible. If \( T(\gamma) \cap R(\gamma) \neq \emptyset \), then we obtain the required result. We proceed assuming otherwise. We claim that there exists \( \epsilon_0 > 0 \) such that for all \( 0 \leq \epsilon \leq \epsilon_0 \), we have \( T(\gamma) \subseteq T(\gamma + \epsilon) \), \( R(\gamma) \subseteq R(\gamma + \epsilon) \), and \( T(\gamma + \epsilon) \cap R(\gamma + \epsilon) \neq \emptyset \).

Suppose we increase \( \gamma \) by a sufficiently small amount \( \epsilon > 0 \). Since \( T(\gamma) \cap R(\gamma) = \emptyset \), for each \( j \in T(\gamma) \), we have \( j \notin R(\gamma) \), and the values of \( \phi(\gamma, j) \) and \( \eta(\gamma, j) \) increase exactly by \( \epsilon \), implying \( r_j = \phi(\gamma, j) - \gamma = \phi(\gamma + \epsilon, j) - (\gamma + \epsilon) \). Thus, \( j \) is in \( T(\gamma + \epsilon) \). Furthermore,

\[ (b(j) - \phi(\gamma, j)) \mod C \tag{2.1} \]

decreases by \( \epsilon \).

For \( j \notin T(\gamma) \), \( \phi(\gamma, j) \) and \( \eta(\gamma, j) \) are unchanged by \( \epsilon \). Thus, if \( j \) is in \( R(\gamma) \), then it is also in \( R(\gamma + \epsilon) \).

We also note that

\[ (\phi(\gamma, j) - \gamma) - r_j \tag{2.2} \]

decreases by \( \epsilon \).

Let \( \epsilon_0 > 0 \) be the smallest \( \epsilon \) for which one of (2.1) and (2.2) vanishes for some \( j \). Then a tight operation in \( T(\gamma) \) becomes right-justified with respect to \( \gamma + \epsilon_0 \), or a right-justified operation in \( R(\gamma) \) becomes tight with respect to \( \gamma + \epsilon_0 \), \( a \) and \( b \). We get \( T(\gamma + \epsilon_0) \cap R(\gamma + \epsilon_0) \neq \emptyset \) proving the claim.
We also note that for all positive $\epsilon \leq \epsilon_0$, we have that $\eta(\gamma + \epsilon, j) \leq \eta(\gamma, j) + \epsilon$ for all $j \in V$. Increasing $\gamma$ by $\epsilon$ does not increase flow time $\eta(\gamma + \epsilon, t) - (\gamma + \epsilon) \leq \eta(\gamma, t) - \gamma$. Thus $\gamma + \epsilon_0$ is in $B$.

Theorem 3 permits us to restrict the set of candidates for an optimal $\gamma$ in Lemma 5, below. In particular, let $\Gamma = \{(b(j) - r_j) \mod C | j \in V\}$. Since there is some optimal $\gamma^*$ in $B$ such that some operation is both tight and right-justified, we have the following corollary:

**Corollary 4** Under Assumption 2, given $a$ and $b$,

$$
\min_{0 \leq \gamma < C} [\eta(\gamma, t) - \gamma] = \min_{\gamma \in \mathbb{R}} [\eta(\gamma, t) - \gamma].
$$

(2.3)

**Proof:** From Theorem 3, we know that there exist $\gamma \in B$ and operation $j$ such that $j \in T(\gamma) \cap R(\gamma)$. Thus, $\gamma + r_j = \phi(\gamma, r_j)$, and $\phi(\gamma, r_j) \mod C = b_j$. We then obtain $r_j \equiv b_j - r_j (\mod C)$.

### 2.3 Minimum Flow Time Given a Fixed Cyclic Ordering

Suppose we fix a cyclic ordering $\sigma$ of the operations on the single machine. For each $j \in V$, we set a “point” restriction. That is, if $\sigma_1, \sigma_2, \cdots, \sigma_n$ is a permutation of operations, then we set $a(\sigma_j) = b(\sigma_j) = (\sum_{h=1}^{j-1} p(\sigma_h)) \mod C$, where $h$ is used to index all operations before $j$ in $\sigma$. The minimum flow time under a given $\sigma$ can be expressed in terms of the epoch bounding functions as follows.

**Lemma 5** Given a cyclic permutation $\sigma$ of operations,

$$
\min_{0 \leq \gamma \leq C} [\eta(\gamma, t) - \gamma]
$$

(2.4)

is the minimum flow time.

**Proof:** By definition of $\phi$ and $\eta$, $\eta(\gamma, t) - \gamma$ is a lower bound on the flow time given $\gamma$. Since $\gamma$ is in $[0, C]$, $\min_{0 \leq \gamma \leq C} [\eta(\gamma, t) - \gamma]$ is a lower bound on the flow time for all $\gamma$.

By choice of $a$ and $b$, the processing interval of $[a(j), a(j) + p(j)]$ for each operation $j$ is fixed up to $\mod C$, and mutually exclusive except possibly at end points. Then, $\phi$ and $\eta$ indicate the actual starting and ending
Figure 1: A Single Machine Example: Processing Times, Minimum Delays, and Precedence of Operations

Figure 2: A Single Machine Example: An Optimal Schedule with Flow Time 16 Units

<table>
<thead>
<tr>
<th>Cyclic Perm. $\sigma$</th>
<th>$\gamma$</th>
<th>$\phi$</th>
<th>$\eta$</th>
<th>Op</th>
<th>s</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>$t$</th>
<th>$\eta(\gamma, t) - \gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 2, 3)</td>
<td>0</td>
<td>$\phi$</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>11</td>
<td>17</td>
<td>21</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta$</td>
<td>0</td>
<td>6</td>
<td>12</td>
<td>20</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$\phi$</td>
<td>3</td>
<td>5</td>
<td>11</td>
<td>17</td>
<td>21</td>
<td></td>
<td>18</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta$</td>
<td>3</td>
<td>6</td>
<td>12</td>
<td>20</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$\phi$</td>
<td>4</td>
<td>5</td>
<td>11</td>
<td>17</td>
<td>21</td>
<td></td>
<td>17</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta$</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>20</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>$\phi$</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>16</td>
<td>20</td>
<td></td>
<td>19</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta$</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>19</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$\phi$</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>16</td>
<td>20</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\eta$</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>19</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Computation of $\phi$ and $\eta$ for Cyclic Permutations on Operations and Some $\gamma$ Values
epochs, respectively, of each operation (as opposed to a lower bound). By setting $\gamma_0 = \arg\min_{0 \leq \gamma \leq C} [\eta(\gamma, t) - \gamma]$, we obtain a cyclic schedule with the minimum flow time $\eta(\gamma_0, t) - \gamma_0$.

From (2.3) and (2.4), we remark that given a cyclic permutation $\sigma$ of operations, the minimum flow time can be computed in $O(|V||A|)$ time.

**Example 6** We continue with Example 1. Since there are 3 operations, there are 2 cyclic permutations of $\{1, 2, 3\}$. For $\sigma = (1, 2, 3)$, the point restriction implies $(a(1), a(2), a(3)) = (b(1), b(2), b(3)) = (0, 1, 2)$. From $(r_1, r_2, r_3) = (1, 3, 7)$, we get $L = \{4, 3, 0\}$. By Corollary 4, Table 1 demonstrates that the minimum flow time given $\sigma = (1, 2, 3)$ is 17 units. For $\sigma = (1, 3, 2)$, we have $(a(1), a(2), a(3)) = (b(1), b(2), b(3)) = (0, 4, 1)$ and $L = \{1, 4\}$. The minimum flow time given $\sigma = (1, 3, 2)$ is 16 units, which is the optimal value.

### 2.4 Computing Minimum Flow Time: Branch and Bound

Let $l^*(\sigma)$ denote the minimum flow time of a cyclic schedule for the single machine problem with cyclic permutation $\sigma$ of operations. Let $l^{**}$ denote the minimum of $l^*(\sigma)$ over all cyclic permutations $\sigma$. In this section, we describe a branch and bound method for determining $l^{**}$.

The branch and bound tree is constructed by progressively fixing the values of $\sigma_i$, for $i = 1, 2, \ldots, n$. A node at level $d \in \{1, 2, \ldots, n\}$ of the tree corresponds to the first $d$ values of the permutation sequence being fixed. Let $\tilde{\sigma}$ denote the partial sequence associated with a node and let $\overline{V}$ denote the operations in $V$ that are not in the partial sequence $\tilde{\sigma}$. The children of node $\tilde{\sigma}$ in the branch and bound tree are all nodes with partial sequences of the form $\text{concat}(\tilde{\sigma}, j)$ for $j \in \overline{V}$.

Let $l(\tilde{\sigma})$ denote a lower bound on any permutation of operations in $V$ that begins with the partial sequence $\tilde{\sigma}$, computed as follows. For $i = 1, 2, \ldots, d$, set $a(\sigma_i) = b(\sigma_i) = \sum_{h=1}^{i-1} p(\sigma_h)$. For each $j \in \overline{V}$, set $a(j) = a(\sigma_d) + p(\sigma_d)$ and $b(j) = C - p(j)$. That is, fix the start and end epochs of the first $d$ operations as per the known sequence and bound the start and end epochs of the remaining operations to occur somewhere in the interval $[\sum_{i=1}^{d} p(\sigma_i), C]$ modulo C. For these constraint functions $a(\cdot)$ and $b(\cdot)$, compute candidate
offset set, $\Gamma$, and using (2.3), the optimal offset, $\gamma^*$. The value of of $l(\bar{\sigma})$ is given by $\eta(\gamma^*, t) - \gamma^*$. Clearly, $l(\bar{\sigma}) \leq l^*(\sigma)$ for any sequence $\sigma$ that begins with $\bar{\sigma}$.

Note that if $\bar{\sigma}$ represents a full permutation (i.e., $d = n$), then $l(\bar{\sigma}) = l^*(\bar{\sigma}) \geq l^*$, and this value can be used to fathom unexplored nodes, $\bar{\sigma}'$, of the tree for which $l(\bar{\sigma}') \geq l(\bar{\sigma})$. Also, note that if the branch and bound algorithm is terminated early, then the minimum value of $l(\bar{\sigma})$ across all leaf nodes of the tree constitutes a lower bound on $l^*$. The rule to pick which node to explore next is simply that the unfathomed node with the minimum value of $l(\bar{\sigma})$, with ties broken arbitrarily.

3 Multiple Machine Identical Job Problem: A Lower Bound

3.1 Problem Statement

We let $G = (V, A)$, $p$ and $q$ as before. Let $M = \{1, 2, \cdots, h\}$ be the set of dedicated machines, and let $m : V \rightarrow M$ be the assignment of operations to machines. As in the single machine problem, the cycle length $C$ is given, and we assume that $\sum_{j \in V : m(j) = m_o} p(j) = C$, for some $m_o \in M$. We say that such a machine $m_o$ is a bottleneck machine. For each machine $m' \in M$, we require $\sum_{j \in V : m(j) = m'} p(j) \leq C$, for feasibility. In this section, we develop a lower bound on the flow time for this problem.

3.2 Lower Bound: Relaxation to a Single Machine Problem

Suppose that the multiple machine identical job problem $P$ specified by $G = (V, A)$, $p$, $q$ and $m$ has minimum flow time $z$. We will specify a related problem $P'$ by $G' = (V', A')$, $p'$, $q'$ and $m'$ such that the minimum flow time $z'$ of $P'$ is a lower bound on $z$.

Let $v \in V - \{s, t\}$ be any operation. Define $V' = V - v$. Let $p'$ and $m'$ be the same as $p$ and $m$ defined on $V'$. Let $A' = A'_1 \cup A'_2$ where $A'_1 = \{u \rightarrow w | u \rightarrow v \in A, v \rightarrow w \in A\}$ and $A'_2 = \{e \in A | e \text{ neither leaves nor enters } v\}$. 


For each $e = u \rightarrow w \in A'$, let

$$ q'(e) = \begin{cases} 
q(u \rightarrow v) + p(v) + q(v \rightarrow w), & \text{if } u \rightarrow w \in A'_1; \\
q(e), & \text{otherwise.}
\end{cases} $$

For $u, v, w \in V$ satisfying $u \rightarrow v, v \rightarrow w \in A$, in any feasible schedule in $P$, operation $w$ can start no earlier than $q(u \rightarrow v) + p(v) + q(v \rightarrow w)$ time units after operation $u$ is finished. Thus the minimum flow time for $P'$ is a lower bound on the minimum flow time for $P$.

If we repeat the reduction process one at a time to operations $\{v \in V | m(v) \neq m_o\}$ not assigned to the bottleneck machine $m_o$, then we eventually obtain a single machine problem $P''$. By finding the minimum flow time to problem $P''$ using the method in Section 2, we derive a lower bound for $P$. We refer to this as the Bottleneck Lower Bound (BLB).

**Example 7** Now consider the three machine example given in Figure 3. The RFLB is 11 units. By applying reductions, we obtain a single machine problem as in Example 1. Thus, the BLB is 16 units. The minimum flow time for this example is also 16 units, as shown in Figure 4.

## 4 Computation

The lower bound techniques described above has been implemented on randomly generated test problems with up to 40 operations on the bottleneck machine. This section reports the findings of those tests.

For any given node set $\{1, 2, \cdots, n\}$, we construct a randomized directed acyclic precedence graph by considering independently all pairs $(v, w)$ of the form $1 \leq v < w \leq n$ and by adding arc $v \rightarrow w$ to the graph according to a Bernoulli random variable with probability of success $p$. Thus, the density of the graph are correlated with the arc probability $p$. We considered three possible values of $p$: 0.2, 0.5 and 0.8.

Randomized processing times for each node are drawn from a uniform distribution over either $\{1, 2, \cdots, 5\}$ ("short" processing times) or $\{1, 2, \cdots, 10\}$ ("long" processing times). Delays on arcs in $A$ are either set to 0 ("no delay"), or are drawn from a uniform distribution over $\{1, 2, \cdots, 5\}$ ("delay").
Figure 3: A Multiple Machine Example: Processing Times, Minimum Delays, and Precedence of Operations

Figure 4: A Multiple Machine Example: An Optimal Schedule with Flow Time 16 Units
For the single machine case, we consider small problems with the number \( n \) of operations to be either 3, 6, or 9. For the multiple machine case, we parametrize problems by the number \( N_o \) of operations on the bottleneck machines, and consider problems with \( N_o = 3, 8, 20 \) and 40 bottleneck operations. We set \( n = N_o + 4 \) and choose the \( N_o \) bottleneck operations at random (uniform without replacement) from the set \( \{1, 2, \cdots, n\} \). Two of the remaining operations are assigned to a second machine at random, and the remaining operations to a third machine.

The purpose of considering single machine problems was to explore the growth in running time of the lower bound algorithm as a function of problem size and constraint density. Table 2 summarizes the running time statistics, both mean and standard deviation, for different combinations of problem size and arc probability. For each combination, 27 problem instances were generated using delays and short processing times. The minimum flow time was computed for each instance using the branch and bound technique of Section 2.4. The running time units are CPU seconds on Sun Microsystems ULTRA 10 with 128 megabytes of memory. As anticipated, the mean running time increases exponentially with problem size. The variability in running time for a given size decreases with the density of the precedence graph (i.e. with the number of constraints).

Since we have no practical way to compute optimal flow times for multiple machine problems, particularly large problems, the main purpose of experiments with multiple machine problems was to contrast the bottleneck lower bound (BLB) with the best available alternative lower bound, the resource-free lower bound (RFLB). Of interest, therefore, is the ratio of BLB to RFLB for each problem instance. Tables 3 and 4 summarize the mean and standard deviation of this ration over combination of problem size (number of operations on the bottleneck machines), arc probability \( (p) \), delay \( (No \ delay \ or \ Delay) \), and processing time \( (Short \ or \ Long) \). For each combination, 27 problem instances were generated and the BLB and RFLB were computed for each problem instance. In the case of large problem sizes \( (N_o = 20 \ or \ 40) \), the branch bound algorithm was terminated after 120 CPU seconds and the best available lower bound reported.

As Table 3 demonstrates, the BLB resulted in an overall average improvement of 68% over the RFLB and the improvement tended to increase with problem size. For the same problem size, the improvement
decreased in variability, and perhaps in mean, with constraint density \((p)\). Given the variability in results (Table 4), no conclusion can be drawn concern the impact of processing and delay time lengths on the magnitude of improvement.

5 Conclusion

Cyclic scheduling has been investigated as a paradigm for understanding efficiency in repetitive manufacturing systems. It is at the heart of cellular manufacturing analysis and has been applied to large scale re-entrant flow problems in semiconductor fabrication. Flow time minimization in cyclic schedules is known to be NP-Hard and so research has been focused on efficient heuristics to find minimum flow time schedules. To date, the evaluation of these heuristics has been hampered by the lack of a good lower bound on minimum flow time. In this paper, we have presented a technique that finds the minimum flow time for a single machine problem and provides a lower bound on the minimum flow time for multiple machine problems. In numerical experiments, this so-called bottleneck lower bound proved to be 68% better, on average, than the weaker resource-free lower bound used in past studies. Future studies on cyclic scheduling heuristics should use the improved bound.
<table>
<thead>
<tr>
<th>Num of Ops</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = 0.2</td>
<td>p = 0.5</td>
<td>p = 0.8</td>
</tr>
<tr>
<td>3</td>
<td>0.0000</td>
<td>0.0011</td>
</tr>
<tr>
<td>6</td>
<td>0.0811</td>
<td>0.0352</td>
</tr>
<tr>
<td>9</td>
<td>29.1730</td>
<td>1.5285</td>
</tr>
<tr>
<td>Overall</td>
<td>9.7514</td>
<td>0.5216</td>
</tr>
</tbody>
</table>

Table 2: Running Times for the Single Machine Problem (in CPU seconds)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p = 0.2</td>
<td>1.2535</td>
<td>1.1716</td>
<td>1.2548</td>
<td>1.2733</td>
<td>1.2383</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>1.1206</td>
<td>1.1789</td>
<td>1.1479</td>
<td>1.2490</td>
<td>1.1741</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>1.0231</td>
<td>1.2229</td>
<td>1.0258</td>
<td>1.1899</td>
<td>1.1154</td>
</tr>
<tr>
<td>8</td>
<td>p = 0.2</td>
<td>1.8693</td>
<td>1.3974</td>
<td>1.9380</td>
<td>1.4211</td>
<td>1.6564</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>1.3247</td>
<td>1.2679</td>
<td>1.3056</td>
<td>1.4206</td>
<td>1.3297</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>1.1023</td>
<td>1.3235</td>
<td>1.0760</td>
<td>1.4677</td>
<td>1.2424</td>
</tr>
<tr>
<td>20</td>
<td>p = 0.2</td>
<td>2.4728</td>
<td>1.9809</td>
<td>2.4951</td>
<td>2.1671</td>
<td>2.2790</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>2.5763</td>
<td>1.7714</td>
<td>2.1929</td>
<td>1.9114</td>
<td>2.1130</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>1.1752</td>
<td>1.6927</td>
<td>1.1324</td>
<td>1.9372</td>
<td>1.4844</td>
</tr>
<tr>
<td>40</td>
<td>p = 0.2</td>
<td>3.0890</td>
<td>2.3339</td>
<td>3.1076</td>
<td>2.6251</td>
<td>2.7889</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>2.3201</td>
<td>1.5096</td>
<td>2.1269</td>
<td>1.7308</td>
<td>1.9218</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>1.7458</td>
<td>1.6706</td>
<td>1.7260</td>
<td>2.0036</td>
<td>1.7865</td>
</tr>
<tr>
<td>Overall</td>
<td>1.7561</td>
<td>1.5434</td>
<td>1.7107</td>
<td>1.6997</td>
<td>1.6775</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Ratio of Bottleneck Lower Bounds over the Resource-Free Lower Bounds: Mean

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p = 0.2</td>
<td>0.2683</td>
<td>0.2006</td>
<td>0.2920</td>
<td>0.2669</td>
<td>0.2586</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>0.1407</td>
<td>0.1367</td>
<td>0.1528</td>
<td>0.1620</td>
<td>0.1540</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>0.0659</td>
<td>0.1604</td>
<td>0.0603</td>
<td>0.1201</td>
<td>0.1421</td>
</tr>
<tr>
<td>8</td>
<td>p = 0.2</td>
<td>0.6513</td>
<td>0.3896</td>
<td>0.4841</td>
<td>0.3028</td>
<td>0.5306</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>0.1855</td>
<td>0.1588</td>
<td>0.2489</td>
<td>0.2195</td>
<td>0.2108</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>0.1260</td>
<td>0.1795</td>
<td>0.0852</td>
<td>0.2003</td>
<td>0.2227</td>
</tr>
<tr>
<td>20</td>
<td>p = 0.2</td>
<td>0.9080</td>
<td>0.6103</td>
<td>1.2974</td>
<td>0.8407</td>
<td>0.9584</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>0.8666</td>
<td>0.3356</td>
<td>0.3433</td>
<td>0.4675</td>
<td>0.6221</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>0.2332</td>
<td>0.2081</td>
<td>0.1829</td>
<td>0.2940</td>
<td>0.4135</td>
</tr>
<tr>
<td>40</td>
<td>p = 0.2</td>
<td>0.9629</td>
<td>0.6796</td>
<td>0.8848</td>
<td>0.7090</td>
<td>0.8702</td>
</tr>
<tr>
<td></td>
<td>p = 0.5</td>
<td>0.6937</td>
<td>0.2667</td>
<td>0.4748</td>
<td>0.3390</td>
<td>0.5654</td>
</tr>
<tr>
<td></td>
<td>p = 0.8</td>
<td>0.3284</td>
<td>0.3067</td>
<td>0.4405</td>
<td>0.4805</td>
<td>0.4110</td>
</tr>
<tr>
<td>Overall</td>
<td>0.5577</td>
<td>0.3622</td>
<td>0.6576</td>
<td>0.4295</td>
<td>0.6038</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Ratio of Bottleneck Lower Bounds over the Resource-Free Lower Bounds: Standard Deviation
References


