SOME PROBLEMS IN ONE-OPERATOR SCHEDULING

Md. Fazle Baki and R.G. Vickson
Department of Management Sciences, University of Waterloo, Canada

Abstract

In this paper, we discuss a class of problems that arises in an m-machine flow-shop operated by a single operator.

1 INTRODUCTION

Suppose that a single operator has to perform some jobs, each of which requires an operation on each of m machines. The operator can perform only one operation at a time. However, there are many possible sequences of operations. For example, suppose that there are two machines and three jobs. The operator may first do all jobs on machine 1 and, then, all jobs on machine 2. As another alternative, the operator may follow the sequence shown in Figure 1. Every node \( v = (i, j) \) represents operation of job \( j \) on machine \( i \) and every arc \( (v, v') \) represents the fact that the operation corresponding to node \( v \) is followed immediately by the operation corresponding to node \( v' \). Hence, the operator processes jobs 1 and 2 on machine 1, then job 1 on machine 2, then job 3 on machine 1 and, finally, jobs 2 and 3 on machine 2.

For every job we shall assume that its operation on machine \( i \) precedes its operation on machine \( (i + 1) \). For \( n \) jobs there exists \( m \times n \) operations and, since operations on machine \( i \) precede those on machine \( (i + 1) \), the \( m \times n \) operations can be carried out in \( (m \times n)! / (m!)^n \) sequences. Our problem is to decide which sequence to choose.

Essentially, our problem is an \( m \)-machine flow-shop problem in which no more than one machine can be run at any time. Such cases are important if both machines and operators are considered scarce. Even if we are interested in a case

\[ t_{ij} \geq 0. \]

for every job \( j \) a release date \( r_j \) and a due date \( d_j \) are also known and fixed. Associated with every job \( j \) is a priority factor \( w_j \), which is

\[ w_j > 0. \]

This is a natural assumption, and it is easy to see that in order to minimize makespan all jobs with \( w_j \) greater than a given \( k \) must be finished before all jobs with \( w_j \) less than or equal to \( k \). If it is required to minimize maximum lateness, there arise no such difficulties.

2 NOTATION AND DEFINITIONS

In every problem we assume that we are given jobs \( J_1, J_2, \ldots, J_n. \) We have to process every job on machines \( M_1, M_2, \ldots, M_m \) in stated order. An operation \( (i, j) \) refers to the processing of job \( j \) on machine \( i \). For every operation \( (i, j) \) the processing time is known and fixed and is given by \( t_{ij} \geq 0. \) For every job \( j \) a release date \( r_j \) and a due date \( d_j \) are also known and fixed. Associated with every job \( j \) is a priority factor \( w_j \), which is

\[ w_j > 0. \]
the importance weight of job $j$ relative to other jobs in the system. In the context of a single machine problem we shall simplify the notation and denote $t_{ij}$ as $t_j$.

For some problem we consider in this paper, there exists an optimal schedule which is called a batching schedule. A batching schedule is a schedule in which: (i) the same job-order, $\psi$ is adopted on all machines; (ii) there exist $k \geq 1$ distinct integers $1 \leq j_1 \leq \cdots \leq j_k = n$ such that on every machine $i$, jobs in batches $\{\psi(j_1), \ldots, \psi(j_i+1), \ldots, \psi(j_k)\}$ are processed contiguously; and (iii) every batch is started on machine $i$ immediately after its completion on machine $(i-1)$. We denote such a batching policy as $\mu = (j_1, \ldots, j_k)$. In Figure 2 we show an example of a batching schedule with two machines, five jobs, job-order $\psi = (1, 2, 3, 4, 5)$, and batching policy $\mu = (2, 4, 5)$. Observe that for every batch, nodes representing its operations are separated from the rest by a pair of vertical lines. Such lines will be called batch separating lines.

Following Graham et al. [3] we describe a scheduling problem by a triplet $\alpha|\beta|\gamma$.

2.1 The alpha field

The first field $\alpha = \alpha_1 \alpha_2$ contains the machine and operator environment. We have $\alpha_1 = \circ, F, 1$ respectively for single machine, flow-shop, and one-operator problem and $\alpha_2 = m$, the number of machines.

2.2 The beta field

The second field $\beta \subseteq \{\beta_1, \ldots, \beta_7\}$ contains processing characteristics and constraints.

$\beta_1 \in \{r_j, \circ\}$, $\beta_1 = r_j$ : Job $j$ is ready for processing at time $r_j$. $\beta_1 = \circ$ : All $r_j = 0$.

$\beta_2 \in \{S_k, \circ\}$, $\beta_2 = S_k$ : Every time an operator changes to machine $i$ from any other machine an $S_i$ units of time is required to set up machine $i$. $\beta_2 = \circ$ : All $S_i = 0$.

$\beta_3 \in \{job \text{ order} = \psi, \circ\}$, $\beta_3 = \text{job \text{ order}} = \psi$ : A known and fixed job-order $\psi$ is employed on all machines. $\beta_3 = \circ$ : Job-orders are not known.

$\beta_4 \in \{\text{batch} = \mu, \circ\}$, $\beta_4 = \text{batch} = \mu$ : Search for an optimal schedule is restricted to batching schedules with batching policy $\mu$. $\beta_4 = \circ$ : No such restriction is specified.

$\beta_5 \in \{l_{ij}, \circ\}$, $\beta_5 = l_{ij}$ : A period of time $l_{ij}$ must elapse between the completion of job $j$ on machine $i$ and the start of job $j$ on machine $(i+1)$. In the context of two machines we shall simplify the notation and denote $l_{ij}$ by $l_j$. $\beta_5 = \circ$ : All $l_{ij} = 0$.

$\beta_6 \in \{\text{perm}, \circ\}$, $\beta_6 = \text{perm}$ : A job cannot pass another while in queue. $\beta_6 = \circ$ : No such restriction is specified.

$\beta_7 \in \{\text{prec}, \circ\}$, $\beta_7 = \text{prec}$ : A precedence relation between jobs is specified. If job $j$ precedes job $j'$ then job $j$ has to be completed on machine $m$ before job $j'$ can start on machine $1$. $\beta_7 = \circ$ : No precedence relation is specified.

2.3 The gamma field

The Completion time $C_j$ of job $j$ is the epoch at which its last operation is finished. A regular objective $\Delta$ is a function that is non-decreasing in completion times [2, 6]. The completion time of the last job processed is called makespan and is denoted by $C_{\max}$. For job $j$, flow-time $F_j = C_j - r_j$ and lateness $L_j = C_j - d_j$. The third field $\gamma \in \{\Delta, C_{\max}, F_{\max}, L_{\max}, \sum C_j, \sum w_j C_j\}$ contains the objective to be minimized.

3 A PRELIMINARY DISCUSSION

Every one-operator problem can be transformed into a single machine problem by adding some precedence relations. Some one-operator problems reduce to single machine problems without the addition of precedence relations. One of our interests is to point out such cases.

For any $\beta$ and for any $\gamma$, preemptive versions of problems 1, $m|\beta|\gamma$ with processing times $t_{ij}$ and 1|$\beta|\gamma$ with processing times $t'_j = \sum t_{ij}$ are equivalent. Henceforth, we shall restrict our attention to non-preemptive cases.

The problem 1, $m|\text{prec}|\Delta$ with processing times $t_{ij}$ is equivalent to the problem 1|\text{prec}|\Delta with processing times $t'_j = \sum t_{ij}$. 

\begin{figure}[h]
\centering
\begin{tabular}{c|c|c|c|c}
M1 & 1.1 & 1.2 & 1.3 & 1.4 & 1.5 \\
M2 & 2.1 & 2.2 & 2.3 & 2.4 & 2.5 \\
\end{tabular}
\caption{A batching schedule.}
\end{figure}
Problem 1, \( m | r_j | \Delta, \Delta \in \{ C_{\text{max}}, F_{\text{max}} \} \) with processing times \( t_{ij} \) is equivalent to the problem \( 1|r_j|\Delta \) with processing times \( t'_j = \sum_i t_{ij} \).

This equivalence of problems 1, \( m | r_j | \Delta \) and \( 1|r_j|\Delta \) does not extend to \( \Delta \in \{ L_{\text{max}}, \sum C_j \} \). However, since the problem \( 1|r_j|\Delta \), \( \Delta \in \{ L_{\text{max}}, \sum C_j \} \) is strongly NP-hard [4] the problem 1, \( m | r_j | \Delta \) is strongly NP-hard for \( m \geq 1 \).

4 TIME LAGS

Since the problem \( 1 | r_j | \Delta, \Delta \in \{ L_{\text{max}}, \sum C_j \} \) is strongly NP-hard [4] the problem 1, \( m | l_{ij}, \text{perm} | \Delta \) and 1, \( m | l_{ij}, \text{perm} | \Delta \) are strongly NP-hard for \( m \geq 2 \) and the problem 1, \( m | l_{ij}, \text{perm} | C_{\text{max}} \) and 1, \( m | l_{ij}, \text{perm} | C_{\text{max}} \) are strongly NP-hard for \( m \geq 3 \).

In this section, we restrict our attention to the two-machine case with makespan objective.

**Theorem 1** (i) The problem 1, \( 2 | l_j | C_{\text{max}} \) can be solved by solving the corresponding problem \( F, 2 | l_j, \text{perm} | C_{\text{max}} \), and (ii) the problem 1, \( 2 | l_j, \text{perm} | C_{\text{max}} \) can be solved by solving the corresponding problem \( F2 | l_j, \text{perm} | C_{\text{max}} \).

The problem \( F, 2 | l_j, \text{perm} | C_{\text{max}} \) is solved [7] by Johnson's algorithm [2, 6] in \( O(n \log n) \) time. Hence, we get the following:

**Corollary 1** The problem 1, \( 2 | l_j, \text{perm} | C_{\text{max}} \) can be solved in \( O(n \log n) \) time.

The problem 1, \( 2 | l_j | C_{\text{max}} \) cannot be solved by solving the problem 1, \( 2 | l_j, \text{perm} | C_{\text{max}} \). In fact, the following holds:

**Theorem 2** The problem 1, \( 2 | l_j | C_{\text{max}} \) is NP-hard.

**Proof:** Consider an instance of the partition problem which is as follows: for set of integers \( A = \{ a_1, \ldots, a_k \} \), \( \sum a_i = 2b \) does there exist \( S \subset A \) such that \( \sum_{a_i \in S} a_i = b \)? Define an instance of the problem 1, \( 2 | l_j | C_{\text{max}} \) as follows:

\[
\begin{align*}
  n & = k + 2 \\
  t_{1j} = t_{2j} & = \begin{cases} 
    a_j & \text{for } j = 1, 2, \ldots, k \\
    b & \text{for } j = (k + 1), (k + 2) \\
    1b & \text{for } j = 1, 2, \ldots, k \\
    b & \text{for } j = (k + 1) \\
    13b & \text{for } j = (k + 2)
  \end{cases} \\
  l_j & = \begin{cases} 
    b & \text{for } j = (k + 1) \\
    13b & \text{for } j = (k + 2)
  \end{cases}
\end{align*}
\]

We shall show that the problem of checking whether there exists a schedule with \( C_{\text{max}} \leq 24b \) is equivalent to the partition problem. If for the partition problem there exists \( S \subset A \) such that \( \sum_{a_i \in S} a_i = b \), then for the problem 1, \( 2 | l_j | C_{\text{max}} \) we obtain \( C_{\text{max}} = 24b \) with an order \((1, k + 2), \{(1, j) : a_j \in S\}, (1, k + 1), \{(1, j) : a_j \in (A - S)\}\), \((2, k + 1), \{(2, j) : a_j \in S\}, (2, k + 2), \{(2, j) : a_j \in (A - S)\}\) of operations.

Now, suppose that for the problem 1, \( 2 | l_j | C_{\text{max}} \) there exists a schedule \( \sigma \) with \( C_{\text{max}} \leq 24b \). Then, the operator has no idle time and \( C_{\text{max}} = 24b \).

Without loss of generality, we can assume that all the operations on machine 1 are processed before any operation on machine 2 (if not, we can modify the given schedule so that the assumption holds and makespan does not increase). Operation \((1, k + 2)\) is processed before \((1, k + 1)\). Else, \( C_{\text{max}} \geq t_{1(k+1)} + t_{1(k+2)} + t_{(k+2)} = 28b > 24b \). Similarly, operation \((2, k + 2)\) is processed after \((2, k + 1)\). Now, define the following:

\( J_1 \): The set of jobs processed before \((1, k + 2)\)
\( J_2 \): The set of jobs processed on machine 1 after \((1, k + 1)\)
\( J_3 \): The set of jobs processed on machine 2 before \((2, k + 1)\)
\( J_4 \): The set of jobs processed after \((2, k + 2)\)

Observe that \( J_3 \subset J_1, J_2 \subset J_4 \), and \( J_2 \cap J_3 = \phi \). Now, if \( \sum_{j \in J_2} t_{1j} + \sum_{j \in J_3} t_{2j} < b \), then the operator will have some idle time and \( C_{\text{max}} > 24b \). If \( \sum_{j \in J_2} t_{1j} + \sum_{j \in J_3} t_{2j} > b \), then \( \sum_{j \in J_2} t_{1j} + \sum_{j \in J_4} t_{2j} > b \) and \( C_{\text{max}} \geq \sum_{j \in J_2} t_{1j} + t_{1(k+2)} + t_{2(k+2)} + \sum_{j \in J_4} t_{2j} > 24b \). Therefore, \( \sum_{j \in J_2} t_{1j} + \sum_{j \in J_3} t_{2j} = b \). Now, \( \{a_j : j \in J_2 \cup J_3\} \) is a subset of \( A \) which solves the partition problem.

5 SETUP TIMES

**Lemma 1** For the problem 1, \( 2 | S_j | \Delta \) there exists an optimal schedule which is a batching schedule.

Lemma 1 cannot be extended to problems \( 1, m | S_j | \Delta \) for \( m > 2 \). Henceforth, we shall restrict our attention to the two-machine case. In view of lemma 1, the problem 1, \( 2 | S_j | \Delta \) can be solved by two types of enumeration:

(i) **Enumeration over job-orders:** For every job-order \( \psi \), solve the problem 1, \( 2 | S_j, \text{jobord} = \psi | \Delta \). For \( n \) jobs there are \( n! \) job-orders.
(ii) **Enumeration over batching policies:** For every batching policy \( \mu \), solve the problem \( 1, 2 \mid |S_\mu| \text{ batch } = \mu \mid \Delta \). For \( n \) jobs there are \( 2^{n-1} \) batching policies.

For any regular objective the problem of arranging the jobs within a batch reduces to a single machine problem. Suppose that a batch comprises jobs 1, 2, ..., \( k \) with processing times \( t_i \). For any objective \( \Delta \) studied in this paper the problem of arranging the jobs 1, 2, ..., \( k \) within the batch is equivalent to the problem \( 1 \mid \Delta \) with processing times \( t_i' = t_{m,j} \).

6 **SETUP TIMES AND MAXIMUM LATENESS**

Throughout this section we consider the problem \( 1, 2 \mid |S_i| \text{ max} = L_{max} \). We develop a polynomial time algorithm for the problem. First, we observe that optimality of Jackson's Earliest Due Date (EDD) rule \( [2, 6] \) extends to the problem \( 1, 2 \mid |S_i| \text{ max} \). This follows by an interchange argument similar to that in French \( [2, \text{ Chap. 3}] \).

Henceforth, in this section, we shall assume that jobs are arranged according to the EDD rule. By relabelling if necessary, we can further assume that the EDD job-order is 1, 2, ..., \( n \). Now, we shall discuss a procedure for checking whether there exists a batching policy such that \( L_{\text{max}} \leq l \) for some given \( l \). To do this we begin by assigning job 1 to batch 1. Then, sequentially for every \( j > 1 \) we either (i) terminate the procedure with the conclusion that \( L_{\text{max}} > l \); (ii) add job \( j \) to the current batch; or (iii) add job \( j \) to a new batch.

If job 1 is the only job in batch 1 its lateness is \( L_1 = S_1 + S_2 + t_{11} + t_{21} - d_1 \). If \( L_1 > l \) the procedure terminates. Otherwise, if \( L_1 \leq l \) we let \( L_1' = L_1 \) be the current \( L_{\text{max}} \) value for batch 1. For \( j > 1 \) we suppose that \( L_{j-1}' \leq l \) is the \( L_{\text{max}} \) value of batch \( 1, \ldots, j - 1 \). If job \( j \) is added to the batch, all \( L_u \) for \( u < j - 1 \) are increased by \( t_{1j} \), so \( L_{j-1}' \) is increased by \( t_{1j} \). Thus, \( j_1 = j - 1 \) if \( L_{j-1}' + t_{1j} > l \). On the other hand, if \( L_{j-1}' + t_{1j} \leq l \) but \( L_j = S_1 + S_2 + \sum_{u=1}^{j-1} (t_{1u} + t_{2u}) - d_j \leq l \), the procedure terminates, because adding job \( j \) to batch 1 or to a new batch both give \( L_{\text{max}} \geq L_j > l \). Finally, if \( L_{j-1}' + t_{1j} \leq l \) and \( L_j \leq l \) we add job \( j \) to batch 1, set \( L_j' = \max(L_{j-1}' + t_{1j}, L_j) \) and then consider job \( (j+1) \) in the same manner.

The resulting first batch \( (1, \ldots, j_1) \) is maximal, in the sense that any first batch containing more than \( j_1 \) jobs would give \( L_{\text{max}} > l \). Furthermore, if there exists a schedule with \( L_{\text{max}} \leq l \) there exists such a schedule with all the jobs 1, ..., \( j_1 \), assigned to batch 1, because assigning fewer than \( j_1 \) jobs to batch 1 cannot decrease the completion times of jobs \( j > j_1 \).

We now suppose that at some stage of the procedure we have constructed \( k \geq 1 \) maximal batches \( (1, \ldots, j_1), (j_1 + 1, \ldots, j_2), \ldots, (j_k + 1, \ldots, j_k) \) for jobs 1, ..., \( j_k \). In the same manner used to obtain \( j_1 \) we determine the maximal \( (k + 1) \) batch \((j_{k+1}, \ldots, j_{k+1})\), or terminate with the conclusion \( L_{\text{max}} > l \). Note that the lateness values of jobs \( v = j_k + 1, \ldots, j \) are \( L_v = (k + 1) (S_1 + S_2) + \sum_{u=1}^{j_k} t_{1u} + \sum_{u=1}^{j_k} t_{2u} - d_k \), when jobs \( j_k + 1, \ldots, j \) are assigned to batch \((k + 1)\).

The whole procedure has at most \( n \) iterations where, in iteration \( j > 1 \) we decide whether the procedure terminates or whether job \( j \) is assigned to the current batch or a new batch. By updating only the \( L_{\text{max}} \) value \( L_{j-1}^* \) of the current batch we can perform each iteration in constant time, so the whole procedure in \( O(n) \) time.

Now, we can minimize \( L_{\text{max}} \) efficiently for integer valued input data \( t_{1j}, t_{2j}, d_j \) and \( S_1 + S_2 \). Let \( L_{\text{max}}^* \) be the minimum value of \( L_{\text{max}} \) over all schedules. We can identify bounds \( a \) and \( b \) such that \( a \leq L_{\text{max}}^* \leq b \). With the procedure discussed above we can perform a binary search for \( L_{\text{max}}^* \) in the interval \( [a, b] \). Hence, we get the following:

**Theorem 3** The problem \( 1, 2 \mid |S_i| L_{\text{max}} \) can be solved in \( O(n \log(b - a)) \) time.

7 **SETUP TIMES AND TOTAL WEIGHTED COMPLETION TIME**

7.1 **The problem \( 1, 2 \mid \sum w_j C_j \)**

We shall show that the problem \( 1, 2 \mid \sum w_j C_j \) is NP-hard. Consider a set of integers \( A = \{a_1, \ldots, a_k\} \), such that \( \sum a_i = 2b \). Define an instance of the problem \( 1, 2 \mid |S_i| \sum w_j C_j \) as follows:

\[
\begin{align*}
n & = k \\
\hat{S} & = S_1 + \hat{S}_2 \\
t_{1j} & = w_j \\
t_{2j} & = 0
\end{align*}
\]

for all jobs.
Let $x_l = \sum (a_j : \text{job } j \in \text{batch } l)$. Then, $\sum_{\text{job } j \in \text{batch } l} w_{lj} = x_l$ and completion time of every job $j \in \text{batch } l$, $C_j = lb + \sum_{i=1}^l x_p$. Hence, $\sum_{\text{job } j \in \text{batch } l} w_{lj} C_j = x_l(lb + \sum_{i=1}^l x_p)$ and if there are a total of $n$ baths then, $\sum w_{lj} C_j = \sum_{i=1}^n \sum_{i=1}^l x_p + b \sum_{i=1}^l (l x_i)$.

In particular, if $n = 1$ then, $\sum w_{lj} C_j = x_l^2 + bx_1$. But, $x_l = 2b$. Hence, $\sum w_{lj} C_j = 6b^2$. If $n = 2$ then, $\sum w_{lj} C_j = x_l^2 + x_2^2 + x_1 x_2 + bx_1 + 2bx_2$.

Now, using the fact that $x_1 + x_2 = 2b$, we get that $\sum w_{lj} C_j = x_l^2 - 3bx_1 + 8b^2$. Setting $d \sum w_{lj} C_j / dx = 0$, it follows that $\sum w_{lj} C_j$ attains a minimum value of $5.75b^2$ if $x_1 = 3b/2$ and $x_2 = b/2$. And for $n > 2$, we show in lemma 2 that the solution $x_1 = 3b/2, x_2 = b/2, x_1 = 0 \forall i > 2$ is the unique optimal with $\sum w_{lj} C_j = 5.75b^2$.

**Lemma 2** Let $f(x_1, x_2, ..., x_n) = \sum_{i=1}^n \sum_{i=1}^l x_p + b \sum_{i=1}^l (l x_i)$. Then, for any $n > 2$ the problem $\min f(x_1, x_2, ..., x_n) : 2b - \sum_{i=1}^n x_i = 0$ and $x_i \geq 0 \forall i \leq n$ has an unique optimal solution $x_1 = 3b/2, x_2 = b/2$, and $x_1 = 0 \forall i > 2$ with $f(x_1, x_2, ..., x_n) = 5.75b^2$.

**Proof:** The hessian matrix $H$ of $f(x_1, x_2, ..., x_n)$ is as follows: $H_{ii} = 2 \forall i$ and $H_{ij} = 1 \forall i \neq j$. Applying the algorithm of Murty [5, pp. 521-523] for checking positive definiteness we obtain an upper triangular matrix $H'$ which is as follows: $H'_{ii} = 1 + 1/i \forall i, H'_{ij} = 1/i \forall i < j$ and $H'_{ij} = 0 \forall i > j$. This shows that $H$ is positive definite and, therefore, our problem is a convex programming problem. Introducing the Lagrange multipliers $\pi_0$ to $\pi_4$ we get the KKT conditions which are as follows:

$$2x_l + \sum_{j \neq i} x_j + ib + \pi_0 - \pi_i \geq 0 \forall 1 \leq i \leq n$$

$$\pi_i x_i \geq 0 \forall 1 \leq i \leq n$$

$$\pi_i x_i = 0 \forall 1 \leq i \leq n$$

Observe that $x_l = 3b/2, x_2 = b/2, x_1 = 0 \forall 2 \leq i \leq n, \pi_0 = 9b/2, \pi_1 = \pi_2 = 0$, and $\pi_i = (i - 5/2)b \forall 3 \leq i \leq n$ satisfy the KKT conditions and, therefore, is the unique optimal solution and the optimal value of $f(x_1, x_2, ..., x_n)$ is $5.75b^2$.

Therefore, the problem of checking whether there exists a schedule with $\sum w_{lj} C_j \leq 5.75b^2$ is equivalent to checking whether there exists $S \in A$ such that $\sum_{a_i \in S} a_i = 3b/2$.

**Theorem 4** The problem $1, 2|S_i, jobord \rightarrow \psi | \sum w_{lj} C_j$ is NP-hard.

**7.2 The problem**

1, 2|\(S_i, j) = \psi | \sum w_{lj} C_j$

By relabelling if necessary, we can assume that the job-order is $1, 2, ..., n$. Consider any batching policy $\mu = (j_1, j_2, ..., j_k)$. Let $S = S_1 + S_2$. For contribution $\eta(j', j) = \{\sum_{j=j'}^{j_k} (t_{ij} + t_{2j}) + S\} \sum_{i=j+1}^{i'} w_i + (\sum_{j=j'}^{j_k} t_{1i} + S) \sum_{i=j'}^{i} w_i + (\sum_{j=j'}^{j_k} (w_i \sum_{i'=j'}^{i} t_{1i'}) t_{1i'}) of batch $(j', j)$, we can obtain $\sum w_{lj} C_j$ by summing up the contributions of all batches.

Define a directed graph $G = (V, E)$ as follows: for every position $0, 1, 2, ..., n$ where we can draw batch separating lines (see Figure 2) define a node. For every $1 \leq v \leq v' \leq n$ define an arc $(v - 1, v')$. Then, every arc $(v, v')$ corresponds to a batch $(v + 1, v')$. Assign weight $w_{v, v'} = \eta(v + 1, v')$ to every arc $(v, v')$. The problem $1, 2|S_i, j) = \psi | \sum w_{lj} C_j$ is equivalent to that of finding a shortest path from $0$ to $n$ in network $(G, r)$.

An example is shown in Figure 3. The shortest path $(0, 2, 3)$ is shown with thick arcs in Figure 3(b). Hence, the batching schedule shown in Figure 3(c) and comprising batches $(1, 2)$ and $(3, 3)$ is optimal.

We have already outlined a polynomial time procedure for the problem $1, 2|S_i, j) = \psi | \sum w_{lj} C_j$. We can improve the running time. First, we observe that it suffices to consider network $(G, r')$ where $r_{v, v'}' = (S_1 + S_2) \sum_{k=v'+1}^{n} w_k + \sum_{k=v'+1}^{n} w_k \sum_{k=v'+1}^{n} t_{1j}$. Now, in $O(n^3)$ time we can construct the network and solve the shortest path problem.

Again, $r_{v, v'+1}' - r_{v, v'}' = t_{1j}(v+1) \sum_{k=v'+1}^{n} w_k - S w_{v, v'+1}$. If $w_j = 1$, then $r_{v, v'+1}'$ can be computed from $r_{v, v'+1}$ in constant time. Also, $r_{v, v'+1}'$
Hence, if \( w_j = 1 \) for all \( j \), then the matrix searching algorithm of Aggarwal et al. [1] applies, and the shortest path problem in network \((G, r')\) can be solved in \(O(n)\) time.

**Theorem 5** The problem \( 1, 2|S_i, joborder = \psi|\sum w_jC_j \) can be solved in \(O(n^2)\) time. The special case with \( w_j = 1 \), can be solved in \(O(n)\) time.

### 7.3 The problem \( 1, 2|S_i, \text{batch} = \mu|\sum w_jC_j \)

First, consider the case with \( w_j = 1 \) for all \( j \).

Consider any batching policy \( \mu = (j_1, j_2, ..., j_k) \) and job-order \( \psi \). For every job \( j \) let \( C_j' \) and \( \bar{C}_j \) be respectively setup time and processing time spent before the completion of job \( j \). Note that \( \sum C_j \) is minimized by minimizing \( \sum \bar{C}_j \).

Let \( j_0 = 0 \). For \( j_{u-1} < j' \leq j_u \) define \( \eta_{jj'} = (n - j_{u-1})t_{ij} + (n - j' + 1)t_{ij'} \). Observe that \( \sum \bar{C}_k = \sum \{\eta_{jj'}: \text{job } j \text{ is assigned to position } j'\} \). Hence, we get the following:

**Theorem 6** The problem \( 1, 2|S_i, \text{batch} = \mu|\sum w_jC_j \) reduces to an assignment problem.

However, the problem \( 1, 2|S_i, \text{batch} = \mu|\sum w_jC_j \) is NP-hard. Consider the set of integers \( A = \{a_1, ..., a_k\} \), such that \( \sum a_i = 2b \). Define an instance of the problem \( 1, 2|S_i, \text{batch} = \mu|\sum w_jC_j \) as follows:

\[
\begin{align*}
    n & = 2k - 2 \\
    \mu & = (k - 1, 2k - 2) \\
    t_{ij} & = \begin{cases} 
        a_j & \text{for } j \leq k \\
        0 & \text{for } j > k 
    \end{cases} \\
    t_{ij'} & = 0 
\end{align*}
\]

Consider any job-order \( \psi \). Let \( J_1 \) and \( J_2 \) be the set of jobs in batches 1 and 2 respectively. Let \( x = \sum_{j \in J_1} t_{ij} = \sum_{j \in J_1} w_j \) and \( \bar{S} = S_1 + S_2 \). Observe that \( C_j = S + x \) if \( j \in J_1 \) and \( C_j = 2(S + b) \) if \( j \in J_2 \). Therefore, \( \sum w_jC_j = x^2 - (S + 2b)x + 4b(S + b) \). Then, \( \sum w_jC_j \) attains a minimum value of \( 4b(S + b) - \left(\frac{S}{2} + b\right)^2 \) if \( x = \frac{S}{2} + b \). Hence, for \( S = 2ab, 0 \leq a < 1 \) the problem of checking whether there exists a job-order with \( \sum w_jC_j \leq 4b\left(S + b\right) - \left(\frac{S}{2} + b\right)^2 = \left(3 + 6a - a^2\right)b^2 \) is equivalent to the problem of checking whether there exists \( S \in A \) such that \( \sum_{a \in S} a_t = b(1 + a) \).

**Theorem 7** The problem \( 1, 2|S_i, \text{batch} = \mu|\sum w_jC_j \) is NP-hard.

![Figure 4](image).

**Figure 4:** (a) Processing times and setup times; (b) the node-packing problem with optimum set of nodes shaded; and (c) optimum order of operations.

### 7.4 The problem \( 1, 2|S_i| \sum C_j \)

We can formulate the problem \( 1, 2|S_i| \sum C_j \) (with both the job-order and batching policy unknown) as follows: A job \( j \) must occupy one of the positions \( j' = 1, 2, ..., n \). If job \( j \) is assigned to position \( j' \), it may be the \( k \)-th job in its batch, where \( k = 1, 2, ..., j' \). Let \( a_{j'k} \) be the contribution to \( \sum C_j \) of having job \( j \) in position \( j' \) as the \( k \)-th job in its batch such that \( \sum C_j = \sum a_{j'k} \). In \( \sum C_j \) \( t_{ij} \) appears \( (n - j' + k) \) times, and \( t_{ij'} \) appears \( (n - j' + 1) \) times. Furthermore, if \( k = 1 \) (i.e., \( j \) is the first job in its group) an additional setup time \( S_1 \) is spent on machine 1 and \( S_2 \) on machine 2. We thus have \( a_{j'1} = (n - j' + 1)(t_{ij} + t_{ij'} + S_1 + S_2) \), and \( a_{j'k} = (n - j' + k) t_{ij} + (n - j' + 1) t_{ij'} \) for \( 1 < k \leq j' \).

Now, the problem can be formulated as a maximum weight node-packing problem in a graph \( G = (V, E_1 \cup E_2 \cup E_3) \), where \( V = \{v_{j'k_j}: 1 \leq j' \leq n, 1 \leq k \leq j', n\} \), \( E_1 = \{(v_{j'k_j}, v_{j'k_j+1})\} \), \( E_2 = \{(v_{j'k_j+1}, v_{j'k_j+2})\} \), \( E_3 = \{(v_{j'k_j+2}, v_{j'+1})\} \) : \( k_2 > l \geq 1, k_2 \neq k_1 + l \) and every node \( v_{j'k_j} \) is assigned a weight \( w_{j'k_j} = M - a_{j'k_j} \) for some \( M \geq \max_{j'k_j} \{a_{j'k_j}\} \).

An example is shown in Figure 4. An optimal solution to the node-packing problem is the set of nodes shaded in Figure 4(b). Hence, an optimal schedule is the one shown in Figure 4(c).

The above formulation can be slightly improved. For example, it suffices to take \( E_3 = \{(v_{j'k_1+2}, v_{j'+1})\} \) \( k_2 \neq 1, k_2 \neq k_1 + 1 \) instead of \( E_3 \).

Let \( x_{j'k_j} = 1 \), if job \( j \) is assigned in position \( j' \) as \( k \)-th job in its group and \( x_{j'k_j} \)
= 0, otherwise. Then, the integer programming formulation of the node packing problem is (P1) \[ \min \sum_{j} \sum_{k < j} \sum_{i} x_{j,i,k}^a \text{subject to} \]
\[ x_{j,k,i}^a + x_{j,k,i}^b \leq 1 \quad \forall (v_{j,k,i}^a, v_{j,k,i}^b) \in E_1 \cup E_2 \]
\[ x_{j,k,i}^a \geq 0 \quad \forall j, k, i. \]
However, we further modify the integer programming formulation. (P2) \[ \min \sum_{j} \sum_{k < j} \sum_{i} x_{j,k,i}^a \text{subject to} \]
\[ \sum_{j} \sum_{k < j} x_{j,k,i}^a = 1 \quad \forall j, \sum_{k < j} \sum_{i} x_{j,k,i}^a = 1 \]
\[ \forall j', \sum_{j} \sum_{k < j} x_{j,k,i}^a - \sum_{j} \sum_{k < j} x_{j',k,i}^a \geq 0 \quad \forall k \leq j' < n, \sum_{j} x_{j,k,i}^a \geq 0 \quad \forall k \leq j' < n, \]
\[ x_{j,k,i}^a \in \{0, 1\} \forall j', k, i. \]

The formulation (P2) has fewer constraints. But, the linear programming relaxation of (P2) gives better bounds. We implement both formulations in GAMS. Formulation (P2) is more efficient both in terms of computational time and memory. We solve problems up to \( n = 30 \) with formulation (P2).

8 CONCLUSION

In this paper we consider various cases of the one-operator scheduling problem. We summarize the major results in the following:

1, 2|\( j \)| \( \text{perm} \) \( C_{\max} \) : Solved in \( O(n \log n) \) time.
1, 2|\( j \)| \( L_{\max} \) : \( \text{NP-hard} \).
1, 2|\( S_i \)| \( \Delta \) : A batching schedule is optimal.
1, 2|\( S_i \)| \( L_{\max} \) : Solved in \( O(n \log(b-a)) \) time, where \( a \leq L_{\max} \leq b \).
1, 2|\( S_i \)| \( \sum w_j C_j \) : \( \text{NP-hard} \).
1, 2|\( S_n \)| \( \text{jobord} \) = \( \psi \) | \( \sum w_j C_j \) : Solved in \( O(n^2) \) time in general and in \( O(n) \) time if all \( w_j = 1 \).
1, 2|\( S_n \)| \( \text{batch} \) = \( \mu \) | \( \sum w_j C_j \) : \( \text{NP-hard} \); reduces to an assignment problem if all \( w_j = 1 \).

The complexity status of the problem 1, 2|\( S_i \)| \( \sum w_j C_j \) remains open.

References


