Scheduling in Flowshops with Flexible Operations: Throughput Optimization and Benefits of Flexibility

Hakan Gultekin*

Department of Industrial Engineering, TOBB University of Economics and Technology,
Sogutozu Cad. No:43, 06560, Ankara, Turkey

Abstract

This study considers the throughput optimization in a two-machine flowshop producing identical jobs. Unlike the general trend in the scheduling literature, the machines are assumed to be capable of performing different operations. As a consequence, one of the three operations that a job requires can only be processed by the first and another operation can only be processed by the second machine. These are called fixed operations. The remaining one is called the flexible operation and can be processed by any one of the machines. The machines are assumed to have different technological properties, i.e. nonidentical, so that the processing time of the flexible operation has different values on the two machines. We first consider the problem of assigning the flexible operations to the machines for each job in order to maximize the throughput rate. We develop constant time solution algorithms for infinite and zero capacity buffer spaces in between the machines. We then analyze the benefits of flexibility. Managerial insights are provided regarding the changes in the makespan as well as the associated cost with respect to the increase in the level of flexibility.

Keywords: Flexible manufacturing systems, flowshop scheduling, makespan, operation allocation.

*e-mail: hgultekin@etu.edu.tr, Tel:+90 312 292 4275, fax +90 312 292 4091
1 Introduction

In order to be successful in today’s highly competitive world requiring high levels of productivity and adaptability to changes, firms increase levels of flexibility in their manufacturing systems. Flexibility in manufacturing is defined as the ability to change or react with little penalty in time, effort, cost or performance ([20]). There are many different types of manufacturing flexibilities such as the machine flexibility defined as the ability of the machines to perform different operations and the operation flexibility defined as the ability to produce a product in different ways ([4]). In order to get the maximum available benefit from flexibility some important problems must be tackled such as the determination of the “optimal” levels of flexibility and the determination of operational rules (e.g. schedules) for such systems. This study considers a flowshop consisting of two workstations which possess machine flexibility. Such situations arise in many different practical settings. For example, if the workstations consist of Computer Numerical Control (CNC) machines which can perform different operations as long as the necessary cutting tools are loaded on their tool magazines or if the workstations consist of manual operators equipped with the necessary tooling and crosstrained to perform different operations. From now on we will refer to the workstations as machines.

We assume identical jobs requiring three operations are to be processed on these machines. The first operation can only be processed by the first machine and the third operation can only be processed on the second machine. These operations are called fixed operations. On the other hand, both of the machines are capable of performing the second operation, which is called the flexible operation. Let $f_1$ denote the processing time of the fixed operation on machine 1 and $f_2$ denote the processing time of the fixed operation on machine 2. The machines are assumed to have different technological properties (nonidentical machines) so that the processing time of the flexible operations take different values depending on the machine it is processed. We denote the processing time of the flexible operations on the first machine by $s_1$ and on the second machine by $s_2$. If the flexible operation is assigned to the first machine for a job, then the total processing time of this job on the first machine is $f_1 + s_1$ and on the second machine is $f_2$. If it is assigned to the second machine then the processing times are $f_1$ and $f_2 + s_2$, respectively. The problem is to determine the assignment of the flexible operations of each job to the machines that maximizes the overall throughput rate (or equivalently minimizes the makespan). Another problem considered in this study is determination of the optimal level of flexibility. Here the decision involves the maximization of the throughput rate while taking the cost of increasing the level of flexibility into account.
There are certain cases in industrial practice that directly correspond to the design mentioned above. For example, the CNC machines are capable of performing different operations provided that the required cutting tools are loaded on their tool magazines. However, having multiple copies of each tool and loading to the magazines of the machines may not be physically possible and/or economically feasible. This is because the total number of tools required to process different operations involved in a job is usually larger than the available magazine storage capacity ([10]). Further, especially the tools used in metal cutting industry may be very expensive. Therefore, because of such tradeoffs some tools may have multiple copies, while some others have only a single copy. Here, operations requiring tools which have multiple copies may be treated as flexible, whereas the others as fixed operations ([11]). A similar discussion also applies to manual operations. It may be possible to train operators to perform some of the necessary operations that a job requires, but training for all the necessary operators may not be possible ([9]). Another example is automotive manufacturing where spot welding robots are used extensively. Due to part geometry, some welding operations can only be performed by special welding guns loaded on specific robots, whereas some other operations can be performed by different gun types. In the assembly of printed circuit boards (PCB), presence or absence of feeder tapes that hold the electronic components to be inserted on machines may lead to a similar circumstance ([7]).

There is an extensive literature on manufacturing flexibility which can be reviewed from the survey of Beach et al. [3]. Daniels and Mazzola ([8]) consider a flowshop with manual operators. The operators are cross-trained to perform all the necessary operations that the jobs require. This situation is called complete flexibility. In another study, Daniels et al. [9] consider a flowshop with partial resource flexibility meaning that the operators are trained to perform a subset of all the operations that a job requires. They assume that the assignment of the operators can be changed dynamically from one station to another and the process times at the stations are functions of the number of operators working on that station. Gultekin et al. [11] make a similar assumption in a robotic cell consisting of two CNC machines and producing identical parts whereas Batur et al. [2] considers the same problem with multiple parts. They aim to determine the assignment of flexible operations to the machines and the robot move cycle that maximize the throughput rate. Anuar and Bukchin [1] consider a flowshop where the assignment of the tasks can be changed dynamically. They aim to balance the line in the long term and investigate the performance of a number of operating rules. Burdett and Kozan [5] consider an m-machine preemptive flowshop producing multiple jobs. The tasks are assumed to be shifted to adjacent stations. Mathematical formulation of the problem and
heuristic procedures are provided. They show that considerable benefits can be obtained by applying task redistribution methodology for a wide range of problem instances, different flowshop types, and task shifting scenarios.

Daniels et al. [9] prove that a large portion of the available benefits associated with labor flexibility can be realized with a relatively small investment in crosstraining. Their results suggest that in order to obtain high-quality solutions, scheduling and resource assignment decisions must be coordinated. Similar conclusions are also made by Jordan and Graves [15], who consider the process flexibility and Nomden and van der Zee [17], who consider routing flexibility. Process flexibility is defined as being able to manufacture different types of products in the same production facility at the same time. Routing flexibility, on the other hand, is defined as the ability to produce a product using different routes. Assuming that the cost of increasing the level of flexibility is directly proportional with the level of flexibility, a low level of flexibility where most of the performance improvements are achieved is the best decision. However, this assumption is not necessarily true under the settings of the current study as shown in Example 2 in Section 4.

In a closely related study Gupta et al. [13] consider a two-machine flowshop with infinite buffer capacity in between the machines that produce multiple jobs having fixed and flexible operations. They prove that the problem is NP-hard, suggest heuristic algorithms and develop a polynomial time approximation scheme (PTAS). Another closely related study is by Crama and Gultekin [6] who consider the problem with identical jobs and identical machines (i.e. $s^1 = s^2 = s$). They develop polynomial or polynomial pointwise algorithms for different assumptions regarding the buffer capacity in between the machines. However, as shown in Example 1 in Section 3.1, their results are not applicable to the non-identical machines case which is a more realistic production setting.

Main contributions of the current study are the development of constant time algorithms for the non-identical machines problem with no-buffer and infinite capacity buffer assumptions and demonstration of the benefits of machine flexibility. Furthermore, the managerial insights towards deciding on the optimal level of flexibility can also be stated as a major contribution of this study. In the next section we define our problem in detail. In Section 3 we develop solution procedures for the no-buffer and infinite capacity buffer cases. Section 4 considers the benefits of flexibility and provides managerial insights. Section 5 is devoted to concluding remarks and future research directions.
2 Problem definition and preliminary results

In this section we define the problem more formally. There are \( n \) identical jobs to be processed on two machines. Let \( N = \{1, 2, \ldots, n\} \) denote the set of jobs to be processed. All jobs are first processed by machine 1 and then by machine 2. The buffer space between the machines is denoted by \( B \). In this study we consider both the no-buffer case \((B = 0)\) and the infinite capacity buffer case \((B \to \infty)\).

In infinite buffer capacity systems, it is assumed that there is always space for an additional part in the buffer between the first and the second machines. Therefore, after the first machine completes processing a part, this part can be placed to the buffer and the machine can start processing the next part immediately. On the other hand, in no-buffer systems, after the processing of a part is completed on the first machine if the second machine is still busy processing another part, first machine cannot be unloaded. As a consequence, this machine cannot start processing the next part. Preemption is not allowed, which means, if one job has started its operation on any machine it must be completed before it leaves the machine. Additionally, each machine can process one job at a time and a job can only be processed by one machine at any time. Each of the identical jobs consists of three operations: The first operation is processed by machine 1 and the third operation is processed by machine 2. The second operation can either be processed by machine 1 or 2. The problem is to determine the assignment of the flexible operations to the machines for each job. These assignments can differ from one job to another. The objective is to minimize the completion time of the last job in the sequence on the second machine that is, the makespan.

We use the following notation and decision variables throughout the text:

- \( p^j_i \): Total processing time of job \( i \in N \) on machine \( j = 1, 2 \). Depending on the assignment of the flexible operation, \( p^j_i \) can either be equal to \( f^j \) or \( f^j + s^j \),
- \( T^j_i \): Starting time of processing of job \( i \in N \) on machine \( j = 1, 2 \),
- \( C^j_i \): Completion time of job \( i \in N \) on machine \( j = 1, 2 \). We have \( C^j_i = T^j_i + p^j_i \),
- \( x^j_i = 1 \) if the flexible operation of job \( i \in N \) is assigned to machine \( j = 1, 2 \) and 0, otherwise.

The Mixed Integer Program (MIP) of the general problem can be formulated as follows for which exact solution algorithms for \( B = 0 \) and \( B \to \infty \) cases are developed in Section 3:

\[
\text{Minimize } \sum_{i=1}^{n} T^2_i + f^2 + s^2 x^2_i
\]  

(1)
In this formulation, the objective function computes the completion time of the $n^{th}$ job on the second machine ($C_{\text{max}}$). Constraints (2) and (3) satisfy that the processing of a job can start if the processing of the previous job is completed on the same machine. Constraint (4) states that the processing of job $i$ on the second machine can start if the processing of this job is completed on the first machine. Similarly, the processing of job $i$ on the first machine can start if job $(i - 1)$ can be transferred to the second machine or to the buffer. Constraint (6) guarantees that the flexible operation for job $i$ is either assigned to the first machine or to the second machine. Constraint (7) states that the schedule starts at time 0. For the no-buffer problem we set $B = 0$ in the formulation, whereas for the infinite capacity buffer case we remove Constraint (5) from the formulation.

In the remainder of this section some basic results will be presented about the problem. The reversibility property which is proved by Muth [16] is still valid for the current study. Given the original problem with parameters $f^1_i, f^2_i, s^1_i, \text{and } s^2_i$, the reversed problem is attained by changing the processing times between the machines as follows:

$$\hat{f}^1_i = f^2_i, \quad \hat{f}^2_i = f^1_i, \quad \hat{s}^1_i = s^2_i, \quad \hat{s}^2_i = s^1_i.$$ 

In the reversed problem, the jobs are first processed by the second machine then by the first one. The importance of this definition comes from the following lemma.

**Lemma 1.** The optimal makespan value of the original problem is identical to the optimal makespan value of the reversed problem.

The proof is a straightforward extension of the proof by Muth [16] for the classical flowshop problem and hence omitted here. This property will be helpful in simplifying the proofs that we make later in this text. Given the optimal schedule for either the original or the reversed problems, the other one can be generated easily.
The following is also an important property used in determining the exact solution procedure:

**Property 1.** There exists an optimal schedule for the problem in which the processing on a machine starts as soon as the machine and the job are both ready.

This property states that there exists an optimal active schedule to the problem, which is well known in scheduling literature and usually valid for regular objective functions (non-decreasing in job completion times) such as the makespan.

In the next section we determine exact and efficient solution procedures for $B = 0$ and $B \to \infty$ cases. In Section 4, we analyze the advantages and disadvantages of such flexibility, demonstrate the increase in the throughput rate, and provide managerial insights. Section 5 is devoted to concluding remarks and future research directions.

### 3 Throughput maximization

In this section, we develop exact solution procedures for $B = 0$ and $B \to \infty$ cases separately.

#### 3.1 No-buffer Case

Such systems are usually faced in automotive and electronics industries. When there are no available buffer space in between the machines, in an optimal schedule both machines can have idle time. The second machine can be idle if the first machine has not completed the processing of the job yet (starving); the first machine can be idle if the second machine is still processing the previous job when the first machine has completed processing the next job in the sequence (blocking). However, both machines can not be idle at the same time in an optimal schedule. The following property is a consequence of these observations and will be helpful in the remainder of the text.

**Property 2.** There exists an optimal schedule in which starting time of job $i$ on the first machine is equal to starting time of job $(i - 1)$ on the second machine, $T^1_i = T^2_{i-1}, i = 2, 3, \ldots, n$.

Crama and Gultekin [6] considered the case where the machines are identical ($s^1 = s^2 = s$) and proved that the flexible operation of the first job is always assigned to the second machine and the flexible operation of the last job is always assigned to the first machine. This result is used extensively to develop solution procedures in that study. However, the following example shows that this result cannot be extended for two non-identical machines case.
**Example 1.** Let \( n = 5, f^1 = 10, f^2 = 20, s^1 = 15, \) and \( s^2 = 22 \). Earlier results for the identical machines case state that the flexible operation of the first job is assigned to the second machine in the optimal solution. With this assignment, the best makespan value for the current problem is 147 units which is depicted in Figure 1b. However, for given parameters the flexible operation of the first job is assigned to the first machine in the optimal solution. The corresponding makespan is 145 units and the Gantt chart is depicted in Figure 1a. Therefore, we can conclude that earlier results cannot be extended for the current study.

As a result of Property 1 and Property 2, a feasible solution can be divided into \((n + 1)\) distinct intervals as shown in Figure 2. The \(i^{th}\) interval starts with the starting time of the \(i^{th}\) job on the first machine and ends with the starting time of the same job on the second machine for \(i = 1, 2, \ldots, n\). Therefore, the \(i^{th}\) interval is represented as \([T^1_i, T^2_i]\), \(i = 1, 2, \ldots, n\). On the other hand, the \((n + 1)^{st}\) interval is represented as \([T^2_n, C^2_n]\). Let \(A = \{I_1, I_2, \ldots, I_n, I_{n+1}\}\) be the set of time intervals and \(\alpha_i\) be the length of the corresponding interval. That is,

\[
\alpha_i = T^2_i - T^1_i, \quad i = 1, 2, \ldots, n,
\]

\[
\alpha_{n+1} = C^2_n - T^2_n.
\]

In a feasible solution depending on the allocation of the flexible operations, within any interval the processing times on the two machines can take only four different values except the first and the last intervals. We will name these four occurrences as patterns. Let \(\beta_h = (P^1, P^2)\) denote the pattern
and \( L(\beta_h) = \max\{P^1, P^2\} \) denote the length of the corresponding pattern for \( h = 1, \ldots, 4 \). The possible patterns are as follows:

\[
\begin{align*}
\beta_1 &= (f^1 + s^1, f^2), \\
\beta_2 &= (f^1 + s^1 + f^2 + s^2), \\
\beta_3 &= (f^1, f^2 + s^2), \\
\beta_4 &= (f^1, f^2).
\end{align*}
\]

Let \( G^k_1 = T^2_k - T^1_{k-2}, k \geq 3 \). From Figure 2, it can be seen that \( G^k_1 = \alpha_{(k-2)} + \alpha_{(k-1)} + \alpha_k \).

The following lemma proves that in an optimal solution to the problem, either all of the intervals \( I_3, I_4, \ldots, I_n \) have pattern \( \beta_1 \), or this pattern is never used.

**Lemma 2.** Given an optimal schedule, if there exist at least one \( k, 3 \leq k \leq n \), such that \( I_k \) has pattern \( \beta_1 \), then \( I_i \) also has pattern \( \beta_1 \) for all \( i = 3, 4, \ldots, n \).

**Proof.** In a feasible schedule, changing the assignment of the flexible operation of job \( k \in \{1, 2, \ldots, n\} \) does not have an effect on the lengths of the intervals excluding \( I_k \) and \( I_{k+1} \). We will make use of this inference throughout the proof. Let us assume we have an optimal schedule in which \( I_k \) has pattern \( \beta_1 \) for some \( k, 3 \leq k \leq n \). Hence, \( \alpha_k = L(\beta_1) \) and the flexible operation of job \( (k-1) \) is assigned to the first machine. As a result, \( I_{k-1} \) has either pattern \( \beta_1 \) or \( \beta_2 \). These two cases are considered separately.

1. If \( I_{k-1} \) has pattern \( \beta_1 \), then \( G^k_1 = \alpha_{k-2} + L(\beta_1) + L(\beta_1) \). Now let us switch the assignment of the flexible operation of \( (k-1) \) so that it is assigned to the second machine. Hence, \( I_k \) does not have pattern \( \beta_1 \) any more. Let this case be denoted by \( \hat{G}^k_1 \). Hence, \( \hat{G}^k_1 = \alpha_{k-2} + L(\beta_4) + L(\beta_2) \).

Since the original schedule is assumed to be optimal, \( G^k_1 \leq \hat{G}^k_1 \) must hold. As a result we have the following:

\[
2L(\beta_1) \leq L(\beta_4) + L(\beta_2). \tag{9}
\]

2. If \( I_{k-1} \) has pattern \( \beta_2 \), then \( G^k_1 = \alpha_{k-2} + L(\beta_2) + L(\beta_1) \). Following a similar series of arguments as before, we compare this value with the one where \( I_k \) does not have pattern \( \beta_1 \), \( \hat{G}^k_1 = \alpha_{k-2} +
\[ L(\beta_3) + L(\beta_2). \]

We must again have \( G^k_i \leq \hat{G}^k_i \) which leads to the following:
\[ L(\beta_1) \leq L(\beta_3). \] (10)

Equations (9) and (10) state the two conditions that must hold if there exist an optimal schedule in which job \( k \) has pattern \( \beta_1 \) for at least one index \( k, 3 \leq k \leq n \). In the following we prove that if \( I_k \) has pattern \( \beta_1 \), then \( I_i \) also has pattern \( \beta_1 \) for different values of \( k \) and \( i \):

**Case 1.** \( 4 \leq k \leq n \) and \( 3 \leq i \leq k - 1 \):

Assume we have an optimal schedule in which \( I_k, 4 \leq k \leq n \) has pattern \( \beta_1 \), but in contradiction with the above statement, \( I_{k-1} \) does not have this pattern. Unless all intervals \( I_i \) for \( 3 \leq i \leq (n - 1) \) have pattern \( \beta_1 \), there will be at least one such occurrence and \( k \) can be selected accordingly. Then, depending on the assignments of the flexible operations, the only feasible alternative for \( I_{k-1} \) is to have pattern \( \beta_2 \). This can only occur when the flexible operation of job \( (k - 2) \) is assigned to the second machine, \( p_{k-2} = f^1 \). Let us consider \( G^k_2 = T^2_k - T^1_{k-3} = \alpha_{k-3} + \alpha_{k-2} + \alpha_{k-1} + \alpha_k, k \geq 4 \).

Depending on the assignment of the flexible operation of job \( (k - 3) \), we have the following cases:

1. If \( I_{k-2} \) has pattern \( \beta_3 \), then we have \( G^k_2 = \alpha_{k-3} + L(\beta_3) + L(\beta_2) + L(\beta_1) \). Let us now switch the assignment of the flexible operation of job \( (k - 2) \) from the second machine to the first so that the pattern of \( I_{k-1} \) changes to \( \beta_1 \). As a result of this change, we have \( \hat{G}^k_2 = \alpha_{k-3} + L(\beta_2) + L(\beta_1) + L(\beta_1) \). From (10), \( L(\beta_1) \leq L(\beta_3) \). Therefore, \( \hat{G}^k_2 = \alpha_{k-3} + L(\beta_2) + L(\beta_1) + L(\beta_1) \leq \alpha_{k-3} + L(\beta_2) + L(\beta_3) + L(\beta_1) = G^k_2 \). Consequently, under the stated condition, if \( I_k \) has pattern \( \beta_1 \), then having the same pattern for \( I_{k-1} \) does not increase the makespan.

2. If \( I_{k-2} \) has pattern \( \beta_4 \), then we have \( G^k_2 = \alpha_{k-3} + L(\beta_4) + L(\beta_2) + L(\beta_1) \). Switching the assignment of the flexible operation of job \( (k - 2) \) from the second machine to the first we have \( \hat{G}_2 = \alpha_{k-3} + L(\beta_1) + L(\beta_1) + L(\beta_1) \). Using this together with (9) we have, \( \hat{G}^k_2 = \alpha_{k-3} + L(\beta_1) + L(\beta_1) + L(\beta_1) \leq \alpha_{k-3} + L(\beta_4) + L(\beta_2) + L(\beta_1) = G^k_2 \). Consequently, if \( I_k \) has pattern \( \beta_1 \), then having the same pattern for \( I_{k-1} \) does not increase the makespan.

These two cases prove that, if \( I_k \) has pattern \( \beta_1 \) for \( k \geq 4 \) then the immediate predecessor of this interval must also have the same pattern. Using the same arguments for all intervals down to the third, all must have the same pattern. This completes the first part of the proof.
Case 2. $3 \leq k \leq n - 2$ and $k + 1 \leq i \leq n - 1$:

Let $H_i^k = T_{k+1} - T_k^1 = \alpha_k + \alpha_{k+1} + \alpha_{k+2} + \alpha_{k+3}$. If $I_k$ has pattern $\beta_1$, then $p_{k+1}^2 = f^2$. As a consequence, possible alternatives for the pattern of $I_{k+1}$ are $\beta_1$ and $\beta_4$. In contradiction to the statement, assume $I_{k+1}$ has pattern $\beta_4$. As a consequence, $p_{k+2}^2 = f^2 + s^2$. Then we have the following alternatives for the pattern of $I_{k+2}$:

1. If $I_{k+2}$ has pattern $\beta_3$, then $H_i^k = L(\beta_1) + L(\beta_4) + L(\beta_3) + \alpha_{k+3}$. Let us switch the assignment of the flexible operation of job $(k + 1)$ so that $I_{k+1}$ now has pattern $\beta_1$. Then, $H_i^k = L(\beta_1) + L(\beta_4) + \alpha_{k+3}$. Using (10), $\hat{H}_1^k \leq H_i^k$.

2. If $I_{k+2}$ has pattern $\beta_2$, then $H_i^k = L(\beta_1) + L(\beta_4) + L(\beta_2) + \alpha_{k+3}$. If the assignment of job $k + 1$ is switched, $\hat{H}_1^k = L(\beta_1) + L(\beta_1) + L(\beta_1) + \alpha_{k+3}$. Using (9), we have $\hat{H}_1^k \leq H_i^k$.

Case 3. $k = n - 1$ and $i = n$:

If $I_{n-1} has pattern $\beta_1$, considering the assignment of the last job, there are two feasible alternatives, namely $p_n^2 = f^2$ case and $p_n^2 = f^2 + s^2$. We will compare these two alternatives using their corresponding $C_n^2 - T_n^1$ values. Since the previous intervals are not affected from any change in the assignment of the last job, $T_n^1$ values are the same for both alternatives. However, $C_n^2$ values depend on the assignment of the last job. Therefore, for the first alternative we use $\hat{C}_n^2$ and for the second one we use $C_n^2$. We have the following:

$$\hat{C}_n^2 - T_n^1 = L(\beta_1) + f^2, \quad \text{if } p_n^1 = f^1 + s^1. \quad (11)$$

$$C_n^2 - T_n^1 = L(\beta_1) + (f^2 + s^2), \quad \text{if } p_n^1 = f^1. \quad (12)$$

Let us add $\max(f^1 + s^1 - f^2, 0)$ to the both sides of the first equation and $\max(f^1 + s^1 - (f^2 + s^2), 0)$ to the both sides of the second equation. Then the first one becomes

$$\hat{C}_n^2 - T_n^1 + \max(f^1 + s^1 - f^2, 0) = L(\beta_1) + \max(f^1 + s^1, f^2) = 2L(\beta_1).$$

On the other hand, the second one becomes

$$C_n^2 - T_n^1 + \max(f^1 + s^1 - (f^2 + s^2), 0) = L(\beta_1) + \max(f^1 + s^1, f^2 + s^2) = L(\beta_1) + L(\beta_2).$$

Using (9), we have $\hat{C}_n^2 - T_n^1 + \max(f^1 + s^1 - f^2, 0) \leq C_n^2 - T_n^1 + \max(f^1 + s^1 - (f^2 + s^2), 0)$. Since $\max(f^1 + s^1 - f^2, 0) \geq \max(f^1 + s^1 - (f^2 + s^2), 0)$, $\hat{C}_n^2 \leq C_n^2$. Therefore, if $I_{n-1}$ has pattern $\beta_1$ in the optimal solution, the flexible operation of the last job is assigned to the first machine. $\Box$
As a consequence of this lemma, if the flexible operations are assigned to the first machine for two consecutive jobs (in such a case, there will be at least one interval with pattern $\beta_1$), then in the optimal schedule, the flexible operations must be assigned to the first machine for all jobs except the first job. It is shown in Example 1 that in the optimal solution the first job need not have this property. However, there are only two alternative assignments for this job which can be seen in Figure 1. Let us define the following for notational simplicity:

\[
\begin{aligned}
\sigma_1 &= f^1 + f^2, \\
\sigma_2 &= f^1 + s^1 + f^2, \\
\sigma_3 &= f^1 + f^2 + s^2, \\
\sigma_4 &= f^1 + s^1 + f^2 + s^2.
\end{aligned}
\]

Then the corresponding makespan values for the two alternatives are as follows:

\[
\begin{aligned}
C_{nb}^1 &= \sigma_2 + (n - 1)L(\beta_1), & \text{if } p^1_1 &= f^1 + s^1, \\
C_{nb}^2 &= \sigma_1 + L(\beta_2) + (n - 2)L(\beta_1), & \text{if } p^1_1 &= f^1.
\end{aligned}
\]

In these equations, $C_{nb}^k$ is used to denote the makespan value for the no-buffer case for alternative solution $k$. The next lemma determines a special case where one of these two alternatives is optimal.

**Lemma 3.** If $f^2 \geq f^1 + s^1$, \(\min\{C_{nb}^1, C_{nb}^2\}\) is optimal.

**Proof.** Under the stated condition, even the flexible operation of a part is assigned to the first machine, the processing time on the second machine is greater than total processing time on the first machine. Therefore, interval lengths are determined by $f^2$. Using this, we can easily calculate a lower bound for the optimal makespan. According to this, in order to minimize the lengths of the intervals, the flexible operations must be assigned to the first machine for all jobs, except the first and the last ones. Additionally, assigning the flexible operation of the last job to the second machine increases the length of the last interval by $s^2$ without changing any other. Therefore, the flexible operation of the last job must also be assigned to the first machine. However, there is no such conclusion for the first job. As a result, the lower bound of the optimal makespan is \(\min\{f^1 + nf^2 + s^1, f^1 + nf^2 + s^2\}\). On the other hand, we have $C_{nb}^1 = f^1 + nf^2 + s^1$ and $C_{nb}^2 = f^1 + nf^2 + s^2$. The minimum of these two values is equal to the lower bound implying that one of these two alternatives is optimal. \(\square\)

The next lemma is a direct result of Lemma 2 and the reversibility property stated in Lemma 1 and thus the proof is omitted here.

**Lemma 4.** Given an optimal schedule, if there exists at least one $k$, $2 \leq k \leq (n - 1)$, such that $I_k$ has pattern $\beta_3$, then $I_i$ also has pattern $\beta_3$ for all $i = 2, 3, \ldots, (n - 1)$. 

12
Similar to the previous case, depending on the assignment of the flexible operation of the last job we have two different alternatives with the following makespan values:

\[
C_{nb}^3 = \sigma_3 + (n-1)L(\beta_3), \quad \text{if } p_n^1 = f_1^1, \tag{15}
\]

\[
C_{nb}^4 = \sigma_1 + L(\beta_2) + (n-2)L(\beta_3), \quad \text{if } p_n^1 = f_1^1 + s^1. \tag{16}
\]

The following lemma determines a special condition similar to Lemma 3 where one of these two alternatives is optimal.

**Lemma 5.** If \( f_1^1 \geq f_2^2 + s^2 \), \( \min\{C_{nb}^3, C_{nb}^4\} \) is optimal.

The proof is very similar to the proof of Lemma 3 and omitted here.

**Lemma 6.** Given an optimal schedule, if there exists at least one \( k, 3 \leq k \leq (n-1) \), such that \( I_k \) has either pattern \( \beta_2 \) or \( \beta_4 \), then the optimal solution is an alternating sequence of \( \beta_2 \) and \( \beta_4 \) patterns for intervals \( 3, 4, \ldots, (n-1) \).

**Proof.** If \( I_k, k = 1, 2, \ldots, (n-1) \), has pattern \( \beta_2 \), this means that the flexible operation is assigned to the first machine for this job. Thus, \( p_k^2 = f_2^2 \) which means that \( I_{(k+1)} \) cannot have pattern \( \beta_2 \). It can either be \( \beta_1 \) or \( \beta_4 \). Lemmas 2 proves that if in an optimal schedule \( I_k \) has pattern \( \beta_1 \) for \( 3 \leq k \leq (n-1) \), then all intervals are \( \beta_1 \). Therefore, if there is an interval \( I_k, k = 1, 2, \ldots, (n-1) \) with pattern \( \beta_2 \) in an optimal schedule, then it must be followed by \( \beta_4 \). A similar reasoning proves that a \( \beta_4 \) pattern must be followed by a \( \beta_2 \) pattern in an optimal schedule, which proves that \( \beta_2 \) and \( \beta_4 \) patterns can only be sequenced alternatingly.

This lemma provides the optimal assignments for jobs \( 3, 4, \ldots, (n-1) \) if there exists at least one \( \beta_2 \) or \( \beta_4 \) pattern. For the remaining jobs (first, second and the last in the sequence) depending on the two possible assignments of the flexible operations there are a total of eight different solution alternatives. Additionally, the makespan values of these alternatives depend also on whether the number of jobs to be produced is even or odd. All possible solution alternatives and their corresponding makespan values are summarized in Table 1. In this table the first three columns show the alternative assignments of the flexible operations of jobs 1, 2, and \( n \), respectively. An entry of “1” (“2”) in these columns means the flexible operation is assigned to the first (second) machine for the corresponding job.

Let us consider two different solutions, \( A \) and \( B \). \( B \) is said to dominate \( A \), if makespan value of \( A \) is greater than that of \( B \) for all possible parameter values. A solution is called nondominated,
Assignment to M/c’s

<table>
<thead>
<tr>
<th>Job</th>
<th>m1</th>
<th>m2</th>
<th>Job n</th>
<th>n = odd</th>
<th>n = even</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>$\sigma_2 + 2L(\beta_2) + \frac{(n-3)}{2}(L(\beta_3) + L(\beta_2))$</td>
<td>$\sigma_2 + L(\beta_1) + \frac{(n-2)}{2}(L(\beta_3) + L(\beta_2))$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>$n$</td>
<td>$\sigma_4 + L(\beta_1) + \frac{(n-1)}{2}L(\beta_4) + \frac{(n-3)}{2}L(\beta_2)$</td>
<td>$\sigma_4 + L(\beta_1) + L(\beta_3) + \frac{(n-2)}{2}L(\beta_4) + \frac{(n-4)}{2}L(\beta_2)$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>$n$</td>
<td>$\sigma_2 + \frac{(n-1)}{2}(L(\beta_3) + L(\beta_2))$</td>
<td>$\sigma_2 + L(\beta_1) + \frac{(n-2)}{2}(L(\beta_3) + L(\beta_2))$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>$n$</td>
<td>$\sigma_4 + L(\beta_1) + \frac{(n-1)}{2}L(\beta_3) + \frac{(n-3)}{2}L(\beta_2)$</td>
<td>$\sigma_4 + \frac{n}{2}L(\beta_2) + \frac{n-2}{2}L(\beta_2)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>$n$</td>
<td>$\sigma_1 + L(\beta_1) + \frac{(n-3)}{2}L(\beta_3) + \frac{(n-1)}{2}L(\beta_2)$</td>
<td>$\sigma_1 + \frac{(n-2)}{2}L(\beta_4) + \frac{n}{2}L(\beta_2)$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>$n$</td>
<td>$\sigma_3 + \frac{(n-1)}{2}(L(\beta_3) + L(\beta_2))$</td>
<td>$\sigma_3 + L(\beta_1) + \frac{(n-2)}{2}(L(\beta_3) + L(\beta_2))$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>$n$</td>
<td>$\sigma_1 + L(\beta_1) + \frac{(n-3)}{2}L(\beta_3) + \frac{(n-1)}{2}L(\beta_2)$</td>
<td>$\sigma_1 + L(\beta_1) + L(\beta_3) + \frac{(n-4)}{2}L(\beta_4) + \frac{(n-2)}{2}L(\beta_2)$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>$n$</td>
<td>$\sigma_3 + 2L(\beta_3) + \frac{(n-1)}{2}(L(\beta_4) + L(\beta_2))$</td>
<td>$\sigma_3 + L(\beta_3) + \frac{(n-3)}{2}L(\beta_4) + \frac{(n-2)}{2}L(\beta_2)$</td>
</tr>
</tbody>
</table>

Table 1: Makespan values for the possible solution alternatives

If there exists no other solutions dominating this one. Once it is proved that a solution is dominated by another one, there is no need to consider this solution any more. We compare the values given in Table 1 with each other to determine the set of nondominated solution alternatives.

Let $C_{nb}^{i,j,k(l)}$ denote the makespan value of the alternative in which the flexible operation of the first job is assigned to machine $i$, the second to machine $j$, and the $n^{th}$ job to machine $k$, where $i, j, k \in \{1, 2\}$. The superscript $l \in \{e, o\}$ denotes whether the total number of jobs to be processed, $n$, is even or odd. As an example, $C_{nb}^{111(o)}$ denotes the makespan value in which the flexible operations are assigned to the first machine for jobs 1, 2, and $n$ and there are an odd number of jobs to be processed. Similarly, $C_{nb}^{212(e)}$ denotes the makespan value in which the flexible operations are assigned to the second machine for the first and the last jobs and to the first machine for the second job; there are an even number of jobs to be processed.

In Table 1, $C_{nb}^{111(e)}$ and $C_{nb}^{121(e)}$ have exactly the same makespan values. Similarly, $C_{nb}^{222(e)}$ and $C_{nb}^{212(e)}$ have identical makespan values. Hence, without loss of generality, we will not consider $C_{nb}^{121(e)}$ and $C_{nb}^{212(e)}$ any more. In the following we will show that $C_{nb}^{211(e)}$ is always smaller than $C_{nb}^{112(e)}$, $C_{nb}^{122(e)}$, and $C_{nb}^{221(e)}$. From Table 1 we have the following:

$$C_{nb}^{211(e)} = \sigma_1 + \frac{(n-2)}{2}L(\beta_2) + \frac{n}{2}L(\beta_2)$$

$$= f^1 + f^2 + \frac{(n-2)}{2} \max\{f^1, f^2\} + \frac{n}{2} \max\{f^1 + s^1, f^2 + s^2\}.$$  \hspace{1cm} (17)
From Table 1 we have:

\[ C_{nb}^{112(e)} = \sigma_4 + L(\beta_1) + L(\beta_3) + \frac{(n-2)}{2} L(\beta_4) + \frac{(n-4)}{2} L(\beta_2) \]

\[ = f^1 + s^1 + f^2 + s^2 + \max\{f^1 + s^1, f^2\} + \max\{f^1, f^2 + s^2\} \]

\[ + \frac{(n-2)}{2} \max\{f^1, f^2\} + \frac{(n-4)}{2} \max\{f^1 + s^1, f^2 + s^2\}. \]

Comparing this with (17):

\[ C_{nb}^{112(e)} - C_{nb}^{211(e)} = s^1 + s^2 + \max\{f^1 + s^1, f^2\} + \max\{f^1, f^2 + s^2\} - 2 \max\{f^1 + s^1, f^2 + s^2\} \]

\[ = \max\{f^1 + s^1 + s^2, f^2 + s^2\} + \max\{f^1 + s^1, f^2 + s^2 + s^2\} - 2 \max\{f^1 + s^1, f^2 + s^2\} = 0. \]

From Table 1 we have:

\[ C_{nb}^{122(e)} = f^1 + s^1 + f^2 + s^2 + \frac{n}{2} \max\{f^1, f^2\} + \frac{(n-2)}{2} \max\{f^1 + s^1, f^2 + s^2\}. \]

Comparing with (17) we have:

\[ C_{nb}^{122(e)} - C_{nb}^{211(e)} = s^1 + s^2 + \max\{f^1, f^2\} - \max\{f^1 + s^1, f^2 + s^2\} \]

\[ = \max\{f^1 + s^1 + s^2, f^2 + s^2\} + \max\{f^1 + s^1, f^2 + s^2\} - \max\{f^1 + s^1, f^2 + s^2\} = 0. \]

From Table 1, \( C_{nb}^{221(e)} \) is equal to the following:

\[ C_{nb}^{221(e)} = f^1 + f^2 + \max\{f^1 + s^1, f^2\} + \max\{f^1, f^2 + s^2\} \]

\[ + \frac{(n-4)}{2} \max\{f^1, f^2\} + \frac{(n-2)}{2} \max\{f^1 + s^1, f^2 + s^2\}. \]

Then we have the following:

\[ C_{nb}^{221(e)} - C_{nb}^{211(e)} = \max\{f^1 + s^1, f^2\} + \max\{f^1, f^2 + s^2\} - \max\{f^1 + s^1, f^2 + s^2\} \]

\[ = \max\{f^1 + s^1, f^1 + s^1 + s^2, f^2 + s^2, f^1 + f^2 + 2f^2 + 2s^2\} - \max\{f^1 + s^1, f^1 + f^2 + s^2\}. \]

If we investigate these two max statements term by term, we can conclude that \( C_{nb}^{221(e)} \geq C_{nb}^{211(e)} \).

For the remaining alternatives, when \( n \) is even, there are no other dominance relations. This means that, depending on the given parameter values, one of these nondominated alternatives can be optimal.

When there are an odd number of jobs to be produced, we will prove that \( C_{nb}^{211(o)} \) is always smaller than \( C_{nb}^{112(o)} \) and \( C_{nb}^{122(o)} \), and \( C_{nb}^{221(o)} \) is always smaller than \( C_{nb}^{122(o)} \) and \( C_{nb}^{212(o)} \). For this purpose, we will use the following equations that we have in Table 1:
For $C_{nb}^{112(o)}$, we have the following:

$$C_{nb}^{112(o)} = f^1 + s^1 + f^2 + s^2 + \max\{f^1, s^1, f^2\} + \frac{(n-1)}{2} \max\{f^1, f^2\} + \frac{(n-3)}{2} \max\{f^1 + s^1, f^2 + s^2\}. \quad (19)$$

Comparing this with Equation (18), we have:

$$C_{nb}^{112(o)} - C_{nb}^{211(o)} = s^1 + s^2 + \max\{f^1, f^2\} - \max\{f^1 + s^1, f^2 + s^2\}$$

$$= \max\{f^1 + s^1 + s^2, f^2 + s^2\} - \max\{f^1 + s^1, f^2 + s^2\} \geq 0.$$

For $C_{nb}^{122(o)}$, we have the following:

$$C_{nb}^{122(o)} = f^1 + s^1 + f^2 + s^2 + \max\{f^1, f^2 + s^2\} + \frac{(n-1)}{2} \max\{f^1, f^2\} + \frac{(n-3)}{2} \max\{f^1 + s^1, f^2 + s^2\}. \quad (19)$$

Comparing this with Equation (19), we have:

$$C_{nb}^{122(o)} - C_{nb}^{221(o)} = s^1 + \max\{f^1, f^2\} - \max\{f^1 + s^1, f^2\} \geq 0.$$
As a consequence, the number of nondominated alternatives is reduced from 16 to 7. Table 2 lists the makespan values for these alternatives.

Table 2: Alternative makespan values for the possible solutions

<table>
<thead>
<tr>
<th>Job 1</th>
<th>Job 2</th>
<th>Job n</th>
<th>n = odd</th>
<th>n = even</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$C_{nb}^{5} = \sigma_2 + 2L(\beta_1) + \frac{(n-3)}{2}(L(\beta_4) + L(\beta_2))$</td>
<td>$C_{nb}^{9} = \sigma_2 + L(\beta_1) + \frac{(n-2)}{2}(L(\beta_4) + L(\beta_2))$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>$C_{nb}^{6} = \sigma_1 + L(\beta_1) + \frac{(n-3)}{2}L(\beta_4) + \frac{(n-1)}{2}L(\beta_2)$</td>
<td>$C_{nb}^{10} = \sigma_1 + \frac{(n-2)}{2}L(\beta_4) + \frac{n}{2}L(\beta_2)$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>$C_{nb}^{7} = \sigma_3 + 2L(\beta_3) + \frac{(n-3)}{2}(L(\beta_4) + L(\beta_2))$</td>
<td>$C_{nb}^{11} = \sigma_3 + L(\beta_3) + \frac{(n-2)}{2}(L(\beta_4) + L(\beta_2))$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>$C_{nb}^{8} = \sigma_1 + L(\beta_3) + \frac{(n-3)}{2}L(\beta_4) + \frac{(n-1)}{2}L(\beta_2)$</td>
<td>——</td>
</tr>
</tbody>
</table>

As a consequence, the number of nondominated alternatives is reduced from 16 to 7. Table 2 lists the makespan values for these alternatives.

Note that, for the alternatives listed in Table 2 there are no dominance relations. Depending on the parameter values anyone of these can be optimal. Table 3 provides example parameter values for which each alternative is optimal when $n$ is odd and even. The bold numbers in each row show the optimal makespan values for the given parameters.

Table 3: Examples where each alternative cannot be dominated for $n = odd$ and $n = even$ cases

As a result, a constant time algorithm that computes the optimal makespan is presented as Algorithm 1. Once the assignments of the flexible operations are known, generating the optimal schedule is straightforward by using Property 2. In the next section we consider the variation of the problem where there is an infinite capacity buffer space in between the machines.

### 3.2 Infinite Capacity Buffer Case

In this section we develop a constant time solution procedure for the infinite capacity buffer case. In an infinite capacity buffer system whenever the processing of a job is completed on the first machine
### Algorithm 1 Algorithm for the No-Buffer Case

**Input:** $n$, $f^1$, $f^2$, $s^1$, and $s^2$.

**Output:** $C^{\ast}_{\text{max}}$.

1. **if** $(f^2 \geq f^1 + s^1)$ **then**
   2. $C^{\ast}_{\text{max}} = \min\{C^1_{\text{nb}}, C^2_{\text{nb}}\}$.
   3. **else if** $(f^1 \geq f^2 + s^2)$ **then**
   4. $C^{\ast}_{\text{max}} = \min\{C^3_{\text{nb}}, C^4_{\text{nb}}\}$.
   5. **else if** $(n = \text{odd})$ **then**
   6. $C^{\ast}_{\text{max}} = \min_{i \in [1,8]} \{C^i_{\text{nb}}\}$.
   7. **else**
   8. $C^{\ast}_{\text{max}} = \min_{i \in [1,4] \cup [9,11]} \{C^i_{\text{nb}}\}$.
   9. **end if**

There is always an available space for this job in the buffer. Therefore, the first machine is never blocked. If the optimal number of jobs for which the flexible operations are assigned to the first machine is known, the optimal schedule can easily be generated using the SPT1-LPT2 algorithm (Johnson’s algorithm [14]). According to this algorithm the jobs for which the processing times on the first machine are less than that of the second machine are sequenced with respect to their processing times on the first machine using the shortest processing time first method. The remaining set of jobs are sequenced with respect to their processing times on the second machine using the largest processing time first method. Afterwards, these two sequences are concatenated.

Our solution procedure relies on determining the number of jobs for which the flexible operations are assigned to the first machine in the optimal schedule. Before going in to details of this procedure let us first handle two special cases: If $f^1 \geq f^2 + s^2$, then even if the flexible operations for all jobs are assigned to the second machine, the processing times on the first machine are still greater than the realized processing times on the second machine. Without considering the last job, this solution is the best that can be achieved. The assignment for the last job need not be to the second machine resulting from Example 1 and the reversibility property. There are two possible assignment alternatives for this job, which depend on the values of $s^1$ and $s^2$. Let $C^k_{\text{inf}}$ denote the makespan value for alternative $k$. 
Figure 3: Minimizing makespan for infinite capacity buffer case

for the infinite buffer case. The corresponding makespan values for these alternatives are as follows:

\[ C_{inf}^1 = nf^1 + f^2 + s^2, \quad \text{if } p_n^1 = f^1, \]  
\[ C_{inf}^2 = nf^1 + s^1 + f^2, \quad \text{if } p_n^1 = f^1 + s^1. \]

When these two makespan values are compared with each other, if \( s^1 > s^2 \) (\( s^1 < s^2 \)), then assigning the flexible operation of the last job to the second (first) machine is optimal. If \( s^1 = s^2 \) both alternatives work equally well. Using the reversibility property, if \( f^2 \geq f^1 + s^1 \), then all flexible operations are assigned to the first machine except the first job. Depending on the assignment of the first job we have the following alternatives:

\[ C_{inf}^3 = f^1 + nf^2 + s^2, \quad \text{if } p_n^1 = f^1, \]  
\[ C_{inf}^4 = f^1 + s^1 + nf^2, \quad \text{if } p_n^1 = f^1 + s^1. \]

Let us now consider the case where \( f^1 < f^2 + s^2 \) and \( f^2 < f^1 + s^1 \). Let \( r^* \) denote the number of jobs for which the flexible operations are assigned to the first machine. Since in this study, jobs are identical, there are only two types of processing time values depending on the assignments: \( (f^1 + s^1, f^2) \) and \( (f^1, f^2 + s^2) \). Then, as a consequence of the Johnson’s algorithm ([14]), the flexible operations are assigned to the second machine for the first \( (n - r^*) \) jobs and to the first machine for the remaining \( r^* \) jobs in the optimal schedule. We present this result as a property of the optimal solutions for further reference in the text.

**Property 3.** *In the optimal schedule, the flexible operations are assigned to the second machine for the first \( n - r^* \) jobs and to the first machine for the last \( r^* \) jobs.*

In the following, we develop a procedure to determine the value of \( r^* \). Observe that, the second machine is idle during the first machine is processing the first job and similarly, the first machine is
idle during the second machine is processing the last job. These idle times are called unavoidable idle times. Without considering these portions on both machines, as it is depicted in Figure 3, the rest of the schedule is minimized when the total processing times on the first and the second machines are as close to each other as possible (the lower bound is attained when these two values are equal to each other). That is, when \(|(C^1_n - T^1_2) - (C^2_{n-1} - T^2_1)|\) is minimized. The decision variables are the number of flexible operations that must be assigned to the first and the second machines. Then, considering all possible assignment alternatives for the first and the last jobs, which effect the unavoidable idle times, the optimal makespan can be found. Hence, we need to evaluate the following alternatives:

1. If \(p_1 = f^1\) and \(p_n = f^1 + s^1\): Let \(r_1\) denote the number of jobs for which the flexible operations are assigned to the first machine. Then, the total processing times on the two machines will be equal if \((n - 1)f^1 + r_1s^1 = (n - 1)f^2 + (n - r_1)s^2\). This yields the following:

   \[
   r_1 = \frac{(n - 1)(f^2 + s^2 - f^1) + s^2}{s^1 + s^2}.
   \]

   By definition, \(r_1\) must be an integer. Then the optimal solution can be found by trying \([r_1]\) and \([r_1]\) values and selecting the one with the smaller makespan value. \([a]\) denotes the largest integer value smaller than \(a\) and \([a]\) denotes the smallest integer value larger than \(a\). The corresponding makespan value is the following:

   \[
   C_{inf}^5 = \min\{nf^1 + [r_1]s^1 + f^2, f^1 + n s^1 + (n - [r_1])s^2\}. \tag{24}
   \]

2. If \(p_1 = f^1 + s^1\) and \(p_n = f^1 + s^1\): Let \(r_2\) denote the number of jobs for which the flexible operations are assigned to the first machine. Then, using similar arguments as above, \((n - 1)f^1 + (r_2 - 1)s^1 = (n - 1)f^2 + (n - r_2)s^2\). This yields the following:

   \[
   r_2 = \frac{(n - 1)(f^2 + s^2 - f^1) + s^1 + s^2}{s^1 + s^2}.
   \]

   The corresponding makespan value is the following:

   \[
   C_{inf}^6 = \min\{nf^1 + [r_2]s^1 + f^2, f^1 + s^1 + nf^2 + (n - [r_2])s^2\}. \tag{25}
   \]

3. If \(p_1 = f^1\) and \(p_n = f^1\): Denoting the number of jobs for which the flexible operations are assigned to the first machine by \(r_3\), we must have \((n - 1)f^1 + r_3s^1 = (n - 1)f^2 + (n - 1 - r_3)s^2\), which gives the following:

   \[
   r_3 = \frac{(n - 1)(f^2 + s^2 - f^1)}{s^1 + s^2}.
   \]

20
The corresponding makespan value is the following:

\[
C_{inf}^{7} = \min\{nf^1 + |r_3|s^1 + f^2 + s^2, f^1 + nf^2 + (n - [r_3])s^2\}. \tag{26}
\]

4. If \(p_1^1 = f^1 + s^1\) and \(p_n^1 = f^1\): Let \(r_3\) denote the number of jobs for which the flexible operations are assigned to the first machine. Then, we must have \((n - 1)f^1 + (r_4 - 1)s^1 = (n - 1)f^2 + (n - 1 - r_4)s^2\). Therefore, we have the following:

\[
r_4 = \frac{(n - 1)(f^2 + s^2 - f^1) + s^1}{s^1 + s^2}.
\]

The corresponding makespan value is the following:

\[
C_{inf}^{8} = \min\{nf^1 + |r_4|s^1 + f^2 + s^2, f^1 + s^1 + nf^2 + (n - [r_4])s^2\}. \tag{27}
\]

In Case 4 above, the unavoidable idle times on both machines are on their maximum values, which increases the makespan value. The following lemma proves that this alternative cannot be optimal for any parameter input.

**Lemma 7.** For any parameter input \(\min\{C_{inf}^{5}, C_{inf}^{6}\} \leq C_{inf}^{8}\).

**Proof.** We will prove this lemma considering the following cases:

1. If \(s^1 \geq s^2\), then \(r_1 \leq r_4 \leq r_1 + 1\). Let us compare \(C_{inf}^{8}\) with \(C_{inf}^{5}\) given in Equation (24). Comparing the first terms of the min functions of both equations, we have \(nf^1 + |r_3|s^1 + f^2 + s^2 \geq nf^1 + |r_1|s^1 + f^2\). When the second terms are compared with each other, \(f^1 + s^1 + nf^2 + (n - |r_3|)s^2 \geq f^1 + s^1 + nf^2 + (n - |r_1| - 1)s^2 \geq f^1 + s^1 + nf^2 + (n - |r_1|)s^2\). Therefore, for \(s^1 \geq s^2\), \(C_{inf}^{8} \geq C_{inf}^{5}\).

2. If \(s^1 < s^2\), then \(r_4 \leq r_2 \leq r_4 + 1\). Let us compare \(C_{inf}^{8}\) with \(C_{inf}^{6}\) given in Equation (25). Comparing the first terms of the min functions of both equations, we have \(nf^1 + |r_4|s^1 + f^2 \leq nf^1 + (|r_4| + 1)s^1 + f^2 < nf^1 + |r_4|s^1 + f^2 + s^2\). When the second terms are compared with each other, \(f^1 + s^1 + nf^2 + (n - |r_4|)s^2 \geq f^1 + s^1 + nf^2 + (n - |r_2|)s^2\). Therefore, for \(s^1 < s^2\), \(C_{inf}^{8} \geq C_{inf}^{6}\).

These two cases prove that \(\min\{C_{inf}^{5}, C_{inf}^{6}\} \leq C_{inf}^{8}\). \(\square\)
In conclusion, the constant time algorithm for the infinite capacity buffer case is presented as Algorithm 2. Note that, this algorithm outputs the minimum makespan values. In order to compute these values, number of flexible operations that are assigned to the first and second machines are also calculated. Using these together with Property 3 one can easily generate the optimal schedule ($T_{ij}^*$ values for $i = 1, 2, \ldots, n$ and $j = 1, 2$).

Algorithm 2 Algorithm for the Infinite Capacity Buffer Case

Input: $n$, $f_1$, $f_2$, $s_1$, and $s_2$.
Output: $C_{\text{max}}^*$.

1: if ($f_1 \geq f_2 + s_2$) then
2: $C_{\text{max}}^* = \min\{C_{\text{inf}}^1, C_{\text{inf}}^2\}$ given in Equations (20) and (21).
3: else if ($f_2 \geq f_1 + s_1$) then
4: $C_{\text{max}}^* = \min\{C_{\text{inf}}^3, C_{\text{inf}}^4\}$ given in Equations (22) and (23).
5: else
6: $C_{\text{max}}^* = \min\{C_{\text{inf}}^5, C_{\text{inf}}^6, C_{\text{inf}}^7\}$ given in Equations (24), (25), and (26).
7: end if

After determining solution procedures for the no-buffer and infinite capacity buffer cases in this section, the next section provides insights about the benefits of flexibility.

4 Managerial Insights

In the previous section we developed the algorithms that produce the optimal throughput rates for given parameter values. This section analyzes benefits of flexibility and the conditions to attain the maximum available benefit. We determine the cases (parameter values) under which a flexible system is more beneficial than an inflexible one and the cases under which there are no benefits at all. Furthermore, we seek answers to the following questions; where should we concentrate if we want to increase the level of flexibility in the system? Is the first or the second machine the best alternative in terms of throughput rate? What about the cost? How much flexibility do we really need? Is it possible to quantify the value of additional flexibility? Answers to these questions are especially important since without a clear understanding of the benefit associated with different levels of flexibility, firms are reluctant to invest in flexibility ([19]).
For this analysis, we consider the option of increasing the level of flexibility in a classical inflexible 2-machine flowshop. If the workstations consist of CNC machines, the level of flexibility can be increased by acquiring extra cutting tools and loading these on the tool magazines of both machines. As a result, both machines are capable of processing the operations that require these tools. On the other hand, if the workstations consist of manual operations, by cross-training the workers and placing the necessary equipment to both of the workstations we transform the corresponding operations to flexible. In conclusion, it is possible to increase the level of flexibility in the system by some capital investment plus annual costs such as the maintenance. Under this setting, increasing the level of flexibility means transforming some of the fixed operations to flexible ones. By this change, the fixed operation time on the related machine reduces while the total processing time of the flexible operations increases and the total processing times on the individual machines depend on the assignment of the flexible operations.

The processing times of the operations are discrete values. When an operation is made flexible, the corresponding processing time is transformed from fixed to flexible. For a given system there is a finite number of alternatives for making the system flexible. This includes all combinations of operations that a part requires. However, in order to deduce conclusions and make generalizations for all problem types, we assume that the level of flexibility can be increased to any value continuously, not limited by discrete processing time values. The only constraint is, flexible processing time value can not be greater than the original processing time of the machine in the fixed design.

For the classical system, the processing times of the machines are assumed to be fixed for all jobs and denoted by $p_a$ for the first and $p_b$ for the second machine. The makespan value for this well known classical manufacturing environment is denoted by $C_{max}^{cc}$. Let $e^i$ denote the processing time of the flexible operations on machine $i$ after the conversion is made. In order to have a meaningful comparison, we assume that the total processing time that a part requires in the flexible system is identical to that of an inflexible system, given that the flexible operations are assigned to the machine they are converted from. Otherwise, technological difference between the machines can cause deviations. In order to satisfy this, if the second machine is used to increase the flexibility then the processing times in the corresponding flexible system are $f^1 = p_a$, $f^2 = p_b - e^2$, $s^1 = e^1$, and $s^2 = e^2$. If, on the other hand, the first machine is used for this purpose, the corresponding processing times are $f^1 = p_a - e^1$, $f^2 = p_b$, $s^1 = e^1$, and $s^2 = e^2$.

Considering a flowshop system with partial resource flexibility where there are $n$ jobs and $m$
machines Daniels et al. [9], concluded that a large portion of the available benefit associated with labor flexibility can be realized with a relatively small investment in crosstraining. Similar conclusions are also made by Jordan and Graves [15] and Nomden and van der Zee [17]. Our results in the current study also support these conclusions, which are especially important when the cost associated with increasing the level of flexibility is directly proportional with the level of flexibility itself. Under such an assumption, since the marginal reduction in the makespan with respect to an increase in the flexibility level is very small, but the increase in the corresponding cost is relatively high, a solution with low flexibility level can be preferable to another solution with a high flexibility level. There are some other cases where the cost is not directly proportional to the level of flexibility. For example a cutting tool associated with an operation with smaller processing time can be much more costly than another one associated with a longer processing time. Hence, a higher flexibility level can be preferable to lower levels in terms of both the makespan and the associated cost. In scheduling literature, most of the studies address only a single criterion. However, most of the real-life problems require multiple conflicting criteria and optimization of these simultaneously yields more realistic schedules for the decision maker. We compare the original system with the flexible one in terms of the makespan and determine the reduction in the makespan for different parameter values. We will comment on the associated cost of increasing the flexibility later on with the help of an example. We denote the makespan of the flexible system by $C_{\text{max}}^f$.

In this section, we consider the no-buffer and the infinite capacity buffer cases and analyze the benefits of flexibility under different parameter values in each case. As mentioned before, the analysis is made for the non-identical machines problem, which is more realistic than the identical machines problem handled by Crama and Gultekin [6]. The identical machines problem is a special case of the current one where $s^1 = s^2 = s$. Therefore, the attained results throughout this section is valid for that problem as well. Furthermore, by assuming the machines to be nonidentical, we could see the effects of technological differences between the machines to the benefits of flexibility.

Throughout the analysis, we assumed that $p_a \geq p_b$. The analysis and the results for the remaining case is just the symmetric of this one as a consequence of the reversibility property.

The following lemma, which is valid for both the no-buffer and infinite capacity buffer cases, proves that the benefits associated with flexibility is limited when the machine with smaller processing time is used to increase the flexibility.

**Lemma 8.** If $p_a \geq p_b$, then converting fixed operations to flexible ones from the second machine
reduces the makespan by \( \max\{0, \epsilon^2 - \epsilon^1\} \).

**Proof.** For \( p_a \geq p_b \), independent of having a no-buffer or an infinite capacity buffer system, the makespan of the inflexible system can easily be calculated to be the following:

\[
C_{\text{max}}^c = np_a + p_b. \tag{28}
\]

If fixed operations from the second machine with total processing time of \( \epsilon^2 \) are changed to be flexible, then \( f^1 = p_a \), \( f^2 = p_b - \epsilon^2 \), \( s^1 = \epsilon^1 \), and \( s^2 = \epsilon^2 \). Since \( p_a \geq p_b \), then \( f^1 \geq f^2 + s^2 \). For the no-buffer case, under the given condition, from Lemma 5, the minimum of \( C_{3, \text{nb}}^3 \) or \( C_{4, \text{nb}}^4 \) will be the optimal solution. From Equation (15), \( C_{3, \text{nb}}^3 = f^1 + f^2 + s^1 + (n - 1) \max\{f^1, f^2 + s^2\} = np_a + p_b \) and from Equation (16), \( C_{4, \text{nb}}^4 = f^1 + f^2 + \max\{f^1 + s^1, f^2 + s^2\} + (n - 2) \max\{f^1, f^2 + s^2\} \) = \( np_a + p_b + \epsilon^1 - \epsilon^2 \). Then, we have the following:

\[
C_{\text{max}}^c - \min\{C_{3, \text{nb}}^3, C_{4, \text{nb}}^4\} = \max\{0, \epsilon^2 - \epsilon^1\}.
\]

For the infinite capacity buffer case, using Algorithm 2, the optimal makespan is the minimum of \( C_{\text{inf}}^1 \) and \( C_{\text{inf}}^2 \). From Equation (20), \( C_{\text{inf}}^1 = nf^1 + f^2 + s^2 = np_a + p_b \) and from Equation (21), \( C_{\text{inf}}^2 = nf^1 + s^1 + f^2 = np_a + \epsilon^1 + p_b - \epsilon^2 \). Then, we have the following:

\[
C_{\text{max}}^c - \min\{C_{\text{inf}}^1, C_{\text{inf}}^2\} = \max\{0, \epsilon^2 - \epsilon^1\}. \quad \Box
\]

As a consequence of this lemma, if the second machine is technologically new, \( (\epsilon^2 \leq \epsilon^1) \), and the processing time on this machine is smaller than the processing time on the first machine, then there is no benefit of converting fixed operations from the second machine. On the other hand, if the first machine has more advanced technology, then the reduction in the makespan is equal to the difference of the processing times of the flexible operations on both machines, \( (\epsilon^2 - \epsilon^1) \). Note that, this lemma includes \( p_a = p_b \) case, which can be rephrased as, if the processing times on both machines are equal to each other, called perfect balance in the literature, then the benefit of increasing the flexibility using the processing times of a single machine is limited.

As a consequence of Lemma 8, we only consider increasing the flexibility level using the operations on the first machine. The processing times of the inflexible system are \( p_a \) and \( p_b \) and the corresponding flexible ones are \( f^1 = p_a - \epsilon^1 \), \( f^2 = p_b \), \( s^1 = \epsilon^1 \), and \( s^2 = \epsilon^2 \). We aim to determine the gain in makespan quantified as \( C_{\text{max}}^c - C_{\text{max}}^f \) for different parameter values. We analyze the effects of the level of flexibility measured by \( \epsilon^1 \), the imbalance between the machine processing times
measured by $p_a/p_b$, and the technological difference between the machines measured by $s^2/s^1$. For $p_a \geq p_b$, the makespan value of the classical system for both no-buffer and infinite capacity buffer problems is equal to $C_{max}^c = np_a + p_b$. For the no-buffer problem, the optimal solution for the flexible system is found by using Algorithm 1 and for the infinite capacity buffer problem, it is found by using Algorithm 2.

Figure 4a uses Algorithm 1 to depict the reduction in the makespan with respect to the flexibility level. Different values are used for technological difference between the machines. In this graph $p_a/p_b$ ratio is fixed to 1.5. Since the graphs for $n = \text{odd}$ and $n = \text{even}$ cases are very similar to each other, we only consider the former one here. According to this figure, the reduction in the makespan is always greater when the second machine is technologically more advanced. In the figure, maximum reduction is attained when $s^2/s^1 = 0.5$. Additionally, whatever the technological difference be, $C_{max}^c - C_{max}^f$ has a steep increase as the level of flexibility increases. After reaching a maximum value for a relatively low level of flexibility, it starts decreasing. Therefore, it is not true that the gain always increases as the level of flexibility increases. In the figure, $s^2/s^1 = 1$ case is identical to the problem considered by Crama and Gultekin [6].

Similar conclusions can also be made for the infinite buffer case. Figure 5a depicts the same graph for this system. This graph shows that most of the benefits associated with the flexibility can be attained with a relatively small flexibility level. The greater the technological difference between the machines, the greater the benefit of flexibility. Different than the no-buffer case, $C_{max}^c - C_{max}^f$ continues to increase slightly with respect to $\epsilon^1$ with small fluctuations.

Figure 4b depicts the graph of $C_{max}^c - C_{max}^f$ with respect to $\epsilon^1$ for different values of $p_a/p_b$. In this
Figure 5: Reduction in makespan with respect to the level of flexibility for infinite buffer systems

graph $s^2/s^1$ is fixed to 0.75. We can see the effect of the imbalance between the machine processing times to the benefits of increasing flexibility. It can be concluded from this figure that the gain is greater when the imbalance between the machine processing times are greater. Maximum reduction is attained when $p_a/p_b = 3$. Similar to the previous case, most of the gain in the makespan is attained for low flexibility levels. When the processing times in the inflexible system are equal to each other, $p_a/p_b = 1$, the benefit of flexibility is very limited as proven in Lemma 8.

Figure 5b depicts the same for the infinite capacity buffer case. Similar to the no-buffer case, the benefits increase when $p_a/p_b$ increases. When the two processing times are equal to each other, $p_a/p_b = 1$, the benefits are limited.

Figure 6 compares the no-buffer and infinite capacity buffer systems with each other with respect to the flexibility level in the system. In this figure $p_a/p_b = 1.5$ and $s^2/s^1 = 0.75$. This figure shows the effects of the additional buffer space to the system. The maximum gain for the infinite capacity buffer systems is greater than the maximum gain of the no-buffer systems. However, the gap between the maximum gain of the infinite capacity and the maximum gain of the no-buffer systems is very small. For low flexibility levels both systems perform equally well. Although in an infinite capacity system, theoretically the buffer capacity must be at least $n-1$ parts, usually this maximum capacity need not be used to get the optimal makespan value. Therefore, one can compare the cost of attaining additional buffer space with the reduction in the makespan to design the production system.

The analysis in this section strengthen the conclusions of Daniels et al. [9] and Jordan and Graves [15] who stated that a large portion of the available benefit associated with flexibility can be realized with a relatively small flexibility level. However, the reduction in the makespan is not the only factor
for deciding the level of flexibility. The cost of increasing the flexibility level is also important for this decision. Considering the cost together with the makespan provides more insights to the decision maker. In some production systems, the cost associated with increasing the level of flexibility is directly proportional with the level of flexibility itself. In such a production system, consider two different flexibility levels, $\delta_1$ and $\delta_2$ such that $\delta_2 \geq \delta_1$. Solution corresponding to $\delta_1$ is better than $\delta_2$, in terms of both makespan and cost, if the reduction in the makespan with respect to $\delta_1$ is not less than the reduction in $\delta_2$. For such systems, the flexibility levels smaller than the points where the reduction in makespan reaches its maximum value are the only alternative solutions. There is no need to consider greater flexibility levels. However, the cost may not be directly proportional to the flexibility level in some systems. The following is an example of such a situation.

**Example 2.** Let us consider the production of 20 identical jobs in a 2-machine no-buffer flowshop. Each job requires 6 operations with corresponding processing times of $o_1 = 3$, $o_2 = 6$, $o_3 = 8$, $o_4 = 13$, $o_5 = 22$, and $o_6 = 42$ units. Assume that the second machine is capable of performing the sixth operation and all remaining ones must be processed on the first machine. Therefore, we have $p_a = 52$ and $p_b = 42$ units. Assume the second machine is more powerful and we have $s^2/s^1 = 0.75$. Using (28), we have $C^c_{\text{max}} = 1082$. Assume that with some capital investment, the operations can be made flexible. Let us suppose that the cost required to make each operation flexible is 2, 10, 7, 5, 12, and 8, respectively. If the first operation is made flexible, since $o_1 = 3$, the makespan of the flexible system is given by Lemma 5 and Equation 15. The reduction in the makespan is equal to 57.5 time units and the corresponding cost is 2 in monetary units. Note that, since $p_a > p_b$ and $s^2 \leq s^1$, from Lemma 8, converting the sixth operation into a flexible one does not reduce the makespan. Table 4
summarizes the costs and the makespan reductions for the alternatives. According to these, maximum reduction in makespan is attained when the second operation is made flexible with a corresponding cost of 10 units. If we compare the third and the fourth alternatives, the fourth alternative has a greater reduction in makespan with a smaller cost, which means the fourth alternative dominates the third one. The same alternative also dominates the fifth one. However, there are no dominance relations among the boldly written remaining three alternatives. Each alternative is better than any other one either in terms of the reduction in makespan or the associated cost.

<table>
<thead>
<tr>
<th>Alternative</th>
<th>Flexible Operation</th>
<th>Reduction in $C_{max}$</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$o_1$</td>
<td>57.5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>$o_2$</td>
<td>106</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>$o_3$</td>
<td>80</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>$o_4$</td>
<td>103</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>$o_5$</td>
<td>47</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 4: Reduction in the makespan and the associated costs for Example 2

For any given parameter values, such bi-criteria analysis considering the makespan and the cost can easily be made by the methodology developed hereby. The increase in the throughput rate can be compared with the capital investment and other relevant costs related with increasing the flexibility level in the system through an engineering economic analysis.

5 Conclusion

In this study we considered machine flexibility as an option to increase the throughput rate in two machine flowshops. As a consequence of this flexibility we assumed that each job has fixed and flexible operations. We first considered the optimization problem where the machines are assumed to be non-identical. For both the no buffer and infinite capacity buffer cases we developed constant time solution procedures. In the second part, we demonstrated the benefits of machine flexibility. We considered the reduction in the makespan corresponding to any flexibility level measured by the processing time of the flexible operations. After this analysis we conclude that most of the benefits in terms of the reduction in the makespan are attained with a relatively small level of flexibility. The cost of increasing the flexibility level need not be directly proportional with the flexibility level as assumed in earlier studies. In such a situation, the optimal flexibility level can be determined through a bicriteria analysis of the makespan and the cost as in Example 2.
This research can be extended by increasing the number of machines in the flowshop. In such a case, different assumptions can be made regarding the flexible operations. For example, it can be assumed that there is a flexible operation between any two adjacent machines. In another system, the flexible operations can be assigned to any one of the machines in the system or to a subset of machines, not necessarily consisting of adjacent machines. Determination of complexities of these alternatives are open research questions. Furthermore, considering machine flexibility in different production settings such as job shops or open shops and considering different objective functions other than the makespan will also contribute to the literature.

Acknowledgments. The author thanks to anonymous referee whose constructive comments improved the quality of the paper significantly. The author acknowledges the support of the Scientific and Technical Research Council of Turkey under grant #110M489.

References


