


Paper Type: Research Paper



An Empirical Analysis of Exact Algorithms for Solving Non-Preemptive Flow Shop Scheduling Problem

Md. Kawsar Ahmed Asif^{1,*} , Shahriar Tanvir Alam², Sohana Jahan¹, Md. Rajib Arefin¹¹ Department of Mathematics, University of Dhaka, Dhaka-1000, Dhaka, Bangladesh; ahmedasif.math.du@gmail.com; sjahan.mat@du.ac.bd; arefin.math@gmail.com.² Department of Industrial and Production Engineering, Military Institute of Science and Technology, Dhaka-1216, Dhaka, Bangladesh; tanvir.shahriar.tro@gmail.com.

Citation:

Asif, Md. K. A., Alam, Sh. T., Jahan, S., & Arefin, Md. R. (2022). An empirical analysis of exact algorithms for solving non-preemptive flow shop scheduling problem. *International journal of research in industrial engineering*, 11(3), 306-321.

Received: 03/07/2022

Reviewed: 05/08/2022

Revised: 25/08/2022

Accepted: 15/09/2022

Abstract

Sequencing and scheduling are the forms of decision-making approach that play a vital role in the automation and services industries. Efficient scheduling can help the industries to achieve the full potential of their supply chains. Conversely, inefficient scheduling causes additional idle time for machines and reduces productivity, which may escalate the product price. This study aims to find the most effective algorithm for solving sequencing and scheduling problems in a non-preemptive flow shop environment where the objective functions are to reduce the total elapsed time and idle time. In this research, four prominent exact algorithms are considered and examined their efficiency by calculating the 'total completion time' and their goodness. In order to demonstrate the comparative analysis, numerical examples are illustrated. A Gantt chart is additionally conducted to exhibit the efficiency of these algorithms graphically. Eventually, a feasible outcome for each condition has been obtained by analyzing these four algorithms, which leads to getting a competent, time and cost-efficient algorithm.

Keywords: Exact algorithm, Flow shop, Makespan, Optimal sequence, Scheduling.

1 | Introduction



Licensee

International Journal of Research in Industrial Engineering. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).

Job sequencing and scheduling problems are one of the elemental and essential applications of operations research [1], [2]. Scheduling problems frequently require the combination of several elements and it is an exceptionally perplexing one because of different limitations, for example, routing, and assignment [3]. However, significant assumptions that exist in real-world scheduling problem are constantly overlooked [4]. Job shop scheduling is an optimization process in which jobs are assigned to handle particular complex sequence [5]. Furthermore, it is an optimization problem in which a collection of n jobs should be processed through a collection of m specific machines [6]. Each job contains a particular set of operations, that needs to be processed in keeping with a given order. When we have m number of machines and n number of jobs in production scheduling, then there are $n!^m$ number of possible sequences. However, the most optimal sequence minimizes the



Corresponding Author: ahmedasif.math.du@gmail.com

<http://dx.doi.org/10.22105/riej.2022.350120.1324>

total elapsed time (makespan) as well as the idle time. Consequently, the sequencing problems are concerned with the suitable choice of a sequence of jobs to be given a finite number of machines [7].

The integral elements in sequencing and scheduling models are resources (machines) and tasks (jobs). Here we are concerned to work with a static system in which the set of tasks available for scheduling does not change over time. We presume that each machine executes one job simultaneously, and each job is processed by a single machine with no pre-emption. That is, we are considering the non-preemptive flow-shop scheduling problem.

It is worth mentioning here that the terms Sequencing and Scheduling are both associated with the job shop (machine order varies) and flow shop (machine order is fixed) process. Sequencing is the serial, where the jobs (tasks) are executed through the machines (resources). On the contrary, scheduling is the process of allocating machines to accomplish a set of jobs over a timeframe. Several notations that we are going to use throughout this paper are given in *Table 1*.

Table 1. Mathematical notations used.

Notation	Description
J	Set of jobs, $J = \{1, 2, 3, \dots, n\}$, assigned by j
M	Set of machines, $M = \{1, 2, 3, \dots, m\}$, assigned by i
n	No. of jobs to be processed
m	No. of machines in a shop
P_{ij}	Job (j) processing time on machine (i)
C_{\max}	Maximum completion time or makespan
UB	Upper bound at the beginning
LB	Lower Bound (LB) of the makespan C_{\max}
PS	Partial sequence
NPS	Not part of the partial sequence PS

Review work with several existing exact algorithms is conducted here. Although the existing algorithms are not recent, the selected algorithms work pretty well in minimizing the makespan and the idle time in a flow shop environment. The minimum makespan and idle time are evaluated by applying the four prominent exact algorithms in this manuscript. Furthermore, the goodness of these existing algorithms is measured by calculating the Lower Bound (LB) which provides a clear idea of their applicability.

This paper will discuss several exact algorithms to solve flow shop sequencing and scheduling problem where the objective is to minimize the total elapsed time and the idle time. The remaining portion of this article is presented as follows: in the next section, the previous works have been summarized on the flow shop scheduling problem conducted by other authors. Section 3 represents a generic flow shop problem along with the mathematical statements. In the following section, the exact algorithm to solve the sequencing and scheduling problem has been discussed. An empirical comparison between these algorithms is documented in Section 5. Investigation of big data has been introduced as an advanced insightful innovation, including the considerable scope and complex applications [8]. A real-life problem has been adopted, where data is collected from a local Ready-Made Garments (RMG) manufacturing company. Finally, Section 6 represents the findings of the study and Section 7 shows the conclusion and future recommendations.

2 | Literature Review

Several studies on job sequencing and scheduling have been conducted so far. According to an analysis of relevant literature, most of the exact and heuristic algorithms have been developed for makespan minimization in a flow shop environment over the past fifty years. The earliest algorithm known as Johnson [9] considered the two-machine flow shop scheduling problem to minimize the total elapsed time [10].

After that, the researchers developed different exact, heuristic, and meta-heuristic algorithms to minimize the total elapsed time for 'm' machine problems.

Palmer [11] utilized 'a single iteration' technique to minimize the total elapsed time. The sequencing was conducted depending on the 'Slope Index (SI)' estimation, which is then arranged in the decreasing order to get the optimal sequence by Palmer's heuristic technique. The objective was to prioritize the jobs so that jobs with processing times that increase from machine to machine can receive higher priority [11]. This approach can be used to solve m -machine and n -job flow shop scheduling problems. Ignall and Schrage [12] developed Branch and Bound (B&B) algorithms for the Permutation Flow Shop Problem (PFSP) with the objective of makespan minimization.

Campbell et al. [13] developed a makespan minimization heuristic for scheduling problems in a flow shop environment that augments Johnson's rule. It uses several iterations to arrive at the eventual solution, unlike the modified Johnson method [13]. This method is known as 'CDS heuristic' and is a widely used heuristic. A variety of distinct sequences are constructed here, among which the optimum sequence is the one that has the least makespan. Furthermore, Johnson's rule is followed at each iteration. Gupta [14] proposed a similar heuristic to Palmer's SI method. He described the SI differently by considering some interesting cases regarding the optimality of Johnson's algorithm for a three-machine scheduling problem [14]. Dannenbring [15] introduced the Rapid Access (RA) procedure which incorporates the benefits of both Palmer's SI method and the CDS method. Its objective is to generate a suitable solution as rapidly as feasible. Rather than solving $m - 1$ fictitious two-machine problems, it uses Johnson's rule to solve only one fictitious problem, with processing times specified by a waiting strategy [15].

Muhammad et al. [16] designed a framework consisting of the sum of individual job processing times. According to this heuristic approach, jobs with a higher total processing time across all machines must be prioritized over the jobs with lower total processing times. This approach does not convert the initial m -machine scheduling problem into a two-machine fictitious problem. The algorithm constructs the eventual sequence, incorporating a new job at each stage and determining the optimal partial sequence [16]. They used this idea and proposed the NEH heuristics algorithm, which minimizes the total elapsed time.

Kusiak [17] implemented Johnson's method more efficiently for solving n -job & two-machine flow-shop scheduling problems. The suggested method is especially effective for scheduling problems involving a large number of jobs n .

Rajendran and Chaudhuri [18] devised a heuristic approach for the flow shop scheduling problem to minimize the elapsed time, the flow time, the idle time, and many other objectives. The initial sequence was derived from the CDS algorithm for this improved heuristic. The heuristic choice correlation is suggested and utilized to minimize the search for feasible improvements in various objectives. In scheduling, it appears that decision-making objectives are widespread [18].

Modrak et al. [19] compared various heuristic algorithms depending on the makespan. Palmer's SI method, CDS method, Gupta's algorithm, NEH algorithm, and MOD algorithm are among the heuristics that have been selected. Palmer used a single iteration to achieve the feasible solution, whereas the NEH approach utilized the maximum number of iterations. Palmer's method was relatively quick; however, it was inaccurate, and the solutions acquired from this method do not correspond to the optimal solution obtained by the other heuristics algorithms [19].

Rao et al. [20] demonstrated a deviation procedure to solve the flow shop scheduling problem. The time deviation method is a modified heuristic strategy for determining the optimum job sequence and the minimum makespan [20]. This heuristic method described the processing of n jobs on 2 machines, n jobs on 3 machines, and n jobs on m machines.

Gupta et al. [21] suggested an approach for generating the ideal job sequence for an n -job & m -machine scheduling problem with the minimum number of iterations. This recent technique is referred to as the SAI method. The method adopted for solving a huge variety of scheduling problems is the most effortless. It takes the fewest iterations to arrive at the best job sequence [22]. The SAI method finds the optimal job sequence by using mathematical reasoning.

3 | Representation of Flow Shop Problem

A mathematical statement of the flow shop problem can be made as follows:

Let $J = \{J_1, J_2, \dots, J_n\}$ and $M = \{M_1, M_2, \dots, M_m\}$ be two finite sets. On account of the industrial origins of the problem, the J_j are called *jobs* and the M_i are called *machines*. In a flow shop, processing all jobs require machines in the identical order and every job is done by every machine exactly once. An instance of a 3-job & 2-machine flow shop scheduling problem is illustrated in *Table 2*.

Table 2. A flow shop scheduling problem with 3 jobs and 2 machines.

P _{ij}	Machine (i)	
Job (j)	M ₁	M ₂
J ₁	7	11
J ₂	13	3
J ₃	5	8

Without changing order, each job must first go to M_1 and then M_2 . That is, the order of machines to process these jobs is the same $M_1 \rightarrow M_2$.

4 | Methodology

A flow shop problem assumption is considered during sequencing the jobs and is illustrated in *Table 3*.

Table 3. Assumptions in sequencing problems.

Hypotheses about jobs (job-related assumptions)	At any time, no more than one machine can process a job. Every job must go through the machines in a specific order. Depending on the machine order, a job is processed as early as possible. There are no priorities; therefore, all jobs are equally essential.
Hypotheses about machines (machine-related assumptions)	Each machine can process a maximum of one job at a time. After starting an operation, it has to be completed. From each category, there will be only one machine. Any machine does not process a job more than once.
Hypotheses about processing times (time-related assumptions)	Each job's processing time on each machine is independent of the sequence in which the jobs are processed. Each job on each machine has a predetermined and integer processing time. The amount of time required to transfer a job from one machine to another is nearly negligible. Transportation times and setup times are incorporated in the processing times.

4.1 | Johnson's Algorithm

Johnson's algorithm is a technique for scheduling tasks in two workstations. Its principal objective is to locate an optimum sequence of tasks to reduce 'makespan'. The technique minimizes the makespan and idle time in the case of two workstations. Moreover, the method determines the smallest makespan in the case of three or more workstations if additional constraints are met [9].

4.1.1 | Johnson's algorithm for n jobs and 2 machines

Assume that jobs must be processed through machine 1 first, then through machine 2. The minimum makespan (or total elapsed time) is determined using Johnson's algorithm.

The steps involved in Johnson's method are as follows:

Let p_{ij} = Processing time for job j on machine i .

Find the job with the least p_{ij} .

If $i = 1$ (machine 1), the job turns into the first job.

If $i = 2$ (machine 2), the job turns into the final job.

Eliminate assigned job from the list and repeat (break ties arbitrarily).

Calculate the total elapsed time as per the sequence determined and calculate the idle time associated with machines.

4.1.2 | Johnson's algorithm for n jobs and 3 machines

Suppose now we have three machines namely, A, B, C and A_i, B_i, C_i are the processing times in machine A, B and C respectively. Johnson's Algorithm may be expanded to resolve the exceptional cases when at least one of the following conditions is met:

Min $A_i \geq \text{Max } B_i$.

Min $C_i \geq \text{Max } B_i$.

Then replace the problem with a similar problem, considering n jobs and two imaginary machines say, G and H and corresponding processing time as G_i and H_i are defined by $G_i = A_i + B_i$ and $H_i = B_i + C_i$.

4.1.3 | Johnson's algorithm for n jobs and m machines

Suppose that expected processing times for n Jobs and m machines are represented in Table 4.

Table 4. Processing times for n jobs and m machines.							
Jobs	Machines						
	M_1	M_2	M_3	...	M_{m-1}	M_m	
1	p_{11}	p_{21}	p_{31}	...	$p_{(m-1)1}$	p_{m1}	
2	p_{12}	p_{22}	p_{32}	...	$p_{(m-1)2}$	p_{m2}	
3	p_{13}	p_{23}	p_{33}	...	$p_{(m-1)3}$	p_{m3}	
...	
n	p_{1n}	p_{2n}	p_{3n}	...	$p_{(m-1)n}$	p_{mn}	

A result for this problem is achievable if and only if any of the following cases are fulfilled:

- The least processing time on machine M_1 is greater or equal to the highest processing time among machines M_2, M_3, \dots, M_{m-1} .
- The least processing time on machine M_m is greater or equal to the highest processing time among machines M_2, M_3, \dots, M_{m-1} .

That is,

Either, $\min(p_{1j}) \geq \max\{p_{2j}, p_{3j}, \dots, p_{(m-1)j}\}$,

Or, $\min(p_{mj}) \geq \max\{p_{2j}, p_{3j}, \dots, p_{m-1j}\},$

The following steps are involved in determining the optimal sequence:

Step 1. Investigate whether it fulfils the previously mentioned case.

Step 2. If the case is fulfilled, continue further, or else the technique fails.

Step 3. Convert the m machines problem into an identical two machines problem, say G and H , so that

$G: G_i = M_1 + M_2 + \dots + M_{m-1}$ (Excluding the time of the last machine),

$H: H_i = M_2 + M_3 + \dots + M_m$ (Excluding the time of the first machine).

Step 4. Using Johnson's method, find the optimal sequence of n jobs through two machines.

4.2 | B&B Algorithm

B&B algorithm substitute the initial problem into a group of sub-problems and can be used in minimizing the total flow time [23]. As per Ignall and Schrage [12], the initial problem is reformulated into a 'solution tree', where every node indicates a LB of the desired 'objective function' [24].

The B&B algorithm was utilized and suggested by Kim [25], where every node indicates the partial sequence of the eventual outcome. This partial sequence is termed as $|PS|$, and the group of jobs that are not part of it is termed as $|NPS|$.

Once a node gets branched, the partial sequence corresponding to the branched node receives a new job from $|NPS|$, which generates a single sequence or several partial sequences [25]. For each node generated, one LB is determined for the total elapsed time [24].

Ronconi [26] used the 'depth-first search' algorithm to select the branching node, which chooses the node in the partial sequence with the most jobs. In the event of a tie, the node with the minimum LB for the 'makespan' is chosen [26].

Ronconi's successful implementation of the node selection algorithm in the permutation flow shop with a blocking problem was the reason for its selection [24]. Fig. 1 presents the 'flow chart' of the B&B algorithm. A pseudo-code for this algorithm is shown below.

B&B Algorithm for the PFSP

Step 1. "Initialization" (setting the root node)

UB \rightarrow Initial Upper Bound ;

$|PS| = \{\emptyset\}$;

$|NPS| = \{1, 2, 3, \dots, n\}$;

Nodes = 0 number of branching nodes) ;

Step 2. "First Level Branching Step"

for Node = 1:n do ;

$|PS|_{Node} = \{j\}$;

```

|NPS|Node = |NPS|Node - |PS|Node ;
LBNode = LB for |PS|Node ;

if LBNode < UB ;

    Nodes = Nodes + 1 ;

```

```

end if ;

```

```

end for ;

```

Step 3. "Other Levels Branching Step"

```

while Nodes > 0 do ;

```

```

    Nodes = 0 ;

```

```

    for Node = 1:Nodes do ;

```

```

        for j ∈ |NPS|Node do ;

```

```

            |PS|Node = |PS|Node + {j} ;

```

```

            |NPS|Node = |NPS|Node - {j} ;

```

```

            LBNode = LB for |PS|Node ;

```

```

            if LBNode < UB ;

```

```

                if Level < n ;

```

```

                    Nodes = Nodes + 1 ;

```

```

                else ;

```

```

                    LBNode = UB ;

```

```

                end if ;

```

```

            end if ;

```

```

        end for ;

```

```

    end for ;

```

```

end while ;

```

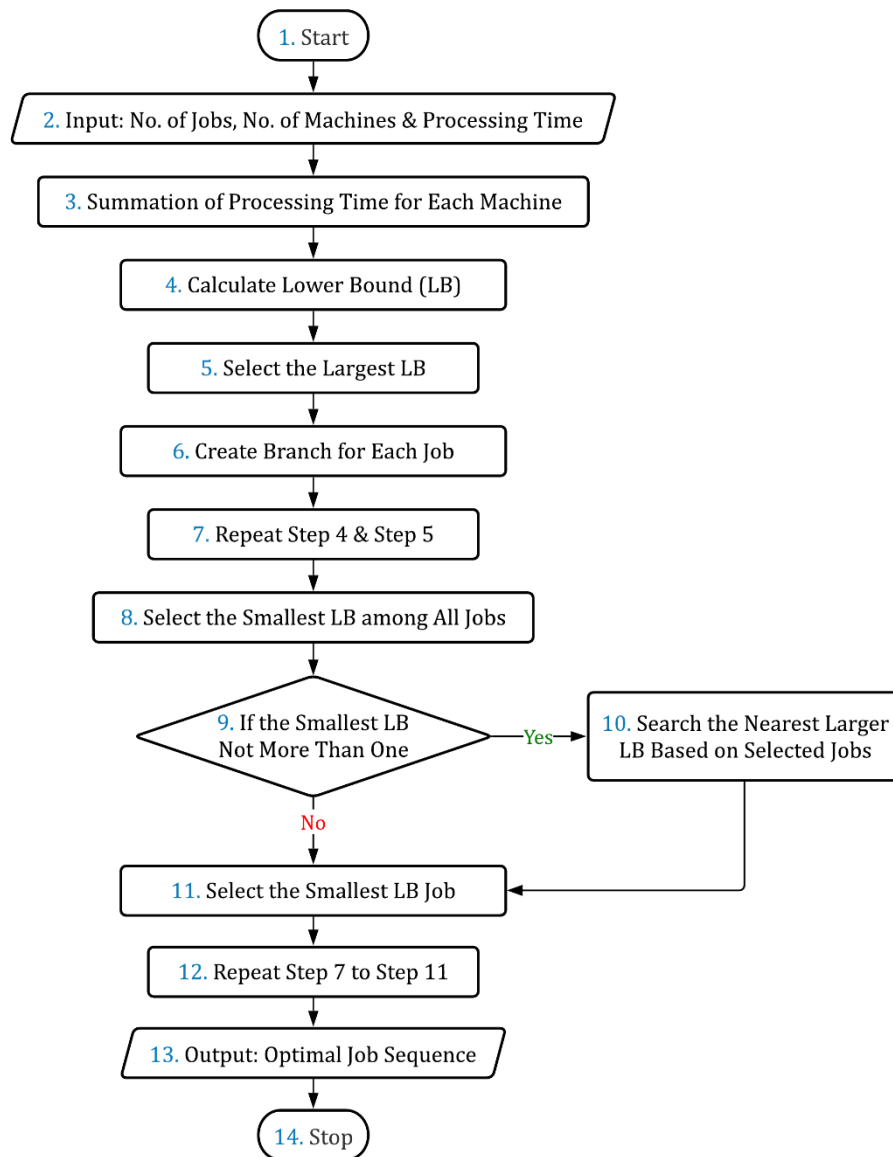



Fig. 1. Flow chart of B&B algorithm for the General Flow Shop Problem (GFSP).

4.3 | Kusiak's Algorithm

Andrew Kusiak introduces this algorithm, which is a more efficient implementation of Johnson's scheduling technique. The steps of the algorithm are as follows:

Step 1. Set $k = 1, l = n$.

Step 2. For each job, store the least processing time and the corresponding machine number.

Step 3. Arrange the resultant list with job number, time, or machine number in ascending order of processing time.

Step 4. For every entry in the arranged list:

- If machine number = 1,
 - (i) Set the associated job number at the place of k ,
 - (ii) $k = k + 1$.
- Else
 - (i) Set the associated job number at the place of l ,
 - (ii) $l = l - 1$.

End if

Step 5. Terminate when the complete list of jobs is filled.

4.4 | SAI Algorithm

The 'step-wise' iterative technique of SAI algorithm to find out the optimal sequence for 'n jobs' on 'm machines' are explained below:

Step 1. The processing time of n jobs (1 to n) on m machines (1 to m) is illustrated in Table 5.

Table 5. Processing time of n jobs on m machines.

Machines \ Jobs	1	2	3	...	j	...	n
1	p_{11}	p_{12}	p_{13}	...	p_{1j}	...	p_{1n}
2	p_{21}	p_{22}	p_{23}	...	p_{2j}	...	p_{2n}
3	p_{31}	p_{32}	p_{33}	...	p_{3j}	...	p_{3n}
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
i	p_{i1}	p_{i2}	p_{i3}	...	p_{ij}	...	p_{in}
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
m	p_{m1}	p_{m2}	p_{m3}	...	p_{mj}	...	p_{mn}

Step 2. Investigate the jobs and choose the least job processing time among all n jobs ($j = 1$ to n) for each machine and later checked it with $-$ sign. Suppose that the 'minimum processing time' is appeared at j^{th} job on i^{th} machine then mathematically, this can be illustrated as;

$$\text{Min}_j \{p_{i1}, p_{i2}, \dots, p_{ij}, \dots, p_{in}\} = p_{ij}.$$

Step 3. In a similar manner, choose the least processing time among all m machines ($i = 1$ to m) for each job and later checked it with $+$ sign. Suppose that the 'minimum processing time' is appeared at i^{th} machine for the j^{th} job then mathematically, this can be illustrated as;

$$\text{Min}_i \{p_{1j}, p_{2j}, \dots, p_{ij}, \dots, p_{mj}\} = p_{ij}.$$

Step 4. Investigate the rows and columns of the table once again, chose the cell with \oplus sign. Suppose that the \oplus sign has appeared at the cell that relates to the i^{th} machine and j^{th} job. Then the j^{th} job is eliminated from the table and is inserted in the optimal job sequence.

Step 5. Step 1 to 4 are continued until every one of the jobs are set in the optimal job sequence. There might be a circumstance when a tie has happened;

- I. When \oplus appears at more than one cell, the job with 'least processing time' is chosen and is set in the optimal job sequence.
- II. When \oplus appears at more than one cell, and the processing time for the assigned jobs is identical. After neglecting the other higher-order machines, the job that will process on the lower-order positional machine is chosen.

Step 6. Finally, we compute the idle time and makespan of machines.

5 | Numerical Experiments

A practical problem has been considered to minimize the makespan of the n -jobs and m -machines flow shop problem. The numerical data is illustrated in Table 6, gathered from a RMG manufacturing

company. The manufacturer has to perform two operations: cutting and sewing on various jobs (products). The required time to execute these operations for each job is specified.

Table 6. 3-jobs and 3-machines flow shop problem.

Jobs	Cutting (Minutes)	Sewing (Minutes)	Packing (Minutes)
J1	3	4	7
J2	8	5	9
J3	7	1	5
Total processing time	18	10	21

Table 7 gives all the possible sequence and the corresponding makespan and idle time.

Table 7. All possible sequences (flow shop scheduling for three machines).

Sequence	Makespan (Mins)	Idle Time (Mins)
J1-J2-J3	30	41
J1-J3-J2	32	47
J2-J1-J3	34	53
J2-J3-J1	34	53
J3-J1-J2	32	47
J3-J2-J1	36	59

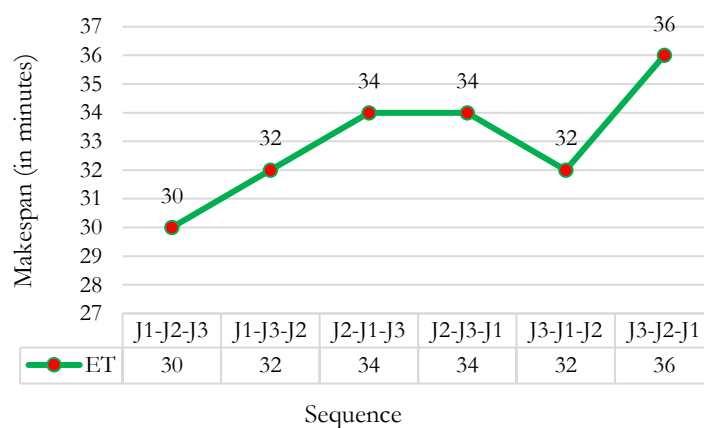


Fig. 2. Sequences and makespan.

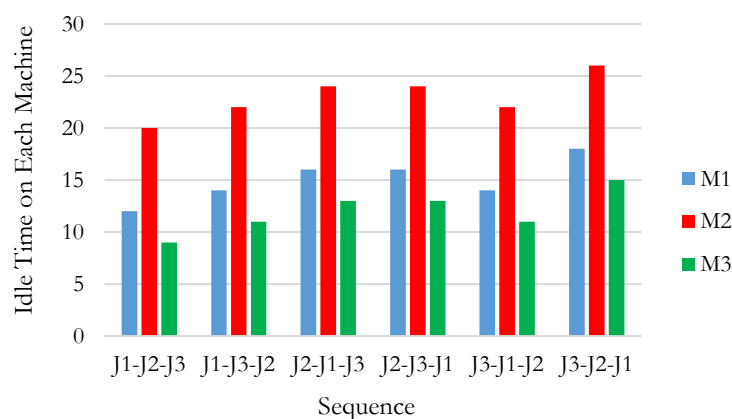


Fig. 3. Sequences and idle time.

A 4-jobs and 4-machines flow shop problem is illustrated in Table 8.

Table 8. 4-jobs and 4-machines flow shop problem.

Jobs	Cutting (Mins)	Sewing (Mins)	Pressing (Mins)	Packing (Mins)
J1	6	5	3	9
J2	7	6	5	7
J3	5	4	6	8
J4	8	3	4	6
Total processing time (in minutes)	26	18	18	30

All possible sequences and the corresponding elapsed time and idle time is tabulated in Table 9.

Table 9. All possible sequences (flow shop scheduling for four machines).

Sequence	Makespan (Minutes)	Idle Time (Minutes)	Sequence	Makespan (Minutes)	Idle Time (Minutes)
J1-J2-J3-J4	45	88	J3-J1-J2-J4	45	88
J1-J2-J4-J3	45	88	J3-J1-J4-J2	45	88
J1-J3-J2-J4	44	84	J3-J2-J1-J4	45	88
J1-J3-J4-J2	44	84	J3-J2-J4-J1	45	88
J1-J4-J2-J3	47	96	J3-J4-J1-J2	45	88
J1-J4-J3-J2	44	84	J3-J4-J2-J1	47	96
J2-J1-J3-J4	48	100	J4-J1-J2-J3	47	96
J2-J1-J4-J3	48	100	J4-J1-J3-J2	46	92
J2-J3-J1-J4	48	100	J4-J2-J1-J3	50	108
J2-J3-J4-J1	48	100	J4-J2-J3-J1	50	108
J2-J4-J1-J3	48	100	J4-J3-J1-J2	47	96
J2-J4-J3-J1	48	100	J4-J3-J2-J1	47	96

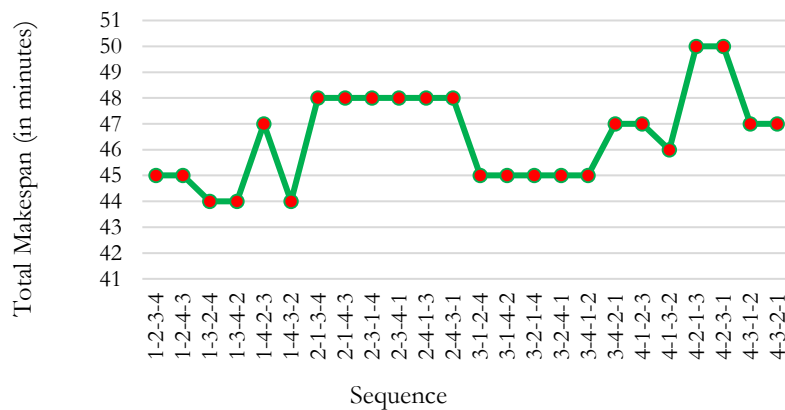


Fig. 4. Sequences and total makespan.

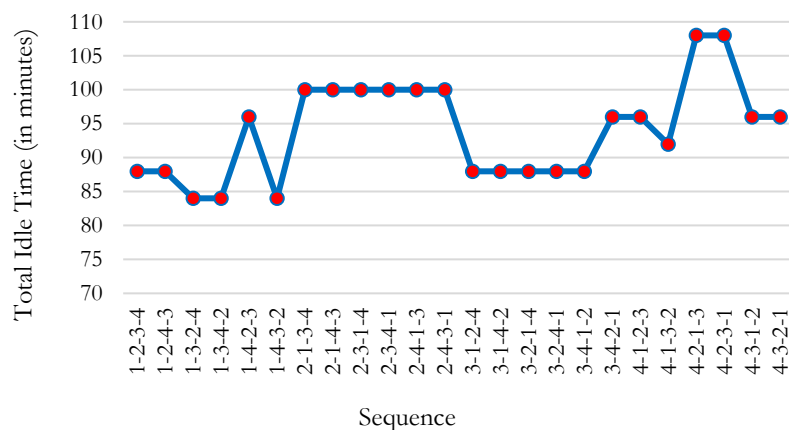


Fig. 5. Sequences and total idle time.

In Figs. 4 and 5, we have shown how the total idle time and total elapsed time vary with sequences' choice. In Table 10, we will apply all four algorithms to determine an optimum or nearly optimum sequence.

Table 10. Consider following 5-jobs and 3-machines flow shop problem.

Jobs	Cutting (Minutes)	Sewing (Minutes)	Packing (Minutes)
J1	3	4	7
J2	8	5	9
J3	7	1	5
J4	5	2	6
J5	4	3	10
Total processing time	27	15	37

The minimum makespan obtained by the four algorithms are as follows,

1. Johnson's Algorithm=44.
2. B&B Algorithm=44.
3. Kusiak's Algorithm=44.
4. SAI Algorithm=46.

Goodness of an algorithm measures the error percentage of the algorithm shown in Eq. (1).

$$\text{Goodness of Exact Algorithms} = \frac{\text{Makespan} - \text{LB}}{\text{LB}} \times 100\%. \quad (1)$$

In order to find out how good the algorithm is, we have to determine the LB for the makespan based on each of the machines. Now,

$$\text{LB based on M1, } LB_{M1} = \sum p_{1j} + \min\{\sum(p_{2j}, p_{3j})\} = 27 + (1 + 5) = 33,$$

$$\text{LB based on M2, } LB_{M2} = \min(p_{1j}) + \sum p_{2j} + \min(p_{3j}) = 3 + 15 + 5 = 23,$$

$$\text{LB based on M3, } LB_{M3} = \min\{\sum(p_{1j}, p_{2j})\} + \sum p_{3j} = 3 + 4 + 37 = 44.$$

Now out of the 3 LBs the maximum one is the best bound.

$$\text{LB} = \text{Max}\{LB_{M1}, LB_{M2}, LB_{M3}\} = \text{Max}\{33, 23, 44\} = 44.$$

Based on the LB, now we know that the optimum makespan cannot be less than 44. So, the makespan can only be 44 or more. The goodness measure in Table 11 shows that for this problem all the three methods except SAI give similar result and achieves the best possible sequence with 0% error. Fig. 6 illustrates the Gantt chart for optimal sequence.

Table 11. Goodness measurement of the selected exact algorithms.

Algorithm/Method	Optimal Sequence	Makespan (Minutes)	Idle Time (Minutes)	Goodness $(\frac{\text{MS}-\text{LB}}{\text{LB}} \times 100\%)$
Johnson's Algorithm	J4-J1-J5-J2-J3	44	53	$\frac{44-44}{44} \times 100\% \approx 0\%$
B&B Algorithm	J1-J3-J4-J5-J2	44	53	$\frac{44-44}{44} \times 100\% \approx 0\%$
Kusiak's Algorithm	J1-J4-J5-J2-J3	44	53	$\frac{44-44}{44} \times 100\% \approx 0\%$
SAI Method	J3-J4-J1-J5-J2	46	59	$\frac{46-44}{44} \times 100\% \approx 4.54\%$

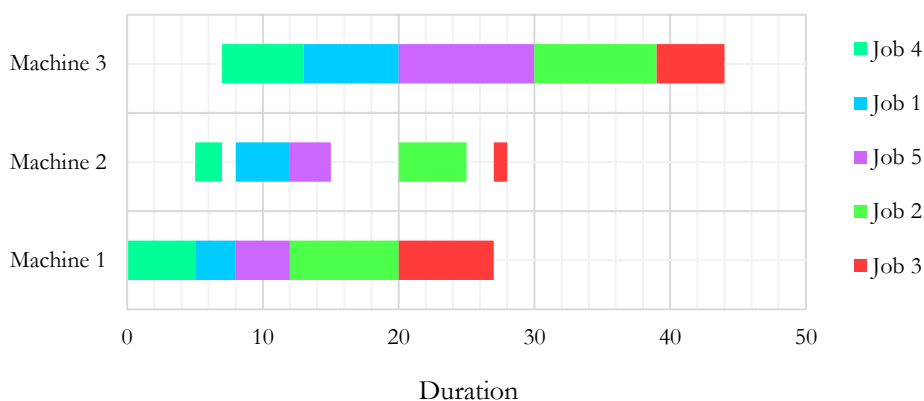


Fig. 6. Gantt chart for optimal sequence (4, 1, 5, 2, 3).

We further considered several problems and applied the algorithms to find the optimal sequence. In Table 12, we have documented some results when the system has identical number of jobs and machines, which can also be observed in Fig. 7.

Table 12. Results obtained by exact methods for identical number of jobs and machines.

Methods	Problem Size	Optimal Sequence	Elapsed Time (Minutes)
Johnson	(2×2)	$1 \rightarrow 2$	16
	(3×3)	$1 \rightarrow 2 \rightarrow 3$	30
	(4×4)	$1 \rightarrow 3 \rightarrow 2 \rightarrow 4$	47
	(5×5)	$1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4$	58
Branch & Bound	(2×2)	$1 \rightarrow 2$	16
	(3×3)	$1 \rightarrow 2 \rightarrow 3$	30
	(4×4)	$1 \rightarrow 2 \rightarrow 4 \rightarrow 3$	47
	(5×5)	$5 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 2$	62
Kusiak	(2×2)	$1 \rightarrow 2$	16
	(3×3)	$1 \rightarrow 2 \rightarrow 3$	30
	(4×4)	$1 \rightarrow 3 \rightarrow 2 \rightarrow 4$	47
	(5×5)	$1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4$	58
SAI	(2×2)	$1 \rightarrow 2$	16
	(3×3)	$3 \rightarrow 1 \rightarrow 2$	32
	(4×4)	$1 \rightarrow 4 \rightarrow 3 \rightarrow 2$	47
	(5×5)	$5 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 2$	62

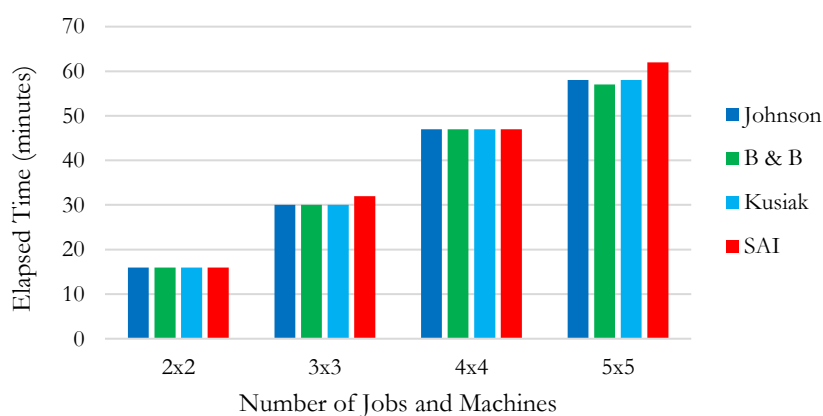


Fig. 7. Comparison diagram of the methods for identical number of jobs and machines.

6 | Findings and Complexity Analysis

The following conclusions are made based on the objective of makespan minimization in flow shop scheduling problems:

- I. Johnson's algorithm gives the best optimal sequences with minimum elapsed time for two machines and n jobs scheduling problem. In the case of three or more machines, if the condition of using Johnson's algorithm does not hold, then Branch & Bound gives the better solution.
- II. One of the advantages of using the SAI algorithm over Johnson's method is that no conversion of machines is required in the SAI algorithm in the case of three or more machines.
- III. Kusiak's algorithm works likewise to Johnson's algorithm. Due to frequent ties, both Johnson's and Kusiak's algorithms produce more than one sequence.
- IV. The Kusiak algorithm is specifically helpful while dealing the scheduling problems involving a large number of jobs.
- V. Though Branch & Bound gives better solutions in some cases but as the number of jobs and number of machines increases, the complexity of the B&B algorithm increases.

For a n job & m machine flow shop problem, number of sequences and computational complexity or time complexity evaluated by the exact algorithms are given in *Table 13*.

Table 13. Computational complexity of the algorithms for n jobs and m machine.

Exact Algorithms	Number of Sequence	Computational Complexity
Johnson's Algorithm	1	$O(n \log(n))$
B&B Algorithm	$n!$ (at most)	$O(n^2)$
Kusiak's Algorithm	1	$O(n \log(n))$
SAI Algorithm	1	$O(n + m \log(n))$

7 | Conclusions

In this paper, several exact algorithms have been analyzed to solve the flow shop problem. The primary objective of this study is to identify the most efficient algorithm for solving sequencing and scheduling problems in a flow shop environment. The objective is to minimize the total elapsed time and the idle time. Four prominent exact algorithms are taken, including Johnson's Algorithm, B&B Algorithm, Kusiak's Algorithm, and SAI Algorithm, and examined their efficiency by calculating the total completion time and the goodness. Numerical results show that Johnson's Algorithm gives the best result in most cases, but when it fails, the B&B algorithm gives a better result than others. A comparison between a stochastic methods with the exact algorithms can be performed in future work. A modified and hybrid exact algorithm may also be proposed later. Unique 'tight LBs and meta-heuristic calculations, like the Genetic Algorithm (GA), and Tabu Search (TS) can be combined for further research.

Abbreviation and Nomenclature

B&B	B&B	GFSP	GFSP
UB	Initial Upper Bound	LB	LB
ET	Elapsed Time	IT	Idle Time

Conflicts of Interest

All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We certify that the submission is original work and is not under review at any other publication.

- [1] Defersha, F. M., & Rooyani, D. (2020). An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time. *Computers & industrial engineering*, 147, 106605. <https://doi.org/10.1016/j.cie.2020.106605>
- [2] Metaxiotis, K. S., Psarras, J. E., & Ergazakis, K. A. (2003). Production scheduling in ERP systems: an AI-based approach to face the gap. *Business process management journal*, 9(2), 221-247. <https://doi.org/10.1108/14637150310468416>
- [3] Samadpour, E., Ghousi, R., Makui, A., & Heydari, M. (2022). Routing and scheduling for a home health care problem considering health workers' skills. *Journal of applied research on industrial engineering*, 9(3), 249-263. <https://doi.org/10.22105/jarie.2022.314240.1397>
- [4] Ruhbakhsh, R., & Adibi, M. A. (In Press). Presenting a model for solving lot-streaming hybrid flow shop scheduling problem by considering independent setup time and transportation time. *Journal of decisions and operations research*. DOI: [10.22105/dmor.2022.296154.1450](https://doi.org/10.22105/dmor.2022.296154.1450)
- [5] Sarfaraj, N., Lingkon, M. L., & Zahan, N. (2021). Applying flexible job shop scheduling in patients management to optimize processing time in hospitals. *International journal of research in industrial engineering*, 10(1), 46-55. <https://doi.org/10.22105/riej.2021.260145.1158>
- [6] Jayasankari, S. (2021). An efficient flow shop scheduling problem with makespan objective. *Turkish journal of computer and mathematics education (TURCOMAT)*, 12(4), 461-466. <https://doi.org/10.17762/turcomat.v12i4.527>
- [7] Michael, L. P. (2018). *Scheduling: theory, algorithms, and systems*. Springer.
- [8] Fakheri, S. (2022). A comprehensive review of big data applications. *Big data and computing visions*, 2(1), 9-17. <https://doi.org/10.22105/bdcv.2022.325256.1041>
- [9] Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1), 61-68. <https://doi.org/10.1002/nav.3800010110>
- [10] Gill, S., & Kumar, R. (2012). Literature review for scheduling problems. *International journal of latest research in science and technology*, 1(1), 98-100. https://www.mnkjournals.com/journal/ijlrst/pdf/Volume_1_1_2012/10018.pdf
- [11] Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. *Journal of the operational research society*, 16(1), 101-107. <https://doi.org/10.1057/jors.1965.8>
- [12] Ignall, E., & Schrage, L. (1965). Application of the branch and bound technique to some flow-shop scheduling problems. *Operations research*, 13(3), 400-412. <https://doi.org/10.1287/opre.13.3.400>
- [13] Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management science*, 16(10), B-630. <https://doi.org/10.1287/mnsc.16.10.B630>
- [14] Gupta, J. N. (1976). A heuristic algorithm for the flowshop scheduling problem. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, 10(V2), 63-73. http://www.numdam.org/item/RO_1976__10_2_63_0.pdf
- [15] Dannenbring, D. G. (1977). An evaluation of flow shop sequencing heuristics. *Management science*, 23(11), 1174-1182. <https://doi.org/10.1287/mnsc.23.11.1174>
- [16] Muhammad, N., Emory, E. J., & Inyong, H. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *The international journal of management science*, 11(1), 91-95. [https://doi.org/10.1016/0305-0483\(83\)90088-9](https://doi.org/10.1016/0305-0483(83)90088-9)
- [17] Kusiak, A. (1986). Efficient implementation of Johnson's scheduling algorithm. *IIE transactions*, 18(2), 215-216. <https://doi.org/10.1080/07408178608975349>
- [18] Rajendran, C., & Chaudhuri, D. (1992). An efficient heuristic approach to the scheduling of jobs in a flowshop. *European journal of operational research*, 61(3), 318-325. [https://doi.org/10.1016/0377-2217\(92\)90361-C](https://doi.org/10.1016/0377-2217(92)90361-C)
- [19] Modrak, V., Semanco, P., & Kulpa, W. (2013). Performance measurement of selected heuristic algorithms for solving scheduling problems. *2013 IEEE 11th international symposium on applied machine intelligence and informatics (SAMI)* (pp. 205-209). IEEE. <https://doi.org/10.1109/SAMI.2013.6480977>
- [20] Rao, N. N., Raju, O. N., & Babu, I. R. (2013). Modified heuristic time deviation technique for job sequencing and computation of minimum total elapsed time. *International journal of computer science & information technology*, 5(3), 67. <http://dx.doi.org/10.5121/ijcsit.2013.5305>

- [21] Gupta, S., Ali, I., & Ahmed, A. (2016). A new algorithm for solving job shop sequencing problem. *International journal for computer science engineering*, 5(2), 93-100. <http://www.ijcse.net/docs/IJCSE16-05-02-019.pdf>
- [22] Geetha, M. (2019). A different technique for solving sequencing problem. *International journal of analytical and experimental modal analysis*, 11(11), 2584–2587. <http://www.ijaema.com/gallery/290-november-2941.pdf>
- [23] Kouider, A., & Haddadène, H. A. (2021). A bi-objective branch-and-bound algorithm for the unit-time job shop scheduling: a mixed graph coloring approach. *Computers & operations research*, 132, 105319. <https://doi.org/10.1016/j.cor.2021.105319>
- [24] Takano, M. I., & Nagano, M. S. (2017). A branch-and-bound method to minimize the makespan in a permutation flow shop with blocking and setup times. *Cogent engineering*, 4(1), 1389638. <https://doi.org/10.1080/23311916.2017.1389638>
- [25] Kim, Y. D. (1995). Minimizing total tardiness in permutation flowshops. *European journal of operational research*, 85(3), 541-555. [https://doi.org/10.1016/0377-2217\(94\)00029-C](https://doi.org/10.1016/0377-2217(94)00029-C)
- [26] Ronconi, D. P. (2005). A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking. *Annals of operations research*, 138(1), 53-65. <https://doi.org/10.1007/s10479-005-2444-3>