Natural Resources for Human Health
eISSN: 2583-1194

Original Research

Received 20 May 2026
Revised 29 June 2026
Accepted 24 July 2026

Natural Resources for Human Health 2026, Vol. 6 (9s): 352–365

KEYWORDS:
Flow-shop, Scheduling, Optimization, Total Elapsed Time, Non-resumable, Unavailability, Johnson's Rule, Branch-and-Bound Method

Flow-shop Scheduling Framework for Optimizing Resource Utilization under Non-Resumable Unavailability Constraints

Laxmi Narain[1], Dev Om Singh[2*], Manoj Kumar Rana[3]

[1]Professor, Department of Mathematics, Acharya Narendra Dev College, University of Delhi, Delhi, India
[2]Assistant Professor, Department of Mathematics, Acharya Narendra Dev College, University of Delhi, Delhi, India
[3]Assistant Professor, Department of Mathematics, Ramjas College, University of Delhi, Delhi, India
*Corresponding Author

ABSTRACT: This study develops a flow-shop scheduling framework for optimizing resource utilization under non-resumable unavailability constraints. These non-resumable interruptions are very common in steel manufacturing industry, chemical processing industry and pharmaceutical industry. Under non-resumable unavailability constraint, if a job is interrupted in mid-processing, then entire work done on the job will be lost and when the machine will be available again then the job has to be started from the scratch. This non resumable interruption can be resolved by two ways either by postponing job's start time so that it starts processing at the end of unavailable interval on the machine or reschedule jobs so that job complete the processing before the beginning of unavailable interval on the machine. Both corrective strategies will affect the completion time of jobs executing on the current machine and latter indirectly to all downstream machines. In this paper, we have considered n-jobs, 2-machines flow-shop problem having a single non-resumable unavailability constraint on each machine. We have developed an algorithm to obtain an optimal sequence which minimize total elapsed time under non-resumable unavailability constraint. This algorithm is implemented in two steps. At the first step, we find a sequence by applying Modified Johnson's method to take into consideration the modified completion times that are induced by the non-resumable interruptions. At the second step, we apply Branch-and-Bound method that systematically enumerates the solution space with node pruning to assure global optimality of the sequence. The developed algorithm is illustrated through examples.

1. INTRODUCTION

Minimizing total elapsed time or makespan to complete all the jobs on the last machine is a major and persistent problem in scheduling theory. The Johnson [6] paper provided the foundation for the scientific study of classical flow-shop scheduling and was the major seminal paper on which later work in the field has rested. He considered n-jobs, 2-machines flow-shop problem and the objective was to minimize total elapsed time. The beautiful and efficient Johnson's rule for obtaining optimal sequence and its mathematical proof became one of the most successful theorems in deterministic scheduling theory. It provided the context for further generalizations and extensions. Ignall and Schrage [5] developed a new method in which they used lower bounds evaluated on partial schedules to prune the search tree and ease the computation compared to the full enumeration for makespan problem. Their approach laid the groundwork for the Branch-and-Bound paradigm as the most common exact method employed in flow-shop scheduling and their bounding procedures (which were partial completion times and idle machines lower bounds) were further developed and strengthened by many later researchers. Narain and Bagga [10] study n-jobs, 2-machine flow-shop problems under no-idle constraints. Under no-idle constraint, a machine continuously works without

break. They develop algorithm using Branch-and-Bound technique to obtain optimal sequence to minimize makespan under no-idle constraints. Narain and Bagga [9] consider three-machine problem to minimize makespan with zero idle time on machines. Allahverdi [2] studied the flow-shop problem under no-wait environment.

Research on the classical flow-shop scheduling problem make explicit assumption that machines will always be available and that processing will never be forced to be interrupted. This assumption that each machine is always available during the entire planning horizon is clearly woefully flawed in production settings where machines may go on a periodic "preventive maintenance" break or thermal cooling down for temperature regulations, and required calibration shutoff. The most systematic planned machine unavailability is preventive maintenance. Machines in constant use need to be regularly pulled out of operation for inspection, lubrication, part replacement and calibration to comply with the manufacturer's instructions and in keeping with reliability engineering best practice.

Recently, the assumption of permanent availability has been relaxed to allow for operational factors involved in scheduling under unavailability of machines into a unique and growing subfield of scheduling. In this subfield resumable and non-resumable interruptions has been shown to be relevant. A fundamental difference between resumable and non-resumable interruptions in the field of machine unavailability constraints is its implications for the mathematical structure of the scheduling problem as well as the algorithm solution methods for the scheduling problem. In the resumable (preemptive) unavailability model, upon a machine's unavailability period if a job is active on the machine, the job will be considered interrupted at the time of the unavailability and is resumed when the machine becomes available. This results in no additional waste because this interruption is not responsible for gaining any extra processing time for the job, just displacing the schedule, so the job's total actual processing time equals its nominal processing time.

Authors [3, 4, 7, 8, 11–20] studied the flow-shop problems under unavailability constraints. Cheng et al. [3] looked at planning work when there are delays and other real-life issues. They mostly studied how different delays change jobs move through a shop. Hidri [4] considers the challenges of incorporating removal times in multi-stage flexible flow shops and proposes an efficient two-phase optimization heuristic, designed to provide near optimal solution. Lee [7] concentrated on those instances in which a machine can be stopped irrespective of the length of time, and no jobs could be done during that time. He used mathematical scheduling models that include machine availability constraints. Li et al. [8] study flow-shop scheduling problem under machine breakdown conditions and use hybrid metaheuristic techniques to solve it. Thangaraj and Kumar [13] address a practical flow shop scheduling problem where specific machines must periodically halt for mandatory maintenance. To prevent massive bottleneck delays, they introduce an advanced NEH based heuristic algorithm that adaptively inserts deterministic downtime blocks to ensure optimal makespan and minimized total tardiness. Utama et al. [14] evaluates the mathematical interplay of no-wait and non-variability intervals. It tracks how meta-heuristics and exact mathematical formulations protect production timelines from sudden machine halting points across industrial environments. Wocker et al. [16] builds an integrated production scheduling framework to proactively secure strict time blocks for machine servicing. It evaluates how proactive maintenance placement influences performance measures like machine idle time and total flow times under strict technical unavailability windows. Yuan et al. [18] considers multi-stage shop scheduling where a flexible maintenance period is imposed on a machine. They present a novel Polynomial-Time Approximation Scheme to minimize makespan. Zhou et al. [20] provide metaheuristic algorithms for flow-shop scheduling problems with machine interruptions.

This paper considers a non-resumable (or non-preemptive) unavailability model, where if a job is interrupted mid-processing, then the entire work done on the job is lost and, when the machine is made available again, the job has to be resumed from scratch. Non-resumable machine interruptions are very common in the steel manufacturing industry, pharmaceutical industry, and chemical batch processing industry. For continuous casting machines in the steel making industry no interrupted pour can be allowed as the partially solidified steel cannot be cast further, it has to be scrapped and the process restarted from the beginning.

We have developed an algorithm to obtain an optimal sequence which minimize total elapsed time under non-resumable unavailability constraint. This algorithm is implemented in two steps. At the first step, we find a sequence by applying Modified Johnson's method to take into consideration the modified completion times that are induced by the non-resumable interruptions. At the second step, we apply Branch-and-Bound method that systematically enumerates the solution space with node pruning to assure global optimality of the determined sequence. The algorithm is illustrated through examples.

2. MATHEMATICAL FORMULATION

3.1 Assumptions

The following assumptions are made in the present study. All assumptions are mathematically substantiated and proven to be relevant and realistic in the context of the straightforward permutation flow-shop model supplemented by deterministic unavailability constraints.

- No Release Date: All jobs are available at time zero.
- No Job Split: Each job can be processed on at most one machine at a time. The assumption is that each job-machine operation is a total unit of work and that this is related to the batch processing environment that motivates the non-resumable constraint.
- Unit Machine Capacity: Each machine processes one (and at most one) job at a time. The simultaneous processing of several jobs on a single machine is not allowed.
- Deterministic Processing Times: All processing times are deterministic, finite, non-negative and known at the time of scheduling.
- Permutation Flow-shop: Processing sequence of job is same between each and every machine in the shop. This simplifies the sequencing problem to the selection of only one of the permutations to be used and is consistent for any sequential production situation where rerouting between machines is impossible, like assembly line and linear production.
- Deterministic Machine Unavailability: The unavailability of each machine is expressed by a pre-specified interval known in advance.

3.2 Notations

The following notation is adopted throughout the study and maintained consistently across all formulations, algorithms, and numerical examples.

Table: T1 Symbols and Definitions

Symbol	Definition
n	Total number of independent jobs to be scheduled
$\{J_1, J_2, \dots, J_n\}$	Set of n jobs
$\{M_1, M_2\}$	Set of 2 machines arranged in linear sequence
p_{ij}	Processing time of job J_j on machine M_i(deterministic, non-negative)
$\pi = \{\pi(1), \pi(2), \dots, \pi(n)\}$	Permutation of n jobs defining the processing sequence
$\pi(k)$	Job occupying position k in the sequence
$\sigma = \{\sigma(1), \sigma(2), \dots, \sigma(r)\}$	Partial sequence of r scheduled jobs
σ'	Set of $n - r$ jobs not included in σ
LB_i	Lower bound of Machine M_i

$[a_i, b_i]$	Unavailability interval of machine $M_i, (0 \leq a_i < b_i)$
$t_{i,\pi(k)}$	Earliest precedence-feasible start time of job $\pi(k)$ on machine M_i
$S_{i,\pi(k)}$	Actual start time of job $\pi(k)$on machine M_i under non-resumable constraint
$C_{i,\pi(k)}$	Completion time of job $\pi(k)$on machine M_i
$C_{max}(\pi)$	Makespan for permutation π
π^*	Optimal schedule minimizing C_{max}

3.3 Objective Function

The optimization criterion is to minimize total elapsed time (makespan), the time in which the last job is completed in the sequence on the last machine.

$$Minimze\ \ C_{max}(\pi) = C_{2,\pi(n)}$$

The objective of a production schedule is to minimize the total time needed to process the first job on first machine M_1 to process the last job on last machine M_2. Minimizing $C_{max}(\pi)$ is the same as maximizing machines utilization, minimizing WIP (wait-in-process) inventory accumulation and minimizing time to delivery of the final job, all of which have direct economic importance in the manufacturing environments considered.

3.4 Standard Flow-shop Recursion

If there are no machine unavailability constraints, then the start time and completion time of job $\pi(k)$ on machine M_i are determined by the classical recursion:

$$S_{i,\pi(k)} = \max(C_{i-1,\pi(k)}, C_{i,\pi(k-1)})$$

$$C_{i,\pi(k)} = S_{i,\pi(k)} + p_{i,\pi(k)}$$

with boundary conditions $C_{0,\pi(k)} = 0$ for all $k \in \{1, 2, \dots, n\}$ and $C_{i,\pi(0)} = 0$ for all $i \in \{1, 2\}$. The first boundary condition is due to the fact that all jobs are available at time zero, while the second is due to the fact that none of the jobs in the sequence is scheduled before the first job. Here perform recursions row-by-row (machine-by-machine), across the matrix of scheduling, according to the standard model, to return different start and completion times for each permutation π.

3.5 Modified Recursion under Non-resumable Unavailability

With the non-resumable constraint, the start time $S_{i,\pi(k)}$ is defined with a conditional logic that takes the 3 steps base. For each i and $\pi(k)$, let $t_{i,\pi(k)}$ be the first time that the job i can start without encountering a preceding job that has been delayed.

$$t_{i,\pi(k)} = \max(C_{i-1,\pi(k)}, C_{i,\pi(k-1)})$$

It is the earliest time job $\pi(k)$ might start to run on machine M_i (excluding the unavailability constraint). The non-resumable feasibility condition states that the processing interval $[S_{i,\pi(k)}, S_{i,\pi(k)} + p_{i,\pi(k)}]$ should not overlap the open unavailability interval $[a_i, b_i]$. These are two possible scenarios:

Case 1: $t_{i,\pi(k)} + p_{i,\pi(k)} \leq a_i$ (job completes before unavailability begins)

Case 2: $t_{i,\pi(k)} \geq b_i$ (job starts after unavailability ends)

If Case 1 or Case 2 are true, the job can be started (made feasible) at $t_{i,\pi(k)}$ with no valorization of infeasibility and $S_{i,\pi(k)} = t_{i,\pi(k)}$.

The job will be delayed; that is, it will start at time b_i instead of start at time $\tau_{i,\pi(k)}$, if it does not satisfy either of the two cases above; that is, $a_i < t_{i,\pi(k)} + p_{i,\pi(k)}$ and $t_{i,\pi(k)} < b_i$. The whole modified recursion is:

$S_{i,\pi(k)} = t_{i,\pi(k)}$; $t_{i,\pi(k)} + p_{i,\pi(k)} \leq a_i$ *or* $t_{i,\pi(k)} \geq b_i$

$S_{i,\pi(k)} = b_i$; *otherwise*

$C_{i,\pi(k)} = S_{i,\pi(k)} + p_{i,\pi(k)}$

Note that this delay in the otherwise case is not a constant but depends on the context, and is similar in value to $b_i - t_{i,\pi(k)}$, where $t_{i,\pi(k)}$ represents the scheduling history of all previous jobs. The major additional computational difficulty arising from the non-resumable constraint as compared to the traditional flow shop model is this context-dependence. The delay gets passed on through the precedence relation, and so will the start times and end times of any remaining jobs on the same machine as well as the delays for the current job on the downstream machines, which can form a chain reaction and introduce significant errors in the overall makespan.

3.6 Constraint Set

The complete constraint set governing the problem may be stated as follows:

(A1): $C_{i,\pi(k)} = S_{i,\pi(k)} + p_{i,\pi(k)}$ for all $i \in \{1,2\}, k \in \{1,2,\dots,n\}$

(A2): $S_{i,\pi(k)} \geq C_{i-1,\pi(k)}$ (job must complete on M_{i-1} before starting on M_i)

(A3): $S_{i,\pi(k)} \geq C_{i,\pi(k-1)}$ (machine M_i must complete the previous job before starting the next)

(A4): $[S_{i,\pi(k)}, C_{i,\pi(k)}] \cap [a_i, b_i] = \emptyset$ (non-resumable feasibility condition for all i,k)

(A5): $S_{i,\pi(k)} \geq 0$ for all i,k

The precedence constraints are (A2) and (A3) which are the two basic flow-shop constraints, followed by (A4) which is the non-resumable unavailability constraint, and (A5) for temporal non-negativity.

The aim is to minimize $C_{max}(\pi^*)$ under conditions (A1) to (A5) for an appropriate choice of π^*.

3. ALGORITHM 4.1

Step 1: Use Johnson's Algorithm to find the upper bound UB

(i) Find the sequence π using Johnson's algorithm.

(ii) Calculate the upper bound UB

a. $t_{i,\pi(k)} = \max(C_{i-1,\pi(k)}, C_{i,\pi(k-1)})$

b. $S_{i,\pi(k)} = t_{i,\pi(k)}$; *if* $t_{i,\pi(k)} + p_{i,\pi(k)} \leq a_i$ *or* $t_{i,\pi(k)} \geq b_i$

$S_{i,\pi(k)} = b_i$; *Otherwise*

c. $C_{i,\pi(k)} = S_{i,\pi(k)} + p_{i,\pi(k)}$

UB = $C_{max}(\pi)$

Step 2: Use Branch-and-Bound Method to find lower bound LB

(i) Calculate LB_1 and LB_2

$LB_1 = \sum_{j=1}^{n} p_{1j} + \min_k\{p_{2,\sigma'(k)}\} + (b_1 - a_1)$

$LB_2 = C_{2,\sigma(r)} + \sum_i p_{2,\sigma'(i)} + (b_2 - a_2)\,; if\ C_{2,\sigma(r)} \leq a_2$

$= C_{2,\sigma(r)} + \sum_i p_{2,\sigma'(i)}\,;\ Otherwise$

(i) $LB = \max(LB_1, LB_2)$

The node with the minimum LB value is always chosen to make the beginning of the next step.

We continue to repeat step 2 until we get one of the following two cases:

Case 1: The LB value of all nodes of the scheduling tree $\geq$ UB value. Here, Johnson sequence π will be the optimal sequence.

Case 2: We reach to the end level of the scheduling tree. Here, $\sigma(n)$ will be the optimal sequence.

4. EXAMPLES

Example 5.1: Considers a flow-shop scheduling problem consisting of five jobs $J_1, J_2, J_3, J_4\ and\ J_5$ to be processed on two machines M_1 and M_2 in that fixed order. Machine M_1 is subject to a non-resumable unavailability interval $[a_1, b_1] = [12, 16]$, meaning M_1 is unavailable from time unit 12 to time unit 16. Machine M_2 is subject to a non-resumable unavailability interval $[a_2, b_2] = [10, 15]$, meaning M_2 is unavailable from time unit 10 to time unit 15. The processing times are specified in Table: T2.

Table: T2 Processing time matrix

Job	Machine M_1	Machine M_2	
J_1	3	6	
J_2	8	2	
J_3	2	4	
J_4	4	7	
J_5	5	4	

Applying Algorithm 4.1:

Step 1: Johnson's Method

(i) Johnson's algorithm for the two-machine flow-shop generates a job sequence
$\pi = (J_3, J_1, J_4, J_5, J_2)$

(ii) **Machine M_1 computations** (unavailability interval [12, 16])

$\pi(1) = J_3$

$t_{i,\pi(k)} = \max\,(C_{i-1,\pi(k)}, C_{i,\pi(k-1)})$

$t_{1,\pi(1)} = \max\,(C_{0,\pi(1)}, C_{1,\pi(0)}) = \max(0, 0) = 0.$

$t_{1,\pi(1)} + p_{1,\pi(1)} = 0 + 2 = 2 < 12 = a_1$

$S_{1,\pi(1)} = 0;$

$C_{1,\pi(1)} = 0 + 2 = 2.$

$\pi(2) = J_1$:

$t_{1,\pi(2)} = \max\,(C_{0,\pi(2)}, C_{1,\pi(1)}) = \max(0, 2) = 2$

$t_{1,\pi(2)} + p_{1,\pi(2)} = 2 + 3 = 5 < 12 = a_1$

$S_{1,\pi(2)} = 2$

$C_{1,\pi(2)} = 2 + 3 = 5$

$\pi(3) = J_4$

$t_{1,\pi(3)} = \max\left(C_{0,\pi(3)}, C_{1,\pi(2)}\right) = \max(0,\ 5) = 5$

$t_{1,\pi(3)} + p_{1,\pi(3)} = 5 + 4 = 9\ < 12 = a_1$

$S_{1,\pi(3)} = 5$

$C_{1,\pi(3)} = 5 + 4 = 9$

$\pi(4) = J_5$

$t_{1,\pi(4)} = \max\left(C_{0,\pi(4)}, C_{1,\pi(3)}\right) = \max(0,\ 9) = 9$

$t_{1,\pi(4)} + p_{1,\pi(4)} = 9 + 5 = 14\ > 12 = a_1\ and\ t_{1,\pi(4)} < 16 = b_1$

$S_{1,\pi(4)} = b_1 = 16$

$C_{1,\pi(4)} = 16 + 5 = 21$

$\pi(5) = J_2$

$t_{1,\pi(5)} = \max\left(C_{0,\pi(5)}, C_{1,\pi(4)}\right) = \max(0,\ 21) = 21$

$t_{1,\pi(5)} = 21 > b_1$

$S_{1,\pi(5)} = 21$

$C_{1,\pi(5)} = 21 + 8 = 29$

Machine M_2 computations (unavailability interval [10, 15])

$\pi(1) = J_3$:

$t_{i,\pi(k)} = \max\left(C_{i-1,\pi(k)}, C_{i,\pi(k-1)}\right)$

$t_{2,\pi(1)} = \max\left(C_{1,\pi(1)}, C_{2,\pi(0)}\right) = \max(2, 0) = 2$

$t_{2,\pi(1)} + p_{2,\pi(1)} = 2 + 7 = 9 < 10 = a_2$

$S_{2,\pi(1)} = 2$

$C_{2,\pi(1)} = 2 + 7 = 9.$

$\pi(2) = J_1$:

$t_{2,\pi(2)} = \max\left(C_{1,\pi(2)}, C_{2,\pi(1)}\right) = \max(5, 9) = 9$

$t_{2,\pi(2)} + p_{2,\pi(2)} = 9 + 6 = 15 > 10 = a_2\ and\ t_{2,\pi(2)} = 9 < 15 = b_2$

$S_{2,\pi(2)} = b_2 = 15$

$C_{2,\pi(2)} = 15 + 6 = 21$

$\pi(3) = J_4$:

$t_{2,\pi(3)} = \max\left(C_{1,\pi(3)}, C_{2,\pi(2)}\right) = \max(9,\ 21) = 21$

$t_{2,\pi(3)} = 21 > 15\ = b_2$

$S_{2,\pi(3)} = 21$

$C_{2,\pi(3)} = 21 + 7 = 28$

$\pi(4) = J_4$:

$t_{2,\pi(4)} = \max(C_{1,\pi(4)}, C_{2,\pi(3)}) = \max(21,\ 28) = 28$

$t_{2,\pi(4)} = 28 > 15 \ = b_2$

$S_{2,\pi(4)} = 28$

$C_{2,\pi(3)} = 28 + 4 = 32$

$\pi(5) = J_2$:

$t_{2,\pi(5)} = \max(C_{1,\pi(5)}, C_{2,\pi(4)}) = \max(29,\ 32) = 32$

$t_{2,\pi(5)} = 32 > 15 \ = b_2$

$S_{2,\pi(5)} = 32$

$C_{2,\pi(5)} = 32 + 2 = 34$

The upper bound UB calculations for $\pi = (J_3, J_1, J_4, J_5, J_2)$ is given in the Table: T3.

Table: T3 Detailed schedule for Johnson's sequence $\pi = (J_3, J_1, J_4, J_5, J_2)$

Job	$t_{1,\pi(k)}$	$S_{1,\pi(k)}$	$C_{1,\pi(k)}$	$t_{2,\pi(k)}$	$S_{2,\pi(k)}$	$C_{2,\pi(k)}$
J_3	0	0	2	2	2	9
J_1	2	2	5	9	15	21
J_5	5	5	9	21	21	28
J_4	9	16	21	28	28	32
J_2	21	21	29	32	32	34

The total elapsed time produced by Johnson's method is $C_{max}(\pi) = 34$ units.

Therefore, UB=34

Step 2: Branch-and-Bound Method

Lower bound evaluation for $\sigma(1) = (J_1), (J_2), (J_3), (J_4), (J_5)$

(i) $\sigma(1) = J_1$

$C_{1,\sigma(1)} = 3$

$C_{2,\sigma(1)} = 9$

$LB_1 = \sum_{j=1}^{n} p_{1j} + \min_k\{p_{2,\sigma'(k)}\} + (b_1 - a_1)$

$LB_1 = (3 + 8 + 2 + 5 + 4) + 2 + (16 - 12)$

$= 22 + 2 + 4$

$= 28$

$C_{2,\sigma(1)} = 9 \ \leq a_2$

$LB_2 = C_{2,\sigma(1)} + \sum_i p_{2,\sigma'(i)} + (b_2 - a_2)$

$= 9 + (2 + 4 + 7 + 4) + (15 - 10)$

$= 9 + 17 + 5$

$= 31$

(ii) $LB = \max(LB_1, LB_2)$

$= (28, 31) = 31$

Applying the above steps for $\sigma(1) = (J_1), (J_2), (J_3), (J_4), (J_5)$ the corresponding lower bounds are given in Table: T4.

Table: T4 Level 1 LB node evaluation

$\sigma(1)$	$C_{1,\sigma(1)}$	$C_{2,\sigma(1)}$	LB_1	LB_2	LB	$UB = 34$
J_1	3	9	28	31	31	Explore
J_2	8	10	30	36	36	Prune
J_3	2	9	28	30	30	Explore
J_4	5	9	28	33	38	Prune
J_5	4	10	28	33	33	Explore

LB value for $\sigma(1) = (J_3)$ is minimum and less than UB value. Therefore, $\sigma(1) = (J_3)$ is the next branching node.

Lower bound evaluation for $\sigma(2) = (J_3\,J_1), (J_3J_2), (J_3J_4), (J_3J_5)$ is given in the Table: T5.

Table: T5 Level 2 LB node evaluation

$\sigma(2)$	$C_{1,\sigma(2)}$	$C_{2,\sigma(2)}$	LB_1	LB_2	LB	$UB = 34$
$J_3\,J_1$	5	21	28	34	34	Prune
$J_3\,J_2$	10	17	30	34	34	Prune
$J_3\,J_4$	6	22	28	34	34	Prune
$J_3\,J_5$	7	19	28	34	34	Prune

LB value for $\sigma(2) = (J_3\,J_1), (J_3J_2), (J_3J_4), (J_3J_5) \geq UB$ value. LB value for $\sigma(1) = (J_1)$ is less than UB value. Therefore, $\sigma(1) = (J_1)$ is the next branching node.

Lower bound evaluation for $\sigma(2) = (J_1\,J_2), (J_1J_3), (J_1J_4), (J_1J_5)$ is given in the Table: T6.

Table: T6 Level 2 LB node evaluation

$\sigma(2)$	$C_{1,\sigma(2)}$	$C_{2,\sigma(2)}$	LB_1	LB_2	LB	$UB = 34$
$J_1\,J_2$	11	17	30	32	32	Explore
$J_1\,J_3$	10	17	28	32	32	Explore
$J_1\,J_4$	6	22	28	32	32	Explore
$J_1\,J_5$	7	19	28	32	32	Explore

LB value for $\sigma(2) = (J_1\,J_2), (J_1J_3), (J_1J_4), (J_1J_5)$ is same and less than UB value. We can take anyone of this as next branching node. Let's take $\sigma(2) = (J_1J_3)$ as the next branching node.

Lower bound evaluation for $\sigma(3) = (J_1$

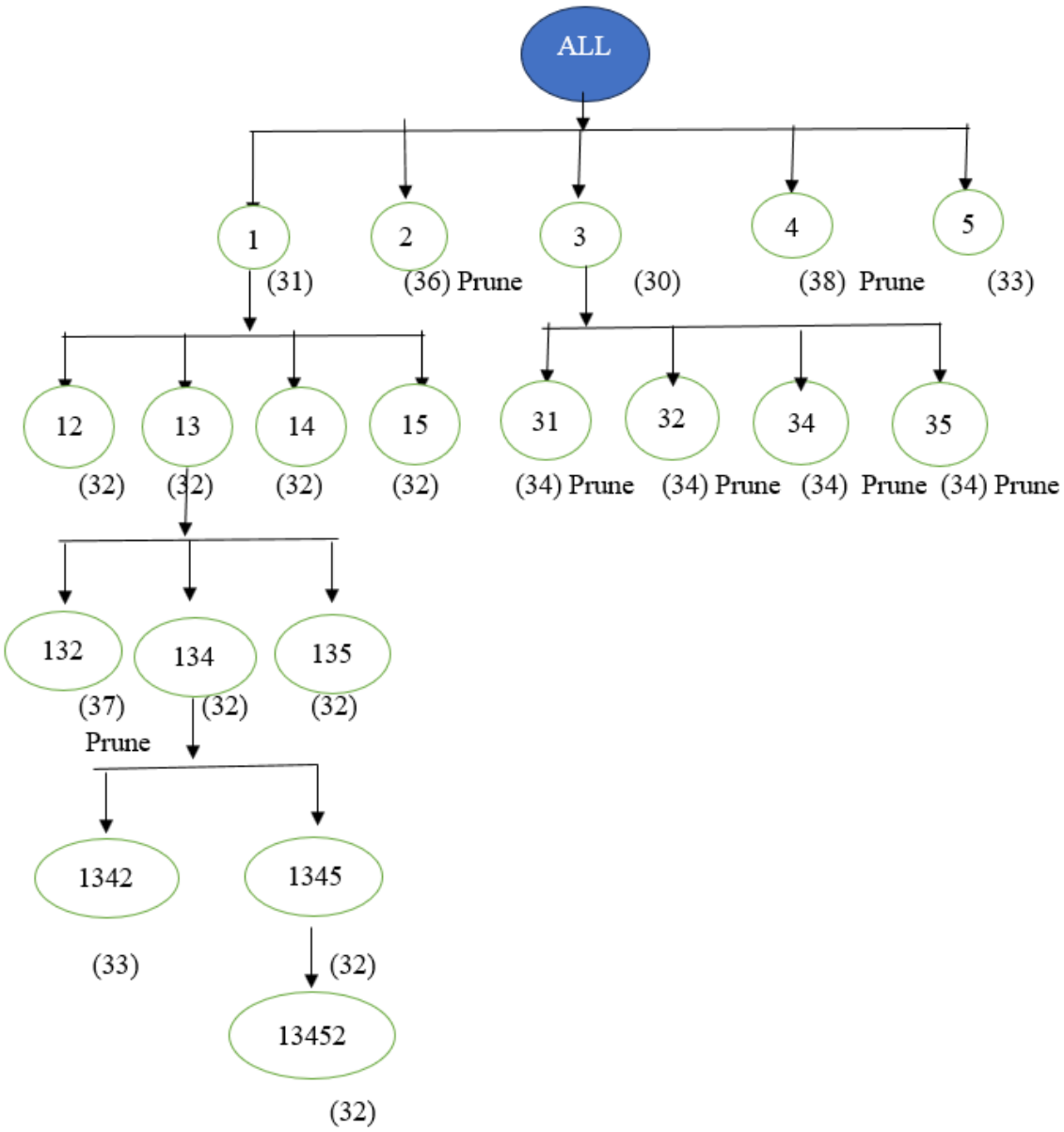

Figure: F1

By applying Algorithm 4.1, we get 1-3-4-5-2 is the optimal sequence and the minimum total elapsed time is 32 units. By applying Johnson's algorithm, we get the sequence 3-1-4-5-2 and the total elapsed time as 34 units. Hence, Johnson's algorithm fails to give optimal solution under non-resumable unavailability constrains on machines.

Example 5.2: Considers a flow-shop scheduling problem consisting of six jobs $J_1,\ J_2,\ J_3,\ J_4,\ J_5\ and\ J_6$ to be processed on two machines M_1 and M_2 in that fixed order. Machine M_1 is subject to a non-resumable unavailability interval $[a_1, b_1] = [10, 16]$ meaning M_1 is unavailable from time unit 10 to time unit 16. Machine M_2 is subject to a non-resumable unavailability interval $[a_2, b_2] = [17, 22]$, meaning M_2 is unavailable from time unit 17 to time unit 22. The processing times are specified in Table: T9.

Table: T9 Processing Time Matrix

Job	Machine M_1	Machine M_2
J_1	5	4
J_2	3	8
J_3	6	3
J_4	2	7
J_5	4	5
J_6	7	2

Applying Algorithm 4.1:

Step 1: Johnson's Method

(i) Johnson's algorithm for the two-machine flow-shop generates a job sequence

$$\pi = (J_4, J_2, J_5, J_1, J_3, J_6)$$

(ii) The upper bound UB calculations for $\pi = (J_4, J_2, J_5, J_1, J_3, J_6)$ is given in the Table: T10.

Table: T10 Detailed schedule for Johnson's sequence $\pi = (J_4, J_2, J_5, J_1, J_3, J_6)$

π	$t_{1,\pi(k)}$	$S_{1,\pi(k)}$	$C_{1,\pi(k)}$	$t_{2,\pi(k)}$	$S_{2,\pi(k)}$	$C_{2,\pi(k)}$
J_4	0	0	2	2	2	9
J_2	2	2	5	9	9	17
J_5	5	5	9	17	22	27
J_1	9	16	21	27	27	31
J_3	21	21	27	31	31	34
J_6	27	27	34	34	34	36

The total elapsed time produced by Johnson's method is $C_{max}(\pi) = 36$ units.

Therefore, UB=36

Step 2: Branch-and-Bound Method

Lower bound evaluations for $\sigma(1) = (J_1), (J_2), (J_3), (J_4), (J_5), (J_6)$ are given in Table: T11.

Table: T11 Level 1 LB node evaluation

$\sigma(1)$	$C_{1,\sigma(1)}$	$C_{2,\sigma(1)}$	LB_1	LB_2	LB	$UB = 36$
J_1	5	9	35	39	39	Prune
J_2	3	11	35	37	37	Prune
J_3	6	9	35	40	40	Prune

Natural Resources for Human Health
eISSN: 2583-1194

J_4	2	9	35	36	36	Prune
J_5	4	9	35	38	38	Prune
J_6	7	9	36	41	41	Prune

LB value for $\sigma(1) = (J_1), (J_2), (J_3), (J_4), (J_5), (J_6) \geq UB$ value.

Therefore, Johnson sequence $\pi = (J_4, J_2, J_5, J_1, J_3, J_6)$ is the optimal sequence.

$C_{max} = 36$ units.

The Complete scheduling tree is shown in Figure: F2.

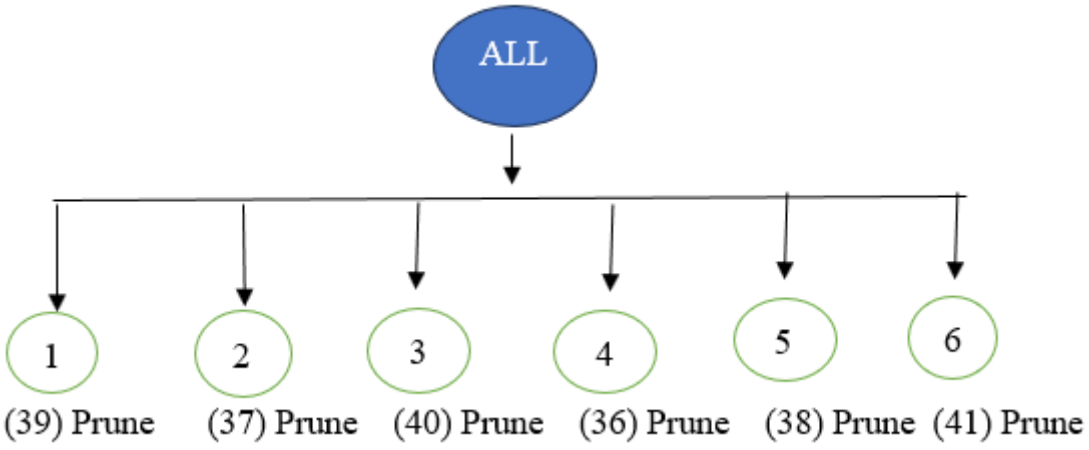

Figure: F2

5. CONCLUSION

The sorting phase of the Johnson's method takes $O(n \log n)$ and the evaluation of the schedule at the end of the method takes $O(n * 2) = O(n)$ time in the two-machine flow-shop problem with non-resumable unavailability. This renders Johnson's approach extremely economical for any practically-important problem size. In the case of n=5 for Example 5.1, Johnson's method only tests one sequence, and instantly returns the result. The same straightforward procedure is used to produce Example 5.2 for n = 6. In Example 5.2, Johnson's method provides optimum sequence but it fails in Example 5.1. Therefore, Johnson's method may fail to give optimum solution in case of non-resumable unavailability constraint flow-shop problems.

The Branch-and-Bound method, on the other hand, will ensure optimality with a greater amount of calculation. The Branch-and-Bound algorithm examined a small number of nodes in Example 5.2. There are 720 possible sequences of 6 jobs. It examined only 6 nodes to conclude that it had found the optimum sequence because of the bound-based pruning. This was also true in Example 5.1 where there are 120 possible sequences of 5 jobs. The observation is that the lower bound expression calculated as per Algorithm 4.1, we can discard lots of operations on the search tree in early stages of the branching process. In Example 5.1, the branches that begin with J_2 and J_4 at position 1 were cut right away at Level 1 and the number of full sequences $2 * 4! = 48$ will not be considered.

Johnson's method and the Branch-and-Bound method are complementary methods for solving non-resumable flow-shop scheduling problems. The Johnson's method offers a quick and effective approach to scheduling decisions in a timely manner. Branch-and-Bound method offers an exact determination of the global optimality of the solution.

6. FUTURE SCOPE

The present study is limited to two-machine permutation flow-shop it can be extended to scheduling problems having more than two machines. In production use, multiple maintenance windows can exist in the same planning horizon, or they can intersect, or become very frequent. So, single non-resumable unavailability intervals per machine can be extended to multiple non-resumable unavailability intervals per machine. All processing time and unavailability parameters are assumed to be deterministic and known. This is the usual assumption in classical scheduling theory but it does not capture the fact that sometimes production processes involve random variability, such as random distribution of processing times over different machines, or random duration of maintenance windows. So, it can be extended to stochastic processing times and probabilistic unavailability constraints.

REFERENCES

[1] Adiri I., Bruno J., Frostig E., & Rinnooy Kan A. H. G. (1989) 'Single machine flow-time scheduling with a single breakdown', *Acta Informatica,* Vol. 26(7), pp. 679–696.

[2] Allahverdi A. (2016) 'A survey of scheduling problems with no-wait in process', *European Journal of Operational Research*, Vol. 255(3), pp. 665–686.

[3] Cheng T. C. E., Gupta J. N. D. & Wang G. (2000) 'Flow shop scheduling with delays and interruptions', *Journal of the Operational Research Society,* Vol. 51(3), pp. 348–357.

[4] Hidri L. (2023), "Flexible flowshop scheduling problem with removal times", *Journal of Engineering Research*, vol 13(4), pp. 3506-3517

[5] Ignall E., & Schrage L. (1965) 'Application of the branch and bound technique to some flow-shop scheduling problems', *Operations Research*, Vol. 13(3), pp. 400–412.

[6] Johnson S. M. (1954) 'Optimal two and three-stage production schedules with setup times included', *Naval Research Logistics Quarterly*, Vol. 1(1), pp. 61–68.

[7] Lee C. Y., & Chen Z. L. (2001) 'Machine scheduling with general availability constraints', *Discrete Applied Mathematics*, Vol. 117(1-3), pp. 65–81.

[8] Li Z., Wang H., & Chen X. (2022) 'Flow shop scheduling optimization under machine breakdown conditions using hybrid metaheuristic techniques', *Applied Soft Computing,* Vol. 120, p. 108694.

[9] Narain L. and Bagga P. C. (2003) 'Minimize total elapsed time subject to zero idle time of machines in *n x 3* flowshop problem', *Indian J. Pure Appl. Math*, Vol. 34, pp. 219-228

[10] Narain L. and Bagga P. C. (2005) 'Flowshop/ no-idle scheduling to minimize total elapsed time', *Journal of Global Optimization*, Vol. 33, pp. 349-367.

[11] Pinedo M. L. (2016) Scheduling: Theory, Algorithms, and Systems, 5th ed., Springer, London.

[12] Schmidt G. (2000) 'Scheduling with limited machine availability', *European Journal of Operational Research*, Vol. 121(1), pp. 1–15.

[13] Thangaraj M. and Kumar T. J. (2025), "An efficient heuristic algorithm for flow shop scheduling problem with periodic machine specific halting and maintenance operations", *International Journal of Services and Operations Management*, Vol 52(3), pp. 275- 292

[14] Utama D. M., Umamy S. Z. and Al-Imron C. N. (2024), "No-wait flow shop scheduling problem: a systematic literature review and biometric analysis", *RAIRO-Operations Research*, 58(2), pp. 1281-1313

[15] Wang J., Li X., & Zhou Y. (2021) 'Optimization approaches for flow shop scheduling under machine breakdown conditions', *Computers & Industrial Engineering, Vol. 158*, p. 107412.

[16] Wocker M. M., Ostermeier F. F., Wanninger T., Zwinkau R. and Deuse J. (2024), "Flexible job shop scheduling with preventive maintenance consideration", *Journal of Intelligent Manufacturing*, Vol 35(4), pp. 1517-1539

[17] Xu D., Zhao F., & Chen Y. (2020) 'Flow shop scheduling with machine availability constraints and resumable processing', *International Journal of Production Research*, Vol. 58(19), pp. 5874–5891.

[18] Yuan Y., Han X., Liu X., Zhou Y. and Lu H. (2026), "Open shop scheduling problem with a flexible maintenance period", *Theoretical Computer Science,* Vol 1069, p. 115794

[19] Zhao F., Tang J., & Wang J. (2020) 'Flow shop scheduling problems with machine availability constraints: a review and future research directions', *Computers & Industrial Engineering,* Vol. 149, p. 106790.

[20] Bationo F., Bayala B. A., Zaida J. T. (2026). Optimization Of The Concrete Mixer Manufacturing Process: An Integrated Approach Combining Mathematical Modeling And Context-Appropriate Technical Solutions For Burkina Faso. International Journal of Engineering Sciences & Research Technology, 15(4), 27-42. https://doi.org/10.64149/j.ijesrt.15.5.27-42

[21] Hassan M., (2026). Rheological Properties and Optimization for Different Types of Concrete: A Review. International Journal of Engineering Sciences & Research Technology, 15(5), 83-93 https://doi.org/10.64149/j.ijesrt.15.5.83-93

[22] Zhou G., Min H., & Gen M. (2021) 'Metaheuristic algorithms for flow shop scheduling problems with machine interruptions', *International Journal of Production Research,* Vol. 59(14), pp. 4215–4231.