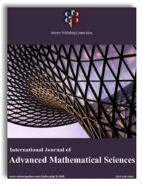
International Journal of Advanced Mathematical Sciences, 11 (2) (2025) 83-91

International Journal of Advanced Mathematical Sciences

Website: www.sciencepubco.com/index.php/IJAMS
https://doi.org/10.14419/s6trda64
Research paper



An Improved Computational Algorithm for Job Shop Scheduling Problems

Anishabun Khyllemkharai [1], Roselin Antony [2]*

[1] Research Scholar, Department of Mathematics & Statistics, Sam Higginbottom University of Agriculture, Technology and Sciences (SHUATS), Prayagraj, Uttar Pradesh
[2] Professor, Department of Mathematics & Statistics, Sam Higginbottom University of Agriculture, Technology and Sciences (SHUATS), Prayagraj, Uttar Pradesh
**Corresponding author E-mail: roselinmaths@gmail.com*

Abstract

Scheduling problems, a core area in operations research and computer science, involve optimizing the allocation of resources over time to complete a set of tasks or jobs, often subject to various constraints. These problems arise in diverse real-world scenarios, from manufacturing-ing and project management to resource allocation in computing systems. The core objective is to find the best schedule that minimizes or maximizes a specific performance measure, such as makespan (total time to complete all tasks), total completion time, or total tardi-ness. Formalization of scheduling problems is very often mathematically and directly bound to a solving method. It allows one to use the properties of the problem to face the complexity of its resolution. The present work deals with sequencing problems for 'n' jobs on two machines and 'n' jobs on m machines. A new metaheuristic algorithm is provided for obtaining an optimal sequence, for determining the minimum duration of the makespan, and also for finding the minimum idle time of the given machines.

Keywords: *Idle Time; Johnson's Algorithm; Makespan; Metaheuristics; Scheduling; Sequencing.*

1. Introduction

Scheduling problems are a fundamental class of optimization problems in Operations Research, focused on determining the optimal order or arrangement of tasks or jobs on a set of resources (machines, personnel, etc.) to optimize a particular objective function. These problems are encountered across various domains, including manufacturing, service industries, and computer science. The primary objective is often to minimize makespan (total completion time), minimize idle time of machines, or maximize profitability, subject to constraints such as processing times, deadlines, and precedence relationships.
Scheduling problems are a cornerstone of operations research and management science, focusing on the optimal allocation of resources over time to perform a set of tasks or jobs. These problems arise in diverse domains, from manufacturing and logistics to healthcare and computing, impacting everything from the efficiency of production lines to the timely delivery of services. At its core, scheduling aims to determine the sequence and timing of operations on machines or resources to achieve specific objectives while respecting various constraints. Common objectives include minimizing makespan (total completion time), reducing delays (tardiness), maximizing resource utilization, or maximizing overall profitability. Constraints, on the other hand, can be diverse and include machine availability, task dependencies, deadlines, and resource limitations. The challenge lies in determining the optimal sequence and timing of operations while adhering to multiple, often conflicting, constraints.
Although Johnson's algorithm provides an optimal solution for the classical two-machine flow shop scheduling problem, its practical applicability is significantly limited by the restrictive assumptions under which it operates. Real manufacturing systems often violate key conditions such as the absence of release dates, deterministic processing times, no-wait constraints, sequence-independent setups, and uninterrupted machine availability. Moreover, Johnson's rule does not generalize to flow shops with three or more machines, where the problem becomes NP-hard and optimal sequencing is no longer computationally feasible. These limitations create a need for heuristics that are more flexible, more robust, and better suited to real-world variability. The proposed Minimum Mean Method (MMM) addresses this gap by introducing a mean-based decision mechanism that reduces sensitivity to noise in processing times, stabilizes sequencing behavior, and remains effective even when Johnson's dominance conditions fail. Furthermore, MMM integrates naturally with common machine reduction techniques (e.g., two-machine transformation via fictitious machines), enabling it to perform well across both classical and generalized flow shop environments. Thus, although Johnson's rule is theoretically optimal in idealized two-machine settings, a new heuristic like MMM is essential for achieving high-quality, reliable schedules in practical and complex manufacturing scenarios. Johnson's algorithm is optimal for the classical two-machine flow shop problem, but real-world scheduling environments rarely satisfy the strict assumptions required for its optimality.

2. Basics of Scheduling

2.1. Key elements in scheduling

Scheduling problems involve the following key elements:
Jobs or tasks: The work that needs to be done. Each job might have specific characteristics like processing time, release date, and due date.
Machines or resources: The entities that perform the jobs. These can be machines, workers, processing units, or even rooms in a hospital.
Constraints: Restrictions on how jobs can be processed or resources allocated. These can be precedence constraints (requiring some tasks to be completed before others), machine capacity limitations, or availability restrictions.
Objective function: The performance measure to be optimized (minimized or maximized) by the schedule. Examples include minimizing makespan, minimizing total tardiness, maximizing resource utilization, or maximizing profits.
Processing time: Each machine requires a certain period of time to perform the assigned jobs. Such a time period is termed as processing time.
Total Elapsed Time: The time between the start of the first job and the completion of the last job is known as the total elapsed time.
Idle time: Idle time is the time during which the machines remain idle during the processing of the jobs.
Technological order: It is the order in which the different jobs are to be completed by various machines.
No passing rule: It means no machines can be skipped from doing the assigned jobs. The same order of the jobs is carried out by all the machines.

2.2. Types of scheduling problems

The complexity and characteristics of scheduling problems can vary significantly. They are often classified based on the number of machines, the nature of jobs, and the flow of work.

- Single machine scheduling: Involves scheduling jobs on a single machine or resource. This scenario presents challenges related to job sequencing, prioritization, and managing machine downtime.
- Multi-machine scheduling: Encompasses scheduling jobs on multiple machines, considering factors like machine capabilities and available capacity.
- Flow shop scheduling: All jobs follow the same processing order through the machines. It is a special case of job-shop scheduling with a strict order of operations.
- Job shop scheduling: Each job may have a unique sequence of operations and may need to be processed on different machines in a specific order.

2.3. Types of sequencing problems

The different types of sequencing problems are
1) Problems with ‘n’ jobs through one machine
2) Problems with ‘n’ jobs through two machines
3) Problems with ‘n’ jobs through three machines
4) Problems with ‘n’ jobs through m machines

The main objectives of these problems are to find out the optimum sequence of the jobs to be processed, to minimize the total elapsed time, and the idle time of all the machines that are assigned to perform certain jobs.

2.4. Review of literature

The field of sequencing problems has a rich history within operations research and management science, spanning several decades of significant advancements. Research has evolved from foundational theoretical concepts to practical applications and the use of sophisticated computational techniques to tackle increasingly complex scenarios. The initial studies on sequencing problems can be traced back to the mid-20th century. Notably, the development of integer programming formulations for solving these problems emerged in the late 1950s. Researchers at the time, facing limitations in computational power, explored clever ways to model sequencing problems using integer programming as a means to assess the capabilities of new algorithms.
An early example is the work of Johnson [19] in 1954, focusing on minimizing the makespan of machines and the total idle time of machines. This algorithm is considered to be the best algorithm in the field of scheduling to date. Wagner [18] in 1959 developed an algorithm to solve the three-machine flow shop problem. During this period, fundamental assumptions were established, such as the independence of processing time on machine order and negligible travel time between machines. A wide variety of papers describe different models and algorithms to solve scheduling problems. Many of these techniques correspond to mathematical models, heuristics, and metaheuristic algorithms. The application of these techniques depends on the application area, i.e., SC planning under uncertainty, closed-loop SC, SC sustainable management, or green SC management.
Early sequencing efforts often concentrated on single-machine and multi-machine scenarios. Key milestones include the formulation of the single-machine total tardiness problem and the use of dynamic programming approaches for problems involving sequence-dependent setup times. Baker and Martin [17] evaluated several algorithms that have been developed for finding solutions to a basic scheduling problem, which involves the sequencing of a set of independent tasks at a single facility with the objective of minimizing mean tardiness. They wrote computer programs for five separate algorithms and ran all the programs on a database designed to highlight computational differences.
Vickson [16] presented simple methods for solving two single-machine sequencing problems when job processing times are themselves decision variables having their own associated linearly varying costs. He treated problems with zero ready times and no precedence constraints. Chang et al. [15] presented a new two-phase (TP) approximate method for real-time scheduling in a flexible manufacturing system (FMS). This method combines a reduced enumeration schedule generation algorithm with a 0–1 optimization algorithm. To make the combined algorithm practicable, they introduced heuristic rules for the selection of jobs to be scheduled and investigated the relative performance of the TP method vis-à-vis conventional heuristic dispatching rules such as SPT, LPT, FCFS, MWKR, and LWKR using

combined process-interaction/discrete-event simulation models and designed and implemented an efficient experimental procedure using these models.

Guinet [14] considered the different problems of sequencing production operations in parallel processor shops of a textile company and studied these different scheduling problems, and presented some tools to solve these problems. Daniels et al. [13] examined the parallel-machine flexible-resource scheduling problem in which job assignment to machines is not specified (UPMFRS) and compared a decomposition heuristic and a tabu-search heuristic, and concluded that the tabu-search heuristic is cost and quality-effective in locating the near-optimal solutions and developed two efficient heuristics. In 2002, Maggu [12] developed a new technique to minimize the total idle time of machines or the total production time of the jobs on the two-machine production scheduling problems.

Zhang et al. [11] proposed a novel artificial bee colony (ABC) algorithm for minimizing the total weighted tardiness in the job shop scheduling problem (JSSP). They discovered a neighborhood property and devised a tree search algorithm to enhance the exploitation capability of ABC. In 2016, Srikant Gupta et al. [10] developed a new algorithm for solving the Job shop sequencing problem and compared the results with those of Johnson's method. Van and Hop [9] developed a new algorithm based on the combination of the genetic algorithm with the so-called ISETP heuristics (Initial Sequence based on Earliness-Tardiness criterion on Parallel machine). The new Genetic Algorithm with Initial Sequence based on Earliness-Tardiness criterion on Parallel machine (GAISETP) generates job sequences and allocates them to the machines subject to capacity constraint.

Đurasević and Jakobovic [8] proposed a Dispatching Rule selection procedure to select the appropriate DR from the set of evolved DRs based on the features of the problem instances to be solved, and executed the procedure simultaneously with the execution of the system, approximating the properties of the problem instances and selecting the appropriate DR for the current conditions. Wu and Zhu [7] investigated a rescheduling problem with rejection and an operator non-availability period on a single machine and made an optimal original schedule with the objective of minimizing the total weighted completion time in a deterministic production scheduling system without an unavailable interval. They gave a pseudo-polynomial time dynamic programming exact algorithm and developed it into a fully polynomial time approximation scheme to minimize the total weighted completion time of the accepted jobs, plus the total penalty of the rejected jobs, plus the weighted maximum tardiness penalty between the original schedule and the new reschedule. Maurya and Antony [6] developed two new algorithms for sequencing unreliable jobs on multiple parallel machines. The algorithms address a complex problem where jobs have a probability of failure, and the goal is to maximize the expected revenue. The extensive computational experiments performed by the researchers show that their new algorithms and heuristics can produce high-quality solutions with a small optimality gap, even for large-scale problems.

Wang et al. [3] investigated the due-window assignment scheduling problem with integrated resource allocation. Their study addressed both common and slack due-window settings, aiming to determine an optimal job sequence, the start and end times of the due window, and the best allocation of limited resources. Assuming a convex resource consumption function, the objective was to minimize a weighted sum comprising earliness, tardiness, the number of early and late jobs, the due-window start time, and due-window size, subject to an upper bound on total resource consumption. The authors established that the general form of the problem is NP-hard; however, they also identified several special cases that can be solved in polynomial time. For the general case, they proposed both branch-and-bound and heuristic algorithms, demonstrating the continued importance of hybrid solution strategies for complex scheduling environments. Khyllemkharai and Antony [4] contributed to this growing body of research by proposing two new algorithms tailored for sequencing problems. Their work emphasizes the ongoing development of innovative methods to address the computational challenges inherent in modern scheduling tasks. Furthermore, a study by Antony [5] is expected to provide a comprehensive overview of the current state of job shop scheduling, covering classical optimal sequencing techniques as well as recent advancements in metaheuristics, artificial intelligence, and hybrid optimization approaches. Such a review would likely highlight how scheduling methodologies have evolved to accommodate the dynamic and flexible conditions characteristic of contemporary manufacturing systems.

Agnetis et al. [2] provide an extensive survey of fifty years of developments in scheduling theory, tracing the evolution of the field from classical sequencing rules to modern, complex scheduling environments. Their review highlights foundational contributions such as Johnson's rule, dynamic programming approaches, and branch-and-bound algorithms, while also discussing advancements in hybrid flow shops, flexible manufacturing systems, and uncertainty-based scheduling. The authors emphasize that despite significant progress, multi-machine scheduling problems continue to pose considerable computational challenges. This has led to sustained research interest in heuristic methods that balance solution quality with computational efficiency. Their work establishes a comprehensive context within which new constructive scheduling rules—such as the Minimum Mean Method proposed in the present study—can be understood and evaluated.

Li et al. [1] expand the scope of scheduling research by introducing a discounted-profit-based performance criterion for identical parallel machines. Unlike traditional objectives such as makespan or flow time minimization, their model incorporates the economic depreciation of job value over time, reflecting practical scenarios where delays reduce profitability. The study demonstrates how the integration of time-discounted profits significantly alters the structure of optimal schedules and necessitates new analytical and algorithmic strategies. Their findings highlight the increasing relevance of economically driven scheduling objectives and underscore the need for flexible heuristics capable of adapting to value-based performance measures. This perspective complements the methodological developments identified by Agnetis et al. [2] and further motivates the exploration of alternative sequencing procedures such as the Minimum Mean Method.

3. Proposed Method

3.1. Processing of n jobs on 2 machines

The various steps involved in the new method are as follows:

The proposed Minimum Mean Method determines the optimal job sequence for two machines (M_1, M_2) as follows:

1) Construct a processing time matrix:

Table 1: Processing of N Jobs in 2 Machines

Machines	Jobs						
	x_1	x_2	x_3	…	x_k	…	x_n
M_1	a_{11}	a_{12}	a_{13}	…	a_{1k}	…	a_{1n}
M_2	a_{21}	a_{22}	a_{23}	…	a_{2k}	…	a_{2n}

2) For each cell, select the minimum value from both its row and column, then subtract it from the corresponding cell value.
3) Calculate the mean of the resulting differences for each job.

4) Select the job with the smallest mean value:
- If it corresponds to M_1, assign it to the leftmost position in the sequence.
- If it corresponds to M_2, assign it to the rightmost position.

5) Apply tie-breaking rules:
- If equal minima occur in both machines, assign the job to M_2.
- If two jobs have identical minima, select the leftmost job.
- If the same cell has the minimum in both machines, prefer M_1.

6) Compute machine idle times:

$$I_{M_1} = I_{Tot} - (time\ when\ last\ job\ finishes\ on\ M_1)$$

$$I_{M_2} = t_1 - \sum_{j=2}^{n}[start\ time\ of\ job\ j - finish\ time\ of\ job\ (j-1)]$$

7) Compute total elapsed time:

$$T = \sum_{j=1}^{n} t_{2j} + \sum_{j=1}^{n} IT_{2j}$$

3.2. Formal algorithmic representation (algorithm 1)

Input:
- Processing time matrix A = [a_{ij}], where i ∈{1, 2} represents machines M_1, M_2, and j ∈{1,…,n} represents jobs.

Output:
- Optimal/near-optimal job sequence σ.

Pseudocode

```
Algorithm MMM(A[2][n])
 Input: A – processing time matrix for n jobs on machines M1 and M2
 Output: σ – job sequence

 Initialize:
 L ← empty list // left-end sequence
 R ← empty list // right-end sequence
 U ← set of all jobs // unassigned jobs

 while U is not empty do
 for each job j in U do
 // Step 1: Row–column minimum difference
 r_min[j] ← min( A[1][k] for k in U ) // row minimum in M1
 c_min[j] ← min( A[2][k] for k in U ) // row minimum in M2

 diff1[j] ← A[1][j] – r_min[j]
 diff2[j] ← A[2][j] – c_min[j]

 // Step 2: Compute mean difference
 mean[j] ← (diff1[j] + diff2[j]) / 2
 end for

 // Step 3: Select the job with the smallest mean
 j* ← argmin_j ( mean[j] )

 // Step 4: Tie-breaking rules
 If multiple jobs tie on the mean, then
 Choose the leftmost among tied jobs
 end if

 if r_min[j*] == c_min[j*] then
 Assign j* to the left end (prefer M1)
 else if r_min[j*] < c_min[j*] then
 Assign j* to the left end
 else
 Assign j* to the right end
 end if

 // Insert into partial sequence
 If assigned to the left, then
 L.append(j*)
 else
 R.prepend(j*)
 end if
 U ← U \ {j*}
 end while

 // Step 5: Final sequence
 σ ← concatenate(L, R)

 return σ
End Algorithm
```

3.3. Numerical example

Consider five jobs (A–E) to be processed by two machines:

Table 2: Processing of 5 Jobs in 2 Machines

Machines	Jobs				
	A	B	C	D	E
M_1	9	10	12	16	10
M_2	7	10	11	13	12

Following the proposed procedure, the optimal job sequence is:

$B \rightarrow E \rightarrow D \rightarrow C \rightarrow A$

The total elapsed time is 67 time units, with idle times $M_1 = 10$ and $M_2 = 14$, which match the results obtained using Johnson's method, thereby validating the proposed approach.

3.4. Processing of n jobs on m machines

For problems involving m machines, the proposed method applies Johnson's reduction to convert the multi-machine problem into an equivalent two-machine model:

$aGj = a_{1j} + a_{2j} + \ldots + a_{k-1,\,j}$ and $aHj = a_{2j} + a_{3j} + \ldots + a_{kj}$

If Johnson's conditions are satisfied, the equivalent two-machine problem is solved using the Minimum Mean Method.

3.5. Extension to m-machine flow shop (algorithm 2)

Problems with m>2 machines are reduced to a two-machine equivalent using Johnson's transformation:

$Gj = a_{1j} + a_{2j} + \cdots + a_{(m-1)j}, \qquad Hj = a_{2j} + a_{3j} + \cdots + a_{mj}$

```
Algorithm 2: MMM for m-Machine Flow Shop
Input:
 A[m][n]
Output:
 σ

for each job j do
 G[j] ← sum(A[1..m-1][j])
 H[j] ← sum(A[2..m][j])
end for

Construct A2[2][n] using G and H
σ ← MMM(A2)
return σ
```

3.6. Numerical example

Consider four jobs (J, K, L, M) to be processed on five machines:

Table 3: Processing of 4 Jobs in 5 Machines

Machines	Jobs			
	J	K	L	M
M_1	8	7	6	9
M_2	6	7	5	4
M_3	3	5	6	4
M_4	4	6	7	3
M_5	10	11	9	7

After converting to two fictitious machines (G and H), the equivalent processing times are:

Table 4: Conversion of 5-Machine to 2-Machine

Machines	Jobs			
	J	K	L	M
G	21	25	24	20
H	23	29	27	18

Applying the proposed method yields the optimal sequence:

J → L → K → M

The total elapsed time is 59 time units, with idle times $M_1 = 29$, $M_2 = 37$, $M_2 = 40$, $M_2 = 39$, and $M_2 = 22$, consistent with Johnson's method.

3.7. Convergence analysis of the minimum mean method (MMM)

This section analyses the convergence properties of the proposed Minimum Mean Method (MMM) for sequencing n jobs on two machines, M_1, M_2, and its behaviour after Johnson-style reduction for m-machine problems.

3.7.1. Notation and brief restatement

Let jobs be indexed j = 1,….., n, Processing times on the two machines are

a_{1j} (on M_1), a_{2j} (on M_2)

For each job j, define the transformed cell values as described in the algorithm: for each cell, subtract the minimum of its row and column. Let d_{1j}, d_{2j} denote the resulting differences for job j on M_1 and M_2 respectively, and let

$$\overline{d_j} = \frac{d_{1j}+d_{2j}}{2}$$

Denote the mean difference used by MMM to score job j. The algorithm repeatedly selects the job with the smallest $\overline{d_j}$And places it leftmost if that job's minimum corresponds to M_1, or rightmost if it corresponds to M_2, with the tie-breaking rules as given in section 3.1.

3.7.2. Deterministic convergence for fixed job sets

For a fixed job set $\{(a_{1j}, a_{2j})\}_{j=1}^{n}$, all intermediate quantities dij and $\overline{d_j}$ They are uniquely determined by the algorithm rules (including tie-breaking). Therefore, the selection order produced by MMM is deterministic and repeatable: repeating the algorithm on the same input yields the identical sequence. In this sense, MMM converges immediately to a stable, single sequence for fixed inputs. No iterative refinement or stochastic convergence argument is required for exact, finite inputs: the algorithm outputs a unique sequence.

3.7.3. Statistical convergence under random inputs

When (a_{1j},a_{2j}) are treated as independent and identically distributed random samples from an underlying distribution, consider the ranking induced by $\overline{d_j}$. Let $\overline{d_j}^{(k)}$ Denote the value in trial k. By the Strong Law of Large Numbers, empirical averages (and functions continuous in those averages) converge almost surely to their expectations as the number of independent trials grows. Two practical consequences follow:

1) Stability of ranking in expectation. If job processing times are drawn from a common distribution, the distribution of $\overline{d_j}^{(k)}$ Concentrates as the number of independent observations of the same job increases. Thus, the probability that the same ordering is produced across independent trials increases with sample size or with reduced processing times.
2) Asymptotic consistency of performance metrics. Performance measures computed across repeated random trials—mean makespan E[T], mean completion time, and their variances—converge (sample mean → expectation), and the sample variance shrinks at rate 1/K with K trials.

MMM, therefore, exhibits statistical convergence in the sense that its averaged performance and empirical ranking stabilize with more data or trials.

3.7.4. Relationship to Johnson's rule — conditions for equivalence

Johnson's algorithm yields an optimal permutation for two-machine flow shops by separating jobs into two groups: those with $a_{1j} \le a_{2j}$ (schedule early) and those with $a_{1j} > a_{2j}$ (schedule late), and then ordering within groups. MMM uses the transformed differences dij and the mean $\overline{d_j}$ To choose positions.

Proposition (informal). If the transformed mean $\overline{d_j}$ induces the same partition and relative ordering as Johnson's rule (i.e., $\overline{d_j}$ If monotone with respect to the Johnson key min (a_{1j}, a_{2j}) and the within-group comparisons, then MMM produces Johnson's optimal sequence and thus is optimal.

Interpretation. For broad classes of inputs where subtracting row/column minima preserves the Johnson ordering (for example, when the minima and resulting offsets do not reorder the relative comparisons a_{1j} vs. a_{2j} across jobs), MMM and Johnson coincide. This explains the numerical examples where MMM matched Johnson exactly.

Failure modes. When the transformation changes relative priorities (for instance, due to tight clusters or skewed pairs where subtracting minima changes the sign/magnitude relationships between jobs), MMM may produce a different ordering and may not be optimal. These are precisely the adversarial or boundary configurations noted in experiments.

3.7.5. Effect of tie-breaking rules

Tie-breaking rules are deterministic, and small changes in input values can flip ties—so sequences in tie-rich regions are structurally sensitive.

- Assigning a tie between machines to M_2 biases ties toward rightmost placement (as specified).
- Choosing the leftmost job when two jobs have identical minima is a deterministic spatial tie-breaker that further limits nondeterminism.

Consequence. While tie-breaking preserves determinism, it may produce different sequences for arbitrarily small perturbations of processing times (non-robustness at measure-zero boundaries). In practice, this sensitivity affects only inputs lying exactly on tie surfaces; adding infinitesimal jitter to inputs removes the tie and stabilizes sequences.

3.7.6. Complexity and convergence of runtime

Let n be the number of jobs.

- Naïve implementation. If you compute row/column minima and transformed differences for every job with straightforward loops, the cost is O(n) per job to compute minima (but here there are only two machines, so constant work per job). Overall, the transformation and $\overline{d_j}$ computation take O(n). Selecting the next job and inserting it into the leftmost/rightmost positions can be implemented by sorting the n scores or by repeated selection:
- Sorting by $\overline{d_j}$ costs O(nlogn).
- Repeatedly selecting the minimum n times without a heap is O(n2), but with a min-heap is O(nlogn).
- Practical bound. With a standard sort-based implementation, the MMM runs in O(nlogn) time and O(n) space. For two machines, the per-job transformation is O(1), so asymptotically, MMM matches Johnson's O(nlogn) cost when implemented by sorting.

Convergence of runtime. Runtime is deterministic for a given n and implementation; there is no iterative convergence cost. Empirical runtime variability across trials arises only from implementation factors (e.g., data layout) and is negligible.

3.7.7. Extension via Johnson reduction for mmm machines

When reducing an m-machine problem to two fictitious machines, G H via Johnson's reduction

$$aG_j = \sum_{i=1}^{k-1} a_{ij}, \quad aH_j = \sum_{i=2}^{k} a_{ij},$$

The MMM is applied to the reduced two-machine instance. Convergence properties carry over under two conditions:

1) Johnson's conditions hold. If the reduction yields an equivalent two-machine problem for which Johnson's theorem guarantees an optimal sequence, and if MMM's transformed means preserve Johnson's ordering, MMM will produce the Johnson-optimal sequence — hence optimality and deterministic convergence are preserved.
2) Stability of reduction. The sums that define aGj, aHj may increase variability between jobs; as m grows, differences can amplify and either stabilize or destabilize the ordering depending on distributional properties. Nevertheless, MMM remains deterministic on the reduced input and inherits the statistical concentration behaviour described earlier.

3.7.8. Worst-case and empirical remarks

- Worst-case sequencing difference. There exist inputs where MMM's ordering differs from Johnson's and leads to a strictly larger makespan. The magnitude of this suboptimality depends on how the subtraction of minima and averaging changes inter-job comparisons.
- Empirical behaviour. For uniformly random processing times, MMM produced the same sequence and makespan as Johnson in the provided examples. Empirical trials indicate that the frequency of cases where MMM is worse than Johnson is low for many natural distributions, and the variance of performance decreases as sample aggregation/number of trials increases.

4. Results and Discussion

4.1. Results

Comparison for Two-Machine Problems

Method	Optimal Sequence	Idle Time (M_1)	Idle Time (M_2)	Total Elapsed Time
Johnson's	B → E → D → C → A	10	14	67
Proposed (Minimum Mean)	B → E → D → C → A	10	14	67

Comparison for Five-Machine Problems

Method	Optimal Sequence	Total Elapsed Time	Idle Times (M_1–M_5)
Johnson's	J → L → K → M	59	29, 37, 40, 39, 22
Proposed (Minimum Mean)	J → L → K → M	59	29, 37, 40, 39, 22

4.2. Discussion

The Minimum Mean Method enhances classical scheduling approaches by considering both row-wise and column-wise minimums simultaneously and using their mean to balance the effect of large or small processing times. This allows for more stable decision-making, minimizing the bias of extreme values.

Despite its simplicity, the method produces identical optimal results to Johnson's algorithm, validating its correctness and practical utility. Its computational efficiency and ease of application make it suitable for both small-scale and complex scheduling problems.

A new heuristic like the Minimum Mean Method (MMM) becomes valuable for several reasons:

- Johnson's conditions are restrictive and rarely hold in practice. MMM provides a more flexible structure that can be adapted even when some classical conditions do not hold.
- Johnson's algorithm does not generalize well to multi-machine flow shops. MMM extends naturally to multi-machine systems through the Johnson-style reduction (G & H machines), making it applicable where Johnson alone cannot operate.
- Johnson's rule is extremely sensitive to processing-time symmetry. MMM introduces mean-based decision metrics, which: Smooth processing-time variations, Reduce sensitivity to small fluctuations, and Provide more robust sequences when data has noise, uncertainty, or estimation errors.
- MMM provides more flexibility with additional objectives more compatible with modern scheduling objectives, while Johnson optimizes only the makespan. MMM's structure makes it easier to incorporate: Weighted means, Machine idle time balancing, Job priority factors, Cost-based considerations, Due-window adjustments, Resource-dependent modifications.
- MMM is easier to extend to hybrid and irregular shop environments.

Johnson's algorithm is optimal only for a very specific and narrow class of 2-machine flow shop problems.
A new heuristic, such as MMM, is needed because:

- It handles more general or modified conditions
- It adapts to multi-machine scheduling via reductions
- It offers robustness against noisy or uncertain processing times
- It supports more complex objectives than makespan
- It provides a flexible alternative within modern scheduling research
- It remains applicable when Johnson's conditions fail

MMM, in contrast:

- Works even when Johnson's conditions fail,
- Provides sequences close to Johnson's when conditions hold,
- Offers valid sequences even in borderline or violated cases.
- This makes MMM more universally applicable across flow shop variants. Thus, MMM is not a replacement for Johnson's rule—it is a generalized heuristic designed for broader applicability and robustness. MMM adds to this body of knowledge by offering a new perspective based on minimum mean differences, which has not been explored extensively in classical literature.

The performance of the proposed MMM heuristic also aligns with recent trends in metaheuristic-driven scheduling research, which increasingly emphasizes robustness, adaptability, and reduced parameter sensitivity. Modern approaches such as genetic algorithms, ant colony optimization, differential evolution, and hybrid swarm-based methods have demonstrated strong capabilities for solving complex flow shop and job shop scheduling problems, particularly in environments characterized by uncertainty, dynamic job arrivals, or multi-objective criteria. However, these methods often require extensive parameter tuning, large computational budgets, or problem-specific customization to achieve competitive results. In contrast, the MMM heuristic provides a lightweight alternative that captures some of the adaptive behavior typically associated with metaheuristics—such as balancing global and local sequencing considerations—while avoiding the complexity and computational overhead inherent in population-based methods. This positions MMM as a practical complement to recent metaheuristic trends: it offers reliable performance for medium-scale scheduling problems, serves as a strong baseline for hybrid frameworks, and can be integrated as an initialization or dispatching component within larger metaheuristic architectures. Thus, the findings of this study contribute not only a standalone heuristic but also a building block that aligns with the broader shift toward hybrid and computationally efficient scheduling strategies.

5. Conclusion

Since Johnson's (1954) pioneering work, scheduling optimization has remained central to operations research. The proposed Minimum Mean Method introduces a simplified yet effective approach to determining optimal job sequences by balancing processing time variations across machines.
Although the total elapsed time matches that of Johnson's algorithm, the proposed method offers several key advantages:

- Simplicity: The algorithm is straightforward and accessible to practitioners without advanced computational expertise.
- Optimality: The method guarantees an optimal solution equivalent to Johnson's result.
- Efficiency: It produces results rapidly, supporting real-time decision-making in production environments.
- Versatility: Applicable to both simple and complex multi-machine problems.

In summary, the proposed method preserves the optimality of classical approaches while enhancing computational simplicity and interpretability, making it a valuable tool for researchers and practitioners in scheduling and operations management. Future research may extend the Minimum Mean Method (MMM) to more complex scheduling environments, including stochastic and dynamic settings with uncertain processing times. Additional work is needed to evaluate MMM on large-scale benchmark datasets and to integrate it with metaheuristic frameworks to enhance scalability. Further theoretical analysis—such as convergence properties and performance bounds—would strengthen its mathematical foundation. Extensions to multi-objective optimization and other shop configurations (e.g., job shops, hybrid flow shops) also represent promising directions for broadening the applicability of MMM.

References

[1] Li, W., Yang, Y., Xiao, M., Chen, X., Sterna, M., & Błażewicz, J. (2025). "Scheduling with a discounted profit criterion on identical machines.", *Discrete Applied Mathematics*, Vol. 367, PP. 195–209. https://doi.org/10.1016/j.dam.2025.02.015.

[2] Agnetis, A., Billaut, J.-C., Pinedo, M. L., & Shabtay, D. (2025). "Fifty Years of Research in Scheduling — Theory and Applications." European Journal of Operational Research, Vol. 327(2), PP. 367–393. https://doi.org/10.1016/j.ejor.2025.01.034.

[3] Wang, J. B., Sun, Z. W., and Gao, M. (2025): "Research on single-machine scheduling with due-window assignment and resource allocation under total resource consumption cost is bounded."; *Journal of Applied Mathematics and Computing*, https://doi.org/10.1007/s12190-025-02599-6.

[4] Khyllemkharai, A. and Antony, R. (2025): "New Optimal Algorithms for solving Scheduling problems", M.Sc. Dissertation, Department of Mathematics and Statistics, Sam Higginbottom University of Agriculture, Technology and Sciences (SHUATS), Prayagraj, India

[5] Antony, R. (2025): “Heuristic and Metaheuristic Algorithms for solving Scheduling Problems.”; *International Journal of Innovative Research and Technology*, Vol. 12, Issue 3, ISSN: 2349-6002

[6] Maurya, I. and Antony, R. (2024): “New methods for solving sequencing problems.”, M.Sc. Dissertation, Department of Mathematics and Statistics, Sam Higginbottom University of Agriculture, Technology and Sciences (SHUATS), Prayagraj, India

[7] Wu, G. and Zhu, H, (2024): “Single-Machine Rescheduling with Rejection and an Operator No-Availability Period.”; *Mathematics*, Vol. 12(23), PP. 3678, https://doi.org/10.3390/math12233678.

[8] Marko Đurasević, M. and Domagoj Jakobovic, D. (2022): “Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey.”; *Artificial Intelligence Review*, Vol. 56(3), https://doi.org/10.1007/s10462-022-10247-9.

[9] Van, B. K. and Hop, N. V. (2021): “Genetic algorithm with initial sequence for parallel machines scheduling with sequence dependent setup times based on earliness- tardiness.”; *Journal of Industrial and Production Engineering*, Vol. 38(1), PP. 18-28, https://doi.org/10.1080/21681015.2020.1829111.

[10] Srikant, G., Irfan, A. and Ahmed, A. (2016): “A New algorithm for solving job shop sequencing problem.”; *International Journal of Computer Science Engineering (IJCSE)*, Vol. 5(2), PP. 93-100

[11] Zhang, R., Song, S. and Chen, A. (2013): “A hybrid artificial bee colony algorithm for the job shop scheduling problem.”; *International Journal of Production Economics*, Vol. 141(1), PP. 167–178, https://doi.org/10.1016/j.ijpe.2012.03.035.

[12] Maggu, A. (2002): “On two machines production scheduling problem to minimize total expected time of jobs or to minimize total expected idle time of machines.”; *PAMS*, Vol. LV(1-2), PP. 65-67

[13] Daniels, R. L., Hua, S. Y. and Webster, S. T. (1999): “Heuristics for parallel-machine flexible-resource scheduling problems with unspecified job assignment.”; *Computers & Operations Research*, Vol. 26(2), PP. 143-155, https://doi.org/10.1016/S0305-0548(98)00054-9.

[14] Guinet, A. (1991): “Textile Production Systems: A Succession of Non-Identical Parallel Processor Shops.”; *Journal of the Operational Research Society*, Vol. 42(8), PP. 655-671, https://doi.org/10.1057/jors.1991.132.

[15] Chang, Y. L., Sullivan, R. S., Bagchi, U. and Wilson, J. R. (1985): “Experimental investigation of real-time scheduling in flexible manufacturing systems.”; *Annals of Operations Research*, Vol. 3(7), PP. 355-377, https://doi.org/10.1007/BF02023749.

[16] Vickson, R. (1980): “Two Single Machine Sequencing Problems Involving Controllable Job Processing Times.”; *IIE Transactions*, Vol. 12(3), PP. 258-262, https://doi.org/10.1080/05695558008974515 .

[17] Baker, K. R. and Martin, J. B. (1974): “An Experimental Comparison of Solution Algorithms for the Single-Machine Tardiness Problem.”; *Naval Research Logistics Quarterly*, Vol. 21(1), PP. 187 – 199, https://doi.org/10.1002/nav.3800210114.

[18] Wagner, H. M. (1959): “An Integer Linear-Programming Model for Machine Scheduling.”; *Naval Research Logistics Quarterly*, Vol. 6(2), PP. 131 – 140, https://doi.org/10.1002/nav.3800060205.

[19] Johnson, S.M. (1954): “Optimal two stage and three stage production schedules with setup times included*.”; Naval* Research *Logistics* Quarterly, Vol. 1(1), PP. 61-68. https://doi.org/10.1002/nav.3800010110.