[1] Madhusudhan Chowdary Koneru[2*]
RM. Nachiappan[1†]
M. Kedar Mallik[2†]

Computation of job idle time with 'n' jobs on 'k' machines – a novel approach with case studies

Abstract: - One of the most common issues with production scheduling is the flowshop sequencing difficulty. In a flowshop, a particular kind of manufacturing procedure adheres to a specified linear framework. The purpose of this study is to minimize job idle time for a series of 'n' activities on 'k' machines to address a flowshop problem in a static workshop. Based on how each job flows as it progresses from the first to the last machine, the proposed approach is formulated. The study's results for 'n*k' flowshop sequencing issues demonstrate that the suggested approach is successful and makes promising strides.

Keywords: Scheduling, idle times, Job idle time, Sequence, Flowshop, Processing time, Heuristic

1. Introduction

Flowshop scheduling involves the management of work that flows uninterruptedly through a series of machines in a specified sequence. This setup ensures that each job follows a linear path through a sequence of machines designed to perform specific tasks. Scheduling itself entails the strategic allocation of resources such as equipment, personnel, and space over time to tasks, projects, or customers. Each work in the flowshop scheduling context must go through a preset order of machines to maximise a preset performance metric. In accordance with this method, a machine can only handle one work at a time, and once a job starts processing, it cannot be stopped or interrupted by another. Flowshop scheduling issues are common in sectors like manufacturing, where meeting customer demands for product quality, flexibility, and production rates depends on effective operation coordination. Because these issues are NP-hard, it is typically not possible to find the global optimal solution in a polynomial amount of time given their complexity. Therefore, the development and implementation of effective flowshop scheduling strategies are crucial for enhancing workflow efficiency and overall production output.

2. Literature survey

To determine the shortest total elapsed time (makespan), Johnson's algorithm (JA) [7] addresses the scheduling problem in two stages ('n' jobs & '2' machines) and three stages ('n' jobs & '3' machines). For problems with more than three steps, however, this strategy might not produce the best results. Many researchers developed new heuristic methods and other extensions for JA. Palmer [10] suggested a heuristic method based on slope index theory to solve 'n' jobs on 'm' machines. As an addition to JA, Campbell, Dudek, and Smith (CDS) [3] presented a heuristic. The initial m-machine issue is split up into a series of m-1 two-machine problems in their method, and each of these is resolved using JA. Palmer and CDS's heuristics were combined in Dannenbring's [4] heuristic to maximize their benefits. According to Nawaz, Enscore, and Ham [9], the fundamental sequencing was established by calculating the total time needed to complete each task and sorting the results from left to right in the sequence based on their highest value. To shorten the make-span of the partial timetable, the first two jobs are then selected and scheduled. To shorten the make-span of the incomplete schedule, the third work is then added to the incomplete sequence. Expand it to include all the planned jobs. A manufacturing company's large-scale flowshop problem was taken on by Gupta et al. [6] in, who used artificial neural networks (ANN) to find the best solution. When setup periods and processing times must be handled independently, Ali Allahverdi et al. [1] created an

[1] *Department of Manufacturing Engineering, Annamalai University, Chidambaram, City, 608001, Tamil Nadu, India.

[2]Department of Mechanical Engineering, Vasireddy Venkatadri Institute of Technology, Nambur, 522508, Andhra Pradesh, India.

*Corresponding email: madhusudhankoneru@hotmail.com;

†These authors contributed equally to this work.

iterated search method with the addition of a block simulation annealing procedure. Park and Choi [11] looked at the'm' machine flowshop scheduling problem in 2015. In order to minimise the make-span and flowtime while addressing a multi-product parallel multi-stage scheduling problem, Kay Saraçolu et al. [5] created a three-phase solution technique. The Hybrid-Heuristic-Metaheuristic-Genetic-Algorithm (HHMGA), proposed by Bari and Karande [2], tackles the issue of minimizing makespan. The suggested HHMGA algorithm is examined with the use of simulation tools and shown with information from the steel sector. Improvement measures which was attained from the Overall Manufacturing Line Effectiveness (OMLE) evaluation and optimization program recommended by Logeshwaran and Nachiappan [8] that improved the bottleneck processes and losses in the product line are validated by the real case study organisation in south India.

Though many researchers have focused on different aspects of scheduling, a novel approach has been proposed in the present work to minimize job idle time.

3. Problem statement

A set of n jobs must be completed in the same order on k separate machines in a flowshop problem. Job j, $j = 1, 2, \ldots, n$, is processed on machines i, $i = 1, 2, \ldots,$ k, with a nonnegative processing time $p_{(i, j)}$.

A maximum of one work can be processed by each machine, and a maximum of one machine can handle a job at any given moment. The jobs are processed by the machine in the order that they are received. This study considers the permutation schedule, where the intermediate storage between consecutive machines is infinite. The idle time of job j, $j = 1, 2, \ldots, n$, on machine i, $i = 1, 2, \ldots,$ k.

In the context of permutation flowshop scheduling with n jobs and k machines, where the objective is to minimize machine idle time, the notation and formulation can be represented as follows:

Notation

- Jobs and Machines:

  $J=\{J_1,J_2,\ldots,J_n\}$: Set of n jobs.

  $M=\{M_1,M_2,\ldots,M_k\}$: Set of k machines

- Processing Times:

  p_{ij} : Processing time of job J_i on machine M_j.

- Schedule:

  $\pi=(\pi_1,\pi_2,\ldots,\pi_n)$: A permutation of the jobs, where π_i denotes the job scheduled in position i in the sequence.

- Completion Times:

  C_{ij}: Completion time of job J_i on machine M_j.

- Make span:

  C_{max}: Completion time of job n on machine k

- Idle Time:

  I_i: Idle time for job J_i (the total time during which job J_i is not being processed).

Problem Formulation

1. **Objective:** Minimize the total idle time of all jobs.
2. **Constraints:**
   - Jobs must be processed in the same order across all machines (permutation flowshop constraint).
   - Each job must be processed in the same order on all machines.

Formulation

1. **Completion Times:**

The completion time for job J_i in position π_i on machine M_j can be determined using the following formula.

$$C_{i1} = \sum_{h=1}^{i} p_{\pi h,1}$$

$$C_{ij} = \max(c_{i,j-1}, c_{i,j}) + p_{\Pi i,j}$$

where $C_{i,0}$ is assumed to be 0 (no machine 0).

2. Idle Time:

The idle time for job Ji is calculated as:

$$I_i = \sum_{j=1}^{k} (C_{ij} - C_{i,j-1} - p_{\Pi i,j})$$

where $C_{i,0}$ is assumed to be 0, and $p_{\pi i,j}$ is the processing time of job J_i on machine M_j.

Objective Function:

Minimize the total idle time of all jobs:

$$\text{Minimize} \sum_{i=1}^{n} (I_i)$$

The task at hand is figuring out the work sequence that minimizes the total idle time of all jobs in a permutation flowshop scheduling problem with n jobs and k machines. The idle time for each job is calculated based on the completion times of that job across all machines and the job's processing times. The objective function is the sum of all individual job idle times, and the solution involves finding a permutation of the jobs that minimizes this sum while adhering to the permutation flowshop constraints.

4. The algorithm

J_n : Job n

M_k : Machine k

P_{nk} : Processing time of job n on machine k

π : Sequence of jobs after applying the algorithm

Table 1 Job-Machine Processing Time Matrix

	M_1	M_2	M_3	M_4	.	.	M_k
J_1	P_{11}	P_{12}	P_{13}	P_{14}	.	.	P_{1k}
J_2	P_{21}	P_{22}	P_{23}	P_{24}	.	.	P_{2k}
J_3	P_{31}	P_{32}	P_{33}	P_{34}	.	.	P_{3k}
J_4	P_{41}	P_{42}	P_{43}	P_{44}	.	.	P_{4k}
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
J_n	P_{n1}	P_{n2}	P_{n3}	P_{n4}	.	.	P_{nk}

Step by step process need to be followed by using Table 1 which represents Job-Machine processing time matrix.

- Step-1: Determine Minimum Processing Sum:

Calculate how long each job will take to process in total on machines 2 through k. Choose the work that takes the least amount of time to process overall across all of these machines and mark it as the last job in the sequence (π_n).

- Step-2: Identify Potential (π_{n-1}) Job:

Examine the difference between the n^{th} job's processing time on machine 1 and the total unallocated job's processing time on machine 2. Locate an unallocated task whose processing time on machine 2 is, with the least amount of variation, either equal to or less than the processing time of the work that was previously assigned on machine 1. The job that is immediately before the π_n job in the sequence, sometimes known as the π_{n-1} job, becomes a candidate for the position.

- Step-3: Validate Processing Time:

Case 1: Verify that the π_{n-1} job on machine 3 and the nth job on machine 2 have comparable processing times. If this requirement is satisfied, move forward with the job assessment.

Case 2: The job that was next better to the previously considered π_{n-1} job should be identified. If the condition is not met, repeat step 2 for the remaining unallocated jobs. Proceed to step 3 and verify that the processing time of the nth job on machine 2 and the π_{n-1} job on machine 3 are compatible after identifying the new π_{n-1} job. Proceed with the job evaluation if this requirement is satisfied.

Examine the task that satisfies both steps 2 and 3 once more if the new π_{n-1} job fails in step 3 as well. After that, move on to step four.

- Step-4: Diagonal Comparison:

If validation in Steps 2 and 3 is successful, continue with diagonal comparisons of the n^{th} job's processing time on each subsequent machine up to machine (k-1), comparing it with the π_{n-1} job's processing time on machine k.

- Step-5: Confirmation of π_{n-1} Job:

If the π_{n-1} job passes all the diagonal comparisons in Step 4, confirm it as the π_{n-1} job in the sequence.

- Step-6: Iterative Sequence Establishment:

For all remaining unallocated jobs, repeat Steps 2 through 5 iteratively to determine their appropriate position in the sequence relative to the already allocated jobs.

Sequence according to proposed algorithm

π : Allocated job 1 (π_1) → Allocated job 2 (π_2) → . → . → Allocated ($n-2$)th job (π_{n-2}) → Allocated $\pi n-1$ job (π_{n-1}) → Allocated nth job (π_n)

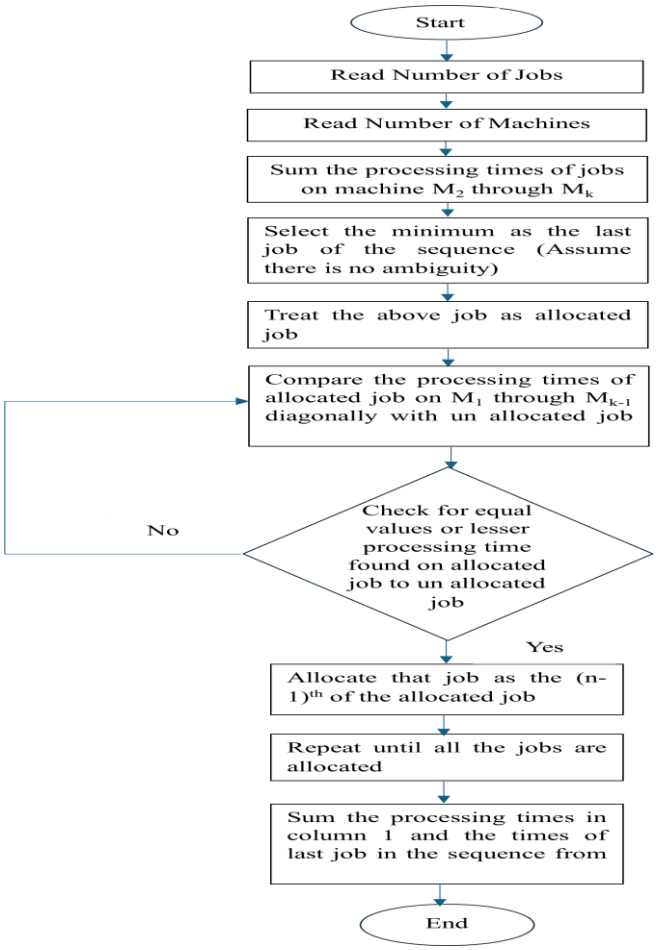

Fig. 1 Flow chart representing the process of sequencing.

5. The Problem

Even though there are several processes in a corrugated box manufacturing organization, the following are the four crucial phases considered in the flowshop scheduling problem (FSSP). Four jobs on these four machines (M_1, M_2, M_3, and M_4) are taken into consideration here, along with their processing timeframes. It is necessary to determine idle time between the jobs.

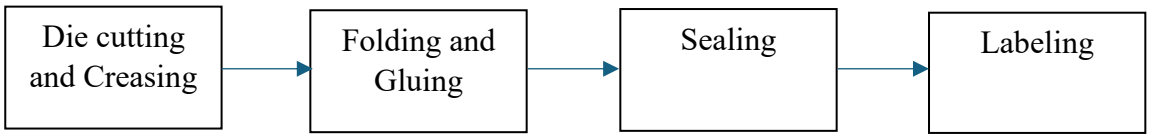

Fig. 2 Flow of operations in the production line

Table 2 Shows the machines employed for the process

Process Number	Name of the process	Machine(s) used for the process
1	Die cutting and creasing	Die-cutting machine (M_1)
2	Folding and glueing	Folding and gluing machine (M_2)
3	Sealing	Tape sealing machine (M_3)
4	Labeling	Labeling machine (M_4)

Consider allocating '4' jobs to '4' machines with the restriction that no work shall be processed before any other job, regardless of the order in which it is to be handled.

The Premises:

- During the whole predetermined operating period, all the machines are continuously accessible at time zero.
- There is zero lag time between machines.
- There is zero time required to load or unload a work onto the machine.
- Once an action has begun in a machine, it must be finished in that machine.
- Only one machine should be processing a task at a time.
- Only one job should be processed by a machine at a time.

The company used two production lines; the processing times of both lines are marginally different. These production line matrices are used as two case studies in the next portion of the study.

First need to apply condition -1 for the given case study, at any given point of time the condition-1 is not satisfied, opt condition-2.

Case study 1:

The 4 × 4 matrix of processing times for the jobs (J_1, J_2, J_3, J_4) on the machines (M_1, M_2, M_3, M_4) is listed in Table 3.

Steps for Condition-1 is as follows:

Step-1: Add the processing times on machines 2 through 4, for each job. Choose the smallest of them and place that job at last in the sequence. Treat that job as allocated one.

Step-2: Compare the processing times of allocated jobs on machine 1 through machine 3 with processing times of unallocated jobs on next machine. If any equal values found, select that job as the previous job of the allocated job.

Step-3: For the remaining jobs that are not in the sequence, repeat Step 2 until all the jobs are allocated.

Table 3 The processing times for each operation on each machine

	Machine 1 (M_1)	Machine 2 (M_2)	Machine 3 (M_3)	Machine 4 (M_4)
Job-1 (J_1)	4	3	1	2
Job-2 (J_2)	1	3	2	4
Job-3 (J_3)	3	2	4	3
Job-4 (J_4)	2	4	3	1

Step-1 is carried out for the current case study, with the corresponding outcomes for J_1, J_2, J_3, and J_4 being 6, 9, 9, and 8, respectively. J_1 is therefore the job that will be processed last in the sequence. Here by J_1 is to be considered as allocated job, whereas J_2, J_3 and J_4 as unallocated jobs.

Table 4 Allocation of last job in the sequence

Sequence:				J_1

Moving forward in step -2, compare the processing time of J_1 on M_1, M_2 and M_3 are compared with the processing time of unallocated jobs on M_2, M_3 and M_4 respectively (Tables 5, 6 and 7). After comparison, as the values of J_4 are coinciding, as shown in Table 7, J_4 is allocated as the last but one operation.

Table 5 Comparison of allocated job with Job-2

	Machine 1 (M_1)	Machine 2 (M_2)	Machine 3 (M_3)	Machine 4 (M_4)
Job-2 (J_2)	1	3	2	4
Job-1 (J_1)	4	3	1	2

Table 6 Comparison of allocated job with Job-3				
	Machine 1 (M_1)	Machine 2 (M_2)	Machine 3 (M_3)	Machine 4 (M_4)
Job-3 (J_3)	1	2	4	3
Job-1 (J_1)	4	3	1	2

Table 7 Comparison of allocated job with Job-4				
	Machine 1 (M_1)	Machine 2 (M_2)	Machine 3 (M_3)	Machine 4 (M_4)
Job-4 (J_4)	1	4	3	1
Job-1 (J_1)	4	3	1	2

The above step is repeated to allocate the other jobs also. Finally, the sequence of jobs to minimize job idle time is identified.

Table 8 Allocation of all jobs				
Sequence:	J_2	J_3	J_4	J_1

Table 9 Computation of the idle time and makespan				
	Machine 1 (M_1)	Machine 2 (M_2)	Machine 3 (M_3)	Machine 4 (M_4)
Job-2 (J_2)	1	3	2	4
Job-3 (J_3)	3	2	4	3
Job-4 (J_4)	2	4	3	1
Job-1 (J_1)	4	3	1	2

t_i =Time at which the job starts processing on respective machine

t_p= Job processing time on respective machine

t_o =Time at which the job completes processing on respective machine

Table 10 Arrangement and calculation of intermittent job idle times for the allocated jobs

	M_1			M_2			M_3			M_4			Intermittent Job Idle Time
	t_i	t_p	t_o	t_i	t_p	t_o	t_i	t_p	t_o	t_i	t_p	t_o	$I_{(I,2,\ldots,n)}$
J_2	0	1	1	1	3	4	4	2	6	6	4	10	0
J_3	1	3	4	4	2	6	6	4	10	10	3	13	0
J_4	4	2	6	6	4	10	10	3	13	13	1	14	0
J_1	6	4	10	10	3	13	13	1	14	14	2	16	0
													$\sum_{i=1}^{n}(I_i)$= 0

Now the $\sum_{i=1}^{n}(I_i)$is zero and C_{max} is 16.

Table 11 Comparison of Total Intermittent Job Idle Time of proposed method with various methods			
Sequence	Name of the Algorithm	C_{max}	$\sum_{i=1}^{n}(I_i)$
J_4-J_3-J_2-J_1	Palmer Algorithm [10]	22	13
J_1-J_2-J_4-J_3	CDS Algorithm[3]	38	1
J_1-J_4-J_2-J_3		37	0
J_1-J_4-J_3-J_2		37	1
J_4-J_1-J_3-J_2		37	1
J_2-J_4-J_3-J_1	DES Algorithm[4]	20	28
J_2-J_3-J_4-J_1	NEH-1983 [9]	16	0
J_2-J_3-J_4-J_1	Proposed Algorithm	16	0

The problem of "4" jobs and "4" machines was examined based on the case study mentioned above. The suggested algorithm produces the best results with zero for total intermittent job idle time along with NEH algorithm. The suggested algorithm was flawlessly executed, adhering to the steps and thumb rules as stated.

Case study 2:

The second production line of the company is the subject of the case study that is presented in table 12. In this case, equal values are not present in comparison (Step 2 in Case 1).

Hence, the job with lesser values than the allocated job values with minimum difference are selected as shown in Fig.2. Because of the difference there can be job idle time would not exist. The total process time is also computed in the same method for the previous case.

Table 12 The processing times for each operation on each machine				
	Machine 1 (M_1)	Machine 2 (M_2)	Machine 3 (M_3)	Machine 4 (M_4)
Job-1 (J_1)	35	38	27	25
Job-2 (J_2)	10	20	30	30
Job-3 (J_3)	40	29	38	10
Job-4 (J_4)	22	32	36	20

Table 13 Computation of the Intermittent job idle time and make span				
	Machine 1 (M_1)	Machine 2 (M_2)	Machine 3 (M_3)	Machine 4 (M_4)
Job-2 (J_2)	10	20	30	30
Job-4 (J_4)	22	32	36	20
Job-1 (J_1)	35	38	27	25
Job-3 (J_3)	40	29	38	10

Table 14 Arrangement and calculation of intermittent job idle times for allocated jobs					
	M_1	M_2	M_3	M_4	

	t_i	t_p	t_o	t_i	t_p	t_o	t_i	t_p	t_o	t_i	t_p	t_o	Intermittent Job Idle Time $I_{(I,2,3,4)}$
J_2	0	10	10	10	20	30	30	30	60	60	30	90	0
J_4	10	22	32	32	32	64	64	36	100	100	20	120	0
J_1	32	35	67	67	38	105	105	27	132	132	25	157	0
J_3	67	40	107	107	29	136	136	38	174	174	10	184	0
													$\sum_{i=1}^{n}(I_i)$ =0

For provided case study, as shown in step 13, all suggested steps with the ideal order J_2-J_4-J_1-J_3 satisfy condition-2 for $\sum_{i=1}^{n}(I_i)$ with zero along with C_{max} value 184 as shown in table 14 , Which was best in both the parameters. Because the machine is prepared to start working as soon as the preceding machine's work is finished, there is no job idle time in the current situation either.

Table 15 Comparison of Total Intermittent Job Idle Time and make span of the proposed method with various methods

Sequence	Name of the Algorithm	C_{max}	$\sum_{i=1}^{n}(I_i)$
J_4-J_2-J_3-J_1	Palmer Algorithm [10]	210	70
J_2-J_1-J_4-J_3	CDS Algorithm [3]	199	31
J_2-J_4-J_1-J_3		184	0
J_2-J_4-J_1-J_3		184	0
J_2-J_1-J_4-J_3	DES Algorithm [4]	199	31
J_2-J_4-J_1-J_3	NEH Algorithm [9]	184	0
J_2-J_4-J_1-J_3	Proposed Algorithm	184	0

Based on the case study provided and with the reference of table 15, the proposed algorithm yielded the most favorable results in terms of make-span, comparable to the CDS and NEH methods. However, its performance was average when considering machine idle time. Nevertheless, it outperformed other methods with the zero-job idle time.

7. Conclusions

A novel algorithm has been proposed to compute the Total intermittent job idle time of ‘n’ jobs on ‘m’ machines, wherein the individual process times are just compared with other and re-arrange in a systematic manner. Two case studies were also presented with the demonstration of computation of idle time of the jobs, and it is compared with popular algorithms mentioned in literature survey.

References

[1] Ali Allahverdi, Harun Aydilek, Asiye Aydilek.:No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan, Applied Mathematics and Computation, Volume 365, 124688, ISSN 0096-3003 (2020) https://doi.org/10.1016/j.amc.2019.124688.

[2] Bari P, Karande P.: Makespan Minimization using Hybrid Heuristic Metaheuristic Genetic Algorithm. IJIEPR 2023; 34 (2):1-16 (2023) http://ijiepr.iust.ac.ir/article-1-1713-en.html

[3] Campbell., H. G., Dudek, R. A., and Smith, M.L.: A heuristic algorithm for the n-job, m-machine sequencing problem, Mgml Sci. B, 10, 630 (1970)

[4] David G. Dannenbring.: An Evaluation of Flowshop Sequencing Heuristics, Management Science, 23, (11), 1174-1182 (1977)

[5] İlkay Saraçoğlu, Gürsel A. Süer, Patrick Gannon.: Minimizing makespan and flowtime in a parallel multi-stage cellular manufacturing company, Robotics and Computer-Integrated Manufacturing, Volume 72, 2021,102182, ISSN 0736-5845 (2021) https://doi.org/10.1016/j.rcim.2021.102182.

[6] Jatinder N. D. Gupta, Arindam Majumder & Dipak Laha.: Flowshop scheduling with artificial neural networks, Journal of the Operational Research Society,71:10, 1619-1637 (2020) https://doi: 10.1080/01605682.2019.1621220.

[7] Johnson, S.M.: Optimal two- and three-stage production schedules with setup times included. Nav. Res. Logist. Quart. 1, 61–68 (1954)

[8] Logeshwaran, J., Nachiappan, R. (2022). 'Optimization of process parameter involved in the effectiveness evaluation of continuous line manufacturing system (CLMS)', International Journal of Nonlinear Analysis and Applications, 13(1), pp. 321-342. doi: 10.22075/ijnaa.2022.5499

[9] Nawaz, M., Enscore, E. E., & Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, 11(1), 91–95 (1983) doi:10.1016/0305-0483(83)90088-9

[10] Palmer, D.S.: Sequencing jobs through a multi-stage process in the minimum total time - A quick method of obtaining a near optimum. Oper. Res. Quart. 16, 101-107 (1965)

[11] Park, M. J., & Choi, B. C.: Min-max regret version of an m-machine ordered flowshop with uncertain processing times. Management Science and Financial Engineering, 21(1), 1-9 (2015). doi:10.2139/ssrn.2537460