International Journal of Physics and Mathematics

E-ISSN: 2664-8644
P-ISSN: 2664-8636
IJPM 2025; 7(2): 37-42

www.physicsjournal.net
Received: 09-05-2025
Accepted: 12-06-2025

Mustafa Talal Kadhim
Ministry of Education, Babil Education Directorate, Iraq

Corresponding Author:
Mustafa Talal Kadhim
Ministry of Education, Babil Education Directorate, Iraq

Some heuristic to solve job shop machine scheduling problems

Mustafa Talal Kadhim

DOI: https://doi.org/10.33545/26648636.2025.v7.i2a.122

Abstract
Presenting a comparatively easy and efficient approach to scheduling issues is the goal of this study. In this study, a novel heuristic approach for workshop scheduling issues has been created and presented with the goal of lowering completion time. A mixture of dispatching rules, including "earliest deadline" for each work, "shortest processing time" for each task, "least tardiness" for the operations in each sequence, and the notion of "first-come, first-served," form the basis of the evaluation. Additionally, in contrast to the majority of heuristic algorithms, the user-specified deadline for each task is taken into consideration while determining the best timetable. This suggested approach has been used to schedule a series of JSP tasks with various properties. One of the most popular and trustworthy heuristic techniques, the shifting bottleneck algorithm, was also applied to solve the models. The results of the output comparison demonstrate the superiority of the suggested algorithm by achieving a substantially shorter and better time period when the number of jobs is less than or equal to the number of machines. Furthermore, the suggested approach produces more efficient results when the ratio of jobs to machines is smaller than 2.1 for the category where the number of jobs exceeds the number of machines. However, when the aforementioned ratio surpasses 2.1, a shorter completion time may be attained by any approach in this category. Since the data do not exhibit any particular pattern, it is not possible to make broad generalizations in this instance.

Keywords: Workshop scheduling, heuristic approach, completion time, dispatching rules, shifting bottleneck algorithm, task deadline

Introduction
Over the past few decades, one of the most delicate subjects in applied mathematics and optimization has been Job Shop Scheduling (JSP). On the other hand, this topic is extremely important because to its extensive use in business and other economic domains, as well as its complexity, particularly when it comes to big issues. JSP's computational complexity makes it a challenging NP-type problem. According to the scheduling literature, Math and Thompson's relatively simple problem—ten tasks and ten machines—has not been resolved for more than 25 years. Furthermore, the complexity of such issues is amply demonstrated by the fact that the scheduling problem involving 15 jobs and 15 computers is now thought to be intractable by accurate techniques. Workshop scheduling issues need processing of 'n' and'm' machines. Every task involves a few activities, all of which require a certain equipment to complete. No machine can be paused or process several jobs at once. Each machine's duty order is determined by minimizing a given character. The goal of this article is "completion time," or the amount of time needed to do all tasks. For efficiently scheduling JSP, a number of methods and techniques have been put forth; these may be categorized into three primary groups: 1) Precise algorithms, such the branch-and-bound method by Laghi and Linstra and Renoykan (1977) and the algorithm of Giffler and Thompson (1960). 2) Heuristic processes, such as Palmer's algorithms, Johnson's, and the Shifting Bottleneck, provide outcomes that are undoubtedly optimum but take a long time to obtain. Some of them provide solutions that are close to ideal, but they are not necessarily the best. 3) Meta-heuristic algorithms, including genetic algorithms and the SA and TS algorithms of Fattahi, Mehrabad, and Golai; these algorithms have a comparatively short calculation time when compared to the other two groups. This group produces better outcomes than the second group, but it does not guarantee the best response. A novel heuristic technique for the best scheduling of concurrent work scheduling issues is presented in this research.

The implementation of this approach is fairly simple since it is the consequence of integrating several distribution rules. It is also crucial to note that the delay is the difference between the time it takes to finish each work (or process) and the job's due date (or relative due date); in other words, the delay is the amount of time it takes for a job to finish beyond its due date. Furthermore, both production line and workshop issues may be scheduled using the new approach. Every machine's operation must proceed in the same order while dealing with production line issues. Workshop scheduling, however, is more generic, and the order It is possible that machines do not always complete tasks in a certain order. As a result, the suggested algorithm can handle shop work scheduling in addition to manufacturing flow difficulties, unlike algorithms like Johnson's and NEH. The next part will go over the specifics of the new algorithm. The moving bottle neck algorithm is regarded as one of the most well-known and trustworthy hoax algorithms currently in use. Consequently, the new algorithm's performance was compared to that of the moving bottle neck method in order to assess its dependability. The two techniques are compared using JSP issues as scheduling models. The results section contains the comparative results. Ultimately, conclusions will be made in light of the observed outcomes.

Materials and Methods

Proposed Algorithm Description

Along with the concept of "priority by arrival," the suggested method was created using certain basic distribution principles, such as "Earliest Due Date" for each work, "Shortest Processing Time" for each task, and "Least Slack" in operations in each sequence. The idea behind using these basic principles is to develop a control system based on a simple rule and use it to solve task scheduling issues with satisfactory results. In comparison to other heuristic algorithms, the outcomes' efficiency is also taken into account. Furthermore, users can provide due date values thanks to the architecture of the suggested technique. With the help of this function, users may choose to employ particular deadlines while solving scheduling issues. Additionally, each function's processing timings and order of operations must be entered. The machine utilized for each action must also be specified by the user. To properly illustrate the stages, a brief example is used to demonstrate the specifics of the new algorithm operations.

Table 1

An example of a JSP with three jobs and three machines is shown in Table 1. Additionally, the delivery dates and processing periods have been mentioned. Every table row contains a definition for each task. Every job has a number of tasks that a particular machine must complete, and each task has a processing time that is stated in the table. For instance, Job 1 has three operations. Machine 1 (M1) must complete the first operation, which has a processing time of seven minutes (you may use any time unit in place of minutes).

Step 1: The relative due dates are the numbers that come from selecting the least of all the due dates that were shown and subtracting that value from the remaining due dates.

Table 1: Step 1

	J_1	J_2	J_3
$Rel\ D_i$	30	30	31
Min D_i	30	30	30
Result of $Rel\ D_i$	0	0	1

Table 2: Step 2

	$J_{1,}$**M1**	J_2**M3**	J_3**M2**
LC_i	20	19	21
Related $Rel\ D_i$	0	0	1
Result T_i	7	6	7
Order Imple	2	1	3

Dates. This is being done for two reasons. The first is that deadline values, which may be quite big, will not need to be dealt with. The second reason is that it is easier to compare the relative significance of tasks. Table 1 shows the subtraction results for the aforementioned scenario.

2) Step 2: The new algorithm is task-oriented, which means that the earliest tasks completed in each job are given precedence. This is based on the basic principle of "first come, first served." The "First Operation" column (the second column in the table) in the aforementioned example is the priority. For their individual duties to go to the next phase, these are the tasks that must be finished. The initial duties are deemed the most crucial based on the principle that has been described. In the optimal scenario, the process's completion time will be equal to the pertinent processing time if there is no ready time. Consequently, the delay for each process will be equal to the relative due date of the process less the processing or completion time of the process in question. Table 3 presents the process. The process with the least delay will be executed first, followed by the process with the biggest delay. This is because the execution order will be determined by the least delay. In the event that a tie may be broken, the algorithm selects the operation according to the function ordering; for instance, in the aforementioned scenario, function 1 is given priority over function 3 among the first operations. Each stage is carried out on a graph that resembles a Gantt chart in order to make the suggested technique more understandable. The order of jobs on each machine is typically displayed on a Gantt chart, which is a graphical representation of a project's timeline used in work scheduling issues. However, the graphs in the charts used in this paper—known as modified Gantt charts—display the order of machines for each activity, taking waiting times and order into account.

Step 3: This step follows the identical process as step 2, with the exception of the authorized completion timeframes. From the start of the task until the completion of the present process, it is equivalent to the processing time of each task. For instance, after completing job 2, the time needed to

complete task 1 is equal to the sum of 7 and 8 (7 + 8 = 15). The specifics of this phase in the aforementioned example are displayed in Table 4. Furthermore, the modified Gantt chart in Figure 1 illustrates how this phase is applied.

4) Step 4: Taking into account the completion time for every job till the end of the present procedure, this step is also comparable to the first two. Table 5 and Figure 1 provide corresponding descriptions of the process and implementation specifics that show the rate for this step on the chart. It is evident from the chart that the total time is thirty-three. It is also important to remember that the execution time, or the time needed to finish all tasks, is the same as the time needed to finish all machines. As previously stated, operations must be performed in a manner that prevents machine conflicts; in other words, before beginning a current operation, the setup time for each operation and the machine's processing time must be considered. As a result, the following is the order of jobs on each machine for the aforementioned case:

The suggested approach may schedule any number of jobs and machines using the same process as in the aforementioned example. In Figure 1, the algorithm was also displayed as a flowchart. A novel algorithm must be evaluated by contrasting it with a trustworthy and well-known current algorithm across many models. Adams, Egan, and Zavack's moving bottleneck method is one of the most widely used and acknowledged inference algorithms, and it is acknowledged as being better than other inference algorithms for static work scheduling.

Shifting Bottleneck Algorithm: This method produces extremely good results, however it does not guarantee the best option.

Table 2

	$J_{1,}$**M3**	J_2**M1**	J_3**M1**
LC_i	20+21=41	19+4=23	21+21=42
Related $Rel\ D_i$	0	0	1
Result T_i	41	23	42
Order Imple	2	1	3

promising, particularly in the collection of literature-based reference problems, which are regarded as a foundational method in the work of several academics, including Schuhen (1995) and Dowser-Pears and Lasser (1993). Nevertheless, a rise in the ratio of machines to jobs might have an impact on this algorithm's efficiency. Machines are used in the moving neck bottle algorithm approach. One machine at a time scheduling for all non-sequential machines is resolved. The task sequence for the top-ranked machine is then established based on the order of the scheduled machines, and the job sequence for the other machines is changed accordingly. This approach was selected in order to compare it with the recently suggested algorithm in this study.

Results

MATLAB was used to code the suggested and variable compressor algorithms.

Table 3

	J_1**,M3**	J_2**M1**	J_3**M1**
LC_i	20+21+10=51	19+4+12=35	21+21+7=49
Related $Rel\ D_i$	0	0	1
Result T_i	51	35	49
Order Imple	3	1	2

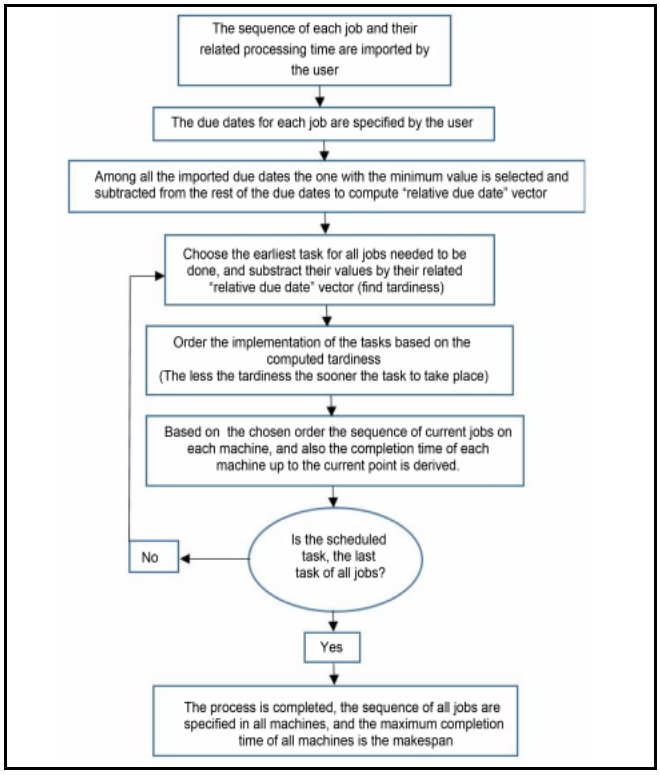

Fig 1: Flow chart of the proposed algorithm.

Table 4: Models with an equal number of machines and occupations are compared.

Models	The proportion of 27 rounds in which the new method yields a shorter completion time (%)	The bottleneck narrowing algorithm's total time was extracted.	The new algorithm's overall time extraction (average of 27 iterative outcomes)	Average congestion conversion algorithm computation time (seconds)	The new algorithm's average computation time (in seconds)
6*6	90%	40	36.383	0.023	0.033
12*12	100%	195	188.343	1.345	0.044
20*20	100%	200	198.333	9.346	0.234
35*35	100%	637	602.433	20.456	0.344
50*50	100%	403	401.678	35.566	1.343
72*72	100%	2120	2112.334	60.344	1.980
82*82	100%	3450	3402.345	75.455	2.344
90*90	100%	4100	3900.344	80.666	2.444
100*100	100%	6001	5990.454	96.657	2.505
130*130	100%	6050	5990.555	102.445	2.990

Table 5: Contrasting models with a higher machine count.

Models	The proportion of 27 rounds in which the new method yields a shorter completion time (%)	The bottleneck narrowing algorithm's total time was extracted.	The new algorithm's overall time extraction (average of 27 iterative outcomes)	Average congestion conversion algorithm computation time (seconds)	The new algorithm's average computation time (in seconds)
2*4	100%	210	198	0.456	0.233
7*8	100%	430	349	0.989	0.434
11*15	100%	980	637	1.987	0.560
14*20	100%	1290	738	2.787	0.990
18*25	100%	1505	890	8.987	1.444
22*30	100%	1900	1190	11.987	1.987
24*34	100%	1998	4500	23.877	1.990
60*72	100%	14900	5670	70.765	5.334
63*80	100%	14909	7680	89.543	20.344
210*250	100%	18098	10200	90.987	25.333

Table 6: A comparison of models with more functions.

Models	Jobs-machine	The proportion of 27 rounds in which the new method yields a shorter completion time (%)	The bottleneck narrowing algorithm's total time was extracted.	The new algorithm's overall time extraction (average of 27 iterative outcomes)	Average congestion conversion algorithm computation time (seconds)	The new algorithm's average computation time (in seconds)
6*4	1.2	100%	210	190.234	0.124	0.334
35*20	1.9	90%	690	590.765	0.993	0.345
200*120	2.8	85%	11898	10282.233	430.443	4.665
10*7	2.9	40%	390	290.233	0.501	0.022
420*200	3.2	0%	40444	39833.899	590.443	9.876
12*9	3.7	0%	401	390.333	0.455	0.433
19*11	3.9	11%	530	501.333	0.984	0.443
12*9	4.1	100%	344	290.223	0.199	0.453
5*3	4.9	100%	310	290.234	0.201	0.982
45*23	5.5	76%	898	790.987	0.567	0.432
43*21	5.9	80%	740	690.455	1.332	0.223
20*11	6.4	87%	547	511.344	1.001	0.432
29*30	6.9	40%	698	601.334	1.933	0.125

A comparison has been made between the program and its outcomes for different challenges. Three major categories may be used to categorize the models under consideration for comparison: those with an equal number of machines and functions, those with more functions than machines, and vice versa. Small, medium, and big issues of various sizes have been covered for each category. Every task is solved 27 times using 27 distinct sets of deadlines that are selected at random. The results obtained from the Shifting Bottleneck algorithm and the suggested technique are compared for each of the 27 categories. Three separate tables display the comparative findings. Table Six has been set aside for models with an equal number of machines and occupations. The models with a higher number of machines are listed in Table 6. Lastly, Table 6 displays the models with a higher number of jobs. Altering the dates might result in different outcomes because the suggested algorithm depends on them to determine the completion time. Nevertheless, the findings of the moving bottleneck method will not alter because it does not take user-specified dates into account. 27 distinct scenarios were examined in each model, each having a set of deadlines that were chosen at random. The outcomes of the shifting bottleneck algorithm were contrasted with the new method's findings in every scenario.

The average outcomes of 27 states for each model are shown here, nevertheless, to conserve space. Furthermore, it is stated what proportion of the time the new algorithm results in a reduced completion time. As a result, two methods were

compared at least 270 times for every set of tasks. This section contains observations from the models that were compared. It is important to note that every processing time under study was selected at random. In the majority of repetitions, the new method results in a shorter execution time, as seen in Table 6. Additionally, it has been noted that the difference in computing time generated by the two methods gets more substantial when the issue size in this category (the number of tasks and machines) rises. Additionally, it is determined that the variety in execution times generated by the two approaches for the same problem increases significantly with the size of the problem. The execution time achieved using the suggested technique is smaller than the equivalent one obtained using the mobile bottleneck algorithm when the number of machines is higher than the number of tasks in all circumstances. Furthermore, as the issue size rises, so does the disparity in computational time. Table 6 provides examples of the specifics. Ultimately, no consistency in the findings was found for the group where the number of jobs exceeded the number of machines. In this instance, expanded test questions were planned, and 350 chosen examples were included in this publication solely to conserve space. It is observed that the suggested approach typically generates results when the number of jobs to machine ratio is less than or equal to 2.1. As a result, the computation time is reduced and the execution time is shortened. But when the aforementioned ratio rises over 2.2, the observed outcomes become erratic, indicating that sometimes the suggested method results in a quicker execution time and other times a bottleneck transformation approach does.

Conclusions

This paper proposes a novel approach for scheduling workshop problems. The foundation of this method is the integration of various fundamental distribution criteria, including the notion of "first come, first served," "earliest due date" for each task, "shortest processing time" for each operation, and "least tardiness" for operations in each sequence. The suggested method's primary benefits are straightforward procedures and simplicity of use. The preceding sections have covered the implementation processes and flowchart in depth. The findings were compared with those obtained using the moving choke algorithm for big issues in order to provide a numerical evaluation and verification of the suggested approach. This report presents a comparison of the outcomes for over 30 models with over 900 iterations (using random deadlines). Additionally, it is observed that the suggested approach yields a shorter completion time in less computing time when the ratio of jobs to machines is smaller than 2.1. However, any approach can achieve a quicker completion time when the previously indicated ratio surpasses 2.1. The findings do not exhibit any particular pattern, thus it is impossible to make broad generalizations in this instance. Additionally, it was noted that the suggested method's computation time was less than the variable bottleneck algorithm's in every evaluated scenario.

References

1. Bagheri A, Zandieh M. An artificial immune algorithm for the flexible job-shop scheduling problem. Future Generation Computer Systems. 2003;26:13-20.
2. Nguyen S, Zhang M, Johnston M, Tan KC. Evolving reusable operation-based due-date assignment models for job shop scheduling with genetic programming. European Conference on Genetic Programming, Malaga, 11-13 April 2012;121-33. doi:10.1007/978-3-642-29139-5_11.
3. Zhang R, Wu C. An artificial bee colony algorithm for the job shop scheduling problem with random processing times. Entropy. 2011;13:1708-29. doi:10.3390/e13091708.
4. Brucker P, Jurisch B, Sievers B. A branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics. 1994;49:107-27. doi:10.1016/0166-218X(94)90204-6.
5. Muth JF, Thompson GL. Industrial Scheduling. Englewood Cliffs, NJ: Prentice-Hall; 1963.
6. Zhang CY, Li P, Rao Y, Guan Z. A very fast TS/SA algorithm for the job shop scheduling problem. Computers & Operations Research. 2008;35:282-94. doi:10.1016/j.cor.2006.02.024.
7. Lageweg BJ, Lenstra JK, Rinnooy Kan AHG. A general bounding scheme for the permutation flow-shop problem. OPNS RES. 1978;26:53-67. doi:10.1287/opre.26.1.53.
8. Adams J, Egon B, Zawack D. The shifting bottleneck procedure for job shop scheduling. Management Science. 1988;34:391-401. doi:10.1287/mnsc.34.3.391.
9. Bhosale PP, Kalshetty YR. Genetic algorithm for job shop scheduling. International Journal of Innovations in Engineering and Technology (IJIET). 2016;7:357-61.
10. Fattahi P, Mehrabad MS, Jolai F. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of Intelligent Manufacturing. 2007;18:331-42. doi:10.1007/s10845-007-0026-8.
11. Abbas M, Abbas A, Khan WA. Scheduling job shop—a case study. IOP Conference Series: Material Science and Engineering. 2016;146:021052. doi:10.1088/1757-899X/146/1/012052.
12. Demirkol E, Mehta S, Uzsoy R. A computational study of shifting bottleneck procedures for shop scheduling problems. Journal of Heuristic. 1997;3:111-37. doi:10.1023/A:1009627429878.