A New Approach to Solve Job Sequencing Problem Using Dynamic Programming with Reduced Time Complexity

Tanzin Ahammad¹, Mahedi Hasan¹, Mohammad Hasan², Md. Sabir Hossain¹, Ariful Hoque³, and Md Mamunur Rashid²

¹ Chittagong University of Engineering and Technology, Chattogram, Bangladesh
² Bangladesh Army University of Science and Technology, Saidpur, Bangladesh
³ Green University of Bangladesh, Dhaka, Bangladesh

Abstract. The classical algorithm which is dedicated to resolve job sequencing problem with a deadline (JSD) needs exponential time $O(n^2)$, where sorting algorithm $[O(n\log n)-(Merge Sort)]$ must have to use to sort all the jobs in decreasing order of their profit and it is a greedy technique. To reduce the complexity of this classical algorithm, we nullify the sorting algorithm using dynamic programming approach in the proposed algorithm. The time complexity after using this approach reduces to $O(mn)$, where no sorting algorithm $[O(n\log n)-(Merge Sort)]$ needed, which has been shown by proper explanation. Here, we were also given a novel approach to resolve the job sequencing problem using Dynamic Programming and it is a unique approach that always finds an optimal solution. By using this approach, a proper algorithm has been developed in this paper. Besides, finding maximum profit and the sequence of the job to obtain maximum profit, this algorithm gives the sequence of jobs for a specific profit or near a specific profit.

Keywords: Job sequencing problem · Dynamic programming · Sorting · Tabulation · 2D array

1 Introduction

Job Sequencing Problem with Deadline (JSD) is a popular algorithm to find the sequence of jobs to obtain maximum profit. In this problem, normally, deadline and profit of each job are given. The profit is achieved, when the job is completed before the given timeframe. Single unit of time is taken by every job. So, the minimum possible deadline for any job is 1. We have to find a maximum profit if only one job can be scheduled at a time. In the classical greedy algorithm, they have to sort the jobs according to profit, they have to sequence the jobs. In our proposed algorithm, we need not sort the job, using the dynamic programming-tabulation method this algorithm reduced this time complexity for sorting. So, a novel approach to solve the job sequencing problem using dynamic programming also proposed in this paper. Besides, reducing time complexity
this algorithm gives the sequence of jobs for finding a specific profit or near the specific profit because in all steps total profit is memorized in the first column of a 2D array. This algorithm is unique and always finds an optimal solution. There are various uses of Job Sequencing problems in real life which are described in the related works section such as, network scheduling technique, flow-shop and job-shop scheduling, task scheduling and many more.

2 Related Works

Minimizing the sum of the overheads of delayed jobs in a machine is given in paper [1]. Here, they proposed a method named as “Range-and-Bound”. The aim of the paper [2] is to develop an improved polynomial-time approximation algorithm. Paper [3] solved the single processor job sequencing with deadlines. In this paper, they have used dynamic programming type algorithms to get best and optimal solutions.

Job-shop-sequencing problem solved by the “network scheduling technique” covered in paper [4]. In this paper, for obtaining least total execution time, a new network scheduling based method with resource constraints has proposed. This paper is best suited where the resource is limited and the jobs are in arbitrary order. This procedure returns an optimal solution with the minimum duration of time. In addition to these features this algorithm also able to solved those problems that are not possible or difficult to solved using a heuristic algorithm or technique to resolve the Job Shop Sequencing problem.

Another new technique for an optimal solution of the Job Sequencing problem using the “path optimization algorithm” proposed in paper [5]. In this paper, they use the Johnson rule to solve this problem. This procedure finds a sequence of substructure and every time they take the best possible sequence that fills up the criteria for job sequencing problem. This procedure ensures an optimal solution for the job sequencing problem.

Finding minimum finish time for both preemptive and non-preemptive is NP-complete. It is the main objective of the paper [6]. This paper mainly discussed the time and space complexity for job-shop and flow shop scheduling. Also, it discusses the time complexity of different techniques for solving job shop scheduling problems such as a heuristic algorithm for job sequencing, Johnson rule for sequencing job shop problem.

An optimal solution for multi-objective distributed permutation Flow Shop Scheduling problem is discussed in paper [7]. Job sequencing problems are of various types, depending on the number of machines, resources and populations this problem varies. Here, they discussed when the number of populations is of two. Here they used the Taguchi method for solving this problem.

Generating a new job one by one is the proposed method for job sequencing problems in paper [8]. The machine for performing jobs may vary according to their characteristics. This method considers the characteristics of the machine and finds the best-suited machine for every position. For example, first, we chose four machines then next machine five then six. And this technique returns a feasible solution and the large computation may be reduced.

In paper [9], a novel heuristic method called “Time Deviation” for finding Job Sequencing problem is used to minimize the total consumed time. This technique is
both applicable for one machine \( n \) jobs and \( n \) machine \( m \) jobs and the time complexity is considerably lower than other proposed techniques. In paper [10] about advanced heuristic technique.

Sequence-dependent set up time means considering the setup time is discussed on paper [11]. In general, we read the papers before we observed that the proposed methodology considers the setup time as processing time but this procedure considers the setup time and analyzes the setup time. But there are some problems with this method. It may not provide an optimal solution.

The technique for job shop for large data is the proposed methodology or procedure in paper [12]. Here Brain k proposed a technique where large data for example data of a company in a spreadsheet is given. This solution generates both optimal and feasible solutions and heuristic solutions.

The intelligence-based genetic algorithm discussed in paper [13]. This procedure is a combination of both heuristic algorithm and the genetic algorithm. When \( n \) machines and \( m \) jobs are given the heuristic algorithm finds the best-suited job at each line of the machine and then the genetic algorithm is applied. The combination of both heuristic algorithm and the genetic algorithm reduces the complexity of the large data input.

In paper [14], Bożejko et al. proposed a “hybrid single-walk distributed tabu search method” to solve flexible job shop problem. In Paper [15], G.S. Paiva et al. have used graph representation, local search and heuristic methods to solve the Job Scheduling problem. \( N \) number of jobs sequencing on a single machine with an obstructive common due window problems has discussed in paper [16]. To solve this problem a new “Backtracking Simulated Annealing (BSA)” algorithm and an efficient coding method is proposed. Paper [17] proposed a new “Tabu Search” algorithm to solve the Job scheduling problems including precedence constraints.

3 Proposed Methodology

To implement this proposed algorithm first read the jobs (means job id, job deadline, job profit for each job). Take a table and fill the first row by 0 and find the maximum deadline. In the table, column numbers are deadline and row numbers are job id, so total columns are maximum deadline and total rows total number of jobs.

For each job, first fill up the present row by the previous row then search maximum deadline/last column no. to 1 and find row’s minimum job profit, if row’s minimum job profit is less than present job profit, then lock this minimum profits box and put the job id in this box. Sum all the job profit of this row and store it in the first column. If all the jobs are traversed, then print the last row, where 1st value is maximum profit and all other are sequence to obtain this profit. If all the jobs are not traversed, then go the next row and repeat this procedure by searching the last column no. to 1. By this process using the Dynamic Programming and tabulation method, we can find the maximum profit among all jobs and the sequence for obtaining maximum profit. Besides all the other things can be found from the table by applying a specific condition. Here the table stores all the information for each step. We have to apply specific conditions in the table for getting a specific solution (Fig. 1). Let’s see the flow chart:-
3.1 Pseudocode of Proposed Algorithm

In this research paper, the following algorithm can solve ‘The Job Sequencing Problem with Deadlines’ and can find the maximum profit with a sequence of the jobs. Here we can also find maximum profit for fixed/flexible number of jobs and also for fixed/flexible deadlines. The following algorithm also gives the solution to obtain a specific profit which jobs had to do in which order if the specific job can’t achieve then it can find the nearest profit (less or greater) of the specific profit and can also find its job sequence.
A New Approach to Solve Job Sequencing Problem

4 Experimental Result and Complexity Analysis

4.1 Step by Step Simulation for Sample Input

Here, given an array of jobs. Every job has a deadline and profit, if the job is finished before the deadline, then the profit is achieved. Every job takes single unit of time, so the minimum possible deadline for any job is 1. Find maximum profit if only one job can be scheduled at a time (Tables 1 and 2).

Table 1. Sample input of Dataset.

| Job id: | 1 | 2 | 3 | 4 | 5 |
|---------|---|---|---|---|---|
| Deadline: | 2 | 1 | 2 | 1 | 3 |
| Profit: | 100 | 19 | 27 | 25 | 15 |
Table 2. Initialization step.

| Jobid (profit, deadline) | 0 | 1 | 2 | 3 |
|--------------------------|---|---|---|---|
| 0(0, 0)                  | 0 | 0 | 0 | 0 |
| 1(100, 2)                |   |   |   |   |
| 2(19, 1)                 |   |   |   |   |
| 3(27, 2)                 |   |   |   |   |
| 4(25, 1)                 |   |   |   |   |
| 5(15, 3)                 |   |   |   |   |

Step 1: Take all the jobs according to jobid in which indicate row number. Take all the deadlines in increasing order as column number. Here every box contains jobid except the 1st column boxes which contain total profit for each row. When jobid = 0, then put 0 in every column of this 1st row. Go to the next row (Table 3).

Table 3. Step 1 simulation.

| Jobid (profit, deadline) | 0 | 1 | 2 | 3 |
|--------------------------|---|---|---|---|
| 0(0, 0)                  | 0 | 0 | 0 | 0 |
| 1(100, 2)                | 100 | 0 | 1 | 0 |
| 2(19, 1)                 |   |   |   |   |
| 3(27, 2)                 |   |   |   |   |
| 4(25, 1)                 |   |   |   |   |
| 5(15, 3)                 |   |   |   |   |

Step 2: Here, first put the previous row values in this row. Then, compare the 1st job’s deadline with the column’s no./deadline from the maximum/last row deadline 3 to 1. If the 1st job’s deadline matches the column no. or greater than the column number then find minimum profit’s job (2, 3) among those jobs and if the 1st job’s profit is greater than this minimum profit’s job (2, 3) (100 > 0), then lock this box (2, 3) and replace this box (2, 3) by 1st job’s id. Calculate the sum of all the job’s profit of this row and put it in this row’s 1st column (100). Go to the next row/job (Table 4).

Table 4. Step 2 visualization

| Jobid (profit, deadline) | 0 | 1 | 2 | 3 |
|--------------------------|---|---|---|---|
| 0(0, 0)                  | 0 | 0 | 0 | 0 |

(continued)
A New Approach to Solve Job Sequencing Problem

Table 4. (continued)

| Jobid (profit, deadline) | 0   | 1   | 2   | 3   |
|--------------------------|-----|-----|-----|-----|
| 1(100, 2)                | 100 | 0   | 1   | 0   |
| 2(19, 1)                 | 100 + 19 = 119 | 2   | 1   | 0   |
| 3(27, 2)                 |     |     |     |     |
| 4(25, 1)                 |     |     |     |     |
| 5(15, 3)                 |     |     |     |     |

Step 3: Here, first put the previous row values in this row. Then, compare the 2nd job’s deadline with the column’s no./deadline from the maximum/last row deadline 3 to 1. If the 2nd job’s deadline matches the column no. or greater than the column number then find minimum profit’s job (3, 2) among those jobs and if the 2nd job’s profit is greater than this minimum profit’s job (3, 2) (19 > 0), then lock this box (3, 2) and replace this box (3, 2) by 2nd job’s id. Calculate the sum of all the job’s profit of this row and put it in this row’s 1st column (119). Go to the next row/job (Table 5).

Table 5. Step 3 visualization.

| Jobid (profit, deadline) | 0   | 1   | 2   | 3   |
|--------------------------|-----|-----|-----|-----|
| 0(0, 0)                  | 0   | 0   | 0   | 0   |
| 1(100, 2)                | 100 | 0   | 1   | 0   |
| 2(19, 1)                 | 100 + 19 = 119 | 2   | 1   | 0   |
| 3(27, 2)                 | 100 + 27 = 127 | 3   | 1   | 0   |
| 4(25, 1)                 |     |     |     |     |
| 5(15, 3)                 |     |     |     |     |

Step 4: Here, first put the previous row values in this row. Then, compare the 3rd job’s deadline with the column’s no./deadline from the maximum/last row deadline 3 to 1. If the 3rd job’s deadline matches the column no. or greater than the column number then find minimum profit’s job (4, 2) among those jobs and if the 3rd job’s profit is greater than this minimum profit’s job (4, 2) (27 > 19), then lock this box (4, 2) and replace this box (4, 2) by 3rd job’s id. Calculate the sum of all the job’s profit of this row and put it in this row’s 1st column (127). Go to the next row/job (Table 6).

Step 5: Here, first put the previous row values in this row. Then, compare the 4th job’s deadline with the column’s no./deadline from the maximum/last row deadline 3 to 1. If the 4th job’s deadline matches the column no. or greater than the column number then find minimum profit’s job (5, 2) among those jobs, but the 4th job’s profit is not greater than this minimum profit’s job (5, 2) (25 < 27), so do not lock this job. Calculate the
Table 6. Step 4 visualization.

| Jobid (profit, deadline) | 0   | 1   | 2   | 3   |
|--------------------------|-----|-----|-----|-----|
| 0(0, 0)                  | 0   | 0   | 0   | 0   |
| 1(100, 2)                | 100 | 0   | 1   | 0   |
| 2(19, 1)                 | 100 + 19 = 119 | 2   | 1   | 0   |
| 3(27, 2)                 | 100 + 27 = 127 | 3   | 1   | 0   |
| 4(25, 1)                 | 100 + 27 = 127 | 3   | 1   | 0   |
| 5(15, 3)                 |     |     |     |     |

sum of all the job’s profit of this row and put it in this row’s 1st column (127). Go to the next row/job (Table 7).

Table 7. Step 5 visualization.

| Jobid(profit,deadline) | 0   | 1   | 2   | 3   |
|------------------------|-----|-----|-----|-----|
| 0(0,0)                 | 0   | 0   | 0   | 0   |
| 1(100,2)               | 100 | 0   | 1   | 0   |
| 2(19,1)                | 100+19=119 | 2   | 1   | 0   |
| 3(27,2)                | 100+27=127 | 3   | 1   | 0   |
| 4(25,1)                | 100+27=127 | 3   | 1   | 0   |
| 5(15,3)                | 100+27+15=142 | 3   | 1   | 5   |

**Step 6:** Here, first put the previous row values in this row. Then, compare the 5th job’s deadline with the column’s no./deadline from the maximum/last row deadline 3 to 1. If the 5th job’s deadline matches the column no. or greater than the column number then find minimum profit’s job (6, 4) among those jobs and if the 5th job’s profit is greater than this minimum profit’s job (6, 4) (0 < 15) then lock this box (6, 4) and replace this box (6, 4) by 5th job’s id. Calculate the sum of all the job’s profit of this row and put it in this row’s 1st column (127). Go to the next row/job.

**Step 7:** If all the jobs/row are traversed, then print the value of the last row, where 1st column of this row (142) which is maximum profit and print all next values of this row (3- > 1- > 5) which is required sequence. Besides, we can find for profit- [127] sequence (3- > 1) 2 jobs needed, for, profit- [119] sequence (2- > 1) 2 jobs needed, if needed profit near 130 then profit– [127] sequence (3- > 1) 2 jobs needed.

**4.2 Time Complexity Analysis**

In this proposed algorithm time complexity depends on input size. Time complexity directly depends on the number of jobs and the maximum deadline. Let, the number of jobs is n and the maximum deadline is m and the function T(n) denotes the number
A New Approach to Solve Job Sequencing Problem

elementary operations performed by the function. Then the recurrence relation for this proposed algorithm is:

For \( i = 1 \) :
\[
T(n) = m \times T(n-1) + m \times c \times 1
\]

For \( i = 2 \) :
\[
T(n) = m \times T(n-2) + m \times c \times 2
\]

For \( i = 3 \) :
\[
T(n) = m \times T(n-3) + m \times c \times 3
\]

For \( i = k \) :
\[
T(n) = m \times T(n-k) + m \times c \times k
\]

**Best Case**
When all the jobs are traversed then, best case complexity:

\[
T(n) = m \times T(n-k) + m \times c \times k
\]

\[
= \Omega(m \times n), \text{ (without sorting)}
\]

Here, all the jobs and deadlines have to traverse. No sorting is needed for this algorithm. The best case of this algorithm is \( m \times n \) where \( n \) is the number of jobs and \( m \) is the maximum deadline.

**Average Case**
The average case complexity of this proposed algorithm has occurred,

\[
T(n) = m \times T(n-k) + m \times c \times k
\]

\[
= \Theta(m \times n), \text{ (without sorting)}
\]

Here all the jobs and deadlines have to traverse. No sorting is needed for this algorithm. The average case of this algorithm is \( m \times n \) where \( n \) is the number of jobs and \( m \) is the maximum deadline.

**Worst Case**
Here, the recurrence relation:

\[
T(n) = m \times T(n-k) + m \times c \times k
\]

for \( i = 1 \) :
\[
T(n) = m \times T(n-1) + m \times c \times 1
\]

for \( i = 2 \) :
\[
T(n) = m \times T(n-2) + m \times c \times 2
\]

for \( i = 3 \) :
\[
T(n) = m \times T(n-3) + m \times c \times 3
\]

When all the jobs are traversed, then: \( (n = k) \) So,

\[
T(n) = m \times T(0) + c \times m \times n \text{ [T(0) is some constant c0]}
\]

\[
T(n) = m \times c0 + c \times m \times n
\]

\[
= O(m \times n), \text{ (without sorting)}
\]

So, Best case of this algorithm is \( m \times n \) where \( n \) is the number of jobs and \( m \) is maximum deadline.
4.3 Space Complexity

In this proposed algorithm space complexity depends on the input data. Inputs are not constant. So, if there are n number of input jobs and maximum deadline m, then space complexity will be n * m. In this proposed algorithm because of using a 2D array, the space complexity of this 2D array is O(n * m).

4.4 Experimental Result Comparison with Classical Algorithms

Here, experiment results show that for the different number of jobs and different maximum deadlines proposed algorithm execution time is less than the classical algorithm. Here, Table 8 shows that for the different number of jobs and different maximum deadlines proposed algorithm execution time is less than the classical algorithm.

Table 8. Execution Time Comparison with other classical algorithms.

| No. | Number of Jobs | Maximum Deadline | Execution time for the proposed algorithm (in µs) | Execution time for the classical algorithms (in µs) |
|-----|----------------|------------------|--------------------------------------------------|--------------------------------------------------|
| 1.  | 4              | 2                | 969.13                                           | 1048.67                                          |
| 2.  | 5              | 3                | 952.35                                           | 999.95                                           |
| 3.  | 6              | 4                | 993.13                                           | 993.56                                           |
| 4.  | 7              | 4                | 1420.4                                           | 1566.55                                          |

The column in Fig. 2 shows with increasing of both the number of jobs and maximum deadlines execution time increasing but the rate of increasing execution time for new algorithms is less than the classical algorithm.

Fig. 2. Comparison Between Classical Algorithm and proposed Algorithm.
The line graph is shown in Fig. 3 with an increasing of both the number of jobs and maximum deadlines execution time increasing but the rate of increasing execution time for the new algorithm is less than the classical algorithm. (Used device in experiment-Intel, CORE I7, 8th Generation, RAM-8 GB, Graphis-4 GB) (Table 9).

![Graph showing execution time comparison between classical and new algorithms](image)

**Table 9.** Overall Complexity Comparisons with previous classical algorithms.

| Algorithm Name | Best Case time complexity | Average Case time complexity | Worst-case time complexity | Space complexity |
|----------------|---------------------------|-----------------------------|---------------------------|-----------------|
| JSD            | $\Omega(n^2)$            | $\Theta(n^2 + n\log n) = n^2 + \Theta(n\log n)$ | $O(n^2 + n\log n) = \Theta(n^2)$ | $O(n)$          |
| Paper [1]      | $\Omega(n^2 \log n + n^2/e)$ | $\Theta(n^2 + n\log n + n^2/e)$ | $O(n^2 + n\log n + n^2/e)$ | $O(n^2/e)$      |
| Paper [2]      | $\Omega(n^2/e)$          | $\Theta(n^2/e)$             | $O(n^2/e)$                 | $O(n^2/e)$      |
| Paper [3]      | $\Omega(\min(2^n, nM))$  | $\Theta(\min(2^n, nM))$     | $O(\min(2^n, nM))$        | $O(\min(2^n, nM))$ |
| Proposed Algorithm | $\Omega(mn)$, (without sorting) | $\Theta(mn)$, (without sorting) | $O(mn)$, (without sorting) | $O(mn)$ |

5 Conclusions and Future Recommendation

The Job Sequencing with Deadline (JSD) is studied and researched by a lot number of researchers. In this paper, we have designed an algorithm by using the dynamic programming approach. By using these approach we have reduced the time complexity...
O(mn), where no sorting [O(nlogn)-Merge Sort] is needed, where n is the number of jobs and m is the maximum deadline. So, the time complexity is reduced and a new approach for solving JSD is proposed in this paper. Normally the time complexity for the general JSD algorithm overall O(n^2) where n is the number of jobs. We are able to show that our algorithm is better which gives results faster and it is a new approach. Besides, solving JSD this algorithm also finds the sequence of jobs for a specific profit or near a specific profit. Because of using 2D array the space complexity of our proposed algorithm is O(mn). In the future, we will try to do further research to reduce the space complexity of JSD as well as time complexity.

References

1. Gens, G.V., Levner, E.V.: Fast approximation algorithm for job sequencing with deadlines. Discrete Appl. Math. 3, 313–318 (1981)
2. Levner, E., Elalouf, A.: An improved approximation algorithm for the ancient scheduling problem with deadlines. In: Proceedings - 2014 International Conference on Control, Decision and Information Technologies, CoDIT 2014, 113–116 (Institute of Electrical and Electronics Engineers Inc., 2014). https://doi.org/10.1109/codit.2014.6996878
3. Sartaj, K.S.: Algorithms for scheduling independent tasks. J. ACM 23(1), 116–127 (1976). https://doi.org/10.1145/321921.321934
4. Akpan, E.O.P.: Job-shop sequencing problems via network scheduling technique. Int. J. Oper. Prod. Manage. 16, 76–86 (1996)
5. Singh, P.K.: Path optimization algorithm for network problems using job sequencing technique. Int. J. Distrib. Parallel Syst. 3(3), 301–309 (2012)
6. Gonzalez, T., Sahni, S.: Flowshop and jobshop schedules: complexity and approximation. Oper. Res. 26(1), 36–52 (1978)
7. Deng, J., Wang, L.: A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. Swarm. Evol Comput. 32, 121–131 (2017)
8. Presby, J.T., Wolfson, M.L.: An algorithm for solving job sequencing problems. Manage. Sci. 13(8), B-454–B-464 (1967)
9. Karthikeyan, K.: Heuristics algorithms for job sequencing problems. Global J. Sci. Front. Res. GJSFR Classif.-D. 10, 76–86 (2010)
10. Iqbal, P.: Job sequencing problem using advanced heuristics techniques. In: Proceedings of the International Conference on Applied Mathematics and Theoretical Computer Science, ICPMT, vol. 12, pp. 89–100 (2013)
11. Das, R.M., Man, M.H.: Job-Shop scheduling problem with sequence dependent setup times. Int. Multi Conf. Eng. Comput. Sci. 12, 67–80 (2008)
12. Baker, K.R.: Solving sequencing problems in spreadsheets. Int. J. Plan. Schedul. 1, 3 (2011)
13. Norozi, A.: Application of intelligence based genetic algorithm for job sequencing problem on parallel mixed-model assembly line. Am. J. Eng. Appl. Sci. 3, 15–24 (2010)
14. Bożejko, W., Hejducky, Z., Uchroński, M., Wodecki, M.: Solving the flexible job shop problem on multi-GPU. Int. Conf. Comput. Sci. 9, 2020–2023 (2012)
15. Paiva, G.S., Carvalho, M.A.M.: Improved heuristic algorithms for the Job sequencing and tool switching problem. Comput. Oper. Res. 88, 208–219 (2017)
16. Lin, S.W., Ying, K.C., Wu, W.J., Cheng, C.Y.: Single machine job sequencing with a restricted common due window. IEEE Access 7, 148741–148755 (2019)
17. Wook, J.S., Woo, K.J.: Solving the constrained job sequencing problem using candidate order based Tabu search. J. Inf. Syst. 25, 159–182 (2016)