Optimising Job Scheduling: A Comparative Study of Greedy and Stochastic Algorithm

Yash Bhargava
Dept. of Computing
National College of Ireland
Dublin, Ireland
x2220@student.ncirl.ie

***Abstract*—Job scheduling is the most common problem in different domains, including manufacturing, computing, and project management, where jobs need to be allocated to be scheduled efficiently to minimize the overall completion time. In this study, we utilize the application of Greedy and Simulated Annealing algorithms for job scheduling on the N_MACHINES= 4 for scheduling N_JOBS= 7, to minimize the overall job completion time. By examining the performance of these algorithms in finding optimal schedules, we have found out that the simulated annealing algorithm outperforms the greedy algorithm in terms of an overall completion time of 62 units of time which was better than the greedy algorithm solution. By leveraging its ability to explore solution spaces more comprehensively and escape local optima, simulated annealing demonstrates its superiority in attaining solutions that exhibit enhanced efficiency and effectiveness. The findings contribute valuable insights into algorithm selection for job scheduling problems and underscore the significance of employing sophisticated optimization techniques to meet stringent performance objectives.**

***Index Terms*—Greedy Algorithm, Stochastic optimisation, Job Scheduling, Optimization**

I. Introduction

Optimal Job scheduling is essential in different domains such as manufacturing, delivery, computing systems etc. The job scheduling process includes resource allocation i.e., allocating jobs in such a way that it optimizes the processing time on different machines and minimizes the wait time and the time at which the machine remains idle until the next job is scheduled for processing. The main goal of optimizing the job sequence is to assign the job sequence to the machine considering the minimum processing time as a job needs to go through the next machine so that it reduces the waiting time and the idle time of the machine. The job scheduling algorithm is one of the most challenging issues around computing complexity. Nowadays, traditional job scheduling systems only consider how we can increase job scheduling efficiency or how to satisfy the QoS requirements and they provide details which allow them to combine these above two aspects together. Various job scheduling algorithms have been proposed previously in the cloud computing area. The development of computing complexity requires technology for virtualization and resource allocation, job scheduling plays an important part in it [1].

A group M = 1, . . . , m of m machines must process a set N = 1, . . . , n of n number of independent jobs in the job scheduling problem. A set, positive processing time pij is required for each job j, j N, on each machine i, where i M. All the n jobs must be completed on all m machines in the same sequence, so that the jobs follow a set machine order through the process, beginning on machine 1 and ending on machine m. The aim is to determine the best order for executing the jobs in the machine to minimize criteria. The decrease in overall waiting and idle time is the criterion that is commonly evaluated in most of the studies. Although some modern metaheuristic algorithms provide good results in acceptable estimation times, some of those recommended may be too complex [2]. This job scheduling problem has various applications in manufacturing industries where the machine is required to schedule the tasks for processing, usually, this type of problem can also arise in data centres where the complexity of computing jobs requires to be assigned to different servers and other domains where jobs required to be assigned effectively.

The motivation behind the choice of the greedy algorithm is the simplicity i.e. it to easy to understand and the implementation was very simple, this algorithm is really time efficient in such a way of computing time complexity and the analysis was easy which allows us to determine the performance of the greedy algorithm. As greedy algorithm shows some promising results over time in previous research works in [1] by reducing the completion time of the assigned jobs and providing good user satisfaction. In the paper [2], the authors have proposed an iterated greedy algorithm based on the NEH heuristic and the performance of the proposed algorithm improved by the consideration of an additional local search phase and the performance of the proposed iterated greedy algorithm was very good as compared to the other state-of-the-art methods. In the paper [3] the study presented the iterated greedy algorithm in which they have combined the destroy and repair operators with a Random Variable Neighbourhood Descent and it also showed very good performance as compared to the other algorithms with 70% of the unique best solutions. The second optimisation we are using is the Stochastic optimisation algorithm because it explores a broad range of solutions and explores beyond the local space which can be advantageous for the job sequencing problem. In the paper [4], the study presented the two modified simulated annealing algorithms in which their performance is robust compared to taboo search in heuristics. The nature of the simulated annealing algorithm

is sequential, which works best for the flow shop sequencing problem. So, the simulated annealing algorithm is the best choice for the job scheduling problem.

In this report, we will apply two optimisation algorithms i.e., the greedy algorithm and the stochastic optimisation algorithm, particularly the Simulated Annealing algorithm which I will further explain in the next section and then summarize the results of this study in the evaluation section of this study.

II. Algorithms

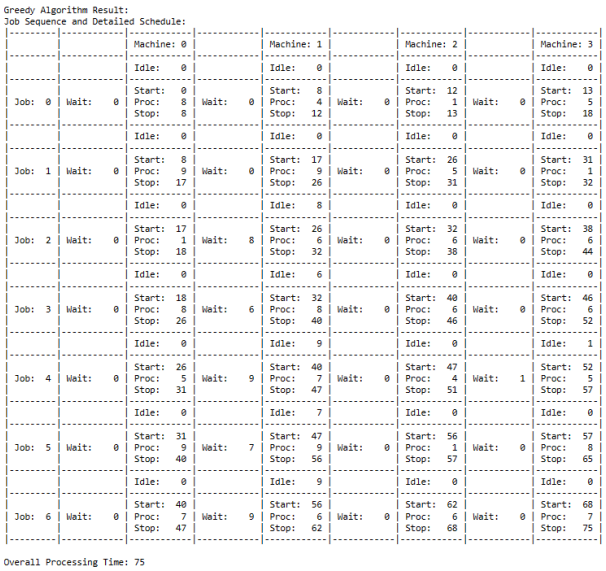
For this job scheduling, we have applied two optimisation algorithms i.e., the greedy algorithm and the stochastic algorithm for scheduling the jobs in an optimised way and then compare the results of the schedule obtained about the integer programming solution.

A. Greedy Algorithm

It is very simple and easy to understand and is mostly used in optimization problems such as scheduling the number of jobs to machines. It focuses on achieving locally optimal solutions at every step in the hope of finding a globally optimised solution [5]. For job scheduling, the greedy algorithm works in such a way that each which takes less processing time needs to be processed first and then in a sequential manner such that it reduces the criteria of waiting and idle time and allocates resources. The derivation of the greedy algorithm can be described in the following steps:

- **Definition of the problem-** This is the first step in which we define the criteria such as minimizing the waiting and idle time of the machine and the job. The main aim is to define the objective i.e., to minimize the idle and the waiting time of the machine.
- **Initialization of the algorithm-** The algorithm is first initialised with an empty schedule and unassigned jobs to the machines.
- **Choice of greed–** At this step of the algorithm the algorithm selects a job that minimizes the processing time when the job is scheduled to the next machine. This decision is based on a rule that prioritizes the jobs based on the processing time of the job.
- **Job assignment-** In this step once the task is chosen then it is assigned to a machine for the execution of the job that reduces the current idle time of the machine. This step is repeated and updates the job schedule and the idle time.
- **Termination of the task-** Once all the jobs are scheduled to the machines then the algorithm ends the final schedule solution is optimal and the idle time is minimized.

In this study, the author has utilised a greedy algorithm for scheduling the number of jobs i.e., N_JOBS = 7 on the number of machines i.e., N_MACHINES = 4 considering that minimizing the overall processing time of the machine concerning the jobs and build a schedule as shown in Fig. 1 considering the greedy choice as shortest time first i.e., which job will be executed first based on the processing time of the



Greedy Algorithm Result:
Job Sequence and Detailed Schedule:

		Machine: 0		Machine: 1		Machine: 2		Machine: 3
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 0	Wait: 0	Start: 0 Proc: 8 Stop: 8	Wait: 0	Start: 8 Proc: 4 Stop: 12	Wait: 0	Start: 12 Proc: 1 Stop: 13	Wait: 0	Start: 13 Proc: 5 Stop: 18
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 1	Wait: 0	Start: 8 Proc: 9 Stop: 17	Wait: 0	Start: 17 Proc: 9 Stop: 26	Wait: 0	Start: 26 Proc: 5 Stop: 31	Wait: 0	Start: 31 Proc: 1 Stop: 32
		Idle: 0		Idle: 8		Idle: 0		Idle: 0
Job: 2	Wait: 0	Start: 17 Proc: 1 Stop: 18	Wait: 8	Start: 26 Proc: 6 Stop: 32	Wait: 0	Start: 32 Proc: 6 Stop: 38	Wait: 0	Start: 38 Proc: 6 Stop: 44
		Idle: 0		Idle: 6		Idle: 0		Idle: 0
Job: 3	Wait: 0	Start: 18 Proc: 8 Stop: 26	Wait: 6	Start: 32 Proc: 8 Stop: 40	Wait: 0	Start: 40 Proc: 6 Stop: 46	Wait: 0	Start: 46 Proc: 6 Stop: 52
		Idle: 0		Idle: 9		Idle: 0		Idle: 1
Job: 4	Wait: 0	Start: 26 Proc: 5 Stop: 31	Wait: 9	Start: 40 Proc: 7 Stop: 47	Wait: 0	Start: 47 Proc: 4 Stop: 51	Wait: 1	Start: 52 Proc: 5 Stop: 57
		Idle: 0		Idle: 7		Idle: 0		Idle: 0
Job: 5	Wait: 0	Start: 31 Proc: 9 Stop: 40	Wait: 7	Start: 47 Proc: 9 Stop: 56	Wait: 0	Start: 56 Proc: 1 Stop: 57	Wait: 0	Start: 57 Proc: 8 Stop: 65
		Idle: 0		Idle: 9		Idle: 0		Idle: 0
Job: 6	Wait: 0	Start: 40 Proc: 7 Stop: 47	Wait: 9	Start: 56 Proc: 6 Stop: 62	Wait: 0	Start: 62 Proc: 6 Stop: 68	Wait: 0	Start: 68 Proc: 7 Stop: 75

Overall Processing Time: 75

Fig. 1. Greedy Algorithm Job Schedule

job and evaluated the overall processing time i.e., 75 units of time and as compared to integer programming solution it is giving 59 units of time which were less than the overall processing time of the greedy algorithm.

B. Simulated Annealing Algorithm (Stochastic Optimisation)

Simulated Annealing algorithm is a type of stochastic optimisation algorithm as the algorithm is based on metallurgy and statistical mechanics concepts. This technique is inspired by the behaviour of the cooling process of physical systems [6]. It is a heuristic-based approach which is focused on finding the global optimal solution for job scheduling problems. Simulated annealing is a physical process of annealing where the material is cooled down slowly until it reaches its lowest energy state, by repeatedly exploring the search space and accepting the solutions with decreasing probability.

The derivation of Simulated Annealing algorithm is described as follows:

- **Definition of an objective function:** In a simulated annealing algorithm, we must first define an objective function that defines the quality of the solution. In terms of job scheduling the objective function can be minimizing the overall processing time.
- **Initialisation of the solution:** In this step, it provides an initial solution which was given using heuristics methods which is the starting point of the algorithm.
- Schedule of Annealing: The schedule defines the rate of the algorithm at which the tries to explore the search space and accepts the solution.
- **Acceptance of the solution:** This algorithm improves the previous solution at each repetition to give another solution. If we adjust the order of the tasks or exchange them between machines it can cause problems. After that, this algorithm utilizes the objective function to check

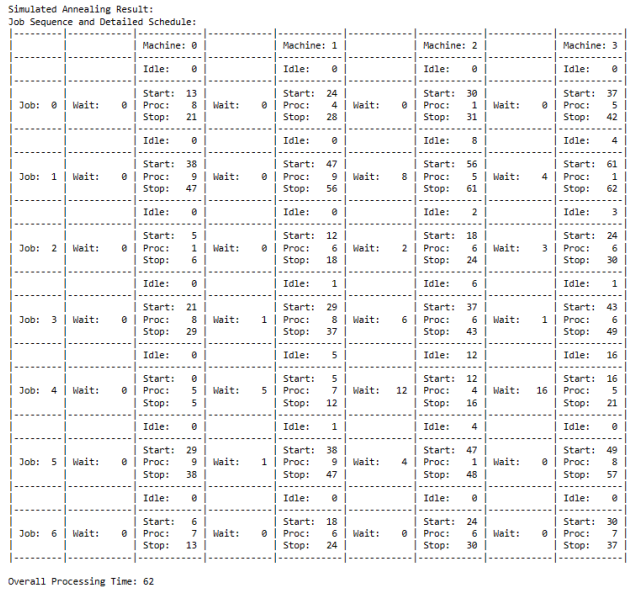


Fig. 2. Simulated Annealing Algorithm Schedule

the proposed solution's quality. The candidate solution is always accepted if it performs better than the existing solution.

- **Termination:** If a termination condition is satisfied, the algorithm keeps searching the solution space and accepting less-than-ideal solutions following the annealing schedule.

In this study, the author has utilised this simulated annealing algorithm as a stochastic optimisation technique for scheduling the jobs as per the objective function i.e., to minimize the overall processing time and provide the initial solution calculated the total makespan time for each of the candidate solutions and check it with the previous solution whether it is improved from the previous one until the optimal schedule is obtained. After that, the overall processing time is 62 units of time which is higher than the integer programming solution i.e., 59 units of time but less than the Greedy algorithm solution. So, as compared to the integer programming solution in which the total processing time is less than the results of simulated annealing which can be shown in Figure 2.

III. Evaluation

In this study, we have done a comparative analysis of both the optimisation algorithms i.e., the Greedy algorithm and Simulated Annealing algorithm for scheduling the jobs to minimize the overall completion of time of all the jobs in all the machines. When we have calculated the overall completion time of the Simulated Annealing algorithm it is found to be 62 units of time and the other one i.e., the Greedy algorithm has an overall completion time of 75 units of time. As we can see the simulated annealing algorithm has performed better than the greedy algorithm as it can explore the search space escape the local optimal solution and focus on the global optimal solution. Finally, when we compare the overall completion time of the Greedy algorithm and the Simulated Annealing algorithm with the integer programming solution, we have got the best results when we have done this job scheduling problem we have got the best result i.e., 59 units of time.

In the future, this study can be extended further to improve the results of the optimisation techniques as the simulated annealing algorithm can improve its results by fine-tuning its parameters such as initial temperature, cooling rate and the job schedule to improve the results of this study we can also combine the simulated annealing algorithm with the other optimisation algorithms which can lead to improving performance and quality of the solution.

References

[1] J. Li, L. Feng, and S. Fang, "An Greedy-Based Job Scheduling Algorithm in Cloud Computing," Journal of Software, vol. 9, no. 4, Apr. 2014, doi: 10.4304/jsw.9.4.921-925. [Online]. Available: http://dx.doi.org/10.4304/jsw.9.4.921-925

[2] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flow-shop scheduling problem," European Journal of Operational Research, vol. 177, no. 3, pp. 2033–2049, Mar. 2007, doi: 10.1016/j.ejor.2005.12.009. [Online]. Available: http://dx.doi.org/10.1016/j.ejor.2005.12.009

[3] D. Mecler, V. Abu-Marrul, R. Martinelli, and A. Hoff, "Iterated greedy algorithms for a complex parallel machine scheduling problem," European Journal of Operational Research, vol. 300, no. 2, pp. 545–560, Jul. 2022, doi: 10.1016/j.ejor.2021.08.005. [Online]. Available: http://dx.doi.org/10.1016/j.ejor.2021.08.005

[4] G. S. Paiva and M. A. M. Carvalho, "Improved heuristic algorithms for the Job Sequencing and Tool Switching Problem," Computers Operations Research, vol. 88, pp. 208–219, Dec. 2017, doi: 10.1016/j.cor.2017.07.013. [Online]. Available: http://dx.doi.org/10.1016/j.cor.2017.07.013

[5] GeeksforGeeks, "Greedy Algorithms," GeeksforGeeks, May 02, 2024. [Online]. Available: https://www.geeksforgeeks.org/greedy-algorithms/

[6] C. Gallo and V. Capozzi, "A Simulated Annealing Algorithm for Scheduling Problems," Journal of Applied Mathematics and Physics, vol. 07, no. 11, pp. 2579–2594, 2019, doi: 10.4236/jamp.2019.711176. [Online]. Available: http://dx.doi.org/10.4236/jamp.2019.711176