Evaluation of Algorithm Performance for the Job Scheduling Problem

Modelling, Simulation and Optimisation (MSCDAD_C)

Debayan Biswas
Student ID: 22242821
Master of Science in Data Analytics
National College of Ireland
x22242821@student.ncirl.ie

***Abstract*—This project report discusses the complexities present in the job scheduling problem to significantly reduce the overall completion time by efficiently assigning tasks to the machines and jobs. The study showcases the ways of overcoming the associated problems of job scheduling by the application of two specific algorithms: greedy algorithm and stochastic algorithm. These two algorithms are considered due to their efficiency in solving complex optimisation problems. The study aims to assess how well the algorithms work and perform in relation to an integer programming solution that is provided. The total completion time, runtime, and relative accuracy of the algorithms will be calculated and a thorough comparative analysis will be performed. This analysis will also reveal certain advantages and drawbacks of the algorithms used for this purpose. The analysis will also open the door to the effective and efficient application of job scheduling algorithms to be applicable in real-world scenarios. Furthermore, the study will also explore ways to improve the execution time of these algorithms based on the provided scenario. By analyzing methods for enhancing the algorithms, the study aims to open new windows for advancing job scheduling methods.**

Keywords– job scheduling, greedy algorithm, stochastic, integer programming, optimisation

[illegible]

I. Introduction

Job scheduling is a technique for the allocation of tasks to a set of machines where different tasks are executed at a defined time on a particular job. The objective of a job scheduling problem is to optimise the total task completion time, reducing runtime and increasing accuracy with proper resource utilization. The job or task scheduling problem is the most significant industrial problem majorly in the manufacturing planning sector [1]. The increase in daily operational complexities also increases the scheduling complexity and hence there is a need for an optimisation of these events.

The motivation for the job scheduling problem lies in the prevalent nature of the challenges and complexities associated with job scheduling. [2] states that these challenges can bring high costs and delays to the operations and there needs to be a method to minimize it by simultaneously selecting and scheduling machines in parallel. There is a possibility of multiple approaches with the end goal of reducing makespan and run time thereby generating profit. This report brings forward two different optimisation algorithms to address the problem of job scheduling. The methods are the greedy algorithm and the genetic algorithm. These methods are selected based on their capabilities in addressing optimisation tasks across diverse sectors such as manufacturing, task scheduling, and service sectors. The job scheduling problem, also known as NP-hard (nondeterministic polynomial time), is a challenging computational problem to solve [3].

The greedy algorithm provides a basic straightforward approach that focuses on optimizing choices at each decision point to find the overall optimal choice thereby solving the entire problem. The greedy algorithm belongs to the class of heuristic algorithms, that helps to solve problems quicker instead of extensively searching for the perfect solution to the problem. [4] shows the application of an iterated greedy algorithm that solves the computational complexity of a job scheduling problem, and how it effectively outperformed other meta-heuristic methods. This stands as a motivation for the application of greedy algorithm in this project.

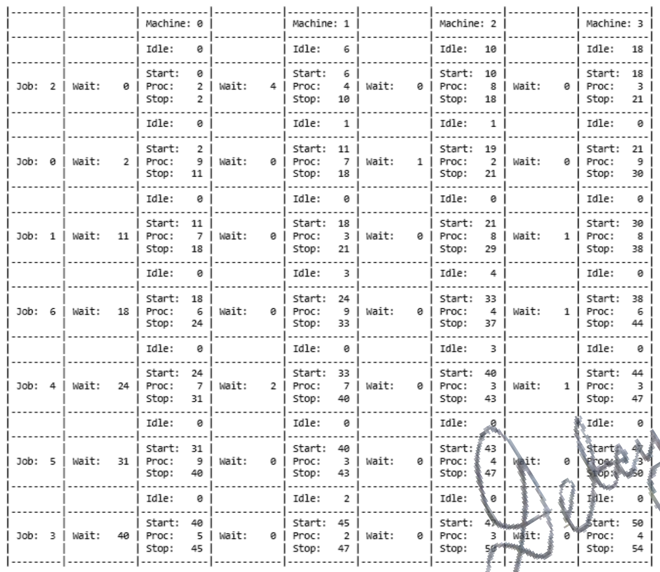
The stochastic algorithm provides randomness in the optimisation process. Stochastic algorithms can solve complex optimisation problems by exploring probabilistic techniques. The application of the Stochastic algorithm draws inspiration from [5] which shows the probabilistic approach of the models in classifying the distribution of makespan variation for scheduling bag-of-words in Hybrid cloud computing. The results outperform all other algorithms used to generate maximum profit and optimise task handling.

Both Greedy and Stochastic algorithms maintain a balance between efficiency and the quality of the obtained solution. This project aims to evaluate the performance of both algorithms in solving the Job scheduling problem to better understand task allocation and scheduling. The knowledge gained will ultimately help in the advancement of advanced job scheduling techniques in the future.

The next section of the paper talks about the Algorithm II section that discusses the outline and results of two algorithms, greedy and stochastic along with their positive and negative aspects. The next section is followed by the Evaluation III section that statistically evaluates the results obtained from the algorithms and also briefly discusses the scope of further improvement.

II. Algorithms

This study discusses job scheduling optimisation by the application of two selected optimisation algorithms, greedy and stochastic algorithm with the provided integer programming algorithm that is set as a benchmark for evaluation. The integer programming algorithm schedules jobs on different available machines keeping makespan at a minimum. The schedule for the integer programming with a seed value of the last 4 digits of student ID 22242821 can be seen in Figure 1 that obtained a makespan value of 54. The schedule is generated perfectly with the current job moving to the previous job only after the successful completion of the current job by all the machines. The greedy and stochastic algorithm will be applied to compare the makespan, and runtime and calculate the relative accuracy.



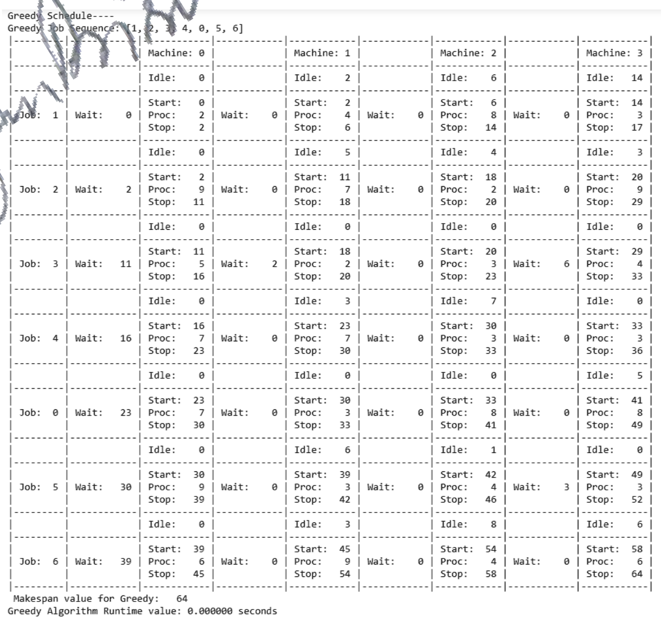
```
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Machine: 0 |           | Machine: 1 |           | Machine: 2 |           | Machine: 3 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    6 |           | Idle:   10 |           | Idle:   18 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:   0 |           | Start:   6 |           | Start:  10 |           | Start:  18 |
| Job:  2 | Wait:   0 | Proc:    2 | Wait:   4 | Proc:    4 | Wait:   0 | Proc:    8 | Wait:   0 | Proc:    3 |
|         |           | Stop:    2 |           | Stop:   10 |           | Stop:   18 |           | Stop:   21 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    1 |           | Idle:    1 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:   2 |           | Start:  11 |           | Start:  19 |           | Start:  21 |
| Job:  0 | Wait:   2 | Proc:    9 | Wait:   0 | Proc:    7 | Wait:   1 | Proc:    2 | Wait:   0 | Proc:    9 |
|         |           | Stop:   11 |           | Stop:   18 |           | Stop:   21 |           | Stop:   30 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    0 |           | Idle:    0 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  11 |           | Start:  18 |           | Start:  21 |           | Start:  30 |
| Job:  1 | Wait:  11 | Proc:    7 | Wait:   0 | Proc:    3 | Wait:   0 | Proc:    8 | Wait:   1 | Proc:    8 |
|         |           | Stop:   18 |           | Stop:   21 |           | Stop:   29 |           | Stop:   38 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    3 |           | Idle:    4 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  18 |           | Start:  24 |           | Start:  33 |           | Start:  38 |
| Job:  6 | Wait:  18 | Proc:    6 | Wait:   0 | Proc:    9 | Wait:   0 | Proc:    4 | Wait:   1 | Proc:    6 |
|         |           | Stop:   24 |           | Stop:   33 |           | Stop:   37 |           | Stop:   44 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    0 |           | Idle:    3 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  24 |           | Start:  33 |           | Start:  40 |           | Start:  44 |
| Job:  4 | Wait:  24 | Proc:    7 | Wait:   2 | Proc:    7 | Wait:   0 | Proc:    3 | Wait:   1 | Proc:    3 |
|         |           | Stop:   31 |           | Stop:   40 |           | Stop:   43 |           | Stop:   47 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    0 |           | Idle:    0 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  31 |           | Start:  40 |           | Start:  43 |           | Start:  47 |
| Job:  5 | Wait:  31 | Proc:    9 | Wait:   0 | Proc:    3 | Wait:   0 | Proc:    4 | Wait:   0 | Proc:    3 |
|         |           | Stop:   40 |           | Stop:   43 |           | Stop:   47 |           | Stop:   50 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    2 |           | Idle:    0 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  40 |           | Start:  45 |           | Start:  47 |           | Start:  50 |
| Job:  3 | Wait:  40 | Proc:    5 | Wait:   0 | Proc:    2 | Wait:   0 | Proc:    3 | Wait:   0 | Proc:    4 |
|         |           | Stop:   45 |           | Stop:   47 |           | Stop:   50 |           | Stop:   54 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
```

Fig. 1: Integer Programming Schedule

A. Greedy Algorithm

A greedy algorithm is a problem-solving heuristic algorithm that makes the local optimal choice at every step to find a global optimum solution. The greedy algorithm selects the job with the shortest processing time on the next available machine. The Greedy Algorithm can provide accurate solutions for certain scheduling problems. [6] discusses the application of an iterated greedy algorithm for non-permutation flow shop scheduling of jobs that are processed by sequential machines to reduce scheduling complexities. The aim of the study revolves around minimizing total completion time. The evaluation of the proposed model outperforms all the existing methods applied to flow shop scheduling. This stands as a motivation for the use of greedy scheduling as greedy performs accurately with less completion time. The approach used in this project is similar to one but uses a heuristic approach rather than a meta-heuristic approach as seen in the paper. Heuristic tends to be faster as it chooses local optimal points at each step without considering the whole solution and hence has lower computational overhead and faster execution.

A constructive greedy algorithm is used in this project for step-by-step decision-making to find an optimal solution. The jobs are selected on criteria such as minimizing the completion time and runtime of each machine before moving to the next available job. [7] defines the steps for creating a greedy algorithm involves defining the job scheduling problem and then identifying the local optimal choice at every step. This is followed by selecting the identified state and updating the present state. Gradually the steps will be repeated for each step until an overall optimal solution is reached. The schedule in Figure 2 shows the complete job schedule of the greedy approach. The greedy with a seed value for the last 4 digits of student id 22242821, produces a makespan value of 64 units with a relative accuracy of 81.48% when compared to the integer programming solution that is provided. This states that the makespan of the greedy algorithm is approximately 81.48% efficient as the makespan produced by the integer programming. The makespan for greedy is higher than that of integer programming however greedy has a runtime of 0.0 seconds which shows exceptional computation speed of greedy. Greedy is a simplistic and fast approach algorithm that may have limitations in exploring more diverse spaces which meta-heuristic algorithms can venture into.



```
Greedy Schedule----
Greedy Job Sequence: [1, 2, 3, 4, 0, 5, 6]
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Machine: 0 |           | Machine: 1 |           | Machine: 2 |           | Machine: 3 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    2 |           | Idle:    6 |           | Idle:   14 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:   0 |           | Start:   2 |           | Start:   6 |           | Start:  14 |
| Job:  1 | Wait:   0 | Proc:    2 | Wait:   0 | Proc:    4 | Wait:   0 | Proc:    8 | Wait:   0 | Proc:    3 |
|         |           | Stop:    2 |           | Stop:    6 |           | Stop:   14 |           | Stop:   17 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    5 |           | Idle:    4 |           | Idle:    3 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:   2 |           | Start:  11 |           | Start:  18 |           | Start:  20 |
| Job:  2 | Wait:   2 | Proc:    9 | Wait:   0 | Proc:    7 | Wait:   0 | Proc:    2 | Wait:   0 | Proc:    9 |
|         |           | Stop:   11 |           | Stop:   18 |           | Stop:   20 |           | Stop:   29 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    0 |           | Idle:    0 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  11 |           | Start:  18 |           | Start:  20 |           | Start:  29 |
| Job:  3 | Wait:  11 | Proc:    5 | Wait:   2 | Proc:    2 | Wait:   0 | Proc:    3 | Wait:   6 | Proc:    4 |
|         |           | Stop:   16 |           | Stop:   20 |           | Stop:   23 |           | Stop:   33 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    3 |           | Idle:    7 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  16 |           | Start:  23 |           | Start:  30 |           | Start:  33 |
| Job:  4 | Wait:  16 | Proc:    7 | Wait:   0 | Proc:    7 | Wait:   0 | Proc:    3 | Wait:   0 | Proc:    3 |
|         |           | Stop:   23 |           | Stop:   30 |           | Stop:   33 |           | Stop:   36 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    0 |           | Idle:    0 |           | Idle:    5 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  23 |           | Start:  30 |           | Start:  33 |           | Start:  41 |
| Job:  0 | Wait:  23 | Proc:    7 | Wait:   0 | Proc:    3 | Wait:   0 | Proc:    8 | Wait:   0 | Proc:    8 |
|         |           | Stop:   30 |           | Stop:   33 |           | Stop:   41 |           | Stop:   49 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    6 |           | Idle:    1 |           | Idle:    0 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  30 |           | Start:  39 |           | Start:  42 |           | Start:  49 |
| Job:  5 | Wait:  30 | Proc:    9 | Wait:   0 | Proc:    3 | Wait:   0 | Proc:    4 | Wait:   3 | Proc:    3 |
|         |           | Stop:   39 |           | Stop:   42 |           | Stop:   46 |           | Stop:   52 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Idle:    0 |           | Idle:    3 |           | Idle:    8 |           | Idle:    6 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
|         |           | Start:  39 |           | Start:  45 |           | Start:  54 |           | Start:  58 |
| Job:  6 | Wait:  39 | Proc:    6 | Wait:   0 | Proc:    9 | Wait:   0 | Proc:    4 | Wait:   0 | Proc:    6 |
|         |           | Stop:   45 |           | Stop:   54 |           | Stop:   58 |           | Stop:   64 |
|---------|-----------|------------|-----------|------------|-----------|------------|-----------|------------|
 Makespan value for Greedy:   64
Greedy Algorithm Runtime value: 0.000000 seconds
```

Fig. 2: Greedy Algorithm Schedule

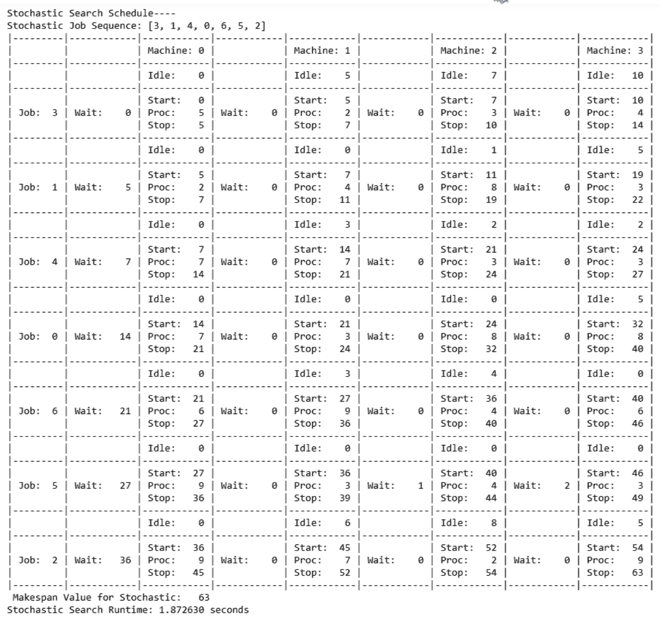
B. Stochastic Optimisation

Stochastic optimisation brings randomness to the job scheduling process and allows the exploration of multiple possible solutions by considering multiple machines and jobs. The algorithm selects the job with the lowest makespan out of the possible solutions obtained. The study [8] presents a simulated annealing-based algorithm for a flexible job shop scheduling problem (FJSSP) which is accelerated using partial scheduling (Partial-S) rather than total scheduling (Total-S). Simulated annealing is a stochastic global search algorithm used for optimisation purposes. The paper details that the results of Partial-S have better efficiency than Total-S when accelerated. An optimised output was obtained in a much shorter time with the use of partial scheduling thus lowering the makespan and runtime. This also allows the simulated annealing to boost the local search to explore over a larger space. The use of the stochastic approach thus provided faster and better solutions with improved efficiency. However, the use of Total-S might make the optimisation process longer with more computation time. The study draws the motivation for using Stochastic algorithm-based simulated annealing scheduling as the algorithm is capable of generating an efficient schedule with lower completion time with boosted local search capabilities.

The stochastic simulated annealing is implemented here in the project based on the study of literature. The name annealing is used for the similarity of the annealing process in metallurgy, where metal is quickly heated and cooled slowly to remove defects [9]. The same process works as a foundation of this algorithm. The process starts with determining the initial temperature to explore the solution area followed by the neighbor generation function that randomly swaps between two jobs. The acceptance probability determines the acceptance or rejection of neighbor solutions based on cost difference and temperature. The cooling rate mimics the slow cooling process in annealing that allows the convergence of solution towards optimal as temperature decreases.

The schedule in Figure 3 shows the complete job schedule of the Stochastic simulated annealing optimisation approach. The seed value of the last 4 digits of student id 22242821 is passed into the algorithm thus generating a makespan value of 63 units with a relative accuracy of 83.33% when compared to the integer programming solution. This means that the algorithm is around 83.33% efficient as compared to integer programming. The makespan for stochastic is higher than that of integer programming with a runtime of 1.872630 seconds. The simulated annealing form of the stochastic algorithm is still a fast approach algorithm while being a probabilistic algorithm. Though being a robust algorithm, certain limitations may arise such as convergence issues and higher runtime due to various factors present within the algorithm parameters. The stochastic optimisation being more complex than the greedy algorithm, still demonstrates better adaptability leading to better all-around performance.

[illegible]

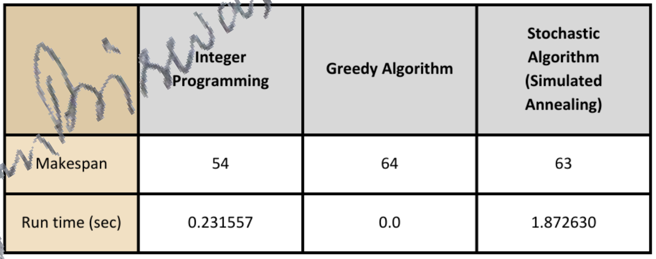


```
Stochastic Search Schedule----
Stochastic Job Sequence: [3, 1, 4, 0, 6, 5, 2]
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Machine: 0 |            | Machine: 1 |            | Machine: 2 |            | Machine: 3 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    5 |            | Idle:    7 |            | Idle:   10 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:   0 |            | Start:   5 |            | Start:   7 |            | Start:  10 |
| Job:  3 | Wait:    0 | Proc:    5 | Wait:    0 | Proc:    2 | Wait:    0 | Proc:    3 | Wait:    0 | Proc:    4 |
|         |            | Stop:    5 |            | Stop:    7 |            | Stop:   10 |            | Stop:   14 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    1 |            | Idle:    5 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:   5 |            | Start:   7 |            | Start:  11 |            | Start:  19 |
| Job:  1 | Wait:    5 | Proc:    2 | Wait:    0 | Proc:    4 | Wait:    0 | Proc:    8 | Wait:    0 | Proc:    3 |
|         |            | Stop:    7 |            | Stop:   11 |            | Stop:   19 |            | Stop:   22 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    3 |            | Idle:    2 |            | Idle:    2 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:   7 |            | Start:  14 |            | Start:  21 |            | Start:  24 |
| Job:  4 | Wait:    7 | Proc:    7 | Wait:    0 | Proc:    7 | Wait:    0 | Proc:    3 | Wait:    0 | Proc:    3 |
|         |            | Stop:   14 |            | Stop:   21 |            | Stop:   24 |            | Stop:   27 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |            | Idle:    5 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  14 |            | Start:  21 |            | Start:  24 |            | Start:  32 |
| Job:  0 | Wait:   14 | Proc:    7 | Wait:    0 | Proc:    3 | Wait:    0 | Proc:    8 | Wait:    0 | Proc:    8 |
|         |            | Stop:   21 |            | Stop:   24 |            | Stop:   32 |            | Stop:   40 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    3 |            | Idle:    4 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  21 |            | Start:  27 |            | Start:  36 |            | Start:  40 |
| Job:  6 | Wait:   21 | Proc:    6 | Wait:    0 | Proc:    9 | Wait:    0 | Proc:    4 | Wait:    0 | Proc:    6 |
|         |            | Stop:   27 |            | Stop:   36 |            | Stop:   40 |            | Stop:   46 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  27 |            | Start:  36 |            | Start:  40 |            | Start:  46 |
| Job:  5 | Wait:   27 | Proc:    9 | Wait:    0 | Proc:    3 | Wait:    1 | Proc:    4 | Wait:    2 | Proc:    3 |
|         |            | Stop:   36 |            | Stop:   39 |            | Stop:   44 |            | Stop:   49 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    6 |            | Idle:    8 |            | Idle:    5 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  36 |            | Start:  45 |            | Start:  52 |            | Start:  54 |
| Job:  2 | Wait:   36 | Proc:    9 | Wait:    0 | Proc:    7 | Wait:    0 | Proc:    2 | Wait:    0 | Proc:    9 |
|         |            | Stop:   45 |            | Stop:   52 |            | Stop:   54 |            | Stop:   63 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
 Makespan Value for Stochastic:   63
Stochastic Search Runtime: 1.872630 seconds
```

Fig. 3: Stochastic Algorithm Schedule

III. Evaluation

The performance of each algorithm for the job scheduling problem is evaluated based on their makespan and runtime, and their relative accuracy is compared to the integer programming solution provided in the Jupyter Notebook. The results show that each algorithm has certain advantages and limitations in solving the Job scheduling problem which can be seen in Figure 4.



	Integer Programming	Greedy Algorithm	Stochastic Algorithm (Simulated Annealing)
Makespan	54	64	63
Run time (sec)	0.231557	0.0	1.872630

Fig. 4: Comparison

The makespan generated by integer programming is 54 units, while greedy and stochastic generate a makespan of 64 and 63 units respectively. This suggests that integer programming is the most efficient in minimizing the makespan and finding the optimal solution. The stochastic falls between greedy and integer programming indicating better performance than the greedy algorithm but slightly worse than the Integer Programming solution in terms of makespan. Integer programming executes with a runtime of 0.231557 seconds which is relatively low compared to the optimal result obtained and the level of complexity present. The greedy approach offers a fast execution time of 0.0 seconds but offers more makespan. The runtime for stochastic is at 1.872630 seconds which is more than the other algorithms. This is because of the fact that the stochastic approach involves random sampling and complex computation because of the use of simulated annealing. The relative accuracy of both the greedy and stochastic models compared to integer programming stand at 81.48% and 83.33% respectively. The greedy algorithm despite its fast execution speed sacrifices the degree of accuracy. The stochastic on the other hand has slightly better-related accuracy which suggests that the stochastic produces results that are closer to the optimal solution despite executing with more runtime.

Factors such as the degree of the optimal solution, makespan, and run-time largely influence the algorithm choices. From the findings of the project, it can be concluded that despite the speed, the greedy algorithm may be more suitable for situations where the optimal solution is a less concerned aspect whereas, stochastic may provide an acceptable optimal solution with little computational expense because of its internal complexities. The integer programming ensures an optimal solution along with far fewer computational resources.

The algorithms although have performed well have certain limitations which can be improved to make them even better. Further fine-tuning and adjusting the algorithm parameters may lead to improved results for optimisation. The implementation of parallel processing can further optimise the solution and run time by simultaneously running the jobs along different machines. Using an ensemble approach can also boost the algorithm's strength and efficiency and make the model more robust in performance. There lies ample scope for refinements in dealing with the optimisation algorithms in the future.

References

[1] A. Arisha, P. Young, and M. E. Baradie, "Job Shop Scheduling Problem: an Overview," *International Conference for Flexible Automation and Intelligent Manufacturing (FAIM 01),Dublin, Ireland, July 2001, pp 682 – 693.*, 1 2001.

[2] D. Cao, M. Chen, and G. Wan, "Parallel machine selection and job scheduling to minimize machine cost and job tardiness," *Computers Operations Research*, vol. 32, no. 8, pp. 1995–2012, 2005.

[3] "Explained: P vs. NP," 10 2009.

[4] J. E. C. Arroyo, J. Y.-T. Leung, and R. G. Tavares, "An iterated greedy algorithm for total flow time minimization in unrelated parallel batch machines with unequal job release times," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 239–254, 2019.

[5] L. Yin, J. Zhou, and J. Sun, "A stochastic algorithm for scheduling bag-of-tasks applications on hybrid clouds under task duration variations," *Journal of Systems and Software*, vol. 184, p. 111123, 2022.

[6] A. Brum, R. Ruiz, and M. Ritt, "Automatic generation of iterated greedy algorithms for the non-permutation flow shop scheduling problem with total completion time minimization," *Computers Industrial Engineering*, vol. 163, p. 107843, 2022.

[7] A. Butch, "What is Greedy Algorithm: Example, Applications, Limitations and More," 2 2023.

[8] M. A. Cruz-Chávez, M. G. Martínez-Rangel, and M. H. Cruz-Rosales, "Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem," *International transactions in operational research*, vol. 24, pp. 1119–1137, 7 2015.

[9] GeeksforGeeks, "Simulated annealing," 4 2024.

[illegible]