Optimizing The Job Scheduling: A Greedy and Stochastic Approach

Vipin Sharma
Dept. of Computing
National College of Ireland
Dublin, Ireland
x22207406@student.ncirl.ie

***Abstract*—This paper explores the different types of job scheduling Problems (JSP). The JSP is an important method used in many different sectors to maximize the use of resources and increase the profit or output. Using the different job scheduling Problems (JSP) and its use in real-world situations focusing on how it is commonly used across all the industries like large manufacturing units, computational work, and logistic work. The Greedy algorithm and the Simulated Annealing algorithm are used to solve the job scheduling Problems (JSP). The greedy algorithm is very popular for its simplicity and efficiency on the other side, the Simulated Annealing algorithm (SA) is flexible. This research gives insights that the Simulated Annealing algorithm performs well as compared to the greedy algorithm. The Processing time of the Simulated Annealing algorithm is 53 and the Greedy algorithm's total processing time is 56. Also, calculating the Start, Stop, Idle, and Wait time for both algorithms for a given problem by analyzing the advantages and disadvantages of both algorithms and also performing the comparison with the Integer programming algorithm, and suggesting ways to improve the algorithms and job scheduling problems like parameter tuning, or any different approach for future work.**

***Index Terms*—Greedy algorithm, Optimization, Simulated Annealing algorithm, Job Scheduling**

I. Introduction

Job Scheduling is the process of optimizing the work and increasing productivity in many different sectors. Many industries which include logistics, manufacturing, and computing are faced with the problem of how to utilize the resources properly so they get the optimum results. [1] The aim objective of the Job Scheduling problem (JSP) is to determine the best possible timetable for allocating shared resources to complete all the activities to minimize the total time required to complete all the tasks. Optimizing the machine schedules helps organizations to quickly respond to market requirements while minimizing production time and costs to maximize profits. Job Scheduling Problem (JSP) means to allocate the various sets of tasks to the machine within a given range of time and resources and then the JSP will determine the best solution for the given tasks in a given time range.

[2]In the JSP find that there are n number of jobs (j1, j2, j3…..j4) that should be finished with the given m number of machines (m1, m2, m3, m4……..mn). Every machine should perform a set of different tasks and they take a certain amount of time and each machine can process only one job at a time. The Flexible job shop scheduling problem started to be studied at the beginning of the 1990s and it is also an important extension of JSP where each operation is processed and there are one or more machines to choose from. The first challenge is to determine which machine is most appropriate for each process, and the second is to set up each machine's workpieces for the best possible processing order. Because it overcomes the limitations of unique resources, multiple machines can complete a given process, and each machine has a different processing time required, JSP becomes more in line with actual production [3].

The FJSP has become an individual shop scheduling issue rather than just another JSP extension due to mass customization and automation. Also, by extending the FJSP, this survey aims to demonstrate the variety of real-world scheduling problem situations that can be modeled. [4] There are many types of algorithms used to optimize the job scheduling problem each of the algorithms has its benefits and weaknesses. Some of the most used algorithms for job scheduling problems are Greedy Algorithms, Genetic algorithms, Ant Colony Optimization (ACO), Simulated Annealing, and Particle Swarm Optimization. Depending on the scheduling problem choose the right algorithm which is suitable and gives the optimal results.

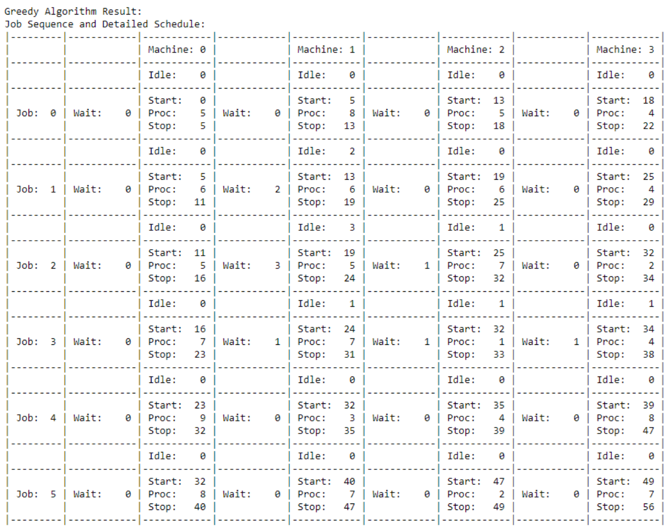
In this paper, the author implements two different important algorithms to solve the job scheduling problem (JSP): Greedy Algorithms and Stochastic Optimization (Simulated Annealing Algorithm). The motivation for choosing a greedy algorithm is that is a problem-solving strategy that tries to find the best option at every step. Greedy algorithms are characterized by their exclusive reliance on the information available at any given time, with no regard for the larger context or possible effects [5]. The second algorithm Simulated Annealing Algorithm was chosen for job scheduling problems because it is very simple and most commonly used for stochastic search and it will give the most efficient global optimum result for a given problem [6]. This introduction, Gives the fundamentals about the Job-Scheduling Problem and shows its importance and incidence in real-world situations. Also, explain the reasoning behind the choice of Greedy Algorithms and Simulated Annealing, citing important works that highlight their usefulness and suitability for addressing the JSP.

II. Algorithms

For the Job Scheduling Problem Use the two algorithms which define in the below subsection. The First algorithm is a Greedy Algorithm and the second algorithm is a Simulated Annealing Algorithm.

A. *Greedy Algorithm*

The Greedy Algorithm for the Job Scheduling Problem works by selecting options that are locally optimal at each step in the effort to find the best solution that is globally optimal. Within a Job Scheduling Problem (JSP), the Greedy Algorithm usually assigns tasks a priority according to predetermined standards, like processing time, idle time, deadline, or depending on different factors. To find the overall best solution, the greedy algorithm solves the problem by selecting the best option at every step. The greedy algorithm does not take the complete data it can divide the data into small segments and make the decision.



```
Greedy Algorithm Result:
Job Sequence and Detailed Schedule:
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Machine: 0 |            | Machine: 1 |            | Machine: 2 |            | Machine: 3 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:   0 |            | Start:   5 |            | Start:  13 |            | Start:  18 |
| Job:  0 | Wait:    0 | Proc:    5 | Wait:    0 | Proc:    8 | Wait:    0 | Proc:    5 | Wait:    0 | Proc:    4 |
|         |            | Stop:    5 |            | Stop:   13 |            | Stop:   18 |            | Stop:   22 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    2 |            | Idle:    0 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:   5 |            | Start:  13 |            | Start:  19 |            | Start:  25 |
| Job:  1 | Wait:    0 | Proc:    6 | Wait:    2 | Proc:    6 | Wait:    0 | Proc:    6 | Wait:    0 | Proc:    4 |
|         |            | Stop:   11 |            | Stop:   19 |            | Stop:   25 |            | Stop:   29 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    3 |            | Idle:    1 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  11 |            | Start:  19 |            | Start:  25 |            | Start:  32 |
| Job:  2 | Wait:    0 | Proc:    5 | Wait:    3 | Proc:    5 | Wait:    1 | Proc:    7 | Wait:    0 | Proc:    2 |
|         |            | Stop:   16 |            | Stop:   24 |            | Stop:   32 |            | Stop:   34 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    1 |            | Idle:    1 |            | Idle:    1 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  16 |            | Start:  24 |            | Start:  32 |            | Start:  34 |
| Job:  3 | Wait:    0 | Proc:    7 | Wait:    1 | Proc:    7 | Wait:    1 | Proc:    1 | Wait:    1 | Proc:    4 |
|         |            | Stop:   23 |            | Stop:   31 |            | Stop:   33 |            | Stop:   38 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  23 |            | Start:  32 |            | Start:  35 |            | Start:  39 |
| Job:  4 | Wait:    0 | Proc:    9 | Wait:    0 | Proc:    3 | Wait:    0 | Proc:    4 | Wait:    0 | Proc:    8 |
|         |            | Stop:   32 |            | Stop:   35 |            | Stop:   39 |            | Stop:   47 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  32 |            | Start:  40 |            | Start:  47 |            | Start:  49 |
| Job:  5 | Wait:    0 | Proc:    8 | Wait:    0 | Proc:    7 | Wait:    0 | Proc:    2 | Wait:    0 | Proc:    7 |
|         |            | Stop:   40 |            | Stop:   47 |            | Stop:   49 |            | Stop:   56 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
```

Fig. 1. Job Sequence scheduled for Greedy Algorithm

The important Steps for creating a Greedy Algorithm [6]:

1) Describe the Issue: In the first step define the problem completely which is to be solved and what the objective is to be optimized
2) Choose the greedy Option: Based on the current state discover the locally best option at every step.
3) Make a Greedy Decision: Update the state while selecting the greedy decision.
4) Again: Until the solution is not found keep making the decision based on greed.

Various algorithms help to understand the greedy algorithm [6]:

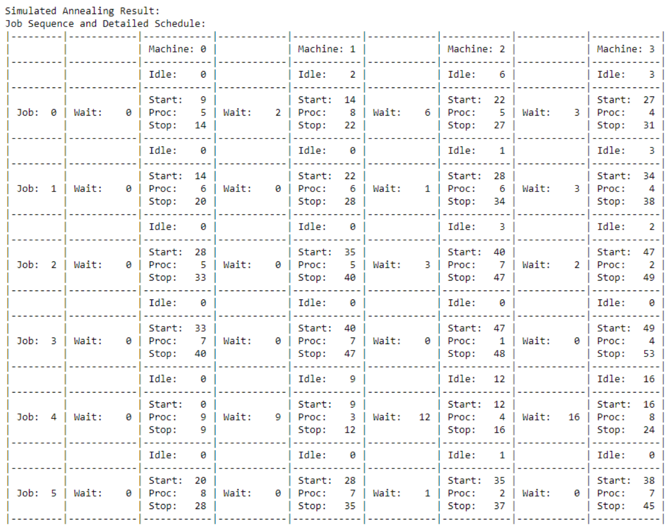
- **Fractional Knapsack:** Improves the value of objects that fit slightly within a knapsack with limited space.
- **Dijkstra algorithm:** This algorithm is used to find the shortest path from the source points to all the other points in a weighted graph
- **Kruskal algorithm:** Find the weighted graphs of the minimum spanning tree.

The are many reasons why Greedy is used in the Job scheduling problem (JSP). The greedy have the nature to be useful in different types of problems also are not always optimal but provide the best possible solution which is close to the best solution. It is finding the solution very fast with taking less time. If talk about the disadvantages of greedy there are some disadvantages like they do not give always an optimal solution and it is not suitable for all types of problems.

When applying the Greedy algorithm to the 6 jobs with 4 machines author found that the job start time is 0, and the last stop time is 56. The total processing time of this algorithm is 56 which is more if we compared this algorithm to Simulated Annealing Algorithm and Integer programming (IP) algorithm. The Idle and wait times is synchronized properly but we can conclude that this algorithm is not getting the best optimal job sequence result

B. *Stochastic Optimization (Simulated Annealing Algorithm)*

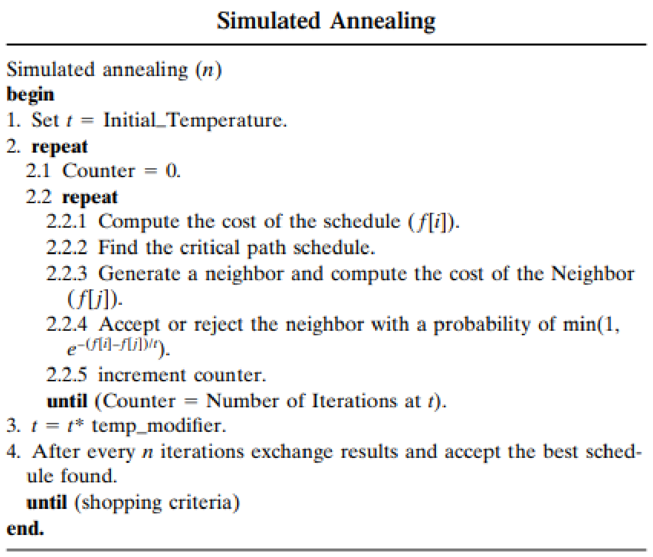
A stochastic optimization technique called ”Simulated Annealing” was inspired by the metallurgical annealing process. By accepting or rejecting candidate solutions by a probabilistic criterion, and iteratively studying the solution space, the algorithm gradually reduces the chance of accepting worse solutions as it goes as well. In the paper [7] author described that the simulated annealing algorithm prevents from getting stuck in a local minimum by additionally accepting cost-increasing neighbors with a certain probability.



```
Simulated Annealing Result:
Job Sequence and Detailed Schedule:
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Machine: 0 |            | Machine: 1 |            | Machine: 2 |            | Machine: 3 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    2 |            | Idle:    6 |            | Idle:    3 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:   9 |            | Start:  14 |            | Start:  22 |            | Start:  27 |
| Job:  0 | Wait:    0 | Proc:    5 | Wait:    2 | Proc:    8 | Wait:    6 | Proc:    5 | Wait:    3 | Proc:    4 |
|         |            | Stop:   14 |            | Stop:   22 |            | Stop:   27 |            | Stop:   31 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    1 |            | Idle:    3 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  14 |            | Start:  22 |            | Start:  28 |            | Start:  34 |
| Job:  1 | Wait:    0 | Proc:    6 | Wait:    0 | Proc:    6 | Wait:    1 | Proc:    6 | Wait:    3 | Proc:    4 |
|         |            | Stop:   20 |            | Stop:   28 |            | Stop:   34 |            | Stop:   38 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    3 |            | Idle:    2 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  28 |            | Start:  35 |            | Start:  40 |            | Start:  47 |
| Job:  2 | Wait:    0 | Proc:    5 | Wait:    0 | Proc:    5 | Wait:    3 | Proc:    7 | Wait:    2 | Proc:    2 |
|         |            | Stop:   33 |            | Stop:   40 |            | Stop:   47 |            | Stop:   49 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  33 |            | Start:  40 |            | Start:  47 |            | Start:  49 |
| Job:  3 | Wait:    0 | Proc:    7 | Wait:    0 | Proc:    7 | Wait:    0 | Proc:    1 | Wait:    0 | Proc:    4 |
|         |            | Stop:   40 |            | Stop:   47 |            | Stop:   48 |            | Stop:   53 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    9 |            | Idle:   12 |            | Idle:   16 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:   0 |            | Start:   9 |            | Start:  12 |            | Start:  16 |
| Job:  4 | Wait:    0 | Proc:    9 | Wait:    9 | Proc:    3 | Wait:   12 | Proc:    4 | Wait:   16 | Proc:    8 |
|         |            | Stop:    9 |            | Stop:   12 |            | Stop:   16 |            | Stop:   24 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Idle:    0 |            | Idle:    0 |            | Idle:    1 |            | Idle:    0 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
|         |            | Start:  20 |            | Start:  28 |            | Start:  35 |            | Start:  38 |
| Job:  5 | Wait:    0 | Proc:    8 | Wait:    0 | Proc:    7 | Wait:    1 | Proc:    2 | Wait:    0 | Proc:    7 |
|         |            | Stop:   28 |            | Stop:   35 |            | Stop:   37 |            | Stop:   45 |
|---------|------------|------------|------------|------------|------------|------------|------------|------------|
```

Fig. 2. Job Sequence scheduled for Simulated Annealing Algorithm

First, an initial solution is randomly generated in Simulated Annealing. Next, a neighbor is found and accepted with a probability of min(1, exp(2d/T)), where T is the control parameter that corresponds to the physical analogy’s temperature and will be referred to as temperature, and d is the cost difference. The algorithm merges to the global minimum with a slow



```
Simulated Annealing
Simulated annealing ($n$)
begin
1. Set $t$ = Initial_Temperature.
2. repeat
  2.1 Counter = 0.
  2.2 repeat
    2.2.1 Compute the cost of the schedule ($f[i]$).
    2.2.2 Find the critical path schedule.
    2.2.3 Generate a neighbor and compute the cost of the Neighbor
      ($f[j]$).
    2.2.4 Accept or reject the neighbor with a probability of min(1,
      $e^{-(f[i]-f[j])/t}$).
    2.2.5 increment counter.
    until (Counter = Number of Iterations at $t$).
3. $t = t$* temp_modifier.
4. After every $n$ iterations exchange results and accept the best sched-
  ule found.
  until (shopping criteria)
end.
```

Fig. 3. [7]Simulated Annealing Algorithm

reduction in temperature, but it takes much longer. Because it is sequential by nature simulated annealing is very slow for the problem that has the big space. The initial schedule goes through to the simulated annealing algorithm. Making and analyzing the neighborhood schedules enables the algorithm to improve upon the original plan. The algorithm approaches an almost ideal solution as the temperature goes down gradually.

When applying the Simulated Annealing algorithm on the same number of jobs which is 6 jobs and 4 machines author found that the job start time is 0 and the last stop time is 53 and the sequence of the jobs is not changed because of the sequential nature of Simulated Annealing algorithm and the optimal job schedule sequence is also not changed and shows sequentially [1, 2, 3, 4, 5, 6] with different idle and wait time and In the integer programming the Optimal job sequence is [0, 1, 4, 5, 3, 2] and the total processing time 52. It means both are working differently and the conclusion comes that integer programming gives a better result than the Simulated Annealing algorithm. Maybe In the future by performing some parameters It would give a good result as compared to the IP programming.

III. Evaluation

The Greedy Algorithm and the Simulated Annealing Algorithm are two different algorithms that we studied in this report for solving the Job Scheduling Problem (JSP). The performance of each algorithm has been determined when compared to the Integer Programming (IP) Solution provided in the JobScheduling IP-Solution.ipynb notebook. Each algorithm has its advantages and disadvantages. Greedy Algorithms and Simulated Annealing are both commonly used for job scheduling problems. Greedy algorithms are very simple and effective, but in complex situations, they might not be able to produce globally optimal solutions as discussed in the Greedy Algorithm section. On the other hand, Simulated Annealing provides a simple and easy method of exploring solution spaces. When calculating the Total processing time for both the algorithms found that the Simulated Annealing algorithm takes less processing time which is 53 as compared to the Greedy algorithm which takes a total processing time is 56. But when comparing the Simulated Annealing algorithm with the given Integer programming(IP) algorithm then found that the IP algorithm takes less processing time and gives the best job scheduling results. There are some methods like parameter tuning, hybrid approaches, and problem-specific heuristics that can be used to improve these implementations and give the ensuring of improved performance and scalability when addressing real-world job scheduling issues.

References

[1] Job shop scheduling - Cornell University Computational Optimization Open Textbook - Optimization Wiki. [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Job_shop_scheduling#cite_note-:0-2

[2] D. Y. Sha and C.-Y. Hsu, "A hybrid particle swarm optimization for job shop scheduling problem," Computers industrial engineering, Dec. 01, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835206001471

[3] "An Improved Ant Colony Algorithm is Proposed to Solve the Single Objective Flexible Job-shop Scheduling Problem," IEEE Conference Publication — IEEE Xplore, Nov. 20, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9305005

[4] S. Dauzère-Pérès, J. Ding, L. Shen, and K. Tamssaouet, "The flexible job shop scheduling problem: A review," European journal of operational research, Apr. 01, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037722172300382X?via

[5] "Job Scheduling with Greedy Approach — CodingDrills." [Online]. Available: https://www.codingdrills.com/tutorial/introduction-to-greedy-algorithms/job-scheduling-greedy

[6] "Greedy Algorithms Introduction - javatpoint," www.javatpoint.com. [Online]. Available: https://www.javatpoint.com/greedy-algorithms

[7] "A Simulated Annealing Hyper-heuristic for Job Shop Scheduling Problems," IEEE Conference Publication — IEEE Xplore, Jun. 01, 2019. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=8790296