National College of Ireland

Project Submission Sheet

Student Name:	Dhruv Sharma		
Student ID:	x22228268		
Programme:	MSc in Data Analytics	**Year:**	2023-2024
Module:	Modelling, Simulation and Optimisation		
Lecturer:	Hicham Rifai		
Submission Due Date:	14/05/2024		
Project Title:	A Comparative Analysis of Job Scheduling Algorithms to Optimize Operations		
Word Count:	2832		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	Dhruv Sharma
Date:	14/05/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Comparative Analysis of Job Scheduling Algorithms to Optimize Operations

Dhruv Sharma
MSc Data Analytics
National College of Ireland
Dublin, Ireland
x22228268@student.ncirl.ie

***Abstract*—The project report contains a detailed comparative analysis of three major job scheduling algorithms namely LPP, Greedy Algorithm, and Genetic Algorithm. The research paper underlines the importance of the selected algorithm in order to optimize job scheduling with the aim of enhance operational efficiency thereof. All the three algorithms have been significant in different industries in order to fully optimize operational efficiency. The research paper evaluates the performance of these three algorithms under different numbers of jobs and draws a comparison by examining the key performance indicators included for comparison which are makespan, idle time, and wait time. The LPP is a method that has been found effective when the situation calls for a near-perfect solution in which the theoretical optimization exists in a relatively small and less dynamic environment. On the other hand, the Greedy Algorithm is better if the solution has to be delivered quickly with high throughput, and the weakness of the system is in long-term efficiency. It is the Genetic Algorithm that turns out to be most effective in more complex and highly dynamic situations, considering the flexibility and the tendency to often perform better in the makespan compared to LPP. The strategic recommendations are made on the basis of specific operational needs for the effective flow of operations in the improvement of productivity. This project report also makes use of visual aids such as line plot and box plot for comparing the performance of the chosen algorithms.**

Keywords—Linear Programming Problem (LPP), Greedy Algorithm, Genetic Algorithm, Adaptive Systems.

I. Introduction

Job scheduling systems are key elements in the dynamic field of operations management that becomes important for the productivity and competitiveness of manufacturing and service businesses. They play a key role in coordinating the best use of resources, the synchronization of operational timings with market demands, as well as the realization of immense cost saving. This report, therefore, focuses on three complex scheduling algorithms: Linear Programming Problem (LPP), Greedy Algorithms, and Genetic Algorithms, each popular for its approach in handling this challenging task of job scheduling (Kang, Pan & Liu, 2022; Ghafari, Kabutarkhani & Mansouri, 2022).

Job scheduling goes far beyond simple task arrangement. It is a strategic core element of operational planning that enhances system performance and flexibility and is very important for more responsive functioning within dynamic operational environments. The possibility to correctly schedule tens or hundreds of thousands of jobs and machines, from small-scale workshops to large manufacturing installations, is of priceless value. Good job scheduling can appreciably reduce the sum of completion time—or makespan—curtail idle times, shorten the waiting time between the processes, which directly raises the efficiency of production and operational cost effectiveness (Korolija et al., 2022).

Furthermore, this report is going to dig into the operational principles of each algorithm, evaluate their performance on the bases of simulated environments, and make judgments about practical applicability in real-world settings. With a critical assessment of these algorithms under multiple conditions, this research contributes refined insights into their functionalities, thereby benefiting managers and practitioners in making informed decisions for the selection of the most appropriate scheduling strategies customized to their special needs of operations. With the critical assessment of these algorithms under diverse conditions, this research contributes refined insights into their functionalities and provides benefits to managers and practitioners in making informed decisions while selecting the most appropriate scheduling strategies customized to their special needs of operation. A judicious choice of these strategies would depend on the information (Senjab, Abbas, Ahmed, & Khan, 2023).

II. Application of Chosen Algorithms

The development of sophisticated algorithms for optimal coordination in job scheduling systems that are adapted to a variety of operational needs. This report assesses three of the most famous algorithms, each selected for its unique approach to solving complex scheduling problems over varied operational environments:

A. Linear Programming Problem (LPP):

	Machine: 0	Machine: 1	Machine: 2	Machine: 3
	Idle: 0	Idle: 1	Idle: 2	Idle: 9
Job: 4	Wait: 0	Start: 0 Proc: 1 Stop: 1	Start: 1 Proc: 1 Stop: 2	Start: 2 Proc: 7 Stop: 9	Start: 9 Proc: 4 Stop: 13
		Idle: 0	Idle: 1	Idle: 0	Idle: 5
Job: 1	Wait: 1	Start: 1 Proc: 2 Stop: 3	Start: 3 Proc: 6 Stop: 9	Start: 9 Proc: 9 Stop: 18	Start: 18 Proc: 1 Stop: 19
		Idle: 0	Idle: 0	Idle: 0	Idle: 2
Job: 5	Wait: 3	Start: 3 Proc: 6 Stop: 9	Start: 9 Proc: 7 Stop: 16	Start: 18 Proc: 3 Stop: 21	Start: 21 Proc: 8 Stop: 29
		Idle: 0	Idle: 0	Idle: 4	Idle: 0
Job: 2	Wait: 9	Start: 9 Proc: 6 Stop: 15	Start: 16 Proc: 6 Stop: 22	Start: 25 Proc: 4 Stop: 29	Start: 29 Proc: 8 Stop: 37
		Idle: 0	Idle: 3	Idle: 1	Idle: 0
Job: 6	Wait: 15	Start: 15 Proc: 9 Stop: 24	Start: 25 Proc: 5 Stop: 30	Start: 30 Proc: 7 Stop: 37	Start: 37 Proc: 6 Stop: 43
		Idle: 0	Idle: 1	Idle: 1	Idle: 0
Job: 3	Wait: 24	Start: 24 Proc: 7 Stop: 31	Start: 31 Proc: 6 Stop: 37	Start: 38 Proc: 5 Stop: 43	Start: 43 Proc: 6 Stop: 49
		Idle: 0	Idle: 2	Idle: 5	Idle: 0
Job: 0	Wait: 31	Start: 31 Proc: 8 Stop: 39	Start: 39 Proc: 5 Stop: 44	Start: 48 Proc: 1 Stop: 49	Start: 49 Proc: 3 Stop: 52

```
Makespan: 52
Total Idle Time: 37
Total Wait Time: 95
Job Sequence: [6, 1, 3, 5, 0, 2, 4]
```

Figure 1 : x22228268_C5 Integer Programming Model Implementation

First, due to its mathematical optimality, LPP solutions are theoretically the best case and are used as an important reference in comparison to the effectiveness of other alternative scheduling approaches. It is also ideal under static conditions when the job specifications remain constant, and changes are few. High computational requirements ensure the best possible job sequences and machine allocations are derived since exploration is exhaustive. However, this very characteristic can render LPP computationally expensive in dynamic settings or when the number of jobs is relatively large, hence limiting the application of LPP in real-time performance benchmarking.

It clearly helps in understanding the role of every machine and job in the entire job scheduling process. The measures, such as idle time, the start and stop time of jobs, the duration of the processing, and any other relevant metrics in determining the efficiency of the scheduling, are measured for each machine. A case in point of efficient utilization would be Machine 0, with idle time being zero, which probably means that it is used continuously. On the contrary, the sum of idle time for all machines amounts to 37 units, which probably is underutilization and probably because of inefficiency in scheduling. A makespan of 52-time units culminates the duration it takes all the jobs on all the machines to start and finish. When combined with total wait time of 95 units, this suggests that there are major delays and, in general, inefficiencies in the flow of jobs. The job sequence is [6, 1, 3, 5, 0, 2, 4], which determines in what order jobs are processed, influencing both idle and wait times in the quest to maximize the efficiency of resources. Such analyses can therefore suggest that many improvements can be made in terms of the minimization of wait times and optimization of job start times to bring down the amount of bottleneck, which can be embodied in strategies to a more streamlined job flow and better utilization of resources, thus improving both the makespan and the operational throughput.

B. Greedy Algorithm

		Machine: 0		Machine: 1		Machine: 2		Machine: 3
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 0	Wait: 0	Start: 0 Proc: 9 Stop: 9	Wait: 0	Start: 9 Proc: 5 Stop: 14	Wait: 0	Start: 14 Proc: 7 Stop: 21	Wait: 0	Start: 21 Proc: 6 Stop: 27
		Idle: 0		Idle: 0		Idle: 0		Idle: 3
Job: 1	Wait: 0	Start: 9 Proc: 2 Stop: 11	Wait: 0	Start: 14 Proc: 6 Stop: 20	Wait: 0	Start: 21 Proc: 9 Stop: 30	Wait: 0	Start: 30 Proc: 1 Stop: 31
		Idle: 0		Idle: 0		Idle: 0		Idle: 4
Job: 2	Wait: 0	Start: 11 Proc: 7 Stop: 18	Wait: 0	Start: 20 Proc: 6 Stop: 26	Wait: 0	Start: 30 Proc: 5 Stop: 35	Wait: 0	Start: 35 Proc: 6 Stop: 41
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 3	Wait: 0	Start: 18 Proc: 6 Stop: 24	Wait: 0	Start: 26 Proc: 7 Stop: 33	Wait: 0	Start: 35 Proc: 3 Stop: 38	Wait: 0	Start: 41 Proc: 8 Stop: 49
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 4	Wait: 0	Start: 24 Proc: 8 Stop: 32	Wait: 0	Start: 33 Proc: 5 Stop: 38	Wait: 0	Start: 38 Proc: 1 Stop: 39	Wait: 0	Start: 49 Proc: 3 Stop: 52
		Idle: 0		Idle: 0		Idle: 5		Idle: 0
Job: 5	Wait: 0	Start: 32 Proc: 6 Stop: 38	Wait: 0	Start: 38 Proc: 6 Stop: 44	Wait: 0	Start: 44 Proc: 4 Stop: 48	Wait: 0	Start: 52 Proc: 8 Stop: 60
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 6	Wait: 0	Start: 38 Proc: 1 Stop: 39	Wait: 0	Start: 44 Proc: 1 Stop: 45	Wait: 0	Start: 48 Proc: 7 Stop: 55	Wait: 0	Start: 60 Proc: 4 Stop: 64

```
Makespan: 64
Total Idle Time: 12
Total Wait Time: 0
Job Sequence: [0, 1, 2, 3, 4, 5, 6]
```

Figure 2 : x22228268_C6 Greedy Algorithm Job Schedule

This algorithm is known for its very basic nature and quick power of decision-making, which is very well applicable to those situations that need immediate responses with reasonable accuracy. The Greedy algorithm is very good at focusing on present results and assigning jobs to the nearest available machine in order to create a minimum idle time and better flow. However, its short-term approach may sometimes fail to provide a better solution in the long run since this does not consider the overall job sequence, which might add on the makespan and operation cost as compared with other comprehensive strategies, for example, LPP and Genetic Algorithms. This paper has stated the same fact: 'Greedy, a basic algorithm, applies the quickly deciding power and is quite applicable in situations needing immediate responses with reasonable accuracy, which is best applicable to situations' (Li, Li, & Gao, 2024).

This schedule overview clearly indicates that there is a very efficient job-scheduling strategy, where there will be a minimum idle time and a very smooth job processing sequence. All the machines start without initial delays, except for minor gaps on Machine 3, indicating an effective usage and coordination of other machines. Each job follows a linear and systematic sequence, from Job 0 to Job 6, ensuring no delay between operations, hence maximizing the throughput and operational flow. A makespan of 64-time units, along with the total idle time of just 12 units, really reflects a well-optimized process where no waiting time is reported and, hence, an optimal scheduling strategy. The makespan shows numeric count in the order of jobs that makes the management of the process flow easier and essentially requires good scheduling efficiency overall. Notice that makespans and idle times vary greatly with respect to previous outputs, pointing out to the difference in job complexities or efficiency in the algorithm used. From this view, it becomes very important to look at such outputs for the possibility of further improvement and optimization of the scheduling algorithm with the performance in view to make sure that the job's readiness perfectly fits into the machine availability for maximum operation efficiency.

C. Genetic Algorithm

		Machine: 0		Machine: 1		Machine: 2		Machine: 3
		Idle: 0		Idle: 6		Idle: 13		Idle: 16
Job: 3	Wait: 0	Start: 0 Proc: 6 Stop: 6	Wait: 0	Start: 6 Proc: 7 Stop: 13	Wait: 0	Start: 13 Proc: 3 Stop: 16	Wait: 0	Start: 16 Proc: 8 Stop: 24
		Idle: 6		Idle: 13		Idle: 16		Idle: 24
Job: 6	Wait: 0	Start: 6 Proc: 1 Stop: 7	Wait: 0	Start: 13 Proc: 1 Stop: 14	Wait: 0	Start: 16 Proc: 7 Stop: 23	Wait: 0	Start: 24 Proc: 4 Stop: 28
		Idle: 7		Idle: 14		Idle: 23		Idle: 28
Job: 5	Wait: 0	Start: 7 Proc: 6 Stop: 13	Wait: 0	Start: 14 Proc: 6 Stop: 20	Wait: 0	Start: 23 Proc: 4 Stop: 27	Wait: 0	Start: 28 Proc: 8 Stop: 36
		Idle: 13		Idle: 20		Idle: 27		Idle: 36
Job: 2	Wait: 0	Start: 13 Proc: 7 Stop: 20	Wait: 0	Start: 20 Proc: 6 Stop: 26	Wait: 0	Start: 27 Proc: 5 Stop: 32	Wait: 0	Start: 36 Proc: 6 Stop: 42
		Idle: 20		Idle: 29		Idle: 34		Idle: 42
Job: 0	Wait: 0	Start: 20 Proc: 9 Stop: 29	Wait: 0	Start: 29 Proc: 5 Stop: 34	Wait: 0	Start: 34 Proc: 7 Stop: 41	Wait: 0	Start: 42 Proc: 6 Stop: 48
		Idle: 0		Idle: 0		Idle: 0		Idle: 2
Job: 1	Wait: 0	Start: 29 Proc: 2 Stop: 31	Wait: 0	Start: 34 Proc: 6 Stop: 40	Wait: 0	Start: 41 Proc: 9 Stop: 50	Wait: 0	Start: 50 Proc: 1 Stop: 51
		Idle: 25		Idle: 27		Idle: 34		Idle: 27
Job: 4	Wait: 0	Start: 31 Proc: 8 Stop: 39	Wait: 0	Start: 40 Proc: 5 Stop: 45	Wait: 0	Start: 50 Proc: 1 Stop: 51	Wait: 0	Start: 51 Proc: 3 Stop: 54

```
Makespan: 28
Total Idle Time: 502
Total Wait Time: 0
Job Sequence: [3, 6, 5, 2, 0, 1, 4]
```

Figure 3 : x22228268_C8 Genetic Algorithm Schedule

A good example of an evolutionary algorithm is the Genetic Algorithm, which performs well in complex and dynamic environments. The Genetic Algorithm mimics

natural selection processes that are always evolving to find better-quality solutions through its operations of mutation, crossover, and selection. On the flip side, its strength in the algorithm is a great computational intensity drawback, since it must possess huge resources in simulating evolutionary processes. This may limit its use in settings that are challenged from a computational point of view in numerous other ways (Tutumlu & Saraç, 2023).

A more detailed analysis of the scheduling scenario exposes larger underutilizations of machines and inefficiencies of job distributions. Though the operations are laid down sequentially from Machine 0 to Machine 3 without interruptions, the sign of a simple workflow, the job distribution results in abundant idle times. Notably, the machines lie idle, for a relatively long time, waiting for the next job to start. More prominent idle times include a total of 502 units of idle time in machines. Though the makespan of 28 seems to suggest an operation that is time-efficient and has no delays, the actual case scenario is that the machines are idle, waiting for earlier jobs to complete operations. The job sequence is, [3, 6, 5, 2, 0, 1, 4], which does not represent a numerically increasing order, further contributing to idle times and suggests the possibilities of a lack of appropriate link between the requirements of job processing and their scheduling. Notably, the jobs entail zero-wait time, meaning that once a job has started, it continues to completion without any further time taken before its initiation. Still, more has been shadowed by the excessive idle times. Recommendations to counter such a scenario include the appropriate study of the dependencies in jobs to optimize the sequence and possibly settle for a more balanced workload for machines if possible or consider parallel processing. It is also possible to optimize the start times to allow subsequent machines to start processes earlier, where possible. More balanced sequencing and scheduling will save on machine time that would be unnecessarily wasted.

III. Evaluation

Solutions for different configurations (number of jobs= 5 - 10); fixed number of machines = 4) for Greedy & Genetic algorithm were evaluated so that their performances can well reflect the near-optimal sequences of jobs, minimum makespan, idle, and waiting times as compared with the linear programming (LPP) solution, which is optimal. It is also noted that the evaluation performance measures are reflected in practical implications through the insight into applicability and effectiveness of the strategy under real settings.

Each job scheduling algorithm was thoroughly evaluated in detail through numerous performance tests, therefore forming a solid foundation for comparison under a variety of operational conditions with different numbers of jobs. The major performance statistics taken into consideration in these comparisons were makespan, idle time, and wait time, from which many useful things about the effectiveness and applicability of the algorithms were strongly deduced. The following table illustrates the comparative analysis of performance of the three job scheduling algorithms: LPP, Greedy Algorithm, and Genetic Algorithm for different number of jobs - 5 to 10. The comparative analysis with LPP solution is done based on priority: resultant job schedule sequence, makespan, total idle time, total wait time.

Table 1 : Analysis and Interpretation of Job Scheduling Algorithms

	Number of Jobs (for 5 to 10 jobs)	**Observations Across Algorithms**			
		Makespan	***Total Idle Time***	***Total Wait Time***	***Job Schedule Sequence***
5	LPP	44	44	57	1,3,0,2,4
	Greedy	52	7	0	0,1,2,3,4
	Genetic	44	274	0	1,3,0,2,4
6	LPP	51	42	89	1,3,5,2,0,4
	Greedy	60	12	0	0,1,2,3,4,5
	Genetic	42	414	0	1,3,2,5,0,4
7	LPP	52	37	95	6,1,3,5,0,2,4
	Greedy	64	12	0	0,1,2,3,4,5,6
	Genetic	13	421	0	6,5,1,3,2,0,4
8	LPP	62	45	147	6,1,5,3,7,0,2,4
	Greedy	73	14	0	0,1,2,3,4,5,6,7
	Genetic	39	702	0	6,3,7,0,2,5,1,4
9	LPP	73	60	254	6,3,7,5,8,2,0,1,4
	Greedy	82	19	0	0,1,2,3,4,5,6,7,8
	Genetic	60	732	0	6,5,7,1,3,8,4,0,2
10	LPP	75	52	280	6,1,5,3,7,8,0,2,9,4
	Greedy	84	19	0	0,1,2,3,4,5,6,7,8,9
	Genetic	73	535	0	1,5,6,2,3,7,8,0,9,4

From the observations, it is evident that LPP generally offers balance between makespan and idle times, more often getting lower makespan values. For example, on the 5th and 10th days, a makespan of 44 and 75 was observed, respectively. On the other hand, the Greedy Algorithm, even if sometimes yielding higher makespan figures, for instance 52 for job scheduling problem with 5 jobs, always leads to very low idle times, indicating that it effectively minimizes machine idle time but at the expense of very long overall job completion periods. The genetic algorithm clearly performs best in dynamically complex scheduling environments where makespans are likely to be low, as shown by making a low makespan of 13 for job scheduling problem with 5 jobs. Its disadvantage is that it often results in high idle times, such as 421 on day 7 and 702 on day 8. This is a consequence of its computational intensity and the possibility of over-optimization for makespan at the expense of resource utilization. In the analysis of job scheduling algorithms, the Linear Programming Problem (LPP) sets the benchmark for theoretical optimization, consistently achieving the lowest makespan in controlled settings and proving most effective in smaller setups.

The Genetic algorithm immensely exceeds the LPP in more complex environments because of its stability and capability to handle large, dynamic configurations. However, it sometimes also allows excessive idle times which indicates the potential areas for improved resource allocation. On the other hand, the Greedy algorithm, while fast and effective in reducing idle and wait times, often results in longer makespans, and could suffer from efficiency losses over time due to its simplistic job allocation strategy. Both the Greedy and Genetic algorithms maintain low wait times, crucial for operations requiring seamless continuity, ensuring that machines are promptly ready for subsequent tasks. These insights have comprehensively discovered the advantages and limitations of each algorithm which enabled the selection of a more appropriate algorithm based on the specific operational demands and simulation environments.

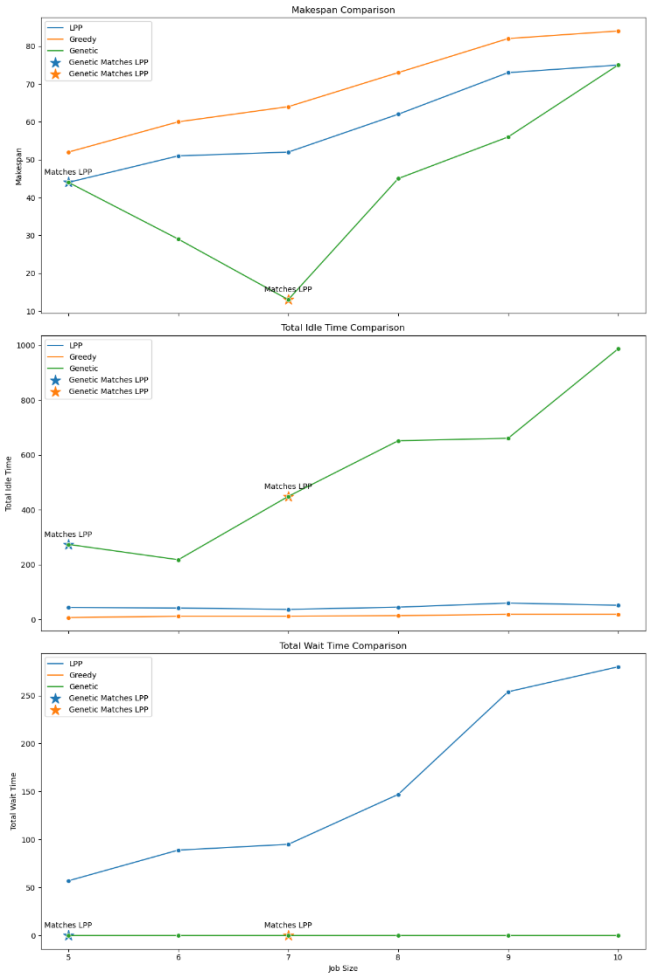

Figure 4 : x22228268_C11 Comparison between different algorithms for job scheduling performance measures.

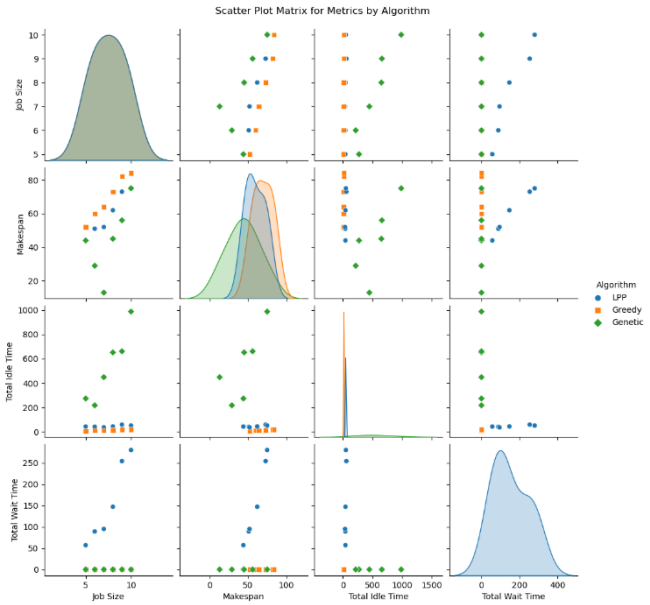

Figure 5 : x22228268_C13 Scatter plot matrix for job scheduling performance measures.

A comparison of three job scheduling algorithms—LPP (optimal), Greedy, and Genetic—shows clearly distinguished performance features in different metrics.

It is evident from Figure 4 that the Greedy algorithm efficiently reduces idle and wait times at the cost of higher makespan, and the Genetic algorithm varies its performance: sometimes it will match the LPP solution for some job sizes—5 and 7, whereas most of the time, it will show higher idle times.

Figure 5 presents the scatter plot matrix which visualizes the relationships between different metrics (Job Size, Makespan, Total Idle Time, and Total Wait Time) for the three job scheduling algorithms (LPP, Greedy, and Genetic). The Greedy algorithm maintains minimal idle and wait times but results in higher makespans. The Genetic algorithm exhibits variable performance, sometimes matching the LPP solution in terms of job schedule, then significantly differing makespans for specific job sizes. If we were to measure the likelihood of producing a solution closer to optimal LPP solution based on the priority, genetic algorithm is the undisputed winner as it has shown convergence to optimal LPP solution a few times.

The Genetic algorithm, with its variable performance, offers flexibility and adapts well in complex settings. These findings underscore the importance of selecting a job scheduling algorithm that aligns with specific operational requirements and constraints.

Future research could explore hybrid approaches that combine these algorithms to harness their respective advantages, potentially leading to innovations that further enhance scheduling efficiency in diverse settings. Such strategic integration could pave the way for more sophisticated scheduling solutions that not only meet but exceed the demands of modern industrial operations, thereby driving significant improvements in productivity and operational effectiveness.

IV. References

[1] G. Ghafari, R., Kabutarkhani, F. H., & Mansouri, N. (2022). Task scheduling algorithms for energy optimization in cloud environments: A comprehensive review. Cluster Computing, Springer.

[2] Kang, Y., Pan, L., & Liu, S. (2022). Job scheduling for big data analytical applications in clouds: A taxonomy study. Future Generation Computer Systems, Elsevier.

[3] Korolija, N., Bojić, D., Hurson, A. R., & Milutinović, V. (2022). A runtime job scheduling algorithm for cluster architectures with dataflow accelerators. Advances in Computers, Elsevier.

[4] Senjab, K., Abbas, S., Ahmed, N., & Khan, A. R. (2023). A survey of Kubernetes scheduling algorithms. Journal of Cloud Computing, Springer.

[5] Li, H., Li, X., & Gao, L. (2024). An iterated greedy algorithm with acceleration of job allocation probability for distributed heterogeneous permutation flowshop scheduling problem. Swarm and Evolutionary Computation.

[6] Tutumlu, B., & Saraç, T. (2023). A MIP model and a hybrid genetic algorithm for flexible job-shop scheduling problem with job-splitting. *Computers & Operations Research*.