Performance Measurement of Nature Inspired Algorithms for Job Scheduling

Divyansh Anand
School of Computing
National College of Ireland
Dublin, Ireland
x22240217@student.ncirl.ie

***Abstract*— This proposed research gives an overview of job scheduling and the challenges we face with the traditional job scheduling approach. This research performs a thorough analysis for the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) algorithm, explains the important functions in these algorithms and gives a detail description of how this inspired in the coding part of this research. This research compares the performance of GA and ACO with given Integer Programming code. Performance time (makespan) shows, for small sample space, Integrated programming works better than GA (54) and ACO (55) with 52 units. Further analysis of these algorithms is done by varying number of jobs to understand the change in behavior of algorithms, processing time of ACO decreases as compared GA when number of jobs increases over 20. This paper also highlights the improvement and future work that can be done to use the real potential of GA and ACO algorithms in different scenarios.**

Keywords—Job Scheduling, Genetic Algorithm, Ant Colony Optimization, Performance Comparison.

I. INTRODUCTION

Job scheduling plays a crucial role in day to day life by handling jobs in a production environment, where thousands of jobs run simultaneously to accomplish a task. Scheduling a job is a challenging task as it involves allocating set of jobs to set of resources, prioritising jobs and making the jobs more efficient to consume more resources and minimize processing time. Job scheduling plays an important role in various industries namely manufacturing, production, computing and logistics, efficient scheduling in these fields can impact the overall production costs.

Two classic problems that have been studied extensively for scheduling are flow shop scheduling and job shop scheduling. In flow shop scheduling, jobs follow a linear path through the machines. All the jobs follow a fixed sequence, they are simpler and easier to solve as compared to job shop. Whereas, jobs can be processed in any order. This is more complex and computationally challenging as compared to flow shop scheduling.[1]

Computation complexity of job scheduling problems are complicated. Most of the problems are classified as NP-hard, which means there is no known algorithm that can solve them in a polynomial time as it scales proportionally with the size of the problem. Some challenges in job scheduling can be job dependencies, resource limitation and varying processing time. Firstly, for job dependency, some jobs might require completion of other jobs before they start, this job dependency restrict the order in which jobs can be scheduled. Secondly, availability of resources like machines or systems can affect how jobs can be scheduled. Finding a feasible solution that manages the available resources efficiently is challenging. Finally, job completion time varies with the complexity of job. This variability in time makes it difficult to predict completion time accurately and can lead to scheduling inefficiencies.

Other than just time complexity of jobs, job scheduling algorithms also depends on desired objectives and computational constraints. For problems that have been proved to only be solved with the limitation of computing power, often heuristic algorithm are used to solve them. These algorithms focused on finding "good enough" solutions at the earliest possible moment which is why they are well suited for real-time scheduling. Illustrations of SPT (Shortest processing time) and EDF (Earliest deadline first) are among them. Precise algorithms although responding optimally as far as resource usage is concerned, they can be computationally expensive for giant size issues. Such scheduling problems with complicated constraints can be resolved by Simulated Annealing or Genetic Algorithms, which are metaheuristics algorithm. These algorithms are very powerful, they try different alternatives inside the solution space iteratively and they often introduce randomness into the process alongside learning mechanisms that help to arrive at the near-optimal solution efficiently.

Recently introduced task scheduling techniques help reduce energy consumptions in modern applications like cloud computing. Cloud data centres consume huge amount of energy in the form of processing power, the proposed model decreases energy consumption and carbon dioxide emission. This study compares different scheduling algorithms and focuses on the fact that machine learning is mainly used for predicting the resources usage and not for the selection of the optimal schedule. Heuristic and exact algorithms are usually used to facilitate scheduling decision in the system based on the prediction of the outcomes. Overall, this research highlights the importance of scheduling tasks efficiently to optimize resource consumption which can be helpful for our research in managing computing power of the jobs. [2]

Jobs scheduling algorithms can also work efficiently with renewable energy sources plants. Since, the integration of renewable energy into the power grid, these sources of alternative energy face challenges of being mutable and not predictable. The job scheduling algorithms can be employed to plan operations in a way that maximizes energy utilization by scheduling jobs using available renewable sources.

The proposed research is motivated by the limitations of traditional job scheduling algorithms in handling complex scheduling problems with factors like dynamic task handling, variable processing times, and to manage complicated job dependencies. Existing approaches like Shortest Processing Time (SPT) and Earliest Deadline First (EDF) may struggle in such scenarios. This research aims to contribute to the field of job scheduling by investigating the application of two powerful metaheuristic algorithms: Genetic Algorithms (GA) and Ant Colony Optimization (ACO). By using the search

capabilities of GA and the cooperative problem-solving nature of ACO, we hope to develop a robust scheduling solution that can efficiently handle complex scheduling problems, leading to improved resource utilization, reduced job completion times.

II. ALGORITHMS

A. *Genetic Algorithm Implementation*

Humans and animals evolve with each passing generations, genetic algorithms is inspired from the same biological process. Genetic algorithms are search algorithms which finds a solution iteratively for any complex problem. Some important components of Genetic algorithm are given below in figure 1.

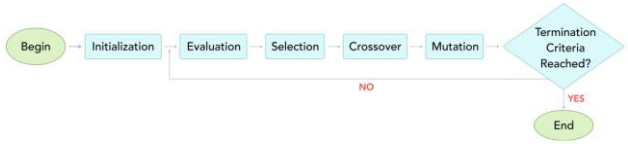

Figure 1. Workflow of Genetic Algorithm

Each component has its importance in the process workflow. In initialization phase, random population of chromosomes are created which represent possible solutions. In the next phase, these chromosomes are evaluated using fitness function. After evaluation phase, selection of the chromosomes with higher fitness value is considered for reproduction(crossover). Crossover phase combines parts of selected chromosomes to generate offspring. In the next phase which is mutation, slight variation is introduced by modifying the portion of offspring. Next, chromosomes for next generation are selected by combining offsprings and some potential parents. Lastly, the iterative process terminates when maximum number of generations are reached. [3]

Inspired from the workflow of the above research, this research has inherited similar kind of flow. Details of components are given below:

1. Initialization of the Population

   The initial population in a GA is usually a set of possible solutions to the problem that is being addressed. It starts by initializing population size for the problem.

   Each schedule is represented by chromosome, schedule is nothing but a permutation of job indices which shows the order in which the jobs will be processed.

   In the initial phase, all the chromosomes in the first population are formed by randomly shuffling the given job sequence. This randomization function facilitates the discovery of different parts of the solution space.

2. Selection Process

   This phase is important to decide genes of which parents will be passed to the next generation. This phase includes two important methods fitness value calculation and selection method.

   Fitness value for each individual schedule is calculated by reciprocal of makespan function. Makespan is total completion time taken by any schedule. Since, it is inversely proportional to Fitness value, schedule having less completion time will have high fitness value.

   In the selection method, job sequences having higher fitness value are selected for the next crossover phase.

3. Crossover Operation

   The job sequences selected after entire selection process are combined to generate offsprings.

   The crossover function is used for this purpose, it is a point where the job sequence is divided and the segments are exchanged between two parents to produce a child. Here, cross over rate play an important role. Cross over rate is an indicator of the frequency of crossover occurrence, if this is not performed then the offsprings will be exact duplicate copies of their parents.

4. Mutation Operation

   Mutation phase introduces random changes in the child to maintain diversity in the population.

   Mutation rate decides the frequency of changes that can be introduced in the parts of chromosome(job sequence). This is done by swapping the sequence of the jobs.

5. Evaluation

   In this phase, fitness level is evaluated for the newly generated offspring.

   First, for each newly generated offspring, makespan is calculated and thus fitness is calculated. The newer generation replaces the old ones, however, the methodology preserves the combination of old parents and new offsprings to maintain diversity.

6. Termination

   The proposed algorithm repeats the selection, crossover, mutation and evaluation phase until the number of generations are completed.

Through this systematic and structured process, Genetic algorithm finds the optimal job sequence while maintaining the balance between solution space of initial job sequences and newly created children or job sequences.

B. *Ant Colony Optimization Technique Implementation*

Ant Colony Optimization technique revolves around the idea that ants even when wandering randomly, find and optimize the paths from their colony to the feeding sources by putting pheromones, which other ants can sense and are likely to follow. With the passage of time, the path that provides the shortest way gathers more pheromones and this in turn, attracts more ants and increases the probability of the path to be optimal.

The concept has been transformed into the algorithmic form to solve different optimization problems. Some important steps given in this research are listed below. [4]

In the initial phase, parameters such as number of ants, pheromone evaporation rate and other algorithmic parameters are initialized. In the second phase which is ConstructAndSolutions, The artificial ants come up with solution by going through the solution space. They pick-up the path according to the pheromone levels and heuristic

information. In the next phase which is ApplyLocalSearch, the solutions created by ants are enhanced by a local search procedure, this is optional phase. Lastly, in UpdatePheromone phase the pheromone levels on the paths are modified according to the quality of the solution. The good solutions gain more pheromone levels while the bad solutions have low pheromone levels.

The steps involved in the ACO algorithm are a part of the iterative process, where the ants construct solutions then refine them through local search and update the pheromone levels to guide the future ants. The algorithm goes through these steps again and again until a termination condition is obtained.[4]

After learning the concept and core fundamental of Ant Colony Optimization technique, below are steps taken in this research for implementing this algorithm.

1. Initialization Phase

   Values of number of jobs and number of machines are assigned. These parameters specify the number of jobs that are to be scheduled and the number of available machines. Number of ants are assigned with value of job number which means each ant will have a full schedule of jobs.

   Number of iterations describes how many rounds of solution construction and pheromone updates will be performed.

   Parameters like evaporation rate control the rate of pheromone evaporation (reducing the influence of older path) and parameter rate of pheromone deposition will increase the influence of successful paths.

2. Initializing Pheromone Level

   Pheromones are set uniformly at the beginning using initial pheromone, which is equal to inversely proportion to the number of jobs.

3. Processing Time Setup

   In this phase, the processing time for each job on each machine are generated randomly which introduces variability and complexity to the scheduling.

4. Solution Construction per Ant

   This step is crucial, each ant begins at randomly selected job and builds a sequence of jobs (schedule) by selecting the next job based on the pheromone trails and probabilistic rule.

   In second round, the next job sequence assigned to the ants is done based on the current pheromone levels and a taboo list which records the already scheduled jobs to avoid any repetition.

5. Makespan and Timeline Calculation

   This function calculates the makespan or processing time for a given schedule and keeps track of start and end time of job on each machine,

6. Update Best Solution

   Once the ants have completed their individual schedules, the makespan is checked and evaluated. If makespan value is the best one that has been found so far (i.e. it's lower than all previous makespan), it update that value to the best known solution.

7. Pheromone Update

   After, all ants have created their schedules for the iteration, the pheromones are evaluated. The pheromone values are higher where the best schedule has been applied, which will attract future ants to consider these paths. The evaporation of pheromones are regulated by evaporation_rate, which in turn ensures that the algorithm does not settle too fast on wrong solutions.

8. Iteration Process

   The entire process is repeated for the given number of iterations and provide the best schedule among all of them.

With this step-by-step process, this research implements the code for Ant Colony Optimization technique to find the optimal job scheduling.

C. Results

For Implementation of code, seed value of 2170 (student ID: x22240217) with 7 number of jobs and 4 number of machines has been used.

1. Integer Programming Results

For this, our research got the optimal job sequence: [1, 4, 0, 6, 5, 3, 2] for the given parameters.
The processing time of the optimal job sequence is 52 units and the detail schedule is displayed in the below figure 2. It represents how the optimal job sequence will start and finish by sharing the machines.

		Machine: 0		Machine: 1		Machine: 2		Machine: 3
		Idle: 0		Idle: 1		Idle: 10		Idle: 11
Job: 1	Wait: 0	Start: 0 Proc: 1 Stop: 1	Wait: 0	Start: 1 Proc: 9 Stop: 10	Wait: 0	Start: 10 Proc: 1 Stop: 11	Wait: 0	Start: 11 Proc: 9 Stop: 20
		Idle: 0		Idle: 0		Idle: 4		Idle: 0
Job: 4	Wait: 1	Start: 1 Proc: 7 Stop: 8	Wait: 2	Start: 10 Proc: 5 Stop: 15	Wait: 0	Start: 15 Proc: 2 Stop: 17	Wait: 3	Start: 20 Proc: 8 Stop: 28
		Idle: 0		Idle: 0		Idle: 2		Idle: 0
Job: 0	Wait: 8	Start: 8 Proc: 7 Stop: 15	Wait: 0	Start: 15 Proc: 4 Stop: 19	Wait: 0	Start: 19 Proc: 9 Stop: 28	Wait: 0	Start: 28 Proc: 9 Stop: 37
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 6	Wait: 15	Start: 15 Proc: 3 Stop: 18	Wait: 1	Start: 19 Proc: 1 Stop: 20	Wait: 8	Start: 28 Proc: 9 Stop: 37	Wait: 0	Start: 37 Proc: 3 Stop: 40
		Idle: 0		Idle: 0		Idle: 2		Idle: 0
Job: 5	Wait: 18	Start: 18 Proc: 2 Stop: 20	Wait: 0	Start: 20 Proc: 5 Stop: 25	Wait: 14	Start: 39 Proc: 1 Stop: 40	Wait: 0	Start: 40 Proc: 5 Stop: 45
		Idle: 0		Idle: 1		Idle: 0		Idle: 0
Job: 3	Wait: 20	Start: 20 Proc: 6 Stop: 26	Wait: 0	Start: 26 Proc: 9 Stop: 35	Wait: 5	Start: 40 Proc: 2 Stop: 42	Wait: 3	Start: 45 Proc: 2 Stop: 47
		Idle: 0		Idle: 0		Idle: 0		Idle: 0
Job: 2	Wait: 26	Start: 26 Proc: 9 Stop: 35	Wait: 0	Start: 35 Proc: 7 Stop: 42	Wait: 0	Start: 42 Proc: 4 Stop: 46	Wait: 1	Start: 47 Proc: 5 Stop: 52

Figure 2. Job Scheduling of Integer Programming

2. Genetic Algorithm Results

The genetic algorithm was executed for 128 generations to find the optimal solution, which is given in below figure 3.

```
Final Best Schedule: [2, 5, 1, 6, 4, 3, 0]
Final Best Makespan: 54.0
Final Best Fitness: 0.018518518518518517
```

Figure 3. Results of Genetic Algorithm

Figure 3, gives the detail about the best schedule [2,5,1,6,4,3,0] which is nothing but the sequence in which jobs are processed across the machines with the best Makespan value which is 54 units. Fitness value is nothing but reciprocal of Makespan time, 1/54= 0.0185186 (approximately) which is the best among all the generations for this given sequence. Gantt chart has been used to display the output of this best schedule which is given below in figure 4.

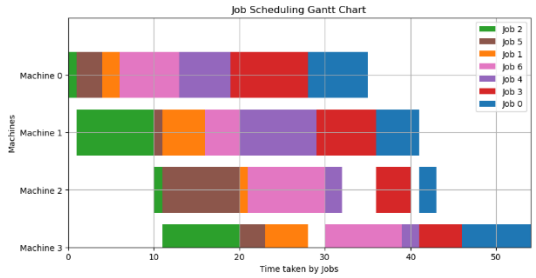

Figure 4. Gantt Chart of Genetic Algorithm

Integer Programming Vs Genetic Algorithm

If we compare the processing time of both the algorithms, we can see there is a difference of 2 units, as Genetic Algorithm took 54 units of time(makespan) more than Integer programming. The slightly higher rate can be due to the GA's stochastic nature (randomness), which leads to the search in broad solution space and probably suboptimal solutions would be attained due to a presence of random processes like mutation and crossovers methodology.

The output schedule of linear programming given in figure 2, shows balanced utilization across machines with minimum idle time. Whereas, from Gantt chart of GA given in figure 4, It shows different job schedule sequence with more variance in idle and jobs start times. This variance can be due to random selection and mutation process.

3. Ant Colony Optimization Results

The Ant Colony Optimization technique was executed for 100 iterations to reach the most optimized job sequence.

Results of ACO are given in below figure 5.

```
Best job sequence: [1, 0, 5, 2, 3, 6, 4]
Best makespan: 55.0
```

Figure 5. Results of Ant Colony Optimization Algorithm

Best job sequence [1,0,5,2,3,6,4] took slightly more makespan time of 55 units as compared to Integer Programming and Genetic Algorithm. For Graphical representation of schedule, Gantt chart for ACO has been given below in figure 6.

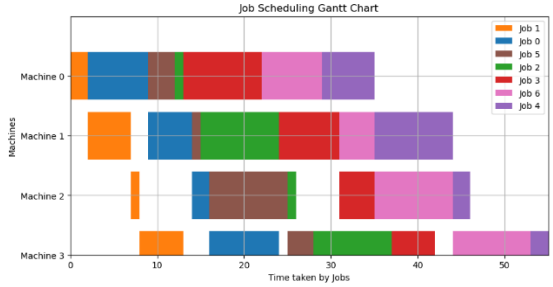

Figure 6. Gantt Chart of Ant Colony Optimization Algorithm

Integer Programming Vs Ant Colony Optimization

ACO provided processing time (55 units), 3 units more than Integer programming results (52 units) which shows ACO has the ability to explore the solution space effectively, but due to its stochastic nature and dependency on pheromone update, the computation complexity increases.

ACO Gantt chart given in figure 5 and 6, shows a different job allocation pattern of machines compared to Integer Programming and Genetic Algorithm. Variations in jobs sequence and some jobs finishing earlier and extended idle times of machine suggests the heuristic behavior resulting due to pheromones of Ant Colony optimization technique.

III. Evaluation

This research has implemented and compared 3 different approaches to solve a job scheduling problem. Below are some evaluation points and future work that can be done to optimize the algorithms better and to test the real potential of these algorithms in different scenarios.

1. Integer programming gave the most optimal solution of 52 units which shows it is comparatively fast in handling well defined and constraint based optimization. But in larger scenarios with huge solution space they tend to fail.
2. Genetic algorithm achieved optimal solution with the makespan (processing time) of 54 units after 128 iterations and proved its ability to find optimal solution even while processing huge solution space in limited time. As per the literature, It has ability to find high quality solutions in the large and complex space. Thus, it agrees with findings we found in this research for the given parameters. [3]
3. Ant colony optimization for this research proved to be inferior as it took 55 units as compared to integer programming and genetic algorithm. But as per the literature, it is perfect due to its complex nature for routing and scheduling problems. This algorithm can find better solutions even in scenarios where traditional approaches might fail. [4]
4. For future scope, Genetic algorithms can be improved by refining the selection, crossover and mutation functions and parameter values. For Ant Colony Optimization, update pheromone logic can be refined to increase search process effectiveness.
5. Further, adapting latest machine learning techniques and using parallel processing can significantly reduce the processing time of the algorithms.

This study has also compared the performance of algorithms by increasing numbers of jobs which is represented in below line graph, given in Figure 7.

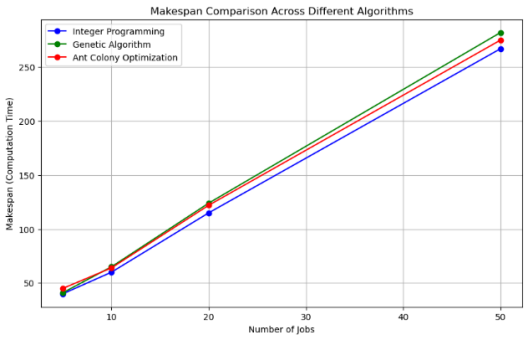

Figure 7. Line graph of algorithms with increase in jobs

Figure 7 represents how computation time changes for algorithms with increase in number of jobs. Major observation is that Processing time of Ant Colony algorithm decreases as compared to Genetic algorithm just when number of jobs increased more than 20, and one can see observable difference when the count of jobs is 50.

IV. References

1. Parveen, S., & Ullah, H. (2011). REVIEW ON JOB-SHOP AND FLOW-SHOP SCHEDULING USING. Journal of the Institution of Engineers Bangladesh, 41(2), 130–146. https://doi.org/10.3329/jme.v41i2.7508

2. Kak, S. M., Agarwal, P., & Alam, M. A. (2022). Task scheduling techniques for energy efficiency in the cloud. ICST Transactions on Energy Web, 9(39), e6. https://doi.org/10.4108/ew.v9i39.1509

3. Genetic Algorithm- a literature review. (2019, February 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8862255

4. Ant colony optimization. (2006, November 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/4129846