Solving Job Shop Scheduling problem using multiple meta heuristic algorithms (Ant colony optimization, Tabu search, Genetic Algorithm)

NAME	STUDENT NUMBER	EMAIL ID
Aishwarya Manapuram	300322316	amana022@uottawa.ca
Ahmed Farooqui	300334347	afaro053@uottawa.ca

ABSTRACT

In this project, our primary objective is to tackle the Job Shop Scheduling problem (JSSP) by employing a range of meta-heuristic algorithms, specifically focusing on Ant Colony Optimization, Tabu Search, and Genetic Algorithm. The JSSP stands as a critical challenge within operations research and production planning, pivotal in the efficient allocation of machines to tasks within manufacturing processes. Notably, such problems exhibit significant computational complexity, largely falling under the classification of NP-hard.

This endeavor holds substantial relevance within the domains of Search-Based Software Engineering (SBSE) and Software Modeling and Testing (SMT/SAT) solvers for Software Testing (ST) or Software Engineering (SE). Our aim is to showcase the practicality of advanced optimization techniques in addressing real-world scheduling dilemmas, extending their potential application across various domains.

Our project methodology revolves around an in-depth analysis of the performance metrics, time complexities, and space complexities inherent in these three distinct algorithms. Our ultimate goal is to discern the most optimized search algorithm tailored for resolving the target problem at hand. Through this comprehensive evaluation, we aspire to contribute to the realm of software engineering and testing by illuminating the efficacy of these optimization approaches in practical settings.

INTRODUCTION

What is JSSP – Job-shop scheduling Problem?

Job-shop scheduling, the job-shop problem (JSP) or job-shop scheduling problem (JSSP) is an optimization problem in computer science and operations research. It is a variant of optimal job scheduling. In a general job scheduling problem, we are given n jobs J1, J2, ..., Jn of varying processing times, which need to be scheduled on m machines with varying processing power, while trying to minimize the makespan – the total length of the schedule (that is, when all the jobs have finished processing). In the specific variant known as job-shop scheduling, each job consists of a set of operations O1, O2, ..., On which need to be processed in a specific order (known as precedence constraints). Each operation has a specific machine that it needs to be processed on and only one operation in a job can be processed at a given time. A common relaxation is the flexible job shop, where each operation can be processed on any machine of a given set (the machines in each set are identical).

Source: https://en.wikipedia.org/wiki/Job-shop_scheduling

Example: Fisher and Thompson 6x6 instance, alternate name (mt06)

```
6   6   1
6   1   3   1   1   1   3   1   2   6   1   4   7   1   6   3   1   5   6
6   1   2   8   1   3   5   1   5   10  1   6   10  1   1   10  1   4   4
6   1   3   5   1   4   4   1   6   8   1   1   9   1   2   1   1   5   7
6   1   2   5   1   1   5   1   3   5   1   4   3   1   5   8   1   6   9
6   1   3   9   1   2   3   1   5   5   1   6   4   1   1   3   1   4   1
6   1   2   3   1   4   3   1   6   9   1   1   10  1   5   4   1   3   1
```

In the first line there are (at least) 2 numbers:

1. the number of jobs
2. the number of machines
3. the average number of machines per operation (optional)

Every row represents one job:

• the first number is the number of operations of that job

• the second number (let's say k>=1) is the number of machines that can process the first operation; then according to k, there are k pairs of numbers (machine, processing time) that specify which

are the machines and the processing times first row: 6 jobs, 6 machines, and 1 machine per operation second row: job 1 has 6 operations; the first operation can be processed by 1 machine, that is machine 3 with processing time 1.

Addressing the Job Shop Scheduling Problem (JSSP) entails using a mix of techniques and algorithms, typically centered around optimization and heuristic strategies given the intricacy of the challenge. The selection of algorithms should align with the problem's complexity. Commonly applied methods include heuristics and meta-heuristics like Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, Tabu Search, or Particle Swarm Optimization.

RELATED WORK

Research into addressing the Job Shop Scheduling Problem (JSSP) through a spectrum of meta-heuristic algorithms, including Simulated Annealing, Ant Colony Optimization (ACO), Tabu Search, and Genetic Algorithms, has been extensive. Our project is founded on pivotal literature within the realm of meta-heuristics for job shop scheduling. Noteworthy studies, such as [1], present comprehensive reviews focusing on algorithms like the Cuckoo Search Algorithm for job shop scheduling. Additionally, a relevant paper [2] proposes a hybrid methodology that merges Genetic Algorithms with Tabu Search to tackle job shop scheduling problems effectively.

GA-based approaches, as detailed in [5], involve the evolution of scheduling populations via genetic operators such as crossover and mutation. These investigations delve into how the genetic evolution process contributes to heightened scheduling efficiency and optimized solutions. In exploring Simulated Annealing (SA) for JSSP, as seen in [3], researchers examine its capacity to navigate solution spaces by embracing probabilistic movements and progressively reducing search temperatures to converge towards optimal or near-optimal solutions. Meanwhile, ACO-based approaches, detailed in [4], strive to replicate ants' foraging behavior to discern efficient schedules in JSSP. These studies explore how emulating pheromone-based communication among artificial ants can yield effective scheduling solutions. Our project builds upon these foundational works by implementing the ACO algorithm, integrating GA with Tabu Search, and amalgamating the

methodologies delineated in these papers. This concerted effort aims to address the JSSP problem in a more holistic and comprehensive manner.

METHODOLOGY

Describes the approach, methods, tools, and techniques used to conduct the project or research.

1. **TABU SEARCH:**

   Tabu Search is a meta-heuristic algorithm used for solving optimization problems, especially those that involve finding the best solution within a large search space. The algorithm operates by navigating through the search space, gradually improving solutions iteratively. It's particularly effective for problems where traditional optimization methods might struggle due to complex solution spaces, constraints, or difficult landscapes.

Algorithm 1: Tabu Search

Data: S - the search space, $maxIter$ - the maximal number of iterations, f - the objective function, the definition of neighborhoods, and the aspiration criteria.
Result: the best solution found
Choose the initial candidate solution $s \in S$
$s^* \leftarrow s$ `// Initialize the best-so-far solution.`
$k \leftarrow 1$
while $k \leq MaxIter$ **do**
  `/* Sample the allowed neighbors of` s `*/`
  Generate a sample $V(s,k)$ of the allowed solutions in $N(s,k)$
    `//` $s' \in V(s,k) \iff (s' \notin T) \vee (a(k,s') = true)$
  Set s to the best solution in $V(s,k)$
  `/* Update the best-so-far if necessary */`
  if $f(s) < f(s^*)$ **then**
    $s^* \leftarrow s$
  end
  Update T and a
  `/* Start another iteration */`
  $k \leftarrow k + 1$
end

2. **ACO:**

   ACO stands for Ant Colony Optimization, a meta-heuristic algorithm inspired by the foraging behavior of ants. It's used to solve optimization problems by simulating the

collective behavior of ants seeking the shortest path between their colony and a food source.

```
Procedure of ACS Algorithm:
Begin
    Initialize
    While stopping criterion not satisfied do
        Position each ant in a starting node
        Repeat
            For each ant do
                Choose next node by applying the state transition rule
                Apply step by step pheromone update
            End for
        Until every ant has built a solution
        Update best solution
        Apply offline pheromone update
    End While
End
```

3. GA:

GA stands for Genetic Algorithm, a powerful optimization and search technique inspired by the principles of natural selection and genetics. It's used to solve optimization and search problems by mimicking the process of natural evolution.

Algorithm 1. Genetic algorithm

```
Data: population of individuals;
Result: Best individuals;
begin
    t = 0;
    Create an initial population– Pop(0);
    Evaluate individuals – calculate the value of the fitness function for each
    individual in the population P0;
    do
        select individuals for the new population Pop(t) – selection;
        perform crossover operation;
        perform the mutation of individuals;
        evaluate individuals;
        replace old population a new one;
        t = t + 1;
    while stop condition reached;
end
```

In the context of these metaheuristic algorithms applied to the Job Shop Scheduling Problem, the fitness function is a crucial component that measures the quality of a particular solution or schedule. The goal of these algorithms is to optimize the scheduling of jobs on machines to minimize certain objective criteria. The fitness function quantifies how well a solution performs with respect to these criteria.

For the Job Shop Scheduling Problem, the common objective is to minimize one or more of the following criteria:

- **Makespan:** This is the total time required to complete all jobs. The fitness function could be defined as the makespan, where shorter makespans represent better schedules.
- **Total Flow Time:** The total time that all jobs spend in the system, which includes the waiting time and processing time. The fitness function could aim to minimize this total flow time.
- **Total Completion Time:** The sum of completion times for all jobs. A good schedule minimizes this value.
- **Maximum Lateness:** The maximum amount by which a job finishes late. The fitness function can aim to minimize the maximum lateness.

The choice of fitness function is a critical aspect of adapting these metaheuristic algorithms to solve the Job Shop Scheduling Problem effectively.

RESEARCH QUESTIONS

Our research questions are as follows:

1. What is the research goal?
    - Understanding the working of the chosen meta-heuristic algorithms (Ant Colony Optimization, Tabu Search, and Genetic Algorithm will) on the selected Dataset and compare their results to identify which algorithm is the most suitable for this specific scheduling challenge.

- We also aim to understand the similarities between them.

2. How can the performance of multiple meta heuristic algorithms be optimized for the Job Shop Scheduling problem?

- The performance of multiple meta heuristic algorithms can be optimized by carefully selecting the algorithms to use and fine-tuning their parameters to fit the specific problem being solved.

3. How do the results of JSSP use multiple meta heuristic algorithms compare to those obtained using traditional optimization techniques?

- The results of JSSP using multiple meta heuristic algorithms may be comparable to those obtained using traditional optimization techniques, but this will depend on the specific problem being solved, and the algorithms being used. In some cases, meta heuristic algorithms may be more effective, while in other cases traditional optimization techniques may be more effective.

4. Can a hybrid approach that combines these meta-heuristic algorithms improve the quality of solutions and convergence speed?

- Combining these meta-heuristic algorithms into a hybrid approach has the potential to improve the quality of solutions and the convergence speed. This will enhance the overall efficiency of solving the Job Shop Scheduling problem.

DATA SET

We plan to use publicly available benchmark data sets for the Job Shop Scheduling problem. These data sets are widely used in literature and will ensure the replicability of our results. We've used the famous **Lawrence Instances**.

SOURCE:
http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html
https://www.ime.usp.br/~cris/fjs/benchmark/

IMPLEMENTATION PROCEDURE

To answer our research questions, we will follow these steps:

Step1: Implement Ant Colony Optimization, Tabu Search, and Genetic Algorithm for the Job Shop Scheduling problem.

Step2: Configure and fine-tune the algorithms using a set of parameters.

Step3: Apply the algorithms to benchmark data instances.

Step4: Evaluate the solutions in terms of makespan, solution quality, and convergence speed.

Step5: Develop an algorithm by combining the best-performing components.

Step6: Conduct experiments to compare individual algorithms.

Step7: Analyze the results and draw conclusions regarding the effectiveness of each algorithm.

EVALUATION

1. TABU SEARCH

The optimum path for Lawrance instances was obtained to be:

```
opt_law = [666, 655, 597, 590, 593, 926, 890, 863, 951, 958, 1222, 1039, 1150, 1292, 1207, 945, 784, 848, 842, 902, 1046, 927, 1032, 935, 977, 1218, 1235, 1216, 1152, 1355, 1784, 1850, 1719, 1721, 1888, 1268, 1397, 1196, 1233, 1233]
```

Using the optimum values,

Zbest: Represents the best or optimal value achieved for the objective function or the best-known solution for the given problem instance.

%Z: Indicates the percentage difference or deviation of the obtained optimal value (Optimum) compared to the best-known solution (Zbest).

Tavg: The average time taken to compute the optimal solution.

Here's an Example: For the 1st Lawrence instance in Lawrence.txt i.e. LA01, the best value is 693 with percentage difference Z=4.05 and tavg= 0.013668

```
instance
C:\Users\aishw\anaconda3\python.exe "E:\MCS - Courses\Courses\AI, Verified software testing\Project\TABU SEARCH\instance.py"
[666, 655, 597, 590, 593, 926, 890, 863, 951, 958, 1222, 1039, 1150, 1292, 1207, 945, 784, 848, 842, 902, 1046, 927, 1032, 935, 977, 1218, 1235, 1
LA01: 693  (optimum: 666 )
Z=4.05, tavg=0.013668

LA02: 685  (optimum: 655 )
Z=4.58, tavg=0.013133

LA03: 697  (optimum: 597 )
Z=16.75, tavg=0.012929

LA04: 1039 (optimum: 590 )
Z=76.10, tavg=0.073149
```

2. GENETIC ALGORITHM

Default values: population_size= 30, mutation_rate= 0.2, iterations=2000

Example: la01, best solution is 712. For la02, best solution obtained in GA is 686

```
genetic_algorithm
C:\Users\aishw\anaconda3\python.exe "E:\MCS - Courses\Courses\AI, Verified software testing\Project\Project Files\GA\genetic_algorithm.py"
la01: 712
End of Instance

Process finished with exit code 0
```

```
genetic_algorithm
C:\Users\aishw\anaconda3\python.exe "E:\MCS - Courses\Courses\AI, Verified software testing\Project\Project Files\GA\genetic_algorithm.py"
la02: 686
End of Instance

Process finished with exit code 0
```

3. ACO

For ACO, la01's best solution is 666. For la02, best solution is 703. The results are tabulated below in Results section. We've taken already existing results for the ACO algorithm from [8] for comparison.

RESULTS

Instance Name	OPTIMUM Makespan	ACO	TABU SEARCH			GA
			Best	Z	t	
LA01	666	666	693	4.05	0.013668	712
LA02	655	703	685	4.58	0.013133	686
LA03	597	680	697	16.75	0.012929	696
LA04	590	688	1039	76.10	0.073149	779

Please find the code for Tabu Search and GA in the folders attached in zip file.

ADDITIONAL WORK:

We tried to compare the performance of a traditional algorithm which is popularly used to solve the JSSP – HEFT.

Based on the paper: An Adaptive Scheduling Algorithm for Dynamic Jobs for Dealing with the Flexible Job Shop Scheduling Problem [7], we implemented the A-HEFT algorithm for JSSP. The paper states that the A-HEFT algorithm performs better than traditional state of the art algorithms such as GA, however the results do not add up for the Lawrence Instances.

Instance Name	A-HEFT
LA01	786
LA02	913
LA03	856
LA04	861

It does not perform better than any of the metaheuristic algorithms on these instances.

The code written for it can be found here

FUTURE WORK:

The advantage of the A-HEFT algorithm is that it provides us with encoded operation and machine sequences which satisfies the precedence constraints of the Job Shop Scheduling Problem. This ensures that any offsprings generated from the parent would not have to be validated to be right. It would be interesting to note the results when genetic algorithm is combined with the A-HEFT algorithm.

REFERENCES

[1] Aziz Ouaarab, Bela¨ıd Ahiod, Xin-She Yang, Mohammed Abbad, “Discrete Cuckoo Search Algorithm for Job Shop Scheduling Problem”, 2014 IEEE International Symposium on Intelligent Control (ISIC).

[2] Olfa Belkahla Driss, Khaled Ghedira, Madiha Harrabi, “Combining Genetic Algorithm and Tabu Search metaheuristic for Job Shop Scheduling problem with Generic Time Lags”, ICEMIS2017

[3] Matrenin P.V., Manusov V.Z., “The Cyclic Job-Shop Scheduling Problem, The New Subclass of the Job-Shop Problem and Applying the Simulated Annealing to Solve It”, 2016 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM).

[4] Cansin Turguner, Ozgur Koray Sahingoz., “Solving Job Shop Scheduling Problem with Ant Colony Optimization”, 15th IEEE International Symposium on Computational Intelligence and Informatics 2014.

[5] Mohamed Ahmed Awad, Hend Mohamed Abd-Elaziz, “An Efficient Modified Genetic Algorithm for Integrated Process Planning-Job Scheduling”, 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC).

[6] S. M. Kamrul Hasan, Student Member, IEEE, Ruhul Sarker, Member, IEEE, and David Cornforth, “Hybrid Genetic Algorithm for Solving Job-Shop Scheduling Problem”, 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)

[7] Cao, Z., Zhou, L., Hu, B. *et al.* An Adaptive Scheduling Algorithm for Dynamic Jobs for Dealing with the Flexible Job Shop Scheduling Problem. *Bus Inf Syst Eng* **61**, 299–309 (2019). https://doi.org/10.1007/s12599-019-00590-7

[8] Shih-Pang Tseng; Chun-Wei Tsai; Jui-Le Chen; “Job Shop Scheduling Based on ACO with a Hybrid Solution Construction Strategy”, 2011 IEEE International Conference on Fuzzy Systems