Final Project
JSSP & FSSP

Xiyao Huang
Department of Computer Science
St. Francis Xavier University
Antigonish, Canada
x2023fkv@stfx.ca
202306631

I. Introduction

Among the many tasks in production logistics, job scheduling plays a vital role. The core of scheduling is how to reasonably allocate limited resources to various activities over time to achieve one or more established optimization goals. In particular, the flow shop scheduling problem (FSSP) involves the process of sequencing activities or operations in a serial flow environment. [1] The relationship between FSSP and Job Shop Scheduling Problem (JSSP) can be seen as an evolution from simple to complex, where FSSP is seen as a specific simplified version of JSSP. Compared with JSSP, FSSP is more stringent in terms of constraints, so it is more suitable for process-based and standardized production scenarios. On the contrary, JSSP provides greater flexibility and scope of application, but also brings higher complexity and challenges, which makes it more suitable for meeting complex and diverse production needs. Many artificial intelligence technologies such as particle swarm optimization, neural networks, and reinforcement learning are now widely used in rescheduling problems. These artificial intelligence solutions not only significantly reduce production costs, but also improve energy efficiency and bring improvements in actual scheduling operations. Faced with the evolving challenges in the contemporary manufacturing environment, the role of artificial intelligence is becoming increasingly critical. [2] Therefore, how to apply technologies such as Evolutionary Computation to problems such as FSSP and JSSP still has certain potential and practical problems.

II. Related Works

Since the job scheduling problem is closely related to real life, and the development of various industries has made job scheduling more complicated, many researchers have conducted in-depth research in this field. Ripon and Jim used a hybrid genetic algorithm combined with variable neighborhood search to simultaneously optimize the completion time and average flow time of the job shop scheduling problem (JSSP), and the total material handling cost and proximity score of the facility layout problem (FLP) in 2014. [3] Caldeira, R.H. and Gnanavelbabu, A. proposed an improved Jaya algorithm in 2019 that addresses the limitations of traditional metaheuristic algorithms by combining an effective initialization mechanism, local search techniques, and acceptance criteria. They evaluated the performance of the algorithm using completion time as the criterion on 203 benchmark instances, demonstrating its effectiveness and robustness in solving the Flexible Job Shop Scheduling Problem (FJSSP). The algorithm benefits from these enhancements to achieve higher solution quality and maintain population diversity. [4] In 2024, Lirui Xue et al. proposed an innovative optimization model for the flexible shop scheduling problem of parallel batch processing machines. The model combines variable neighborhood search technology with a multi-population genetic algorithm and effectively reduces the risk of falling into a local optimal solution by specifically executing a neighborhood search strategy on the elite group. And their computational experiments in an actual production environment have verified that compared with the traditional standard method, this new algorithm has shown significant advantages in reducing the target value, with an improvement of up to 15%. [5]

III. Problem description

A. Flow Shop Scheduling Problem

Since the flow shop scheduling problem has a wide range of applications in industry and economics, it has been extensively studied by many researchers. The standard flow shop problem consists of jobs with the same production process on a set of machines. [6] The goal of FSSP is to optimize the overall working time to minimize it while ensuring that all given constraints are met. Each job consists of multiple processes, which can be flexibly assigned to any machine for processing, but the time it takes for each machine to complete the same process may be different.

B. Job Shop Scheduling Problem

The classic Job Shop Scheduling Problem (JSSP) has been extensively studied over the years. It is often considered to be an NP-hard (non-deterministic polynomial time) problem due to its high computational complexity. [4] The goal of JSSP is similar to that of FSSP, but it places more emphasis on machine utilization. The order of processes for each job is fixed, and each process must be assigned to a specific machine.

IV. Algorithm Description

A. Flow Shop Scheduling Problem

1) Implementation:

- Data processing: Load the FSSP data from the Excel file and identify the unique job ID and machine. Next, create an operation list for each job using job operation mapping, which includes the operation ID, machine, and processing time.
- Scheduling Function: The completion time matrix is used to record the completion time of each job on each machine to calculate the makespan for a given sequence of jobs. Jobs and operations are processed one by one, and the matrix is updated to reflect the dependencies between machines and jobs.
- Initialization: Randomly generate multiple job sequences as the initial population, and set the default population size to 30.
- Fitness Function: Call 'fssp_schedule' function to calculate the completion time of a given job sequence
- Selection: Implement tournament selection and select the top k individuals based on the fitness score, which is the minimum completion time.
- Crossover: The sequential crossover algorithm first selects a subsequence from the first parent, and then fills the remaining unselected positions according to the gene order of the second parent to ensure that each gene appears only once in the offspring and maintains the relative relationship of the original order, thereby generating a new offspring individual.
- Mutation: Randomly swap two tasks in the sequence with a probability less than 10%.
- Visualization: Generate a Gantt chart for optimal job scheduling. Use different colors to represent different jobs and use each bar to represent the operation performed on a specific machine.

2) Enhancements:

- Fitness Tracking: At each iteration, the minimum completion time in the population is calculated and stored in a list. The purpose is to analyze the convergence of the genetic algorithm by recording the best fitness of each generation, that is, the minimum completion time.
- Visualization Enhancements: To further improve the readability and aesthetics of the Gantt chart, I used the Seaborn color palette to enhance color differentiation and ensure that the colors of different jobs are clearly contrasted and easy to identify. At the same time, annotations of the job and operation ID are added to each horizontal bar of the Gantt chart. These annotations clearly mark the job to which each operation belongs and its unique identifier in the entire scheduling process. Through this improvement, the Gantt chart is more intuitive and can better analyze and optimize the job scheduling plan.

B. Job Shop Scheduling Problem

1) Implementation:

- Data processing: Almost the same with FSSP.
- Metrics Calculation: Two metrics are calculated for a given schedule: Makespan: The total time required to complete all jobs. Utilization: The ratio of the machine's actual processing time to the total available time.
- Initialization: To ensure that the operations of each job are executed in order, generate an initial population of random schedules.
- Fitness Function: For multi-objective optimization, fitness returns a tuple containing two elements: completion time and idle rate.
- Crossover: Performs uniform crossover, randomly selecting the start time of the operation from the parent.
- Mutation: Randomize the start time of the operation
- Execution: Run for 30 generations, evolving the population generation by generation and tracking the best completion time.
- Visualization: Visualize with dynamic Gantt charts.

2) Enhacement:

- Pareto Front Optimization: Non-dominated sorting is used to identify the Pareto front. Multiple non-dominated solutions are determined and different optimization schemes are provided, while minimizing completion time and maximizing machine utilization. This multi-objective optimization method based on the Pareto front improves the efficiency and quality of job scheduling while enhancing the flexibility of decision-making.
- Gantt Chart Animation: A dynamic Gantt chart is generated in the update function, and FuncAnimation is used for animation display. By dynamically displaying the scheduling plan, you can more intuitively understand the distribution of jobs on the machine.
- Fitness Predictor: RandomForestRegressor was used to predict the completion time of the JSSP solution. The schedule was converted into a feature vector and trained. The model performance was evaluated and a comparison chart of the predicted and actual completion times was plotted. The mean absolute error (MAE) was calculated to evaluate the prediction effect. The machine learning model was used to quickly evaluate new schedules, reducing the reliance on actual schedule calculations.

V. Results

During the execution of the FFSP genetic algorithm, 30 generations of iterations was performed. In each generation, the fitness of the current population is first evaluated, that is, the completion time of each individual is calculated to measure its performance. Subsequently, the best individuals are selected from the current population, and these individuals will participate in subsequent genetic operations as parents. Next, new offspring individuals are generated by applying crossover (such as sequential crossover OX and other strategies) and mutation operations to increase the diversity of the population and explore the space of potential better solutions. At the same time, the best fitness score in each generation, that is, the completion time of the best individual in that generation, is recorded in order to track the optimization process of the algorithm. After 30 generations of iterations, I obtained a population that gradually approaches the optimal solution and retains the record of the best fitness score of each generation. In order to intuitively display the results of the optimal job scheduling, a Gantt chart was generated. In this chart, each horizontal bar represents an operation performed on a specific machine, and different jobs are distinguished by different colors. Through this visual presentation, can clearly see the execution order, start time, end time, and parallel and serial relationships of each job on the machine, so as to more intuitively understand the specific situation of the optimal job scheduling. For the FSSP, the GA determined the best sequence of the following jobs: [1, 5, 2, 3, 4]. This sequence resulted in an optimal completion time of 96.0, which is the total time required to complete all tasks in the shop under the optimal schedule. For the JSSP problem, in order to more

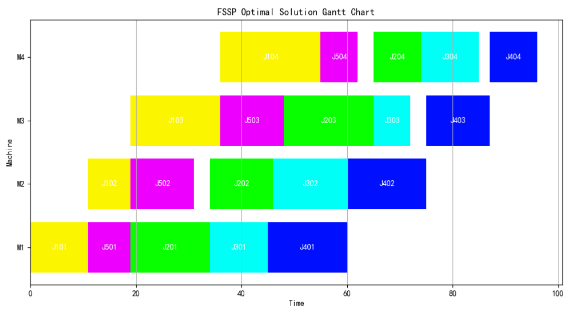

Fig. 1. FFSP_Gantt

intuitively show the scheduling process of the optimal solution, using the FuncAnimation in the Matplotlib library to generate a dynamic Gantt chart. This Gantt chart will clearly visualize the start time, end time of each task and the dependencies between them. Fig. 2. and Fig. 3. show one frame of the animation and the last frame, which is the final result.

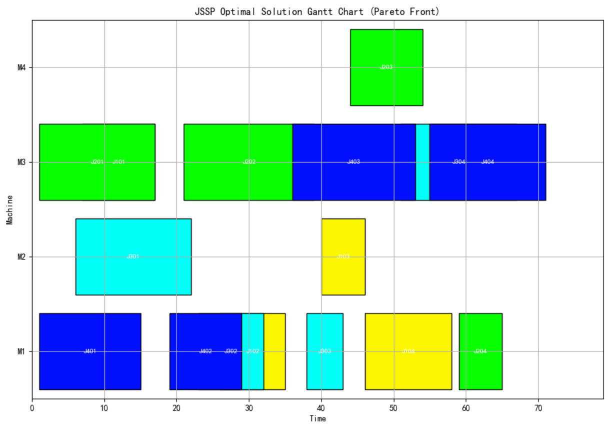

Fig. 2. JSSP Gant 1

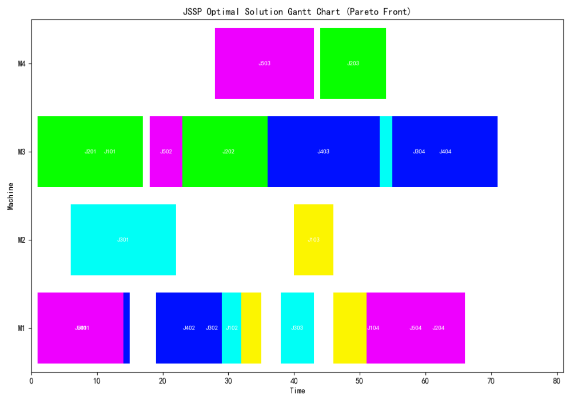

Fig. 3. JSSP Gant 2

In the multi-objective optimization process of the job shop scheduling problem (JSSP), the core goal is to simultaneously pursue minimization of completion time and maximization of resource utilization. After analysis and calculation, the following results were obtained: 1) A solution on the Pareto frontier was successfully identified. 2) The completion time of this solution reached 71 units, showing an efficient job execution speed. 3) Its average resource utilization rate also reached 83.8%, indicating that resources have been fully utilized and configured.

For a more intuitive display, Fig. 4. is drawn, which clearly depicts the distribution of all solutions in the target space. In the figure, the gray points represent all evaluated solutions, which constitute a set of solutions. The red points specifically mark the Pareto optimal solution.

The evolution of the optimal completion time on the Pareto

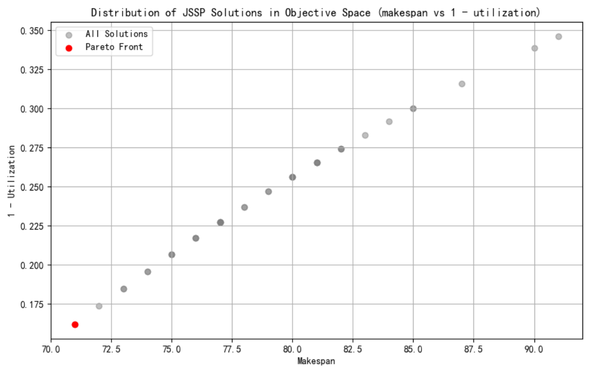

Fig. 4. JSSO Distribution

front is depicted in Fig. 5., which shows the convergence performance of the genetic algorithm (GA) in 30 consecutive generations. It is worth noting that throughout this iterative process, the determined optimal completion time remains at a stable level, 71 units of time.

A fitness predictor was constructed using the random forest

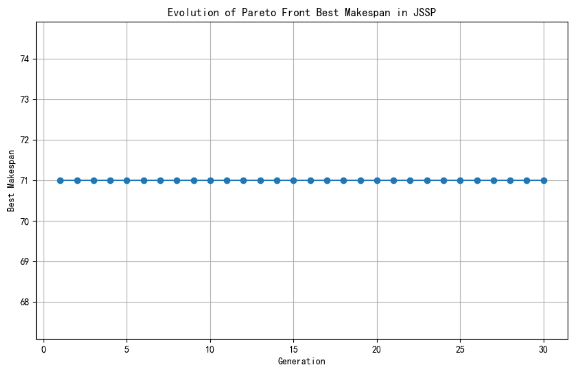

Fig. 5. JSSO Evolution

algorithm. The core function of this predictor is to accurately estimate the completion time of a new solution. In order to fully verify its prediction performance, the predicted completion time was compared with the actual completion time of a set of test samples. It can be clearly seen in Fig. 6. In this figure, the red dotted line represents the ideal prediction line, that is, the state of perfect prediction. The points in the figure are arranged relatively closely along this ideal line, showing the accuracy of the predictor. From the data in Table 1, it can be concluded that its mean absolute error (MAE) is 0.83

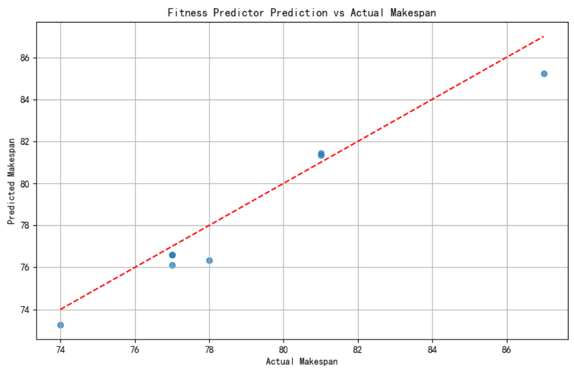

Fig. 6. Fitness Predictor

TABLE I
FITNESS PREDICTOR: PREDICTED VS ACTUAL MAKESPAN

Actual Makespan	Predicted Makespan
78.00	76.33
77.00	76.11
77.00	76.62
81.00	81.36
74.00	73.27
77.00	76.61
87.00	85.23
81.00	81.45

VI. CONCLUSION

In this project, I studied the application of genetic algorithms (GA) in solving flow shop scheduling problems (FSSP) and job shop scheduling problems (JSSP). The results show that genetic algorithms are a powerful tool to deal with the complexity of scheduling tasks. Specifically, the implementation of FSSP successfully optimized the job sequence and achieved a completion time optimization of 96.0%, while in the implementation of JSSP, a completion time of 71 units and a resource utilization rate of 83.8% were achieved through a multi-objective optimization strategy. To further improve the performance of the algorithm, I used dynamic Gantt chart visualization, fitness tracking, and a fitness predictor based on random forests. The use of these techniques significantly enhances the interpretability and efficiency of the scheduling solution. At the same time, the Pareto front optimization method also gives greater flexibility in decision-making based on different scheduling objectives. The performance of genetic algorithms is still affected by the hyperparameter settings, such as population size, crossover rate, and mutation rate. Therefore, future research can explore adaptive algorithms to dynamically adjust these parameters according to the actual situation of the scheduling process in order to improve the convergence speed and the quality of the solution. In addition, hybrid methods that combine genetic algorithms with local search techniques or machine learning models are also effective ways to overcome the limitations of traditional genetic algorithms in complex scheduling environments. As the manufacturing environment continues to evolve, combining genetic algorithms with advanced machine learning models is expected to further improve scheduling efficiency, reduce production costs, and improve resource utilization. Therefore, future work can focus on hybrid algorithms, adaptive parameter adjustment, and real-time scheduling applications to expand the practical application value and scope of these methods.

REFERENCES

[1] González-Neira, Eliana María, et al. “Flow-Shop Scheduling Problem under Uncertainties: Review and Trends.” International Journal of Industrial Engineering Computations, vol. 8, no. 4, 2017, pp. 399–426, https://doi.org/10.5267/j.ijiec.2017.2.001.
[2] Del Gallo, Mateo, et al. “Artificial Intelligence to Solve Production Scheduling Problems in Real Industrial Settings: Systematic Literature Review.” Electronics (Basel), vol. 12, no. 23, 2023, pp. 4732-, https://doi.org/10.3390/electronics12234732.
[3] Ripon, Kazi Shah Nawaz, and Jim Torresen. “Integrated Job Shop Scheduling and Layout Planning: A Hybrid Evolutionary Method for Optimizing Multiple Objectives.” Evolving Systems, vol. 5, no. 2, 2014, pp. 121–32, https://doi.org/10.1007/s12530-013-9092-7.
[4] Caldeira, Rylan H., and A. Gnanavelbabu. “Solving the Flexible Job Shop Scheduling Problem Using an Improved Jaya Algorithm.” Computers & Industrial Engineering, vol. 137, 2019, pp. 106064-, https://doi.org/10.1016/j.cie.2019.106064.
[5] Xue, Lirui, et al. “Flexible Job-Shop Scheduling Problem with Parallel Batch Machines Based on an Enhanced Multi-Population Genetic Algorithm.” Complex & Intelligent Systems, vol. 10, no. 3, 2024, pp. 4083–101, https://doi.org/10.1007/s40747-024-01374-7.
[6] Komaki, G. M., et al. “Flow Shop Scheduling Problems with Assembly Operations: A Review and New Trends.” International Journal of Production Research, vol. 57, no. 10, 2019, pp. 2926–55, https://doi.org/10.1080/00207543.2018.1550269.