BIRZEIT UNIVERSITY

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

Artificial Intelligence

ENCS3340

Project #1:

Optimizing Job Shop Scheduling in a Manufacturing Plant using Genetic Algorithm

---

Prepared by:

Mohammed Abed Alkareem **ID: 1210708**

Mosa Sbeih **ID: 1211250**

Instructor: **Dr. Yazan Abu Farha**

Section: **1**

Date: **19 May 2024**

1. Contents

2. Figures

3. Introduction

Job shop scheduling is a key challenge in manufacturing where multiple products need different operations on various machines. The goal is to optimize the sequence and timing of these operations to minimize production time and maximize efficiency.

In a manufacturing plant, each product follows a specific order of operations on machines like cutting, drilling, and assembly stations. For example, a job might start on a cutting machine, move to a drilling machine, and finish on an assembly station. Scheduling these jobs efficiently is crucial for improving productivity and reducing downtime.

This project uses a genetic algorithm to solve the job shop scheduling problem. Genetic algorithms are good at finding near-optimal solutions for complex problems by mimicking natural selection. Our approach includes:

- Representing the scheduling problem for the genetic algorithm.
- Creating different crossover and mutation methods.
- Defining fitness functions to evaluate schedule quality.
- Analyzing different combinations of crossover and mutation methods to find the best setup.

We aim to show how genetic algorithms can improve job shop scheduling, leading to better efficiency in manufacturing operations.

4. Problem Formulation

4.1 Description of the Problem

In a manufacturing setting, job shop scheduling involves assigning a series of operations to various machines. Each job consists of a sequence of operations, and each operation must be performed on a specific machine for a certain duration. The challenge is to determine the optimal schedule for these jobs to meet specific goals.

For instance, consider a manufacturing plant with machines like cutting machines, drilling machines, and assembly stations. Each job requires these machines in a particular order. For example:

- **Job 1**: M1[10] -> M2[5] -> M4[12]
- **Job 2**: M2[7] -> M3[15] -> M1[8]

In this example, Job 1 starts at machine M1 and takes 10 time units, then moves to machine M2 for 5 time units, and finally to machine M4 for 12 time units. Job 2 starts at machine M2, and so on. The goal is to find a schedule that optimally assigns these jobs to the machines

4.2 Goals

The genetic algorithm aims to find a schedule that satisfies the following fitness functions:

- **Minimize Total Completion Time**: Reduce the overall time required to complete all jobs.
- **Minimize Average Working Time:** The working time in the fitness function aims to minimize the total time each machine spends on jobs (end time - start time) and maximize machine utilization. This ensures efficient use of resources and timely completion of all tasks.
- **Minimize Standard Deviation of Machine Completion Time:** Ensure that the workload is evenly distributed among all machines by reducing the variation in completion times.

4.3 Problem Formulation Components

- **Initial State**: A randomly generated schedule where each job has a sequence of operations on different machines.
- **Actions**: Applying crossover and mutation operations to generate new schedules.
- **Transition Model (Successor Function)**:
    - **Crossover**: Combine parts of two parent schedules to create a new schedule.
    - **Mutation**: Modify parts of a schedule to explore new possibilities.
- **Goal State**: A schedule that optimally satisfies the defined fitness functions.

5. Genetic Algorithm Implementation

5.1 Problem Representation

The problem of job shop scheduling is represented in the context of genetic algorithms as follows:

1. **Initial Population**: We start by creating an initial population of chromosomes. Each chromosome represents a possible schedule of jobs.
2. **Chromosome Structure**: A chromosome consists of multiple phases, each containing:
    - **Job ID**: The identifier for the job.
    - **Machine Order**: The sequence in which the job will be processed on different machines.
    - **Machine**: The specific machine on which the job phase is to be executed.
    - **Time Required**: The duration needed to complete the job phase on the machine.

For example, a chromosome might look like this:

[(Job1, 1, M1, 10), (Job1, 2, M2, 5), (Job1, 3, M4, 12), (Job2, 1, M2, 7), (Job2, 2, M3, 15), (Job2, 3, M1, 8)

3. **Fitness Calculation:** We calculate the fitness of each chromosome using the defined fitness functions (described below). The fitness functions determine how well each chromosome meets the scheduling goals.

4. **Selection:** Based on the fitness values, we select pairs of parent chromosomes to reproduce. The selection is often done using techniques such as roulette wheel selection, where the probability of selecting a chromosome is proportional to its fitness.

5. **Crossover:** The selected parent chromosomes undergo crossover to produce offspring. We implemented four different crossover methods.
6. **Mutation:** The offspring chromosomes undergo mutation to introduce variability. We implemented two mutation methods.
7. **Survivor Selection:** After crossover and mutation, the new chromosomes are evaluated for fitness. The best n chromosomes (depending on the population size) are selected to form the next generation. This ensures that only the most fit individuals survive to the next generation.
8. **Iteration:** Steps 3-7 are repeated for a number of generations or until a termination condition is met (e.g., a maximum number of generations or a satisfactory fitness level is achieved).

5.2 Fitness Functions

The genetic algorithm aims to find schedules that optimize the following fitness functions:

- **Minimize Total Completion Time**: This function reduces the overall time required to complete all jobs, ensuring efficiency in the production process.
- **Minimize Average Working Time**: The working time in the fitness function aims to minimize the total time each machine spends on jobs (end time - start time) and maximize machine utilization. This ensures efficient use of resources and timely completion of all tasks.
- **Minimize Standard Deviation of Machine Completion Time**: This function ensures that the workload is evenly distributed among all machines, reducing the variation in completion times and avoiding bottlenecks.

5.3 Crossover Methods

Four different crossover methods were implemented to generate new offspring from parent chromosomes:

1. **Single Segment Crossover**: A single crossover point is chosen, and segments of parent chromosomes are swapped to create new offspring.

2. **Double Segment Crossover**: Two crossover points are chosen, and segments between these points are swapped between parents to produce offspring.
3. **Partially Mapped Crossover (PMX)**: Parts of the parent chromosomes are mapped to each other to ensure that each job appears only once in the offspring.
4. **Alternating Parental Gene Crossover**: Genes from two parents are alternately chosen to create the offspring.

5.4 Mutation Methods

Two different mutation methods were implemented to introduce variability in the population:

1. **Swapping Mutation**: Two genes in a chromosome are selected at random and their positions are swapped.
2. **Scramble Mutation**: A subset of genes in a chromosome is randomly shuffled to create a new sequence.

These crossover and mutation methods work together to explore the solution space and find high-quality schedules that meet the defined fitness functions.

6. Analysis

6.1 Data Collection

For the analysis of the genetic algorithm, we explored various combinations of crossover and mutation methods. Each combination was evaluated through extensive simulations:

- **Iterations**: 100 iterations were performed for each combination.
- **Generations**: Each iteration ran for up to 500 generations.

This comprehensive data collection allowed us to thoroughly assess the performance and effectiveness of each combination.

6.2 Descriptive Analysis

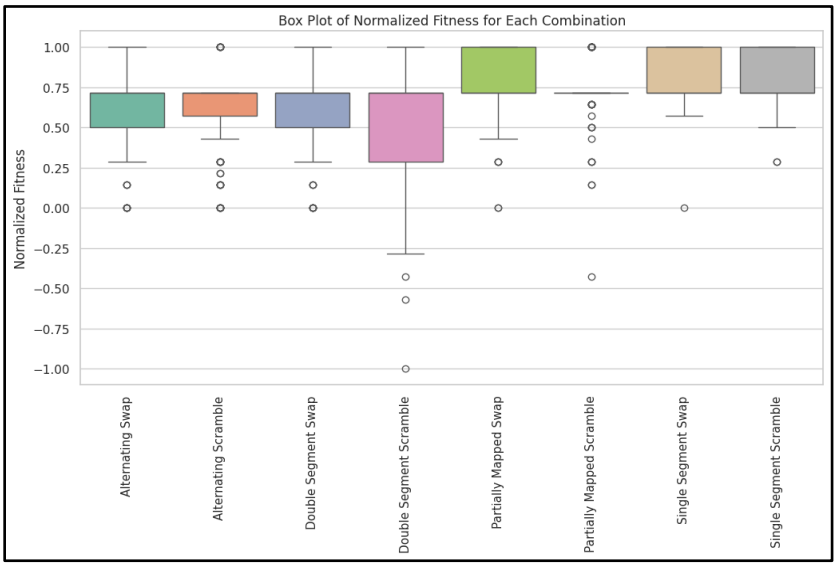

Figure 6-1 Box Plot of Normalized Fitness for Each Combination

1. **Single Segment Scramble**:
    - **Performance**: This combination has the highest median normalized fitness, indicating consistently high performance across different runs. The IQR is narrow, showing low variability, and there are relatively few outliers, which indicates stable performance.
2. **Single Segment Swap**:

- **Performance**: This combination also shows a high median normalized fitness, with a narrow IQR, suggesting consistent performance. The whiskers are short, indicating that most data points are close to the median, and the presence of few outliers suggests stability.

3. **Partially Mapped Swap**:

    - **Performance**: This combination displays a high median fitness value and a relatively narrow IQR, indicating good performance consistency. The presence of a few outliers indicates some variability, but overall, the method performs well.

Conclusion: Single Segment Scramble is the most effective and reliable method, consistently achieving high fitness values with low variability.

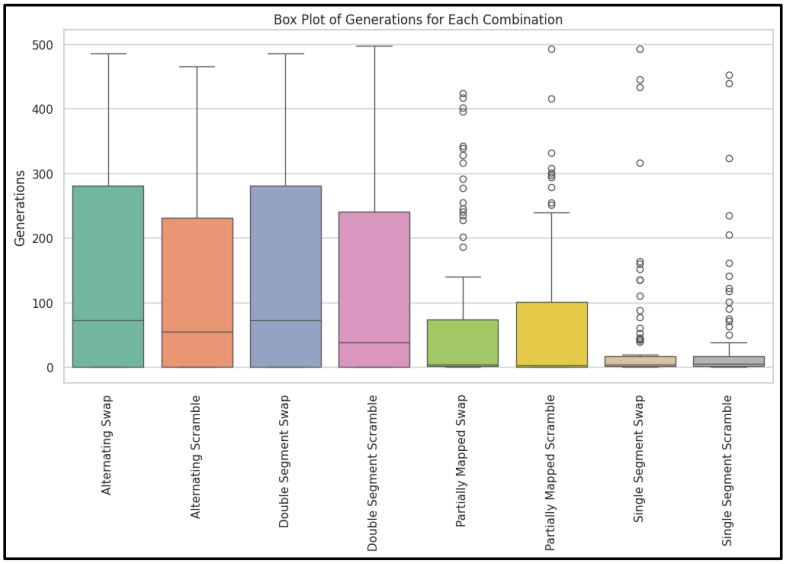

Figure 6-2 Box Plot of Generations for Each Combination

1. **Single Segment Scramble**:

    - **Performance**: This combination shows the lowest median number of generations, indicating it requires fewer generations to achieve the fitness goals. The IQR is narrow, suggesting low variability. The presence of some outliers indicates occasional runs that require more generations, but overall, the performance is efficient.

2. **Single Segment Swap**:

    - **Performance**: This combination also has a low median number of generations and a narrow IQR, suggesting consistent performance with fewer generations required. The whiskers are short, indicating that most data points are close to the median, and the presence of few outliers suggests stability.

3. **Partially Mapped Swap**:

    - **Performance**: This combination displays a relatively low median number of generations and a moderate IQR, indicating good performance consistency. There are a few outliers, suggesting some variability, but overall, the method is efficient.

Conclusion: Single Segment Scramble is the most efficient method, achieving fitness goals quickly and consistently with minimal variability.

6.3 Inferential Analysis

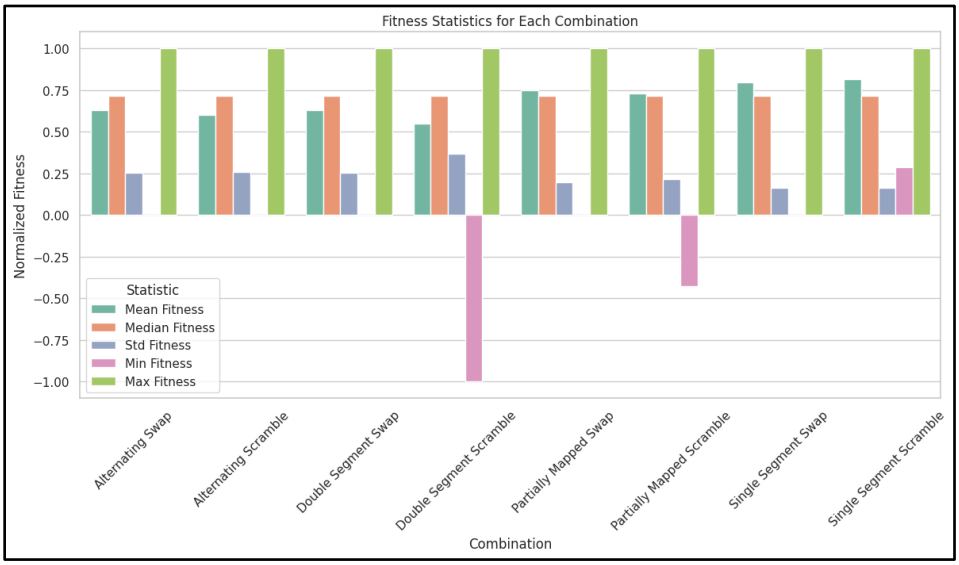

Figure 6-3 Fitness Statistics for Each Combination

1. **Single Segment Swap**:
    - **Mean Fitness**: Highest among all combinations, indicating consistently strong performance.
    - **Standard Deviation**: Low, indicating minimal variability and high reliability.
2. **Single Segment Scramble**:
    - **Mean Fitness**: Very high, close to Single Segment Scramble.
    - **Standard Deviation**: Low, showing consistent performance.
3. **Partially Mapped Swap**:
    - **Mean Fitness**: High, though slightly lower than the Single Segment methods.
    - **Standard Deviation**: Moderate, indicating some variability.

Conclusion: Single Segment Swap shows high mean fitness with low variability, making it a highly reliable method for achieving optimal fitness values.

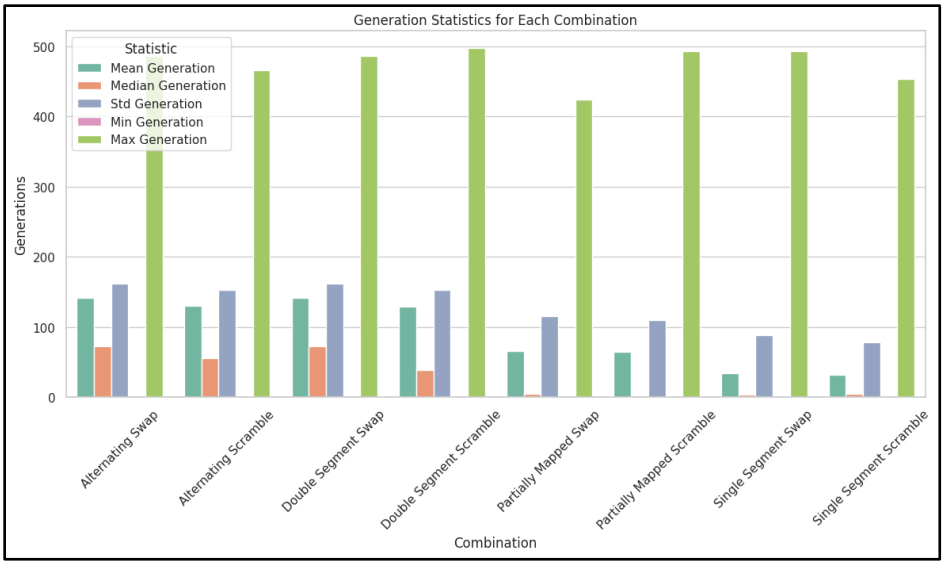

Figure 6-4 Generation Statistics for Each Combination

1. **Single Segment Scramble:**
    - **Performance:** This combination shows the lowest mean generation with low variability. Most runs achieve the fitness goals quickly, indicating high efficiency.
2. **Single Segment Swap:**
    - **Performance:** This combination also has a low mean generation with minimal variability. It demonstrates consistent and quick achievement of fitness goals.
3. **Partially Mapped Swap:**
    - **Performance:** This combination has a moderate mean generation with some variability. It performs well, achieving fitness goals in a reasonable number of generations.

Conclusion: Single Segment Scramble is the most efficient method, consistently requiring fewer generations to achieve fitness goals.

6.4 Confidence Interval Analysis

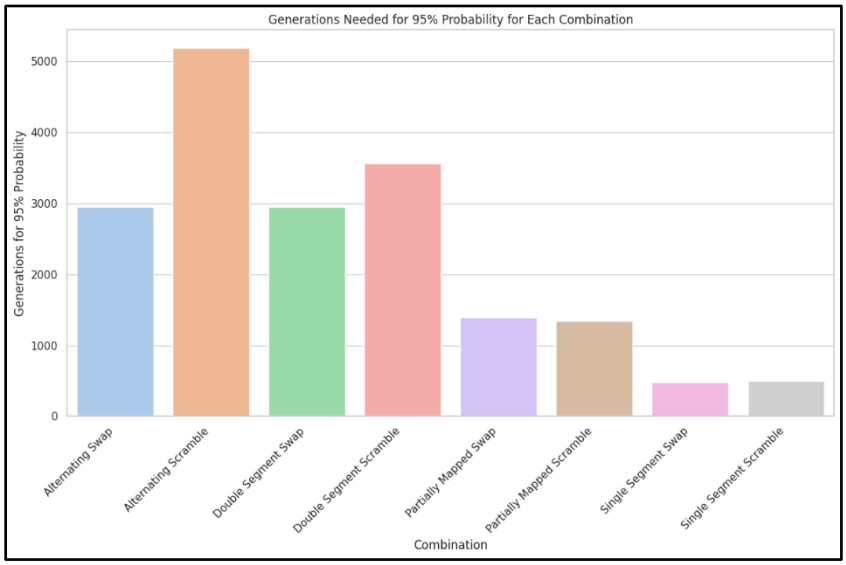

Figure 6-5 Generations Needed for 95% Probability for Each Combination

The bar chart provides a visual summary of the number of generations needed to achieve a 95% probability of reaching the optimal fitness value for each combination of crossover and mutation methods. This analysis highlights the efficiency of each method in terms of the generations required to consistently reach the best fitness value.

1. **Single Segment Swap**: The most efficient method, requiring the fewest generations to achieve a 95% probability.
2. **Single Segment Scramble**: Highly efficient, requiring slightly more generations than Single Segment Swap.
3. **Partially Mapped Scramble**: Moderately efficient, requiring fewer generations than most other methods but more than the Single Segment methods.

Conclusion: Single Segment Swap is the best performing combination, achieving the desired probability with the least number of generations.

7. Test Cases

7.1 Make Span

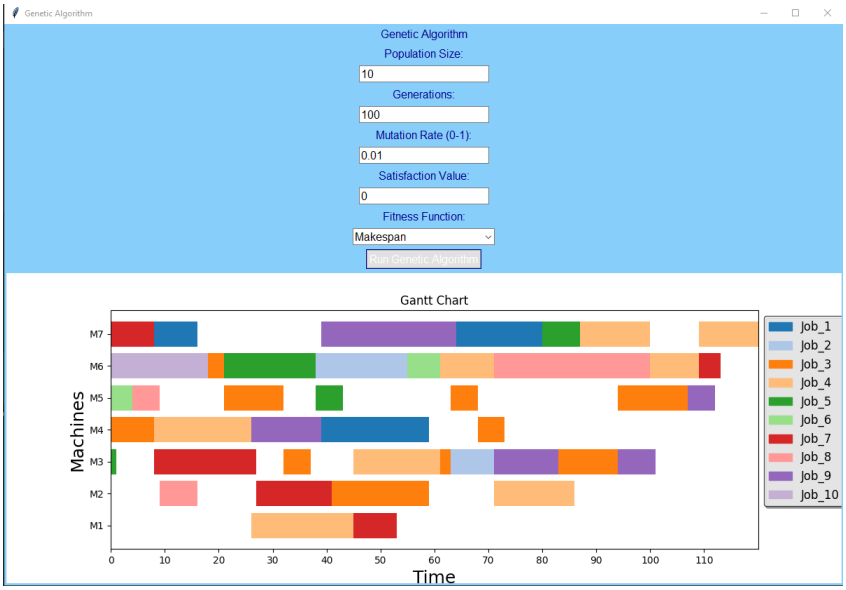

Figure 7-1 Makespan Gantt-Chart

7.2 Working Time

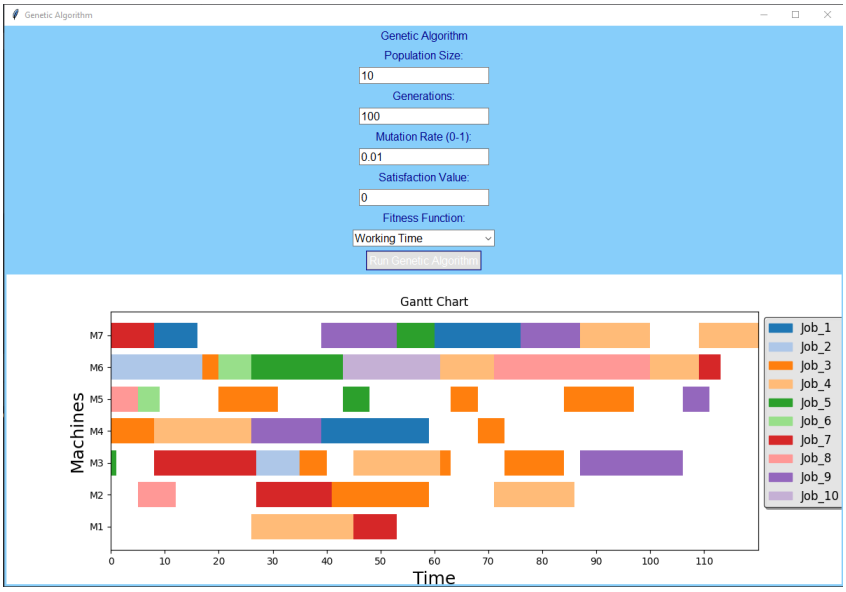

Figure 7-2 Working Time Gantt-Chart

7.3 Standard Deviation

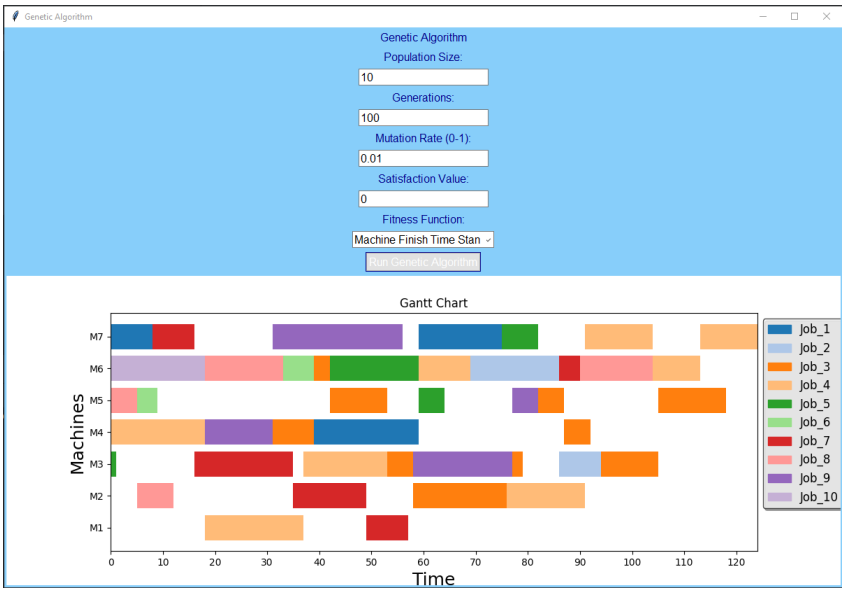

Figure 7-3 Standard Deviation Gantt-Chart

Conclusion

Based on the analysis of fitness statistics and the generations needed to achieve a 95% probability of reaching the optimal fitness value, **Single Segment Scramble** emerges as the best combination. This method consistently shows the highest mean fitness with minimal variability and requires a low number of generations to achieve a high probability of reaching the optimal solution. Therefore, Single Segment Scramble is the most efficient and effective method for optimizing job shop scheduling in this study.

One of the main technical challenges faced during this study was the small dataset size. Each combination was tested with only 100 iterations, and each iteration ran for up to 500 generations. This limitation was due to the slow runtime of the genetic algorithm, making it time-consuming to collect more extensive data. As a result, the dataset may not capture all possible variations and performance characteristics of each combination, potentially affecting the robustness of the conclusions.

References

[1] JOB SCHEDULING WITH GENETIC ALGORITHM BY Debarshi Barat.

[2]Improved Genetic Algorithm for Solving Flexible Job Shop Scheduling Problem Xiong Luo, Qian Qian, and Yun Fa Fu.

[3]GENETIC ALGORITHMS FOR SHOP SCHEDULING PROBLEMS: A SURVEY Frank Werner.

[4]Job Shop Scheduling Problem Using Genetic Algorithms Sahar Habbadi, Brahim HERROU and Souhail SEKKAT.

[5]A Case Study: Using Genetic Algorithm for Job Scheduling Problem Burak Tagtekin, Mahiye Uluya ˇ gmur Öztürk and Mert Kutay Sezer.