BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering

Artificial Intelligence - ENCS3340

Project #1: Optimizing Job Shop Scheduling in a Manufacturing Plant using Genetic Algorithm

Prepared by:

Sara Ewaida 1203048

Instructor: Dr. Yazan Abu Farha

Section: 4

Date: 19/5/2024

Program Implementation

The genetic algorithm (GA) based job shop scheduling system developed for this project is designed to optimize the scheduling of manufacturing jobs across various machines. The implementation incorporates several components: data structures for jobs and machines, the genetic algorithm itself, and a user interface for interaction and visualization. Below is a detailed explanation of these components and their interactions:

Data Structures

- **Job:** Represents a manufacturing job that requires processing on multiple machines. Each job consists of a sequence of operations where each operation specifies the machine it must be executed on and the duration it takes.
- **Machine:** Represents a machine in the manufacturing plant. Each machine can handle one job operation at a time.
- **Individual (or Chromosome):** Represents a potential solution in the genetic algorithm, which is a particular arrangement of jobs on machines. Each individual has a "fitness" value indicating how good the solution is based on criteria like total completion time or machine utilization.

Genetic Algorithm Components

- **Initialization:** Generate an initial population of individuals. Each individual's gene sequence represents a specific job scheduling across all machines.
- **Fitness Evaluation:** Each individual's fitness is evaluated based on the efficiency of the job scheduling. The efficiency could be measured by the overall completion time, the balance of workload across machines, or other relevant metrics.
- **Selection:** Select individuals based on their fitness scores to act as parents for the next generation. Techniques like tournament selection, roulette wheel selection, or rank-based selection can be used.

- **Crossover:** Create new individuals (children) by combining the genes of parent individuals. This might involve methods like one-point crossover, two-point crossover, or more complex schema depending on the problem specifics.
- **Mutation:** Randomly alter the genes of new individuals to maintain genetic diversity within the population. This could involve swapping operations between jobs or reassigning operations to different machines.
- **Replacement:** Generate a new generation by replacing some of the older individuals with new ones based on their fitness values.

User Interface

- **Input Specification:** The user can input the number of jobs, the number of machines, and the specifics of the jobs including the sequence of operations required for each job.
- **Run Genetic Algorithm:** A button to start the genetic algorithm process. Once clicked, it runs the genetic algorithm and generates an optimized scheduling of jobs.
- **Output Display:** The results are displayed in two formats:
  - **Textual Output:** Shows the best schedule found, listing each job's start and end times on respective machines.
  - **Gantt Chart:** Visual representation of the job schedule across machines, showing when each job operation starts and finishes.

The Heuristic of my code

The heuristic employed in this genetic algorithm is designed to optimize job scheduling in a manufacturing setting. The algorithm focuses on minimizing the total production time (makespan) by efficiently scheduling jobs on available machines, considering the sequence of operations required for each job. Each individual in the population represents a possible scheduling solution, with genes corresponding to job sequences on specific machines. Fitness is evaluated based on the makespan, where a lower makespan indicates a more efficient schedule. The genetic operations—selection, crossover, and mutation—are tailored to explore the space of potential schedules, ensuring diversity and preventing premature convergence to suboptimal solutions. The heuristic is crucial in balancing

exploration and exploitation, guiding the algorithm towards the most effective scheduling arrangements while considering machine capacities and job dependencies. This approach is instrumental in achieving optimal throughput and efficiency in the production process.

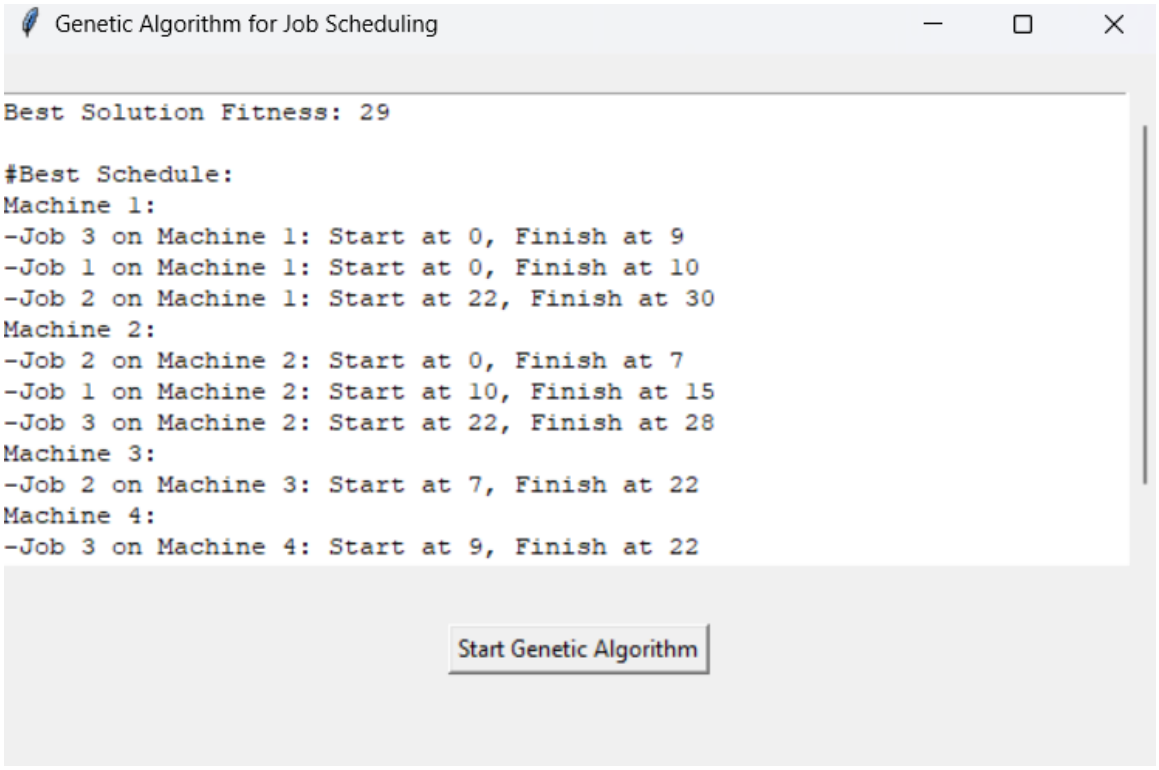
My Tournament with the AI

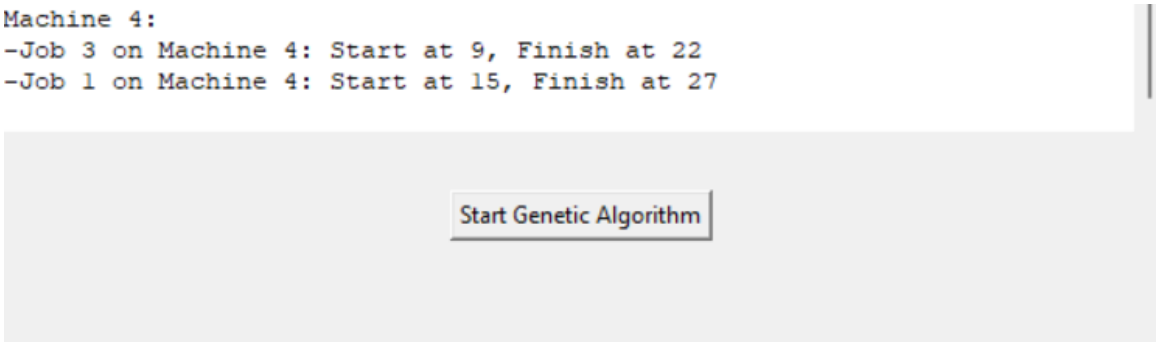
In the tournament, I initiated the genetic algorithm job scheduling system to see how it effectively handles job sequencing across multiple machines to optimize processing time. This demonstration is crucial to showcase the genetic algorithm's ability to solve complex scheduling problems in a manufacturing setup.

First Run: In the initial test, I selected a predetermined set of jobs with predefined operations and processing times. The genetic algorithm was then run to determine the optimal scheduling of these jobs on available machines. The aim was to minimize the overall job completion time, ensuring that each job followed its specific sequence of operations across different machines.

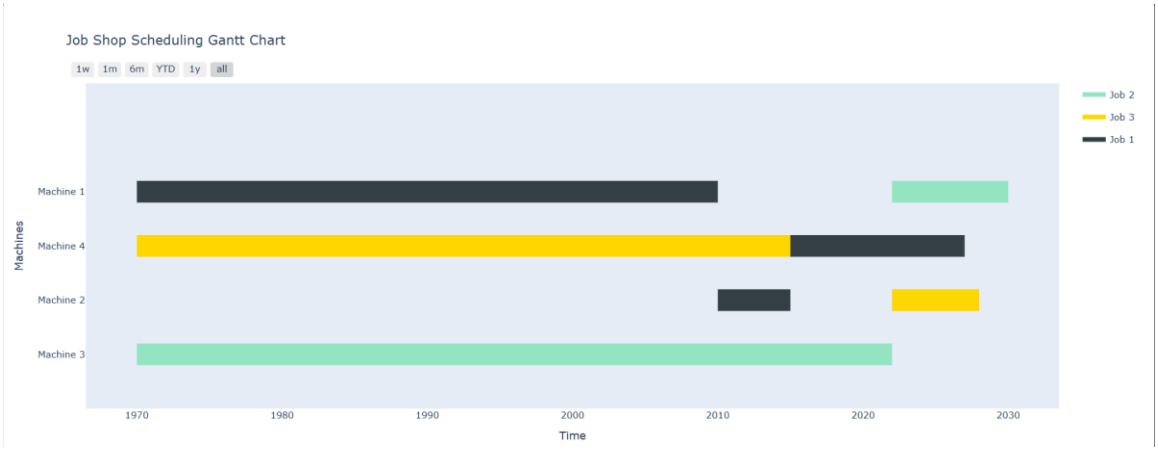
The output clearly indicates the best possible schedule with minimal job completion time. Each machine's job sequence and respective start and finish times were displayed in the UI, confirming that the algorithm successfully calculates the least time-consuming schedule.

Evaluation: The system's effectiveness was evaluated by the genetic algorithm's ability to find the best chromosome—representing the optimal job sequence with the lowest fitness score. The fitness score here corresponds to the total time taken to complete all jobs, reflecting the schedule's efficiency.






```
2024-05-20 22:40:00,424 - INFO - Chromosome: (J1: (M1, 10), (M2, 5), (M4, 12)), (J2: (M2, 7), (M3, 15), (M1, 8)), (J3: (M1, 9), (M4, 13), (M2, 6)) => fitness: 31
2024-05-20 22:40:00,424 - INFO - Chromosome: (J3: (M1, 9), (M4, 13), (M2, 6)), (J2: (M2, 7), (M3, 15), (M1, 8)), (J1: (M1, 10), (M2, 5), (M4, 12)) => fitness: 29
2024-05-20 22:40:00,424 - INFO - Chromosome: (J2: (M2, 7), (M3, 15), (M1, 8)), (J1: (M1, 10), (M2, 5), (M4, 12)), (J3: (M1, 9), (M4, 13), (M2, 6)) => fitness: 33
2024-05-20 22:40:00,424 - INFO - Chromosome: (J1: (M1, 10), (M2, 5), (M4, 12)), (J2: (M2, 7), (M3, 15), (M1, 8)), (J3: (M1, 9), (M4, 13), (M2, 6)) => fitness: 31
2024-05-20 22:40:00,424 - INFO - Best chromosome: (J3: (M1, 9), (M4, 13), (M2, 6)), (J2: (M2, 7), (M3, 15), (M1, 8)), (J1: (M1, 10), (M2, 5), (M4, 12)) => fitness: 29
```



The Gantt chart displayed above shows the scheduling of three different jobs across four machines. Each job is denoted by a unique color, making it easy to track their respective schedules across the timeline:

Job 1 (Black) is scheduled on Machine 1, then moves to Machine 4.

Job 2 (Green) follows its sequence starting on Machine 2, then moving to Machine 3.

Job 3 (Yellow) is allocated to Machine 4 and Machine 2.