Faculty of Engineering & Technology Electrical & Computer Engineering Department

Artificial Intelligence-ENCS3340

PROJECT "ONE"

Prepared by:

Dana Assad **ID: 1211452**

Yara Darabumukho **ID:** 1211269

Instructor:

Dr.Yazan Abu Farha

Section: 1

Date: May/20/2024

➢ Contents

Idea of the project: **3**

The genetic Algorithm In general: **3**

Problem formulation **4**

Test Cases: **5**

➢ Table of Figures:

Figure 1: Cross-over & Mutate Functions Code 4
Figure 2: Fitness Function Code 4
Figure 3: Input 1 Job sequence of operations 5
Figure 4: Gantt chart output of Input1 5
Figure 5: Input 2 Job sequence of operations 6
Figure 6:Gantt chart output of Input2 6

Idea of the project:

The project aims to develop a scheduling system using a **genetic algorithm** to optimize job shop scheduling in a manufacturing plant. This system handles various machines and aims to determine the optimal sequence and timing for each job, minimizing production time or maximizing throughput.

The genetic Algorithm In general:

It is an AI tech that creates efficient heuristic algorithms, inspired by natural selection, to solve optimization and challenging search problems.

The genetic algorithm consists of:

- Population: Subset of solutions that can solve a given problem.
- Chromosomes: are individual solutions in the population, consisting of genes
- Genes: are the elements within a chromosome.
- Fitness function: determines individuals' fitness levels in a population, reflecting their ability to compete. It is used for evaluation in each iteration.
- Genetic operators: in a genetic algorithm improve offspring quality by changing genetic composition, yielding a better generation.
- A selection process: determines which individuals in the population will reproduce and produce the seed for the next generation based on their fitness.

How Genetic Algorithm Work?

Genetic algorithms generate high-quality solutions by utilizing an evolutionary generational cycle. These algorithms enhance or replace the population to produce improved fit solutions. The process involves five phases:

- Initialization
- Fitness Assignment
- Selection
- Reproduction
- Termination

Problem formulation

- Chromosome representation:
  Is a potential solution to scheduling. Each chromosome is a list of job sub-lists, where each sub-list represents a job's sequence of operations. A named tuple, including job ID, machine ID, and duration represents each operation.

- Crossover: The crossover operation exchanges subsequences between parent chromosomes to produce offspring with improved scheduling arrangements in genetic algorithms.
- Mutation: introduces diversity into a population by making small changes to individual chromosomes. It operates probabilistically on randomly selected genes within a chromosome, swapping the positions of two operations. This brings randomness and allows exploration of new search space regions.

```python
def mutate(chromosome):
    """ get any two index randomly from the chromosome, one at variable i and the second one at ii """
    """ then change their places"""
    i, ii = random.sample(range(len(chromosome)), 2)
    temp = chromosome[i]
    chromosome[i] = chromosome[ii]
    chromosome[ii] = temp


def crossover(parent_i, parent_ii):
    """let part variable have a random value between 1 and the length of the parent -1 "due to the indexing"""
    part = random.randint(1, len(parent_i) - 1)
    child_i = parent_i[:part] + parent_ii[part:]
    child_ii = parent_ii[:part] + parent_i[part:]
    return child_i, child_ii
```

Figure 1: Cross-over & Mutate Functions Code

- The objective function: The fitness function evaluates the scheduling efficiency of a genetic algorithm's generated schedule by simulating task scheduling on machines. It calculates the completion time for all tasks based on the chromosome's configuration, aiming to minimize it for a more efficient schedule

```python
def fitness_function(chromosome):
    end_time = defaultdict(int)  # creates a dictionary with default integer value 0
    next_time = defaultdict(int)
    plan = defaultdict(list)  # creates a dictionary with an empty lest
    # determine the schedule of our machines
    for machine in chromosome:  # ************
        for m in machine:
            start_time = max(end_time[m.machine_id], next_time[m.job_id])
            time = start_time + m.duration  # like an end time of the operation
            end_time[m.machine_id] = time
            next_time[m.job_id] = time
            plan[m.machine_id].append((start_time, time, m.job_id))

    score = max(end_time.values())  # return the best value of the successor to be the new parent
    return score, plan
```

Figure 2: Fitness Function Code

Test Cases:

```
jobs = [
    Job(1, [Machine(1, 1, 10), Machine(1, 2, 5), Machine(1, 4, 12)]),
    Job(2, [Machine(2, 2, 7), Machine(2, 3, 15), Machine(2, 1, 8)]),
    Job(3, [Machine(3, 1, 9), Machine(3, 3, 6), Machine(3, 4, 8)]),
    Job(4, [Machine(4, 2, 4), Machine(4, 1, 11), Machine(4, 3, 7)]),
    Job(5, [Machine(5, 4, 5), Machine(5, 2, 8), Machine(5, 3, 10)]),
    Job(6, [Machine(6, 3, 12), Machine(6, 1, 6), Machine(6, 2, 9)]),
    Job(7, [Machine(7, 4, 7), Machine(7, 3, 4), Machine(7, 1, 10)]),
    Job(8, [Machine(8, 2, 11), Machine(8, 4, 8), Machine(8, 3, 6)]),
    Job(9, [Machine(9, 1, 13), Machine(9, 2, 7), Machine(9, 3, 9)]),
    Job(10, [Machine(10, 3, 5), Machine(10, 4, 9), Machine(10, 1, 8)])
]
```

Figure 3: Input 1 Job sequence of operations

Output:

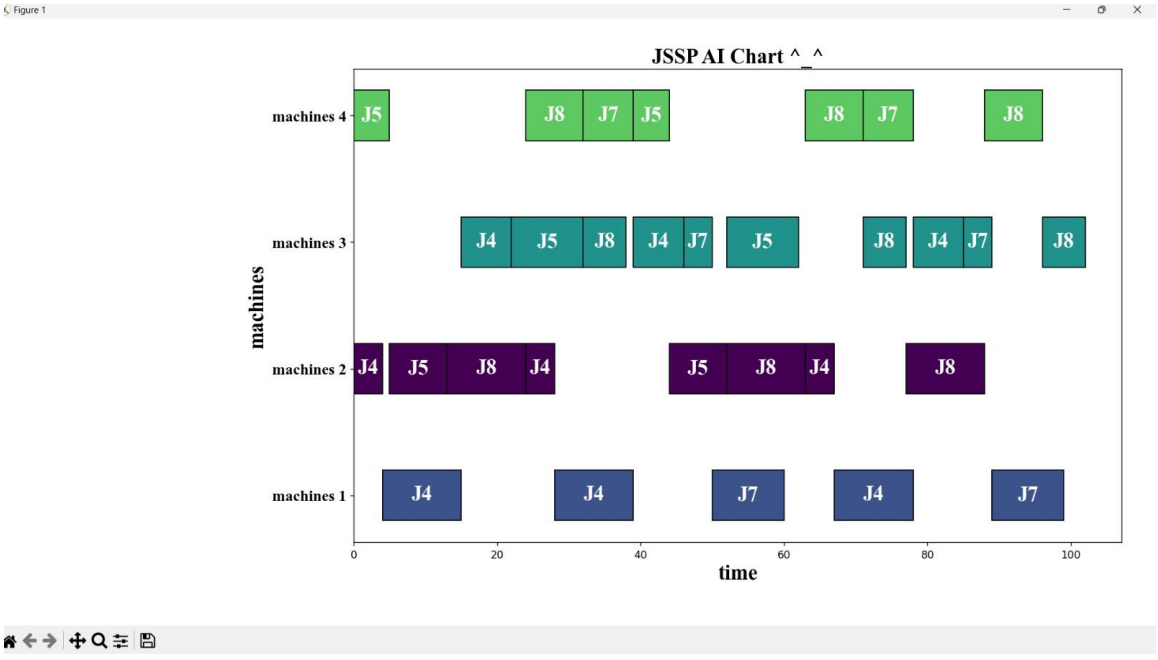

Figure 4: Gantt chart output of Input1

```
jobs = [
    Job(1, [Machine(1, 8, 10), Machine(1, 2, 15), Machine(1, 4, 12)]),
    Job(2, [Machine(2, 5, 7), Machine(2, 3, 15), Machine(2, 8, 8)]),
    Job(3, [Machine(3, 6, 9), Machine(3, 3, 20), Machine(3, 6, 8)])
]
```

Figure 5: Input 2 Job sequence of operations

Output:

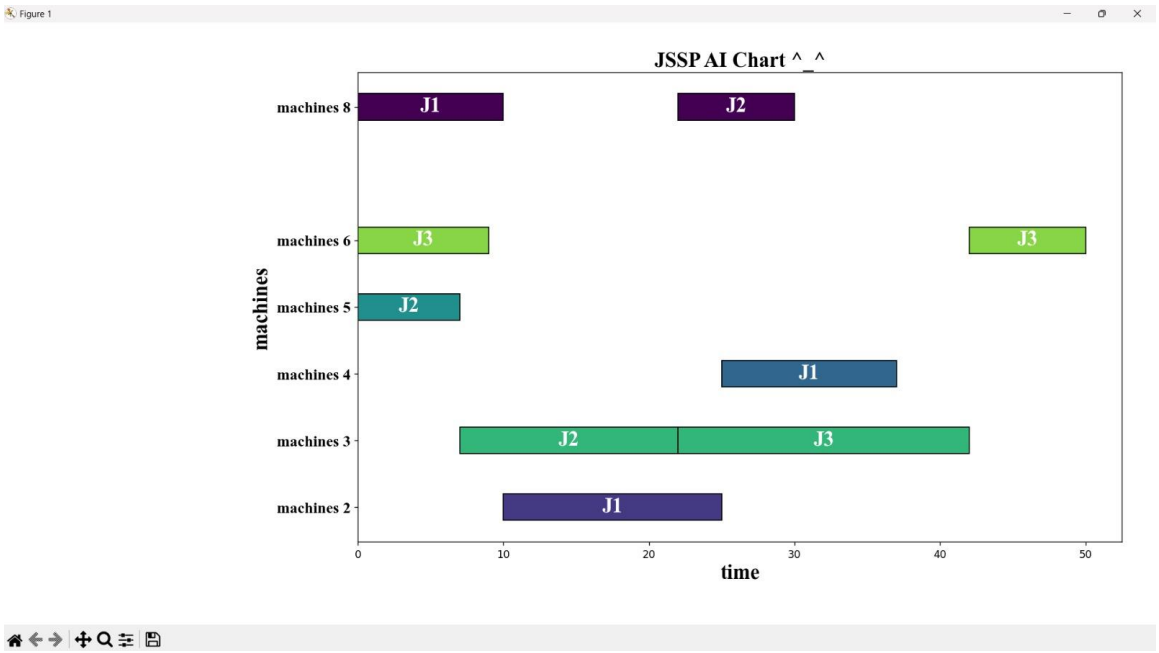

Figure 6:Gantt chart output of Input2

NOTE:

There is a function to read the data from the user, but there is a run time error and there is no time to fix it unfortunately. In the examples above we define random data to text the project functions.