Faculty of Engineering and Technology
Electrical and Computer Engineering Department

Artificial Intelligence
ENCS3340

Project 1

Optimizing Job Shop Scheduling in a Manufacturing Plant using Genetic Algorithm

---

Prepared by:

Qusay Taradeh	**ID: 1212508**	**Section: 4**
Ali Khalil	**ID: 1210750**	**Section: 1**

Instructor: **Dr. Yazan Abu Farha**

Date: **Monday, May 20, 2024**

Problem Formulation:

The Job Scheduling Problem set in the project involves scheduling a set of jobs on a set of machines that are inputted by the user, where each job has a sequence of operations that must be processed in specific order and each order in the sequence requires a specific machine and has a specific processing time. The objective is to determine a schedule that minimizes the make span, which is the total time required to complete all jobs. The objective is done by using Genetic Algorithm.

Chromosome Representation and Objective Function:

The chromosome in our project is the schedule of jobs done with their sequence and start and end time for each operation in the schedule. We can get different chromosomes by changing the order of jobs that are used to create it e.g. (chromosomes1 jobs order [(job1, (job1 sequence)), (job2, (job2 sequence))], chromosomes2 jobs order [(job2, (job2 sequence)), (job1, (job1 sequence))]). This will give us different schedules for each chromosome especially if both job1 and job2 need the same machine to complete the first order of operations.

The objective function or fitness is done at the same time as creating a chromosome because in our implementation we chose the make span to be the fitness of the chromosomes. So, we just take the maximum end time of the last order of operations for all jobs.

The data or the list of jobs and their operations is read from an excel file, its then stored into a list of tuples where each job and it sequence of operation is stored together, the sequence of operations is also a list of tuples where each machine and it proccing time are stored together.

Parents Selection:

We used tournament selection to select the parents, we take a random subset of the population then we compare the fitness of the selected subset and take the best one and repeat the process to select the other parent. Although we select the best schedules, we don’t send them to the cross-over function instead we send the job order used to make said schedule.

Cross-Over:

The cross-over function takes the job order of the parents and cuts them from the start to the cut point in the code e.g. (we had 10 jobs so we set the cut point to 5) So, the first 5 jobs of each parent is stored in a child. Then we use a for loop to check the parent not used to create the child e.g. (child1 will use parent2) if there is a job that's in the second parent and not in the child, we add it to the child if the job is already there, we don't add it. This ensures that all the jobs are stored in the child and that there is no duplication of jobs in it.

Then we send the children to the mutation function, then we create a schedule using the final job order of children, we don't add them immediately to the population, instead we compare them with the schedules in the population to find if there are any schedules that have a worst fitness than them, if there is, then remove them from the population and add the children to it, if there are no worse schedules in the population we don't add them.

Mutation:

The mutation function takes a job order as input not schedule and choses two random indices to swap in the job order. It has a set mutation rate e.g. (mutation rate = 0.1), it chooses a random value if its less than the mutation rate, then it swaps the chosen indices in the job order, otherwise it does nothing.

Generations:

We set the population limit to 20 and the generations limit to 30 and we do one crossover each generation from generation zero to 29. Then we chose the best schedule in the last generation and print it in detail with its fitness and we also generate a Gantt chart for it.

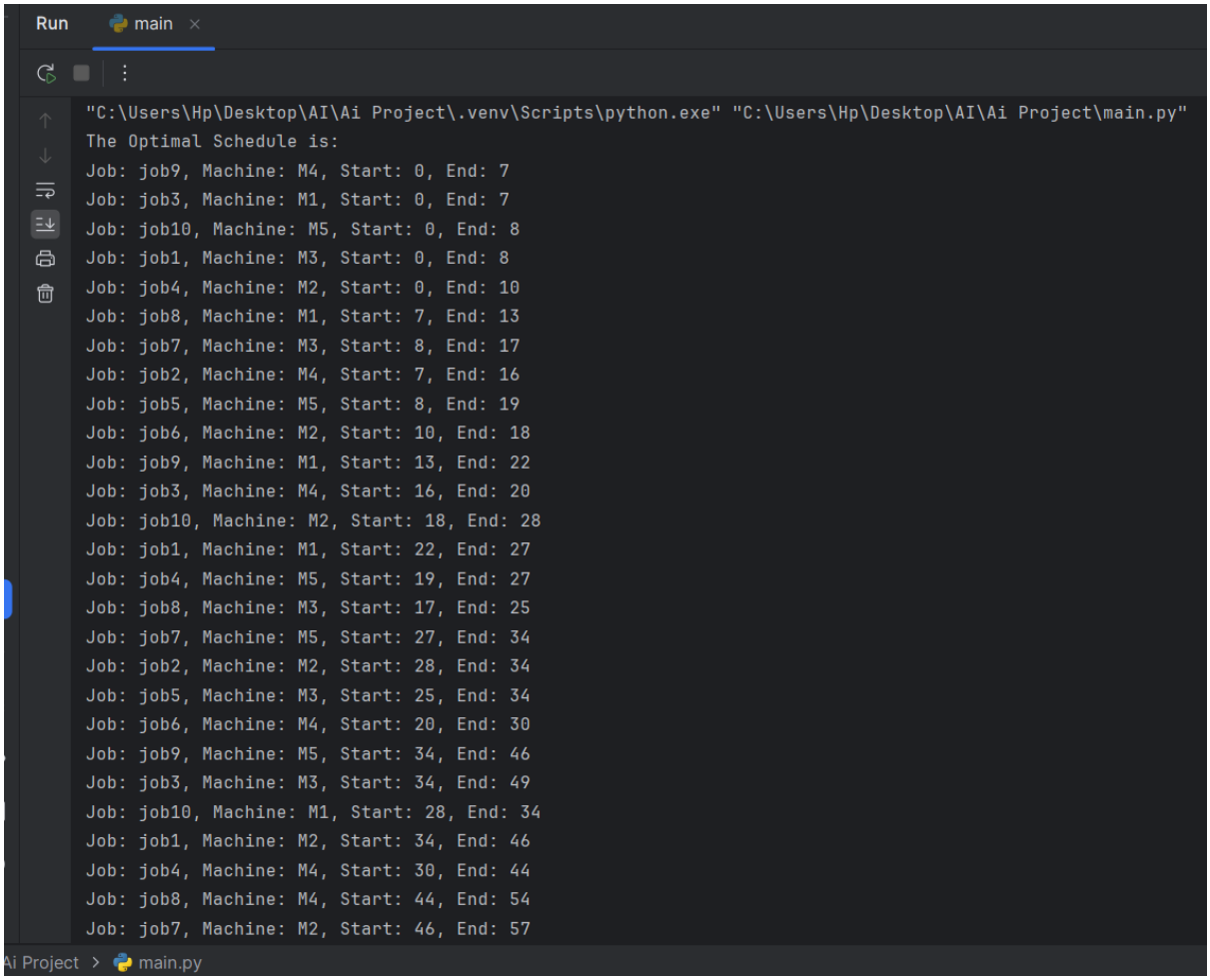
Test Cases:

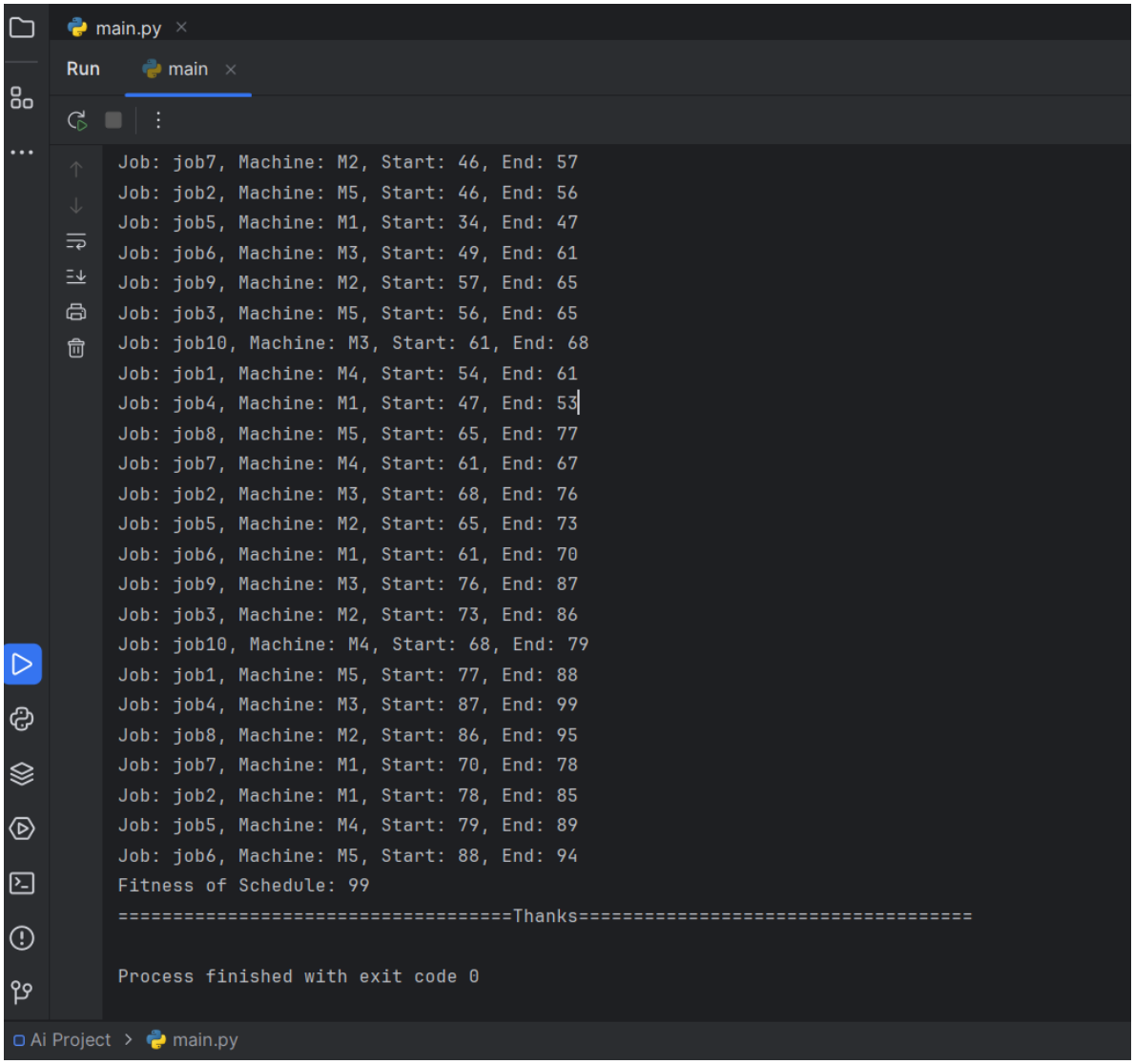
First test case:

We have 10 jobs, 5 machines and 5 orders of operation as shown below:

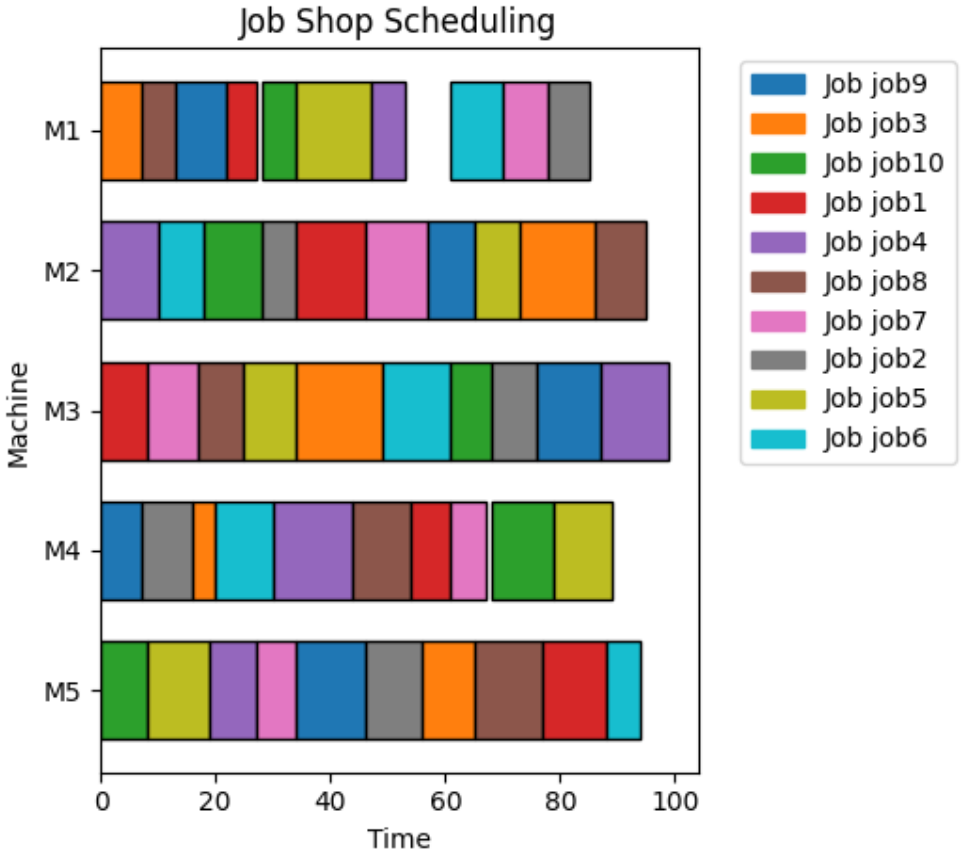
	A	B	C	D	E	F	G	H	I	J	K
1	Job	Order 1	Time 1	Order 2	Time 2	Order 3	Time 3	Order 4	Time 4	Order 5	Time 5
2	job1	M3	8	M1	5	M2	12	M4	7	M5	11
3	job2	M4	9	M2	6	M5	10	M3	8	M1	7
4	job3	M1	7	M4	4	M3	15	M5	9	M2	13
5	job4	M2	10	M5	8	M4	14	M1	6	M3	12
6	job5	M5	11	M3	9	M1	13	M2	8	M4	10
7	job6	M2	8	M4	10	M3	12	M1	9	M5	6
8	job7	M3	9	M5	7	M2	11	M4	6	M1	8
9	job8	M1	6	M3	8	M4	10	M5	12	M2	9
10	job9	M4	7	M1	9	M5	12	M2	8	M3	11
11	job10	M5	8	M2	10	M1	6	M3	7	M4	11

The output for the test case is:




main.py
Run
main
Job: job7, Machine: M2, Start: 46, End: 57
Job: job2, Machine: M5, Start: 46, End: 56
Job: job5, Machine: M1, Start: 34, End: 47
Job: job6, Machine: M3, Start: 49, End: 61
Job: job9, Machine: M2, Start: 57, End: 65
Job: job3, Machine: M5, Start: 56, End: 65
Job: job10, Machine: M3, Start: 61, End: 68
Job: job1, Machine: M4, Start: 54, End: 61
Job: job4, Machine: M1, Start: 47, End: 53
Job: job8, Machine: M5, Start: 65, End: 77
Job: job7, Machine: M4, Start: 61, End: 67
Job: job2, Machine: M3, Start: 68, End: 76
Job: job5, Machine: M2, Start: 65, End: 73
Job: job6, Machine: M1, Start: 61, End: 70
Job: job9, Machine: M3, Start: 76, End: 87
Job: job3, Machine: M2, Start: 73, End: 86
Job: job10, Machine: M4, Start: 68, End: 79
Job: job1, Machine: M5, Start: 77, End: 88
Job: job4, Machine: M3, Start: 87, End: 99
Job: job8, Machine: M2, Start: 86, End: 95
Job: job7, Machine: M1, Start: 70, End: 78
Job: job2, Machine: M1, Start: 78, End: 85
Job: job5, Machine: M4, Start: 79, End: 89
Job: job6, Machine: M5, Start: 88, End: 94
Fitness of Schedule: 99
==========Thanks==========
Process finished with exit code 0
Ai Project
main.py

The Gantt Chart of the test case:

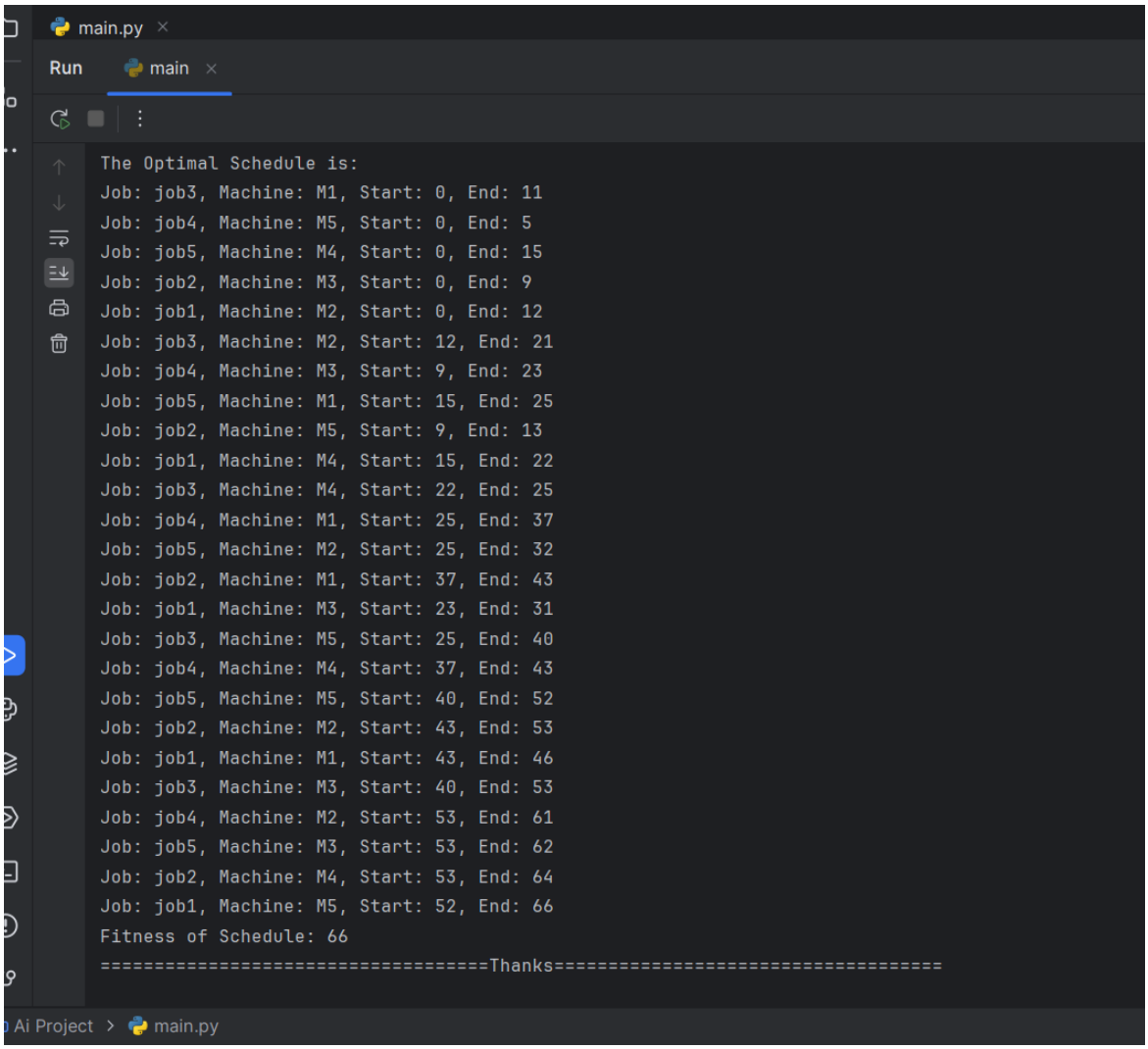


Second Test Case:

We have 5 jobs, 5 machines and 5 orders of operation as shown below:

	A	B	C	D	E	F	G	H	I	J	K
1	Job	Order 1	Time 1	Order 2	Time 2	Order 3	Time 3	Order 4	Time 4	Order 5	Time 5
2	job1	M2	12	M4	7	M3	8	M1	3	M5	14
3	job2	M3	9	M5	4	M1	6	M2	10	M4	11
4	job3	M1	11	M2	9	M4	3	M5	15	M3	13
5	job4	M5	5	M3	14	M1	12	M4	6	M2	8
6	job5	M4	15	M1	10	M2	7	M5	12	M3	9

The output of the above test case is:



The Gantt Chart of the test case: