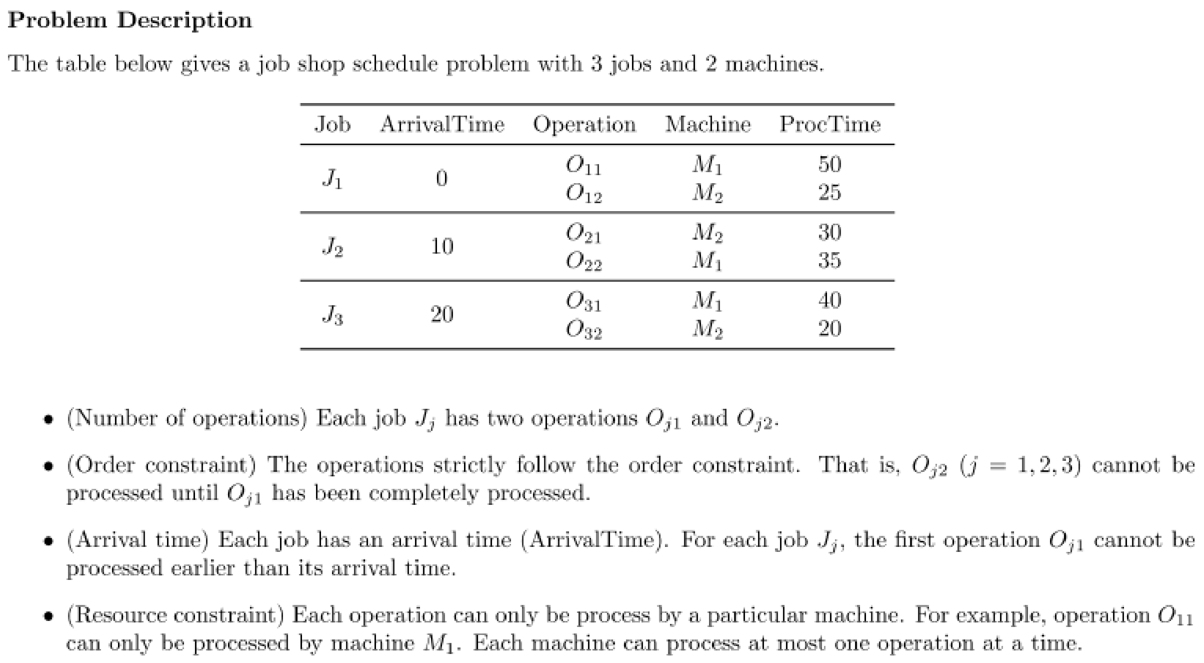
Yun Zhou 300442776

Part 1: Job Shop Scheduling

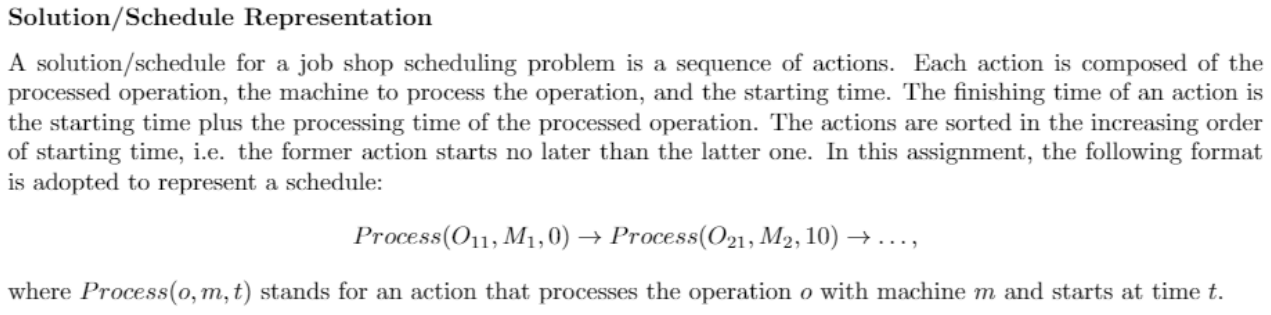


Problem Description

The table below gives a job shop schedule problem with 3 jobs and 2 machines.

Job	ArrivalTime	Operation	Machine	ProcTime
J_1	0	O_{11}	M_1	50
		O_{12}	M_2	25
J_2	10	O_{21}	M_2	30
		O_{22}	M_1	35
J_3	20	O_{31}	M_1	40
		O_{32}	M_2	20

- (Number of operations) Each job J_j has two operations O_{j1} and O_{j2}.
- (Order constraint) The operations strictly follow the order constraint. That is, O_{j2} ($j = 1, 2, 3$) cannot be processed until O_{j1} has been completely processed.
- (Arrival time) Each job has an arrival time (ArrivalTime). For each job J_j, the first operation O_{j1} cannot be processed earlier than its arrival time.
- (Resource constraint) Each operation can only be process by a particular machine. For example, operation O_{11} can only be processed by machine M_1. Each machine can process at most one operation at a time.



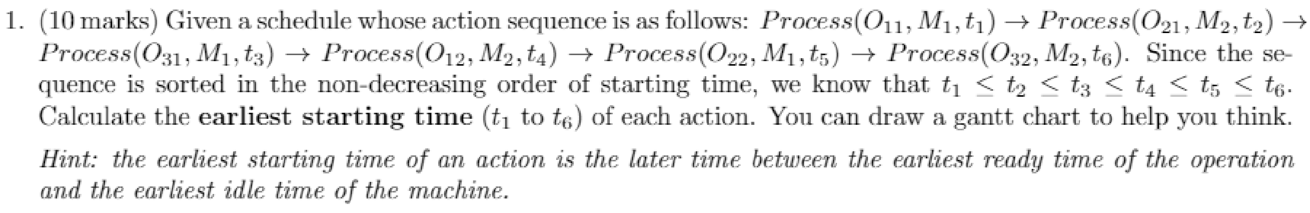
Solution/Schedule Representation

A solution/schedule for a job shop scheduling problem is a sequence of actions. Each action is composed of the processed operation, the machine to process the operation, and the starting time. The finishing time of an action is the starting time plus the processing time of the processed operation. The actions are sorted in the increasing order of starting time, i.e. the former action starts no later than the latter one. In this assignment, the following format is adopted to represent a schedule:

$$Process(O_{11}, M_1, 0) \rightarrow Process(O_{21}, M_2, 10) \rightarrow \ldots,$$

where $Process(o, m, t)$ stands for an action that processes the operation o with machine m and starts at time t.

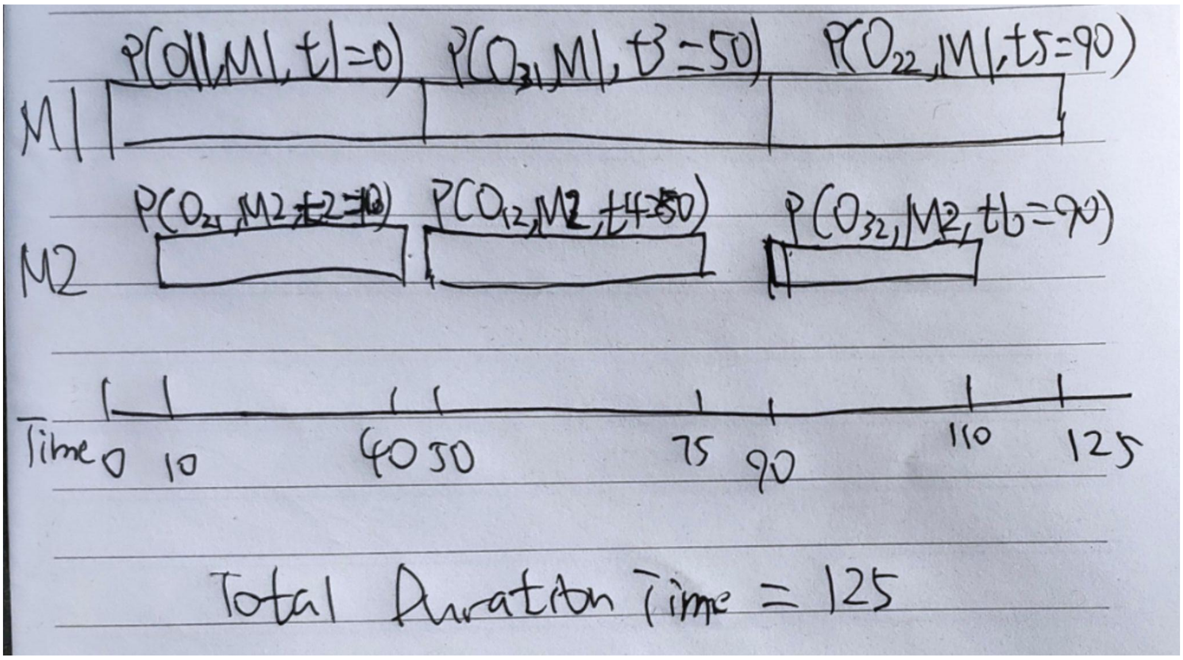
Q1:



1. (10 marks) Given a schedule whose action sequence is as follows: $Process(O_{11}, M_1, t_1) \rightarrow Process(O_{21}, M_2, t_2) \rightarrow Process(O_{31}, M_1, t_3) \rightarrow Process(O_{12}, M_2, t_4) \rightarrow Process(O_{22}, M_1, t_5) \rightarrow Process(O_{32}, M_2, t_6)$. Since the sequence is sorted in the non-decreasing order of starting time, we know that $t_1 \leq t_2 \leq t_3 \leq t_4 \leq t_5 \leq t_6$. Calculate the **earliest starting time** (t_1 to t_6) of each action. You can draw a gantt chart to help you think.

   Hint: the earliest starting time of an action is the later time between the earliest ready time of the operation and the earliest idle time of the machine.

Ghatt chart:

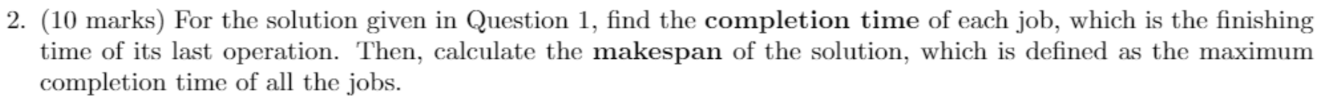


Earliest Starting Time table:

	Earliest Starting Time
Process(O11, M1, t1)	t1= 0
Process(O21, M2, t2)	t2 = 10
Process(O31, M1, t3)	t3 = 50
Process(O12, M2, t4)	t4 = 50
Process(O22, M1, t5)	t5 = 90
Process(O32, M2, t6)	t6 = 90

We can see the earliest starting time is shown above.
And the total duration is 125.

Q2:

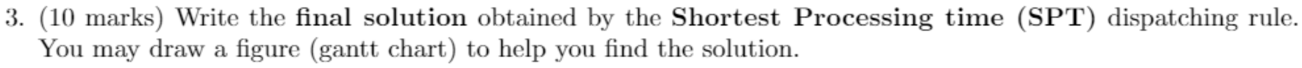


2. (10 marks) For the solution given in Question 1, find the **completion time** of each job, which is the finishing time of its last operation. Then, calculate the **makespan** of the solution, which is defined as the maximum completion time of all the jobs.

The completion time of Job 1 is 75. (which is starting time of Process(O12, M2, t4)+ProcTime = 50+25=75)
The completion time of Job 2 is 125. (which is starting time of Process(O22, M1, t5)+ProcTime = 90+35=125)

The completion time of Job 3 is 110. (which is starting time of Process(O32, M2, t6):+ProcTime = 90+20=110)
Therefore, the makespan is 125, since 125 is the time that all 3 jobs are finished.

Q3



3. (10 marks) Write the **final solution** obtained by the **Shortest Processing time (SPT)** dispatching rule. You may draw a figure (gantt chart) to help you find the solution.

Initialise state

empty schedule, all operations unprocessed, time = 0, machine idle time = 0, first operation ready time = arrival time, other operation ready time =∞
According to the slides, we can know that the Priority() is calculated as 0-ProcTime(O). So, we can get the full priority table which is shown below:

	Priority of Process(Operation,machine,time)		Arrival time
Process(O11, M1, t)	-50	=0-50	0
Process(O21, M2, t)	-30	=0-30	10
Process(O31, M1, t)	-40	=0-40	20
Process(O12, M2, t)	-25	=0-25	
Process(O22, M1, t)	-35	=0-35	
Process(O32, M2, t)	-20	=0-20	

Step 1:

According to the arrival time, the first operation is: Process(O11, M1, t1=0)
Since the arrival time for this **is 0** which is the earliest, meet the dispatch rule which is non-delay. So, now the schedule is Process(O11, M1, t1=0)

Then, for the following steps, we're:

- finding the earliest applicable actions;
- Select the next action by the dispatching rule
- and Add the selected action into the schedule, update the state

Step 2:

For this step, after apply the dispatching rule, the possible applicable actions are:
Process(O21, M2, t2=10)
Then, we can get that the next action is: Process(O21, M2, t2) .

Therefore, the schedule now is: Process(O11, M1, t1=0) ->Process(O21, M2, t2=10)

Step 3:

After counting the processing time taken on M1 &M2, the available time for M1 is 50(i.e.0+50), and for M2 is 40(i.e.10+30). But, due to the order constrain, the M2, for now, should also be 50, which is the same as M1.

For this step, the possible applicable actions are:

Process(O12, M2, 50)

Process(O31, M1, 50)

Process(O22, M1, 50)

Then, apply the dispatching rule, we can get that the next action is:

Process(O12, M2, 50) since it got the higher **priority**(-25>-35>-45).

Therefore, the schedule now is: Process(O11, M1, t1=0) ->Process(O21, M2, t2=10) -> Process(O12, M2, t3=50)

	Priority	
Process(O31, M1, t)	-40	=0-40
Process(O12, M2, t)	-25	=0-25
Process(O22, M1, t)	-35	=0-35
Process(O32, M2, t)	-20	=0-20

Step 4:

For this step, the possible applicable actions are:

Process(O31, M1, 50)

Process(O22, M1, 50)

Since the priority of Process(O22, M1, 50) is higher(-35>-45), so the next action is Process(O22, M1, 50)

Therefore, the schedule now is: Process(O11, M1, t1=0) ->Process(O21, M2, t2=10) -> Process(O12, M2, t3=50) ->Process(O22, M1, t4=50)

Step 5:

For now, after counting the processing time, the M1 time is 85(i.e.50+35), and the M2 is 75(i.e. 50+25), but also due to the order constraints (O32 can only be done after O31 is finished), so:

For this step, the only possible applicable action is:

Process(O31, M1, 85)

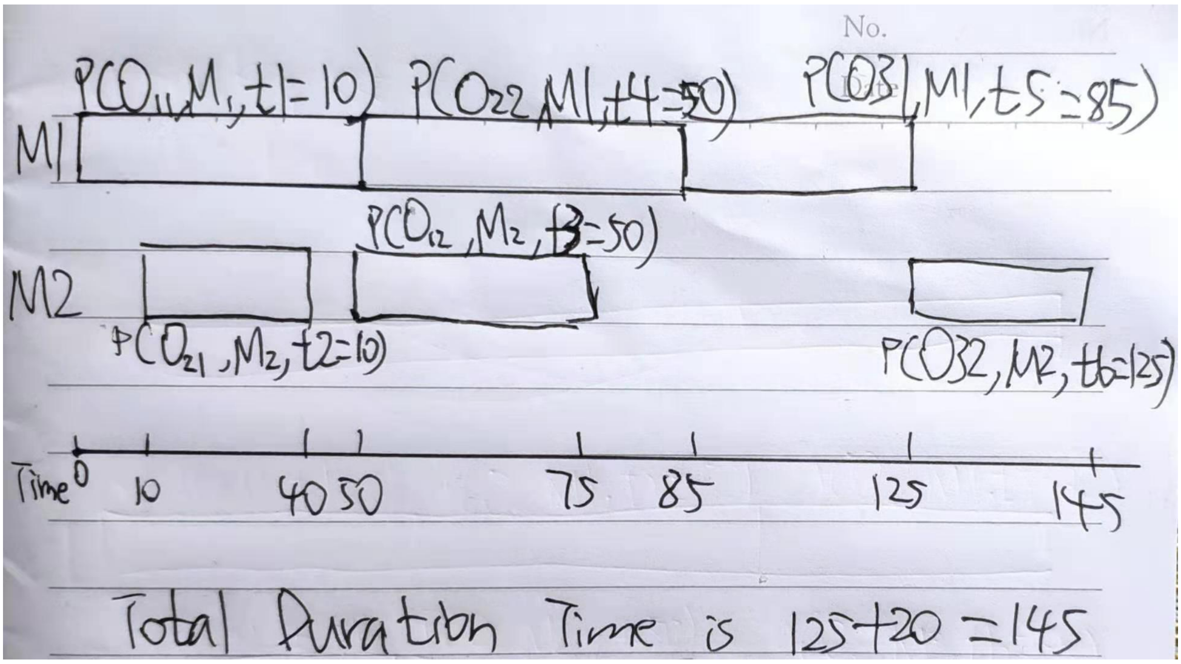
Therefore, the schedule now is: Process(O11, M1, t1=0) ->Process(O21, M2, t2=10) -> Process(O12, M2, t3=50) -> Process(O22, M1, t4=50) -> Process(O31, M1, t5=85)

Step 6:

For this step, the possible applicable action is:
Process(O32, M2, 125)
So, add into the schedule, we can get that the final solution is:

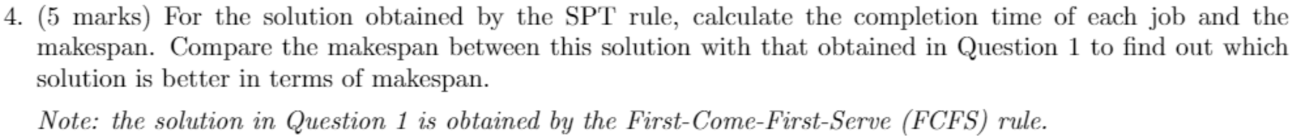
Process(O11, M1, t1=0) -> Process(O21, M2, t2=10) -> Process(O12, M2, t3=50) -> Process(O22, M1, t4=50) -> Process(O31, M1, t5=85) -> Process(O32, M2, 125)

Also, from this, we can get that the total duration is 125+20=145



Total Duration Time is 125+20=145

Q4:



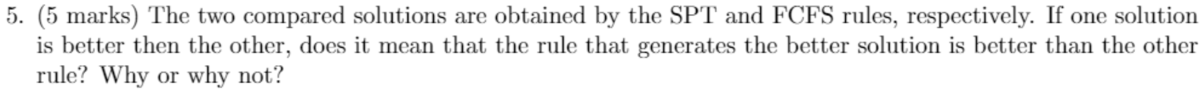
4. (5 marks) For the solution obtained by the SPT rule, calculate the completion time of each job and the makespan. Compare the makespan between this solution with that obtained in Question 1 to find out which solution is better in terms of makespan.

Note: the solution in Question 1 is obtained by the First-Come-First-Serve (FCFS) rule.

From the solution which is obtained by the SPT rule above, we can get that:
The completion time of Job J1 is 75.(which is starting time of Process(O12, M2, t3=50)+ProcTime = 50+25=75)
The completion time of Job J2 is 85.(which is starting time of Process(O22, M1, t4=50)+ProcTime = 50+35=85)
The completion time of Job J3 is 145.(which is starting time of Process(O32, M2, t6=125)+ProcTime = 125+20=145)
Therefore, the makespan for this SPT is 145, since 145 is the time that all 3 jobs are finished.

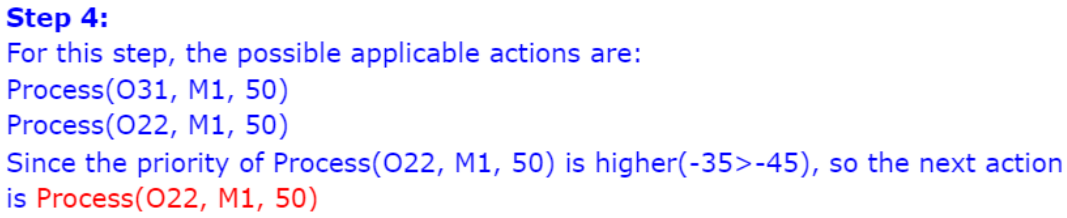
By comparing this to the result from Q1 which is the FCFS one, we can obtain that the solution based on the FCFS rule is better than the SPT one, since the makespan of FCFS is 125 which is lower than the FCFS's 145.

Q5:



5. (5 marks) The two compared solutions are obtained by the SPT and FCFS rules, respectively. If one solution is better then the other, does it mean that the rule that generates the better solution is better than the other rule? Why or why not?

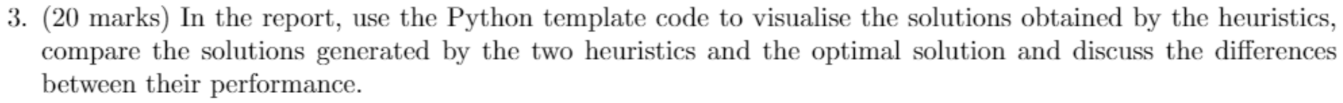
Although in this case, the FCFS rules perform a better solution than SPT, but it's not guaranteed that the FCFS rule is the better rule than the other.
That's because the First-Come-First-Serve rule only considers the earliest applicable actions that can be taken in order to minimise the waiting time, so, it can not handle the case that there are multiple applicable actions that can take at the same time. However, the SPT rule can not only do what FCFS rule can do, but it also can do what FCFS rule can not do. SPT rule will choose the next action based upon the priority function among all multiple applicable actions and choose the action with the highest priority. For instance, in our assignment, the processing time is used as the priority function. As we can see the Q3 Step 4 (which is also shown below), there is more than 1 applicable action, and Process(O22,M1,50) got the higher priority than the other, so it's chosen as the next action.



Step 4:
For this step, the possible applicable actions are:
Process(O31, M1, 50)
Process(O22, M1, 50)
Since the priority of Process(O22, M1, 50) is higher(-35>-45), so the next action is Process(O22, M1, 50)

Except this, the SPT also got other advantages that the First-Come-First-Serve rule does not have. For instance, it can be applied at ANY time point to change the remaining schedule; it is very fast in real-time, can handle dynamic environment very well since at each time point, complexity = #unprocessed ops * O(priority).
In conclusion, according to these advantages of SPT, I think the SPT rule is better than the FCFS rule even though FCFS generate a better solution in this schedule. I believe that in a much bigger job shop schedule problem, the SPT can generate a better solution than FCFS.

Part 2: Vehicle Routing



3. (20 marks) In the report, use the Python template code to visualise the solutions obtained by the heuristics, compare the solutions generated by the two heuristics and the optimal solution and discuss the differences between their performance.

As we can see in the graphs and the data are shown below, for both vrp files, we can see that the saving Heuristic perform and get a better result than the Nearest Neighbour Heuristic since it gets closer to the best VRP distance which can be seen from the below snippets (my algorithm sampleoutput)

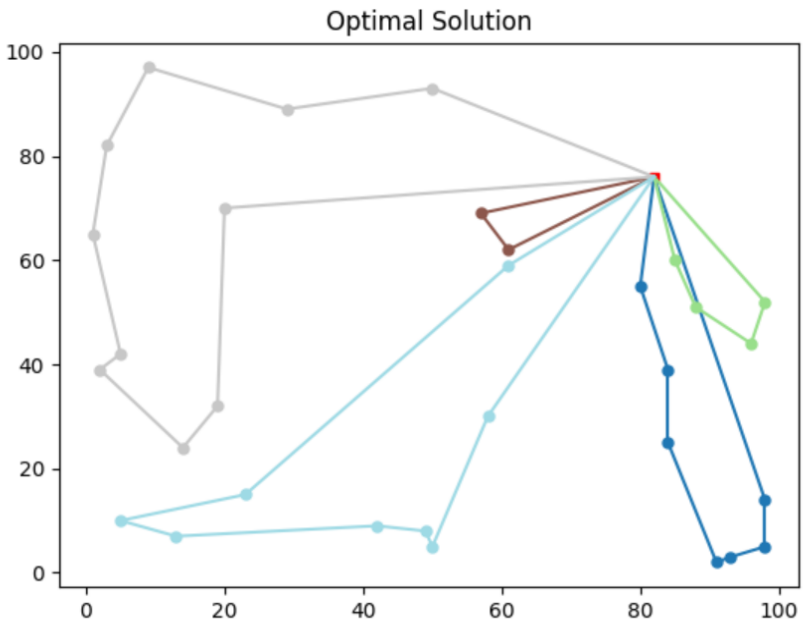
```
n32-k5:
   Best VRP Distance: 787.8082774366646
   Nearest Neighbour VRP Heuristic Distance: 1146.3996317253793
   Saving VRP Heuristic Distance: 994.3450215884798

n80-k10:
   Best VRP Distance: 1766.4999433704206
   Nearest Neighbour VRP Heuristic Distance: 2255.461886842153
   Saving VRP Heuristic Distance: 2112.4403288388535
```

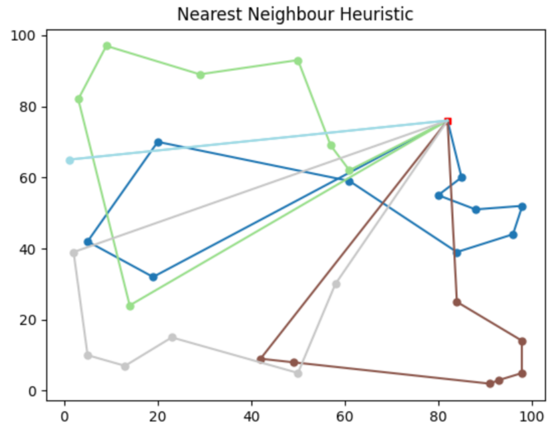
Therefore, we can get that the saving heuristic is a better algorithm than the NN. It is because the NN expand the next node by looking at which node is feasible(i.e. don't exceed the capacity) and got the closest distance, which means for the worst case which is the next feasible node is too far away from the current node may occur and so that it can cause the performance pretty low. However, the Saving Heuristic expand and merge based on the saving cost, which means that it will go through all feasible merges and merge two routes based on the priority of saving cost(higher saving cost get higher priority), which make sure that each step can find the best choice, try the best to avoid the worst-case from NN heuristic, so that's why it can get a closer result to the optimal solution.

(P.S. I find that I didn't get the correct result for saving heuristic since others got a better result. I think that's because, for the merge step, we should merge the last node from route1 and the first node from route2, but I fail to do it and merge the head node from both routes. I am not familiar to do the python, the variable doesn't need to specify the type, strange language.
)

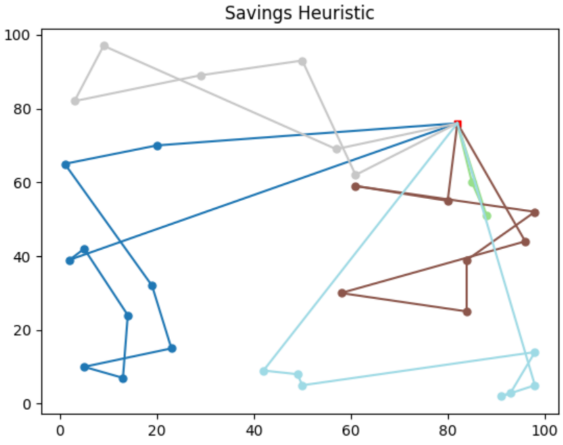
The graphs for both VRP files are shown below.



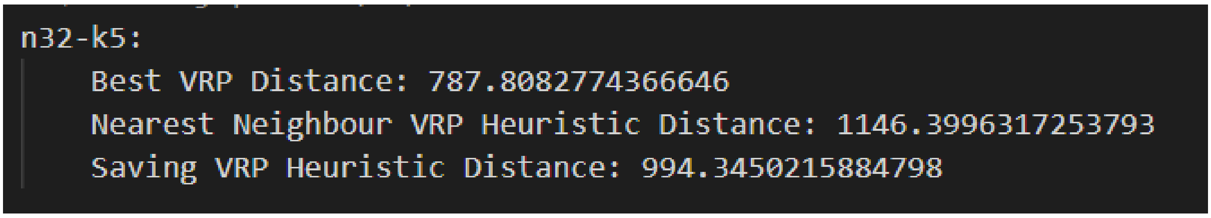
Optimal Solution



Nearest Neighbour Heuristic



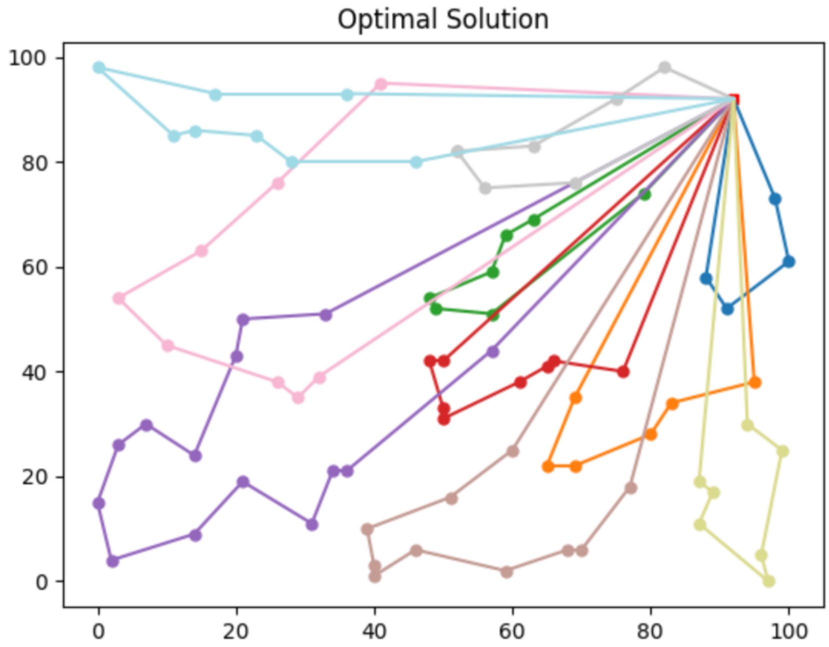
Savings Heuristic

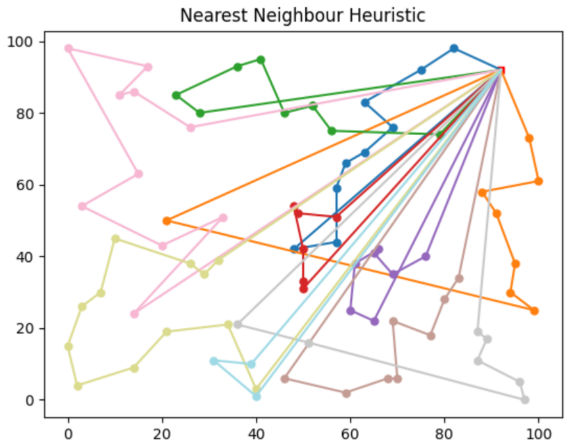


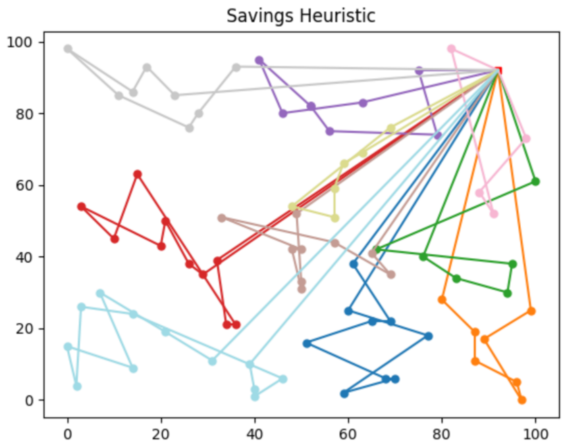
```
n32-k5:
    Best VRP Distance: 787.8082774366646
    Nearest Neighbour VRP Heuristic Distance: 1146.3996317253793
    Saving VRP Heuristic Distance: 994.3450215884798
```

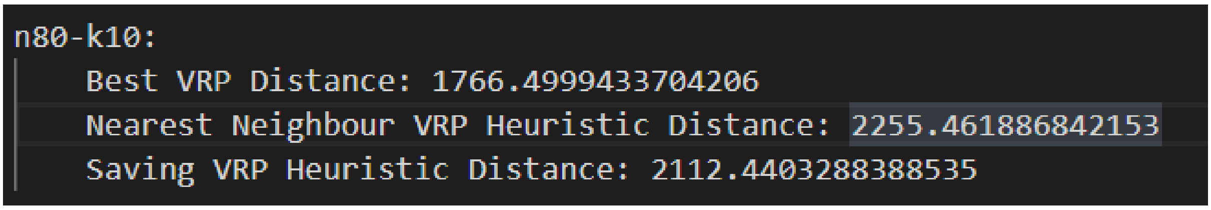
Above are my N32-k5.

Below are n80-k10









```
n80-k10:
    Best VRP Distance: 1766.4999433704206
    Nearest Neighbour VRP Heuristic Distance: 2255.461886842153
    Saving VRP Heuristic Distance: 2112.4403288388535
```

Above are my n80-k10