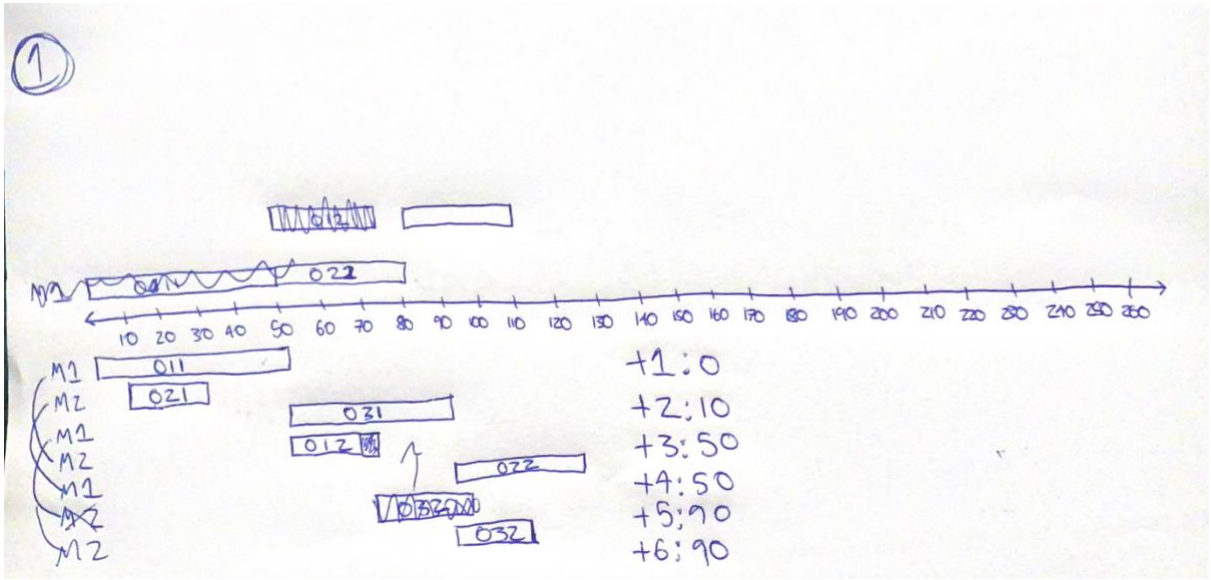
COMP307 – Assignment 4
greenthom – 300536064

Part 1: Job Shop Scheduling

1. **(10 marks) Given a schedule whose action sequence is as follows: Process(O_{11}, M_1, t_1) → Process(O_{21}, M_2, t_2) → Process(O_{31},M_1,t_3) → Process(O_{12},M_2,t_4) → Process(O_{22},M_1,t_5) → Process(O_{32},M_2,t_6). Since the sequence is sorted in the non-decreasing order of the 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.**



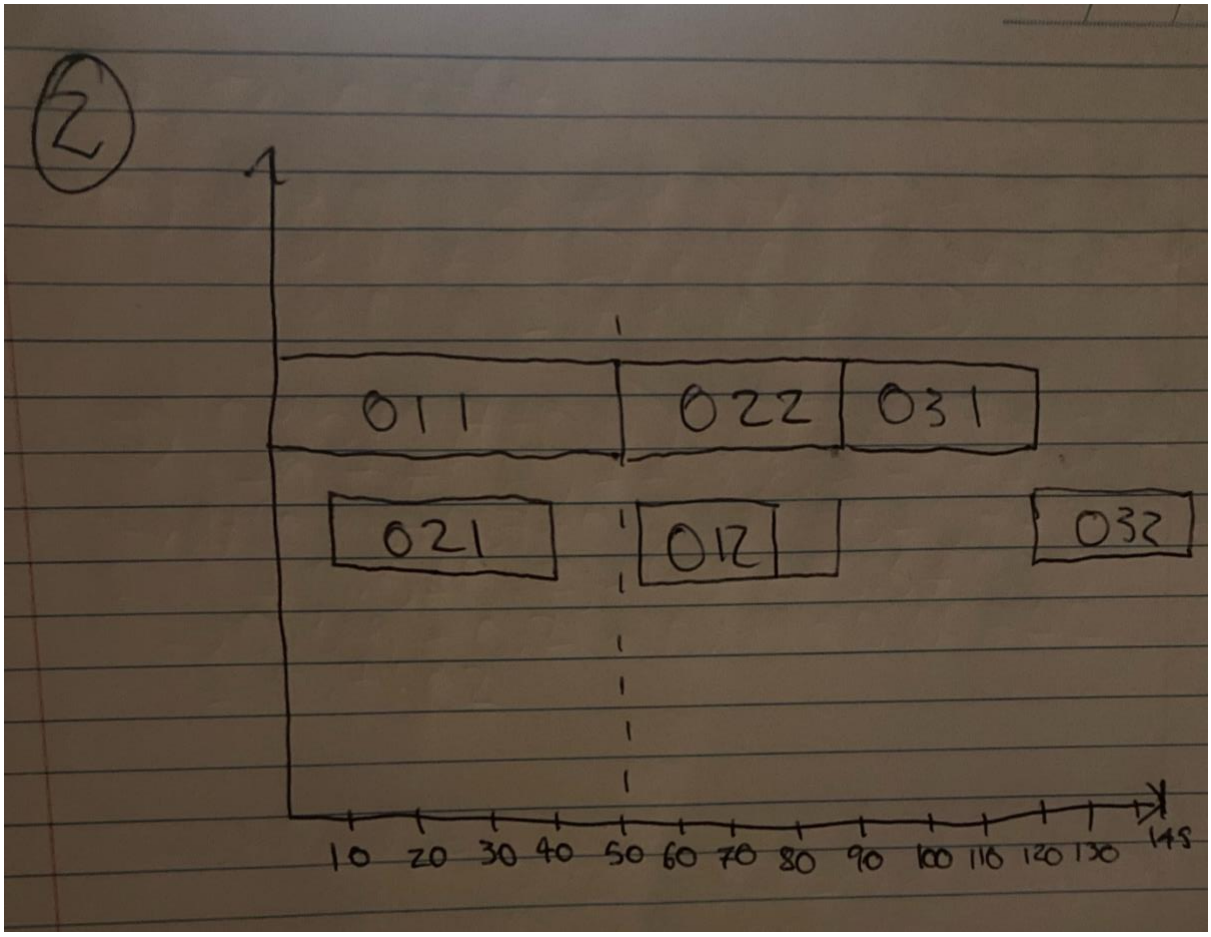
- Process(011, M1, t1) -> earliest starting time = t1= 0.
- Process(021, M2, t2) -> earliest starting time = t2 = 10.
- Process(031, M1, t3) -> earliest starting time = t3 = 50.
- Process(012, M2, t4) -> earliest starting time = t4 = 50.
- Process(022, M1, t5) -> earliest starting time = t5 = 90.
- Process(032, M2, t6) -> earliest starting time = t6 = 90.

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.**

- Process(012, M2, t4) + Time = t4(50) + 25 = 75
- Process(022, M1, t5) + Time = t5(90) + 35 = 125
- Process(032, M2, t6) + Time = t6(90) + 20 = 110

Makespan(FCFS) = max of 75, 125, 110 -> = 125 -> time it takes for all three jobs to be finished up.

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

Process(011, M1, 0) -> earliest arrival time is 0 -> meets non-delay.

Find earliest applicable actions:

Process(021, M2, 10) -> next action is Process(011, M1, 0) -> Process(021, M2, 10)

Select the next action by the dispatching rule:

Order constraint for M2 = 50 (due to processing time of M1 = 50. Possible actions are Process(012, M2, 50) or Process(031, M1, 50) or Process(022, M1, 50). Priority of 012 is 25 which is a higher priority, therefore Process(012, M2, 50)

Process = Process(011, M1, 0) -> Process(021, M2, 10) -> Process(012, M2, 50)

Next possible actions are: Process(031, M1, 50), Process(022, M1, 50). Priority of 022 is higher than 031 (35 and 40):

Process = Process(011, M1, 0) -> Process(021, M2, 10) -> Process(012, M2, 50) -> Process(022, M1, 50)

M1 time is 50+35 and M2 is 50+25 -> 032 can only be done once 031 is finished therefore -> Process(031, M1, 85)

Process = Process(011, M1, 0) -> Process(021, M2, 10) -> Process(012, M2, 50) -> Process(022, M1, 50) -> Process(031, M1, 85)

032 only process left so:

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

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.**

J1 = start time of Process(012, M2, 50) + ProcTime = 50 + 25 = 75
J2 = start time of Process(022, M1, 50) + ProcTime = 50 + 35 = 85
J3 = start time of Process(032, M2, 125) + ProcTime = 125 + 20 = 145

The makespan of the FCFS schedule is 125. The makespan of the SPT schedule is 145. FCFS has a better makespan in this instance.

5. **(5 marks) The two compared solutions in Question 4 are obtained by the SPT and FCFS rules, respectively. If one solution is better than the other, does it mean that the rule that generates the better solution is better than the other rule? Why or why not? Provide a short explanation of your answer.**

In this case FCFS rule performs a better outcome compared to SPT. However that does not mean that the FCFS rule is better compared to SPT. The FCFS rule is only considering the earliest time that can be taken to reduce the waiting time which results in it not handling cases where multiple actions are happening at the same time. The SPT rule can also do what FCFS does -> can choose the next action based on the priority function as well as all applicable actions which it then chooses the action with the highest ‘priority’. However this might result in higher waiting times for jobs which can increase the time it takes for jobs to be completed. FCFS does not take factor of these cases and mainly prioritizes the earliest times that an operation can be taken which can result in poor resource utilization, while SPT handles the priority of operations which is vital for a consistent result.

Scheduling rules can vary based on the specific problem it is handling, including factors that are present in this situation, such as job arrival times, processing times, and order

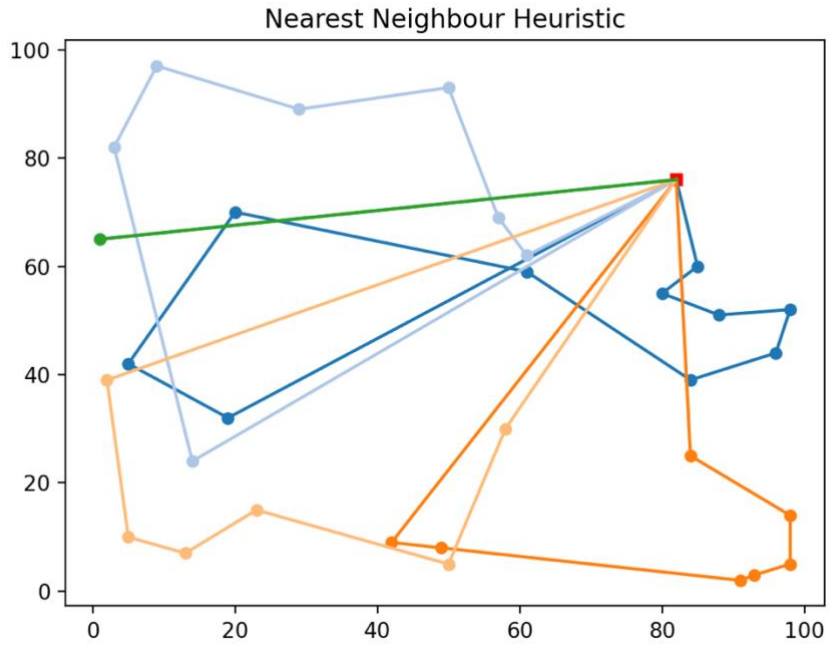
constraints. I believe SPT is better than FCFS in this situation due to the emphasis on order constraints at when a job should be processed due to its structuring around processing orders and resources. Even though the FCFS rule does generate a better result compared to the SPT solution, I believe SPT can handle structuring a bigger job schedule to generate a better solution compared to FCFS. However, this does depend on specific operational goals, the nature of the jobs, and the performance of the system. Each rule does have a scenario where it might outperform the other.

Part 2: Vehicle Routing

1. **Implement the nearest neighbor heuristic to generate a VRP solution. The distance between two nodes is defined as the Euclidean distance. That is, given two nodes v1 = (x1, y1) and v2 = (x2, y2), the distance is calculated as: dist(v1, v2) = square root((x1 – x2)^2 + (y1 – y2)^2)**

Implemented in main.py
N32 (Picture, routes, and distance below):

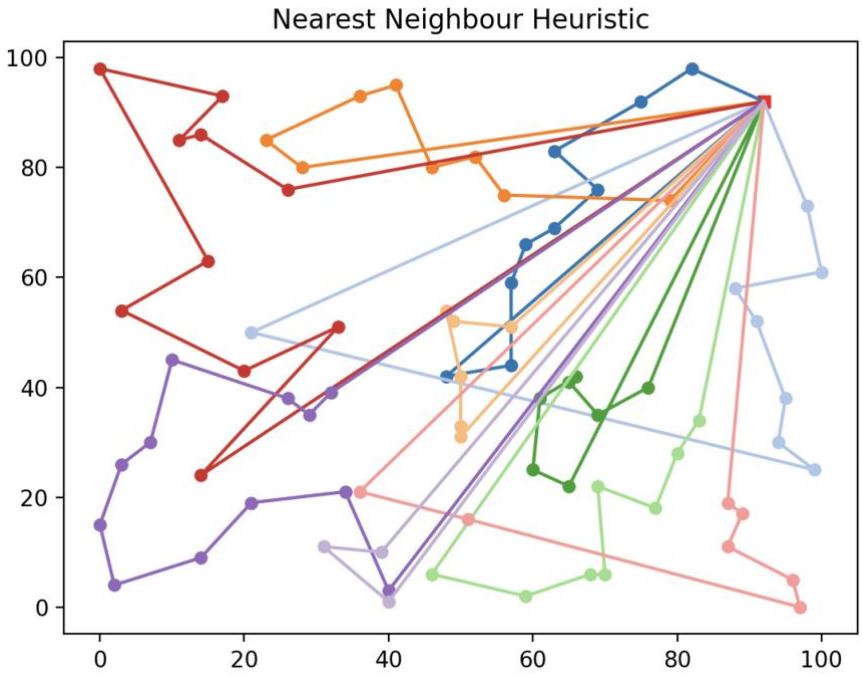
ROUTE 1 : [30, 26, 16, 12, 1, 7, 14, 29, 22, 18]
ROUTE 2 : [24, 27, 20, 5, 25, 10, 8]
ROUTE 3 : [13, 21, 31, 19, 17, 3, 23]
ROUTE 4 : [6, 2, 28, 4, 11, 9]
ROUTE 5 : [15]
Nearest Neighbor VRP Heuristic Distance: 1146.399631725379



N80 (Picture, routes, and distance below):

ROUTE 1 : [49, 73, 36, 42, 51, 77, 3, 29, 31]
ROUTE 2 : [40, 21, 1, 7, 10, 71, 14, 33]

ROUTE 3 : [13, 53, 66, 67, 70, 38, 58, 50, 76]
ROUTE 4 : [74, 60, 39, 17, 27, 59]
ROUTE 5 : [62, 23, 44, 12, 5, 30, 6]
ROUTE 6 : [63, 11, 34, 24, 2, 37, 8, 68]
ROUTE 7 : [72, 45, 22, 32, 4, 54, 9, 15, 64, 47]
ROUTE 8 : [52, 28, 79, 48, 18, 78, 20]
ROUTE 9 : [46, 25, 41, 55, 56, 69, 65, 35, 26, 19, 75, 16]
ROUTE 10 : [61, 57, 43]
Nearest Neighbor VRP Heuristic Distance: 2255.461886842152

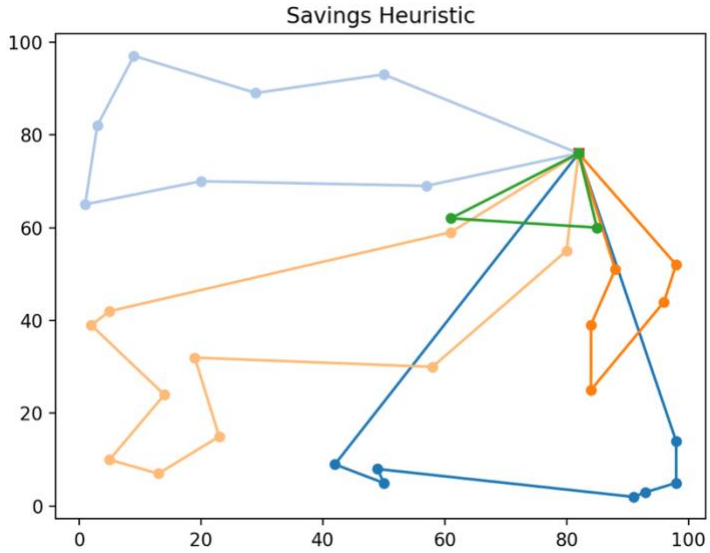


2. Implement the savings heuristic to generate a VRP solution.

N32 (Picture, routes, and distance below):

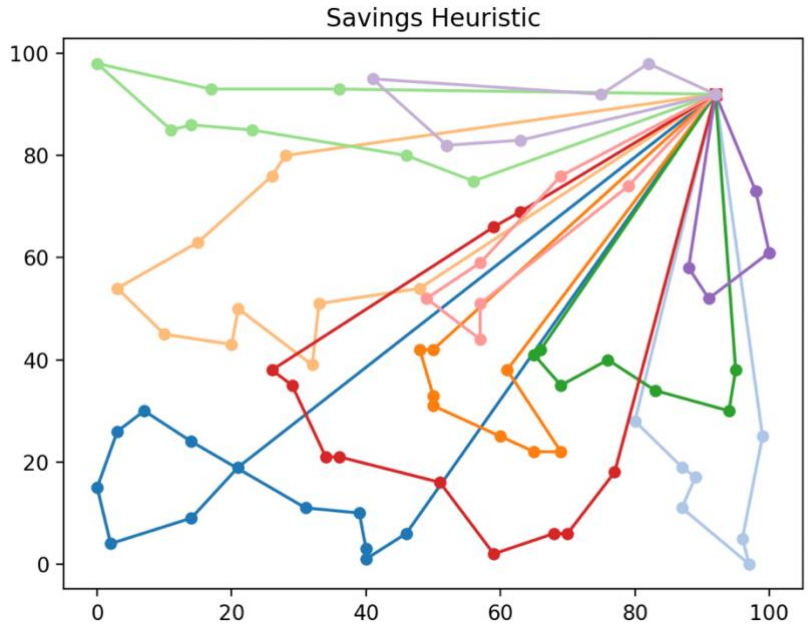
Route 1 : [0, 23, 2, 3, 17, 19, 31, 21, 0]
Route 2 : [0, 20, 5, 25, 10, 15, 29, 27, 0]
Route 3 : [0, 16, 7, 13, 1, 12, 0]
Route 4 : [0, 26, 6, 18, 28, 4, 11, 8, 9, 22, 14, 0]
Route 5 : [0, 24, 30, 0]

Saving VRP Heuristic Distance: 843.688169346627



N80 (Picture, routes, distance below):

Route 1 : [0, 19, 26, 35, 65, 69, 56, 47, 57, 61, 16, 43, 68, 0]
Route 2 : [0, 11, 52, 28, 79, 18, 48, 14, 0]
Route 3 : [0, 17, 31, 27, 59, 30, 6, 24, 5, 0]
Route 4 : [0, 76, 72, 54, 9, 55, 15, 33, 46, 64, 39, 0]
Route 5 : [0, 12, 44, 23, 62, 63, 71, 10, 0]
Route 6 : [0, 58, 32, 4, 22, 45, 50, 70, 53, 0]
Route 7 : [0, 34, 2, 37, 8, 78, 20, 75, 25, 41, 77, 51, 0]
Route 8 : [0, 13, 74, 29, 60, 3, 42, 0]
Route 9 : [0, 1, 7, 21, 40, 0]
Route 10 : [0, 49, 73, 38, 66, 67, 36, 0]
Saving VRP Heuristic Distance: 1836.8389976301548



3. **In the report, use the Python code template to visualize the solutions obtained by the heuristics, compare the solutions generated by the two heuristics and the optimal solution. Discuss the differences among these solutions in terms of their total distances.**

Comparing the two heuristics against the optimal solution (N32 = 787), it is clear that the savings heuristic was much more effective at gaining an optimal solution (N32 = 843, N80 = 1836) compared to nearest neighbor (N32 = 1146, N80 = 2255). The savings heuristic effectively gains a better optimal solution due to minimizing the total travel distance by optimizing the order of node visits. When comparing against large datasets such as N80, where there is a slightly higher dataset compared to N32, the need to manage increased complexity and larger sets of data shows the advantages that the Savings heuristic has over the nearest neighbor. One issue with savings heuristic is that it can be limited by the starting state -> savings is built on this state by merging routes.

The nearest neighbor, while faster in generating initial routes, results in a higher distance due to how it is handling optimization which overlooks potential savings in distance due to it trying to find the nearest node. It starts from the starting point (depot) to then looks for the closest/nearest point. One issue with nearest neighbor is progression searching as it is prone to investigating irrelevant actions. This can be an issue as it will just look for just the closest neighbor, which is a massive problem with datasets such as N80 due (larger datasets), leading to significantly longer routes which affects the cost of the algorithm compared to the savings heuristic and the optimal solution.

While the nearest neighbor heuristic does offer speed and simplicity, the savings heuristic is the better choice in scenarios where route optimization is critical due to its linkages between nodes to reduce distance cost which achieves a better optimization. Savings heuristic is designed for multiple paths within the context where nearest neighbor only considers one path at a time, making it a better choice for routing with larger datasets such as N80, and N32.