# Assignment 4

## Part 1: Job Shop Scheduling

| 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 |

### *1. Calculate the earliest starting time ($t_1$ to $t_6$) of each action.*

$P(O_{11},M_1, t_1)$ -> $P(O_{21},M_2,t_2)$ ->$P(O_{31},M_1,t_3)$-> $P(O_{12},M_2, t_4)$->$P(O_{22},M_1, t_5)$->$P(O_{32},M_2,t_6)$.

| | t1 | | t2 | | | | | | | | t3,t4 | | | | | | | | t5,t6 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 | 80 | 85 | 90 | 95 | 100 | 105 | 110 | 115 | 120 | 125 |
| M1 | O11 | | | | | | | | | | O31 | | | | | | | | O22 | | | | | | | |
| M2 | | | O21 | | | | | | | | O12 | | | | | | | | O32 | | | | | | | |

| t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|
| 0 | 10 | 50 | 50 | 90 | 90 |

### *2. 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.*

$t_{o1}$=75
$t_{o2}$=125
$t_{o3}$=110
makespan =125

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

| | t1 | | t2 | | | | | | | | t4,t5 | | | | | | | t3 | | | | | | | | t6 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 | 80 | 85 | 90 | 95 | 100 | 105 | 110 | 115 | 120 | 125 | 130 | 135 | 140 | 145 |
| M1 | | | | | | | | | | | O22 | | | | | | | O31 | | | | | | | | | | | | |
| M2 | | | O21 | | | | | | | | O12 | | | | | | | | | | | | | | | O32 | | | | |

$P(O_{11},M_1, t_1)$ -> $P(O_{21},M_2,t_2)$ ->-$P(O_{22},M_1, t_5)$-> $P(O_{12},M_2, t_4)$-> $P(O_{31},M_1,t_3)$-> $P(O_{32},M_2,t_6)$.

***4. 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***

$t_{o1}$=75
$t_{o2}$=85
$t_{o3}$=145
makespan =145

For this task FCFS rules is better as a makespan (125) is shorter then the makespan for second SPT (145). The difference is 20.

***5. The two compared solutions 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?***

For particular task FCFS method appeared more effective and gave better solution in case of makespan as criteria of effectiveness, however it could be different with another dataset and also it could be more effective even in our case and we would consider the second solution better if the criteria of success would not be a makespan.

To choose which rule from this two is better we should look at the purpose of the process. If the purpose is to complete operations on those job which come first (for example if it related with orders from clients) the FCFS rule might be more effective. However, some short processes might be pushed back in the queue. With SPT is opposite. Shorter processes are going first. The method normally used for minimizing number of jobs in the process or average flowtime for jobs . The disadvantage that it is moving the largest jobs to the end.

There are also other priority rules which are used for scheduling. For example :

- Earliest Due Date (EDD)
- Slack Time Remaining (STR)
- Maximum Work Remaining(MWR)
- Critical Ratio (CR)

Each of them has advantages and disadvantages and should be chosen to serve the purpose of the system.

***6. Often in practice, neither the SPT nor FCFS rules are good enough for solving the job shop scheduling problem. Suggest two methods to solve the job shop scheduling problem. One method should consider the problem to be static (all information is known in advance), and the other method should consider it to be dynamic (e.g. unpredicted job arrivals can happen in real time). Clearly describe your methods (e.g. algorithm framework, solution representation/encoding, search operators).***

## 1. Simulated annealing

Simulated annealing is a meta-heuristic method and can be initialized from the previous heuristic solutions(in our case FCFS and SPT) and improve them with each iteration.

### *Initial information:*

- Arrival Time
- Number of operations

### *Constraints:*

- Order constrains
- Recourse constrain (machines)

### *Objective function:*

- Makespan

### *Random component:*

- Initial T (temperature)
- T decreasing schedule

### *Steps:*

1. Initialisation. Choose an initial schedule S to be the current solution (Or generate random) and compute the value of the objective function F(S).

2. Neighbour Generation. Select a neighbour of the current solution and compute F(S').

-Swap machines

-Swap arrival time

-Change Jobs by swapping or inserting.

3. Acceptance Test. Test whether to accept the move from S to S'.

To jump from local minimum (maximum) SA suggests the following algorithm.

- If S' better than S, then a move to schedule S'
- if S better than S' move to S' with probability $e^{-S-S'/T}$

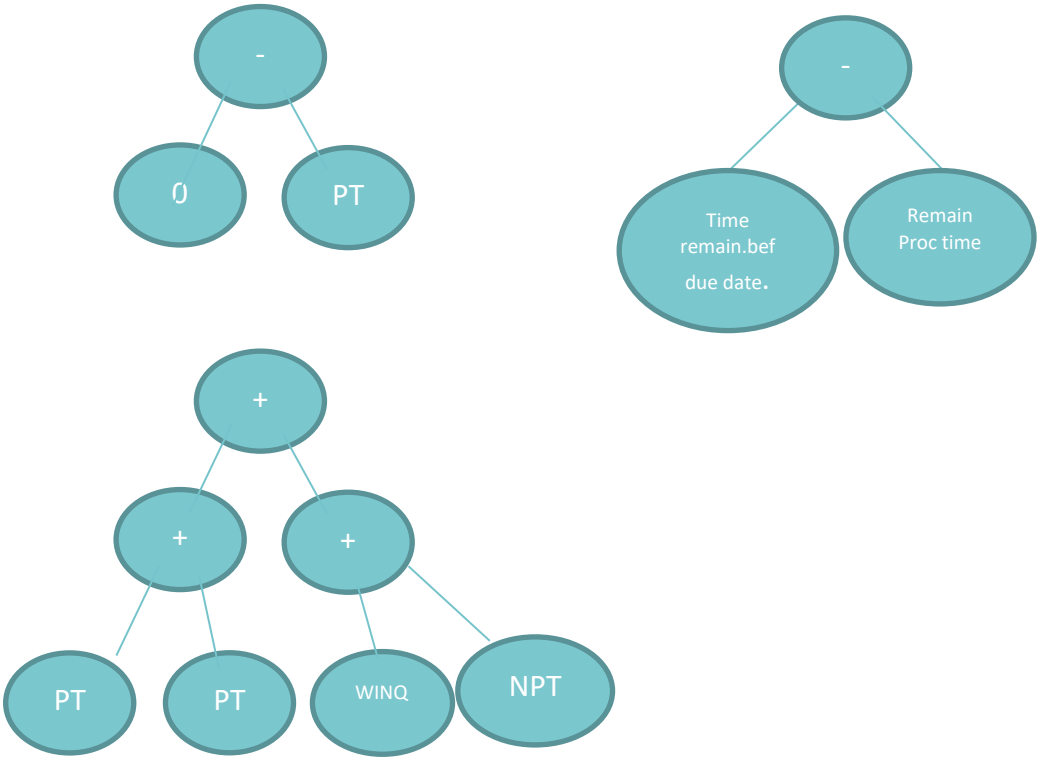
4. Termination Test. Terminates with output or goes to 2.

*Resource used for steps:*

*http://web-static.stern.nyu.edu/om/faculty/pinedo/scheduling/shakhlevich/handout10.pdf*

## 2. Genetic programming

With genetic programming we can find the best priority function. Solution will be represented in best composite dispatching rules. I our case it could be




***Terminal set:***

-Processing time of o

-Processing time of the next operation

-Work remaining

-Arrival time

-Random number

-Next processing time

-Work in next queue

***Functions set:***

-{+,-,x,/}

-{max, min}

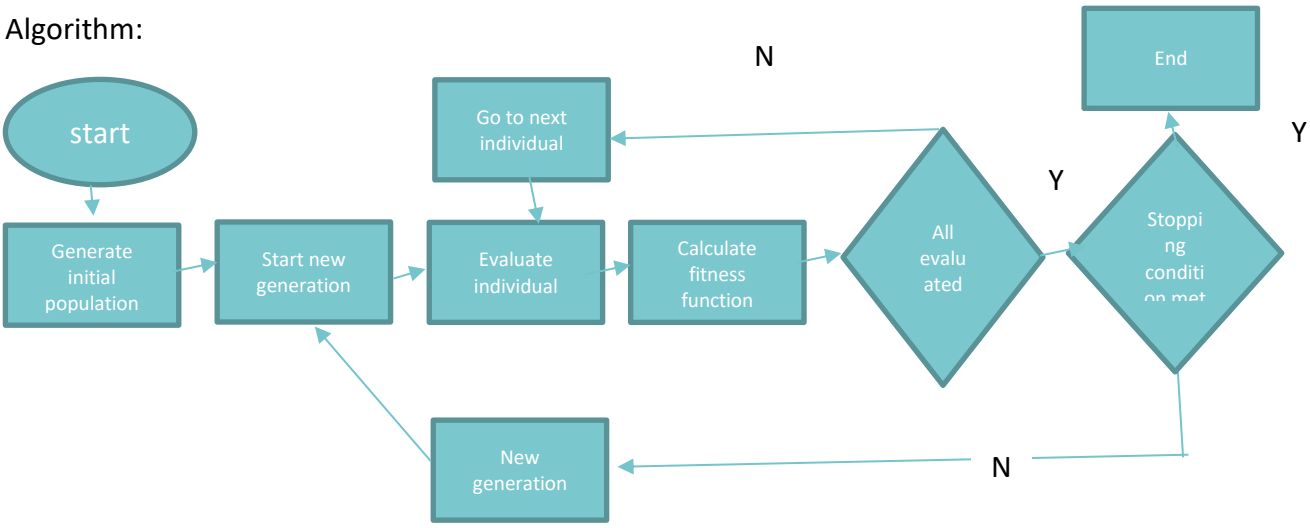
### *Fitness function:*

Makespan (minimisation)

### *Parameters:*

Initialisation
Crossover rate
Mutation rate
Reproduction rate
Max depth
Number of generations
Population size
Tournament selection

Algorithm:



# Part 2: Part 2: Vehicle Routing

### *0. Script summary overall:*

The modified sections:

Main.py

- nearest_neighbour_heuristic

-saving_heuristic

-get minimum (helper function)

-find maximum(helper function)

Utility.py

- calculate_euclidean_distance

- calculate_total_distance

- save_solution (saves the results in files)

### *1. Implement the nearest neighbour heuristic to generate a VRP solution*

The NN method implemented in the main.py script. It is based on euclidian and total distance functions.

[90.0, 94.0, 80.0, 89.0, 57.0]

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

Nearest Neighbour VRP Heuristic Distance: 991.6705767881119

***Script comments:*** The algorithm is slightly different from what I guess it should be as it is closing route based on capacity limit without checking next suitable solution. However, the total distance is shorter in this case then otherwise.

### *2. Implement the savings heuristic to generate another VRP solution*

97.0, 99.0, 93.0, 77.0, 44.0]

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

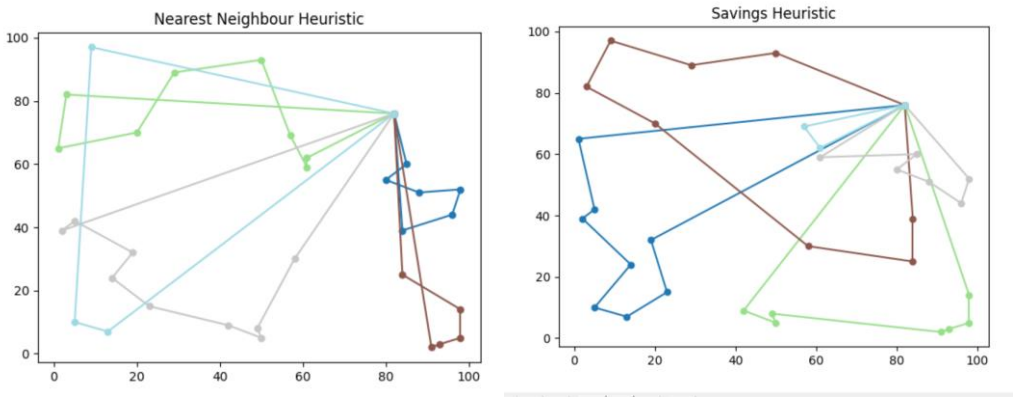
[[0, 18, 28, 4, 11, 8, 9, 22, 15, 0], [0, 23, 2, 3, 17, 19, 31, 21, 0], [0, 7, 13, 6, 29, 10, 25, 5, 20, 0], [0, 14, 30, 26, 16, 1, 12, 0], [0, 24, 27, 0]]

Saving VRP Heuristic Distance: 896.9186943468951

***Script comments:***

Here the algorithm is also slightly different, as merging is happening not with all routes together but growing from both side of first merged nodes and then open the new route and finds the maximum savings from remaining nodes. So order is different and performance worse than it should be.

3. ***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.***




Best VRP Distance: 787.8082774366646

Nearest Neighbour VRP Heuristic Distance: 991.6705767881119

Saving VRP Heuristic Distance: 896.9186943468951

The Nearest Neighbour method as greedy search remains isolated nodes and the last solutions are not optimal. That is why the performance is  worse than optimal.

The Savings Heuristic performs slightly better as should work simultaneously on all the routes.  However, the local search methods have disadvantage and have worse results in compartment with optimal solution. To receive a better solution simulated annealing, tabu search and genetic algorithm could be used. The  specific components of tabu search and simulated annealing allow to jump off local maximum(minimum) and genetic algorithm aimed the best performance using composite solutions which makes it more flexible.

4. ***Suggest a more advanced method that can be better than the above two heuristics. Clearly describe your methods (e.g. algorithm framework, solution representation/encoding, search operators).***

## Tabu search

Tabu Search starts just as an ordinary local search, proceeding iteratively from one solution to the next until some stopping criteria is satisfied while making use of some strategies to avoid getting trapped in a local optima.

Tabu search is based on a tabu list. Tabu list stores attributes of the previous few moves. It has a fixed number of entries and it is updated each time a new schedule S' is accepted. Transformation is entered at the top of the tabu list to avoid returning to the same solution , all other entries are pushed down one position and the bottom entry is deleted.

### *Initialisation:*

- Set of customer nodes
- Set of demands
- Set of x and y coordinates

### *Constrains:*

- Capacity

### *Features of tabu search*

Search space – feasible routes satisfied constrains

Neighbourhood structure – moving from one customer to another place in the route

Tabus – set of acting preventing rules

Aspiration criteria – removing tabu

Stopping criteria - number of iteration, number of iteration without improvements, threshold

Steps (Similar to local search)

1. Initialisation. Choose an initial solution i in S. Could be generated by NN method.
2. Generate subset with other solution violated or satisfied
3. Choose the best j and set i=j. Determine $\Delta$= F(S') - F(S). which met constraints.

   If $\Delta<0$ and S' is " non-tabu", then a move to S' is always accepted.

   If $\Delta<0$ and S' is " tabu", then a move to i=j may be

   If $\Delta\geq0$ and S' is " tabu", then a move i=j is always rejected.

   If $\Delta\geq0$ and S' is " non-tabu", then a " wait and see" if no immediate accepting neighbours, a move to the best candidate i=j

4. Update tabu list and aspiration conditions

5. Stopping criteria met otherwise goes to 2

*Resource used:*

*https://www.ajol.info/index.php/umrj/article/view/131122*

*http://web-static.stern.nyu.edu/om/faculty/pinedo/scheduling/shakhlevich/handout10.pdf*