Heuristic approach:

My goal is to minimize the routing time and to reduce the number of agents used in routing.

The idea of the code is to start the iteration with number of vehicles equals to number of nodes in the network. Let say 9. I need to have iteration loop for vehicle schedules, nodes are to be scheduled to vehicles. In each iteration will have multiple variables that will be updated each time the iteration happen. Let me explain what happen in the iteration in simple way. At the **first** iteration, because the vehicles number is the same as number of nodes, each vehicle will route one node only. And this will be done by Insertion based solution construction algorithm (1.1 look below). And in the next iteration, vehicle numbers will decrease by one and another vehicle will route two nodes. And so on and so forth, so in each iteration the following will happen:

1. vehicle numbers will decrease by one.
2. Each vehicle will have a schedule (data structure: array , let say)
3. A node will remove from the removed vehicle to another schedule (local search in 1.2).
4. Calculate routing time for each vehicle starting from source and end at destination
5. Remove a vehicle depend on the largest routing time.
6. Update the time spent in the routing for each vehicle (apply A* search algorithm for each vehicle)

The road network is like a graph, creating the network is already done as a function or method in the jsprit. I want to have nodes as the same number as the vehicles. And the links between them (roads) has time as weight (in minutes). I should edit the weight(time in minutes) of links/edges between the nodes/ vertices. And the roads that go out from the source have the same weight (larger than the largest weight of all roads. For instance, if a road has weight of 10 minutes, these roads will have 15 minutes as weight, called upfront cost).

The vehicles will start from a point (source) and end at different point (destination).

Each vehicle will have at least one node to route in roads. All vehicle located in the source as a starting. We have to use A* search algorithm to find the shortest time path for each vehicle to route from source to destination.

The variables are:

1- number of agents/vehicle.
2- Routing times.
3- Total routing time of all vahicles
4- Temp routing time
5- Varience (updated in each iteration)
6- Schedule for each vehicle, so, 15 schedules in this example.
7- Var. to save largest routing time
8- Counter starts from 0.

OUTPUT NEEDED:

Graph and tables shows:

Number of agents, total shortest routing time for all agents, solution table , detailed solution table, problem table, graphical presentation plotter,

I need you to add weights on edges. I don't think Jsprit offer adding weight on edges. So, you have to make a new class to do that job, to add time on edges and let the vehicle take the short time according to that time we add by ourselves, not according the locations or time windows. In other words, the vehicles take the job/ services depending on the weight of the edge (time) and the shortest time path is taken from the time on edges also. I don't need distance calculating. All the mater is time and only time. that will be used in local search algorithm.

You have to comment the code very well and locate where is the algorithm used.

Important Algorithm will use:

1.1. Insertion based solution construction heuristic:

Its main feature is the generation of the general solution of scheduling all basic nodes to all agents we start the solution of n nodes and n agents. The solution will start generating a schedule for each agent and assign a node to each agent. Thus, each agent will have a node in its schedule. When all nodes are scheduled, then the construction ends. Local search algorithm will start with schedules of one node at each and gradually decrease number of agent tours by moving nodes at each step from a schedule to another according to local search algorithm until all customers are served with the minimal number of agents and minimal routing time.

1.2.Local Search algorithm

Local Search LS algorithm is used to minimize the number of agents' routes and decrease the total travelled time. In our heuristic solution, moving nodes depends on number of operators: such as, $\tau\alpha_k$which is the number of nodes in an agent's schedule, $Q_{\alpha_k}$is the capacity of an agent α_k, and i which is a counter. The basic idea of local search is that a node, which has the largest routing time, will be transferred into the next lexicographic neighbor or next lexicographic un-neighbor if the capacity of that neighbor's agent is enough to have it in its schedule.

Because the goals of the project are to minimize number of agents and minimize routing time, in all scenarios, we will have only one solution to achieve both goals. Is that one agent will pass by all nodes in the minimum routing time in the road network. That only happens when the capacity of the agent is infinity and the number of nodes are infinity also. But that will never happen. Reasons:

1. Because there is upfront cost that will effect on the agents' tour time, the less use the upfront cost, the less time spent in routing as total time for all trips. Upfront cost is the reason that makes the total time increase high when we increase the number of agents used in routing. So in all cases, the less number of agents used, the less use of upfront cost.

2. The more agents used, the more upfront cost will be spent, the more total time for all agents. (the minimal time spent in routing per agent is more than or equal to upfront cost).
3. The more agents used, the faster each node will be helped, the less time spent in roads per agent,

   HOW TO USE BOTH POINTS TO SOLVE THIS ISSUE? Is to take the average number of agents so all we be average used.

The heuristic is considering the following:

1. The capacity of the vehicles is limited and all the same size let say Q.
2. The service time in each node is 0.
3. Agents are available 24 hours, means the time window of agents are open.
4. Time window of nodes are 24 hours available to be served.

A TS-based heuristic approach:

1. Let *N* be number of basic nodes and agents, *RT* array is the routing times array for all agents of size *N*. Each agent will have an array, *TT* integer variable for total routing time, *TempTT* integer variable for temporary total time, variance integer array to calculate the difference of time between routing time, *N* number of *AgentSchedule* array is agents' schedule. The arrays size will be equal to the capacity *Q* of the agents. *TempLargeTime* variable be the largest routing time from the array *RT,* counter *i,*
2. Set the edges weights and upfront cost for every edge connected to source b0 and set source and destination.
3. Do Constructive Heuristic algorithm
4. For each agent, iteration start:
   - 4.1. Set *i*=1. This is a counter for number of agents' arrays that has i nodes, the i increment when *TempLargeTime* hold the last large routing time of agent that has i nodes.
   - **4.2.** Calculate each agent shortest routing time by using A* search algorithm and save it in *RT*, save the largest value of *RT* in *TempLargeTime.*
   - 4.3. Calculate the summation of all tour times of array 2 and save it in *TT (*and *TempTT* for the first iteration*)*.
   - 4.4. Use Local Search Algorithm.
     - 4.4.1. Move node/s that has the value in *TempLargeTime*, name it *S,* to the agent's schedule that has the next lexicographic neighbor node, or the next lexicographic un-neighbor node and name it $\grave{S}$, and the capacity of that agent is enough to move the node/s into it's schedule.
     - 4.4.2. Update the $\grave{S}$ schedule.
     - 4.4.3. Delete *S* the old agent schedule.
     - 4.4.4. Decrease the number of agents *N.*
   - 4.5. Compare *TT* and *TempTT* and add the difference into *variance* array. then add *TT* into *TempTT* if this is the second iteration.