Full Name (as shown in attendance sheet):

[Optional] LumiNUS Account:

[Optional] NRIC / Passport / NUS Matriculation No.:
(Select and provide the applicable)

Institute of Systems Science
National University of Singapore

GRADUATE CERTIFICATE
INTELLIGENT REASONING SYSTEMS

Assessment

Subject: ***Reasoning Systems***

SECTION A

Question	Marks
1	**/1**
TOTAL	**/1**

Instructions for Paper

Duration: Fifteen minutes exam

This is an *OPEN BOOK* examination. This examination paper consists of *one* Section and *one* Question. You are to answer *ALL* questions. There are a total of *1 Mark* for this paper.

1. Read **ALL** instructions before answering any of the examination questions.
2. Write your Student ID number on the **front page** of this examination paper in the box provided.
3. This is an **Open Book** examination. If you wish, you may use reference materials to answer a question. Reference materials can be *books, manuals, handouts* or *notes*.
4. Answers are to be written **only** in this **examination paper** and any **attachments** provided and will be considered for credit. Answers written in any appendices will **NOT** be marked.
5. Use a pen for writing your answers. Pencil may only be used for drawing diagrams and writing program code.
6. Non-programmable calculators may be used if required. **However, computers of any form (laptops, tablets, smart watches etc.) are not permitted to be brought into the examination hall.**
7. State clearly any assumptions you make in answering any question where you feel the requirement is not sufficiently clear.
8. At the end of the examination:
   a) Hand-in the examination paper for **each** section **separately**, any appendices and attachments.
   b) You are **not** allowed to remove the examination paper, appendices or attachments from the examination hall.

REMEMBER:

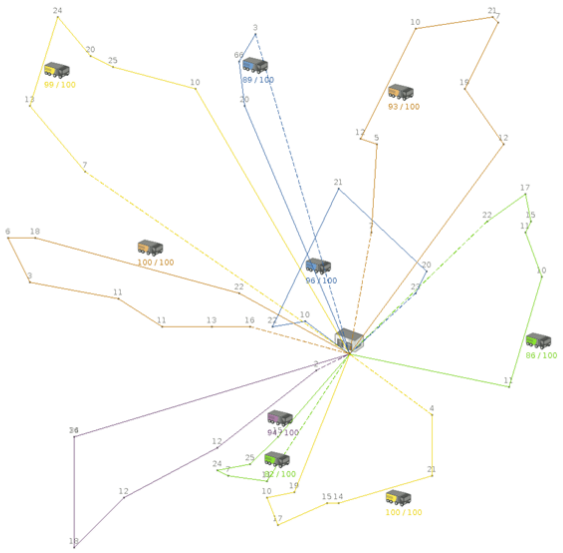
- ***This is an OPEN BOOK exam.***
- ***There are a total of 1 Mark for this paper.***
- ***You are required to answer ALL questions.***
- ***State clearly any assumptions you make in answering any question where you feel the requirement is not sufficiently clear.***

SECTION A

Question 1 *(Total: 1 Mark)*

Optimizing Vehicle Route Planning

- We are a logistic company owning a warehouse and 9 delivery trucks. This morning we received 54 customer orders, with different load demand, and different locations. Our truck's maximum load capacity is 100 TVs.
- We want to delivery all customer orders using fewer gasoline. Hence, we'd like to have shortest distance of combined truck delivery routes.



References:

VRP can be solved using ***State Space Search***, which is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or states of an instance are considered, with the intention of finding a goal state with a desired property. Depth-first search and Breadth-first search are forms of state space search.

Problems are often modelled as a state space, a set of ***states*** that a problem can be in. The set of states forms a graph where two states are connected if there is an ***operation*** that can be performed to transform the first state into the second.

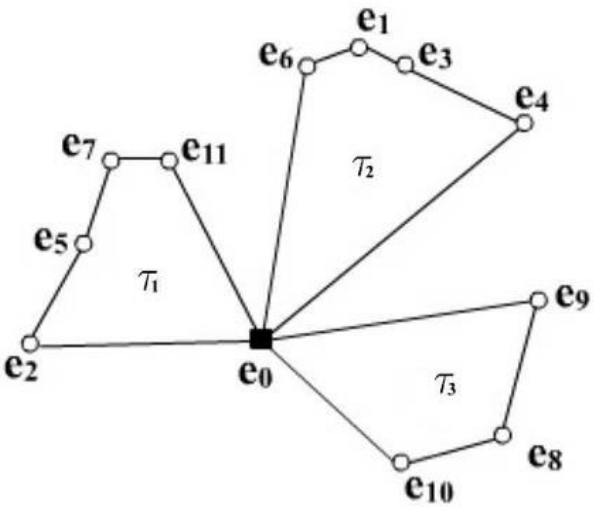
https://en.wikipedia.org/wiki/State_space_search

Answer the following questions:

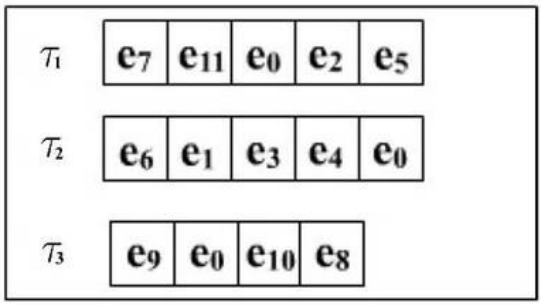
1.1 Propose a *vector or graph or tree* representation of **state (VRP solution representation)** to carry out the **state space search**, and suggest an initial state. (Hint: There are 9 trucks, thus 9 delivery routes; for 54 unique customers/locations)

(0.5 Mark)

[Answer]



a: Graph Representation



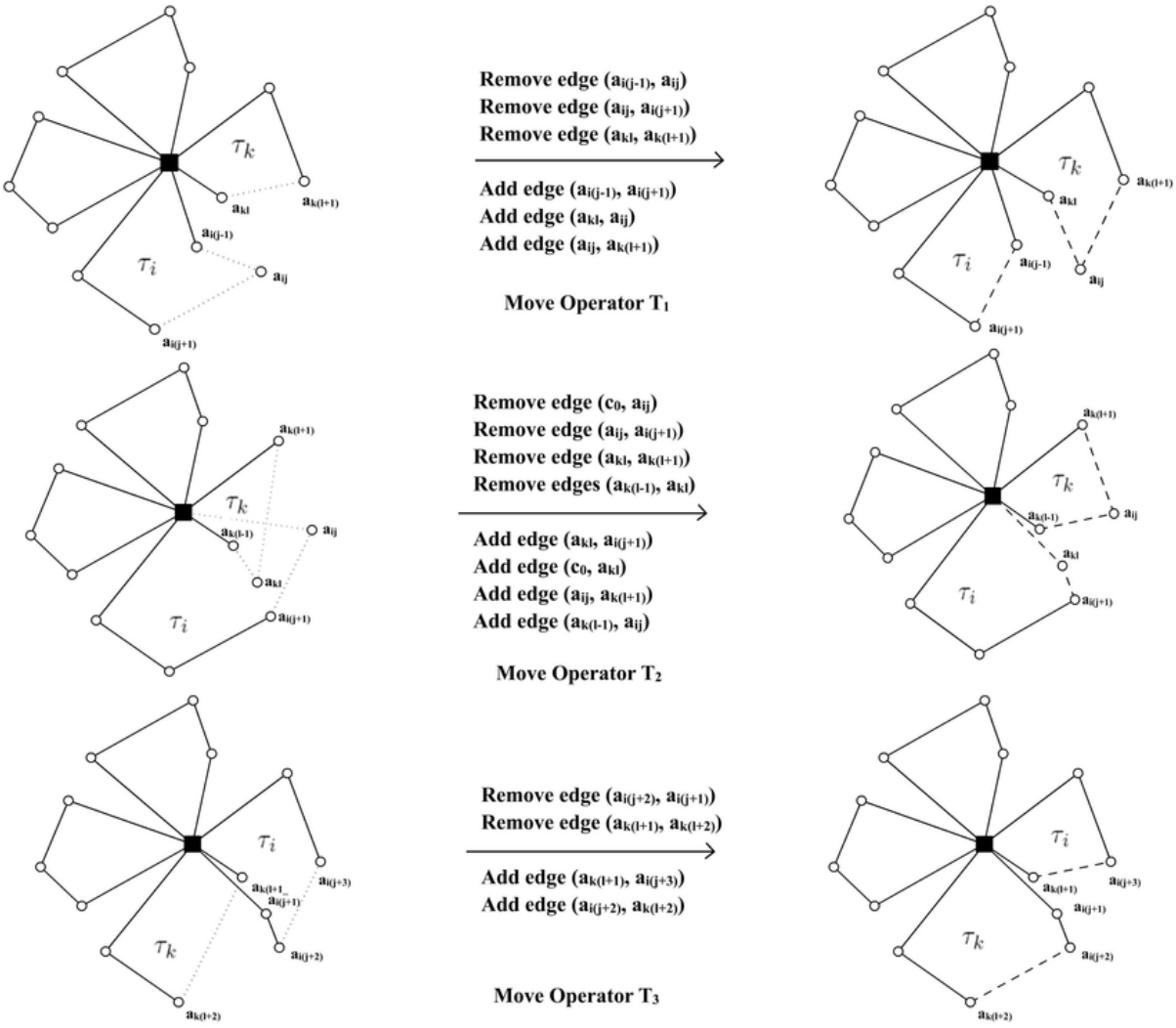
$S = \{ T_1, T_2, T_3 \}$

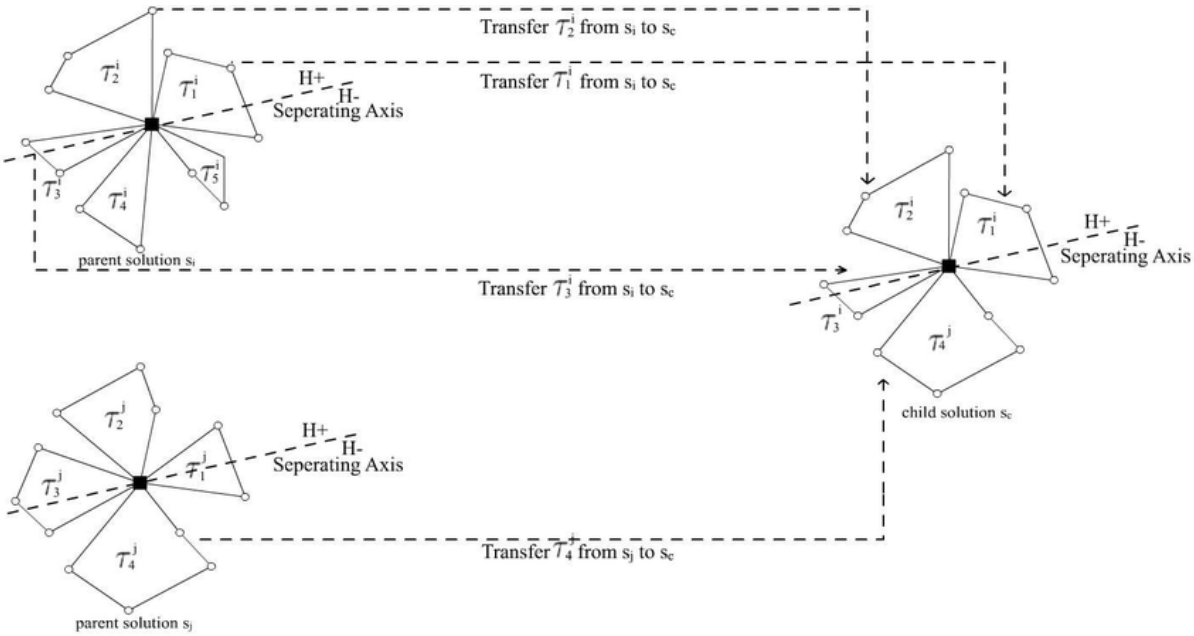
b: Multi-Route Solution Representation

1.2 Define *one or more* possible **search operator/action** to permutate above designed state/solution of VRP.

(0.5 Mark)

[Answer]





https://dial.uclouvain.be/memoire/ucl/fr/object/thesis%3A4615/datastream/PDF_01/view

4.1 Initial solution

As the local search is a perturbative method, it needs to start from an initial solution. This solution can be generated using different approaches:
Random generation: This approach consists in generating the initial solution randomly. The number of routes can be either chosen randomly or based on the total demand of the customers and the capacity constraints of the vehicles. The customers are assigned at random positions in random routes. While this solution most likely violate some of the constraints and can be far from the optimum, it is easily computable and presents the interest to offer a good diversification when coupled with a restart strategy.
Construction method: Another approach is to use a construction method such as the ones presented in Section 4.3.2. These methods can introduce random decisions to offer a better diversification. This approach has the advantage to provide a better initial solution than the random generation. However, it has a higher computational cost.
Mixed techniques: The idea here is, rather than using only local search, to use it to improve a solution found by another technique. Typically, a relaxed exact technique is used to find a first solution in an acceptable time and then this solution is improved through local search.
An important factor to take into account is whether the initial solution is feasible or violates some constraints. In the second case, this has to be taken into account during the search by selecting moves that lower the violation before minimising the cost to obtain a feasible solution.
Finally, another objective can be added in some variants of the VRP: depending on the specifications of the problem, the number of routes can either be constrained with a maximum or represented as an additional value to minimise. The difficulty here is to weight this new constraint or this new cost accordingly to the cost or constraint function. For this reason, some methods consider the number of vehicles as an additional objective which can then be used in a lexicographic optimisation. For example, the iterated minimum routes method described in Section 4.3.1 tries to minimise the number of routes before the cost.
21

4.2 Neighbourhood generation

As explained in Chapter 3, at each iteration, the neighbourhood is generated by the operators. Each operator generates a type of move to apply to the current state to obtain a neighbour.
The success of local search is highly dependant on the neighbourhood and thus on the operators used. Ideally, more than one operator is used to generate the neighbourhood and offer

different kind of moves thus improving the connectivity and the diversification of the neighbourhood. However, having more operators leads to a higher use of resources to both generate the neighbourhood and explore it. It is therefore critical to choose which operators are used inside the search with great care.

In this section we present some of the most common operators [3, 4, 27] used in local search approaches for the VRP. Note that despite being presented in terms of vehicle routing problem, some of these operators are not specific to the VRP and can be used for other problems such as the Travelling Salesman Problem.

4.2.1 Relocate operator

This operator consists in relocating one customer from one route to another. Formally, the customer i from route r_0 is removed from r_0 and inserted in route r_1 at a position p where $r_0 6= r_1$ and $0 \_ p \_ jr_1j$.

For example, in Figure 4.1, the customer c_4 is relocated from route r_1 to route r_0.

Figure 4.1: Example of relocate operator

4.2.2 Swap operator

This operator consists in swapping two customers from different routes. It can be seen as a double relocation in which the customers are inserted at their counterpart's position in the route. More formally, the customer c_i at the position p_i in route r_0 is relocated at position p_j in route r_1 while the customer c_j at the position p_j in route r_1 is relocated at position p_i in route r_0.

In Figure 4.2, the customer c_6 from the route r_1 and the customer c_5 from the route r_0 are swapped according to their respective positions.

22

Figure 4.2: Example of swap operator

4.2.3 K-Exchange operator

This operator, which is also called k-opt, consists in dropping k edges in the same route and then reconnecting the resulting segments by other edges. As a result, some segments might be reversed. Formally, it consists of removing k edges in which the endpoints are at positions $< p_0; p_1:::p_k >$ in the route. The goal is to reconnect the segments in a different order.

This operator might be defined for any number of edges starting from two. However, computing all the possible exchanges would be highly demanding in terms of resources. The operator is thus usually restricted to the 2-exchange or 3-exchange.

Figure 4.3: Examples of the 2-exchange (left) and the 3-exchange (right) operators

Examples of the 2-exchange and the 3-exchange operators are depicted in Figure 4.3. In the first case, two edges are removed with c_1 and c_7 as endpoints. The resulting route is a new route containing the segment starting at c_1 and ending at c_3 in its reversed form.

23

In the second case, these edges have c_1, c_4 and c_6 as endpoints. We obtain thus two segments starting at c_1 and c_4 and ending at c_2 and c_7 respectively. The second one is followed first in the same order. Then, the first one is followed in the reverse order.

4.2.4 Cross operator

This operator cuts two different routes in two parts and recompose them with crossing edges. In formal terms, it exchanges the segments from route r_1 and r_2 starting at customer c_i and c_j respectively and ending at the depot where $r_1 6= r_2$.

Figure 4.4: Example of cross operator

END OF ASSESSMENT PAPER