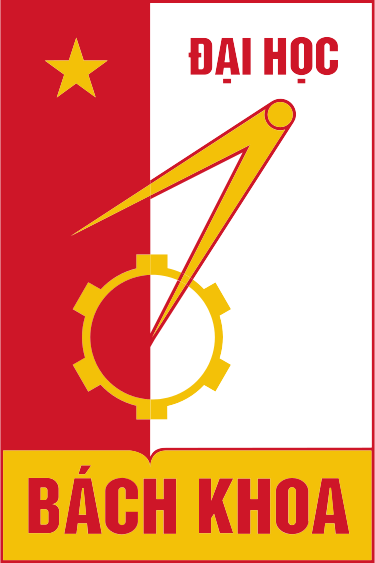
Hanoi University of Science and Technology



PATIENTS DELIVERIES PROBLEM

|  |  |
| --- | --- |
| Nguyen Trung Truc | 20214936 |
| Pham Duc Thanh | 20210795 |
| Nguyen Quang Hung | 20214961 |
|  |  |

## ACKNOWLEDGEMENT

We would like to express out appreciation to Dr Bui Quoc Trung who gave us opportunity to work on this project. This work could not be finished without your helpful advices and thoroughly guidance.

## ABSTRACT

This report works on the problem of solving a variant of Capacitated Vehicle Routing Problem (CVRP) with more constraints which involves sets of hospitals specified to each of the patients. Then, we illustrate our approaches to the problem, including some approaches we have coded and some other approaches we have come up with ideas without coding.

**DESCRIPTION OF THE PROBLEM**

**1.1 Problem description**

N patients need to be transported from their houses to M hospitals. Due to specific issues of the patients, each patient i can only go to a set of hospitals H(i).

There are K cars. The capacity of each car is under or equal to c(k). Supposed that K cars are all at the depot 0.

Patient i lives in point i, for i = 1, 2, … N.

Hospital j is at point N+j, for j = 1, 2, … M.

Given that t(i, j) is the time going from point i to point j.

Find a way to carries all the patients to proper hospitals in the shortest time.

Following the evaluation of our team, this is a variant of CVRP problem with more constraints. Concretely, patients i can only be transported to a set of hospital H(i).

**1.2 Math modelling**

In this report, we will model the problem for **Constraint programming**:

* The patients are located at point 1, 2, .. N
* The Hospitals are located at point N+1, N+2, …N+M
* Depot locations of the cars when leaving: N+M+1, N+M+2, … N+M+K (these points refers to one location in real life)
* Depot locations of the cars when they turn back: N+M+K+1, N+M+K+2, …N+M+K
* Car k have indexes k.
* H is the set of specified hospitals of the patients.

H(i) is the set of specified hospitals of patient i.

We have: B = {1, 2, 3, … N+M+2K} (set of all the points)

vehicles = {1, 2,… K}

F1 = {(i, k+M+N) for i in B for k in vehicles}

F2 = {(k+M+K+N, i) for i in B for k in vehicles}

F3 = {(i, i) for i in B}

A = B\*B \ F1 \ F2 \ F3

A\_plus(i) = {j | (i,j) in A}; A\_minus(i) = {j| (j, i) in A}

Decision variables:

X(k, i, j) = 1 if car k travels from point i to point j; k = 1, 2,.. K, (i, j) in A

Y(k, i): number of items on car k after leaving point i; k = 1, 2,… K, i in B

Z(i): index of truck visiting point i; i in B

Constraints: = = 1, i = 1, 2, ..N+M (only one car enters a node and leaves that node)

= , i = 1, 2,… N+M, k = 1, 2, … K

= = 1, k = 1, 2,.. K (all cars must leave depot and arrive depot)

M(1- x(k,i,j)) + z(i) >= z(j); (i, j) in A, k = 1, 2,…K (if x(k, i, j) = 1 then the index of the cars travel from point i to point j must be equal)

M(1- x(k,i,j)) + z(j) >= z(i); (i, j) in A, k = 1, 2,…K

M(1- x(k,i,j)) + y(k, j) >= y(k, i) + r(j); (i, j) in A, k = 1, 2,…K

M(1- x(k,i,j)) + y(k, i) + r(j) >= y(k, j); (i, j) in A, k = 1, 2,…K

Y(k, k+K+M+N) <= c(k), k = 1, 2, … K (the load of each car must not exceed its capacity)

Y(k, k+M+N) = 0, k = 1, 2, … K (the loads of all cars when they leave depot are 0)

Z(k+M+N) = Z(k+K+M+N) = k, k = 1, 2… K (the indexes of the car leaving point k+M+N and arriving point k+K+M+N must be equal)

= 0, i = 1, 2, … N (a patient i must be transported to a hospital in specific set H(i))

Objective function: F = 🡪 min

* The implementation of the constraints above will be shown in the following chapter)

**SOLVING METHODS**

**1. Backtracking**

Part 1: In the first part, we will find the optimal path of Travelling Salesman Problem (TSP)

For example, we have the input data:

* matrix\_distance=[[0, 3, 2, 4, 3],​

                                        [4, 0, 5, 3, 2],​

                                        [2, 1, 0, 6, 4],​

                                        [1, 1, 3, 0, 5],​

                                        [3, 4, 5, 3, 0]]  ​

The value of matrix\_distance[i][j] is the time to travel from point i to point j. Point 0 is depot.

After that, we have the output: optimal\_path: [9, [0, 2, 1, 4, 3, 0]]

Part 2: generating data

In part 2, we will consider the carriage limitation and the specific set of h(i) hospitals constraints.

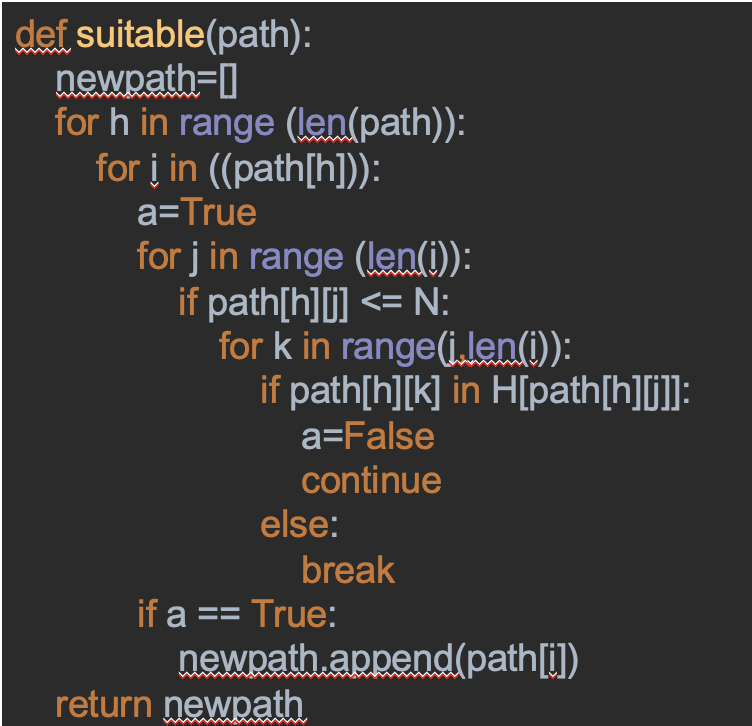
* At first, we generate all forms of all kinds of vehicle can carry by function “genrerate\_carriages”

Take one example: if there are 3 carriages at point (1, 2, 3) and there are 2 cars to transport them from depot 0 to depot 0, the function “generate\_carriages” will give us the list:  [ [[0, 1, 2, 0], [0, 3, 0]], [[0, 1, 3, 0], [0, 2, 0]], ​

                  [[0, 1, 0], [0, 2, 3, 0]], [[0, 2, 3, 0], [0, 1, 0]], ​

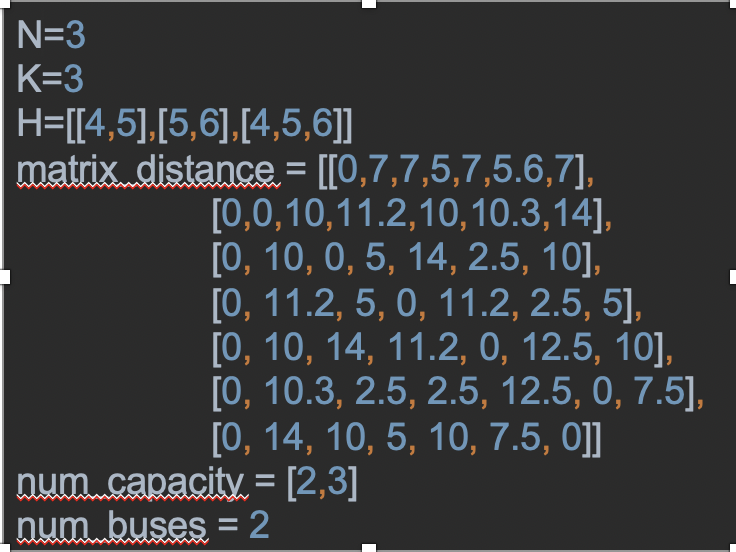
                  [[0, 2, 0], [0, 1, 3, 0]], [[0, 3, 0], [0, 1, 2, 0]] ]​

* Secondly, we select suitable path which satisfies 2 conditions:

1. The total carriage of each car in every path is smaller or equal c(k)
2. For every patient i in each path, there exists a hospital in their H(i).
3. 

* Thirdly, select the best path:

Example:



We have output: Vehicle 1 ‘s path: [0, 1, 4, 0], the cost is 17

Vehicle 1 ‘s path: [0, 2, 5, 3, 6, 0], the cost is 17

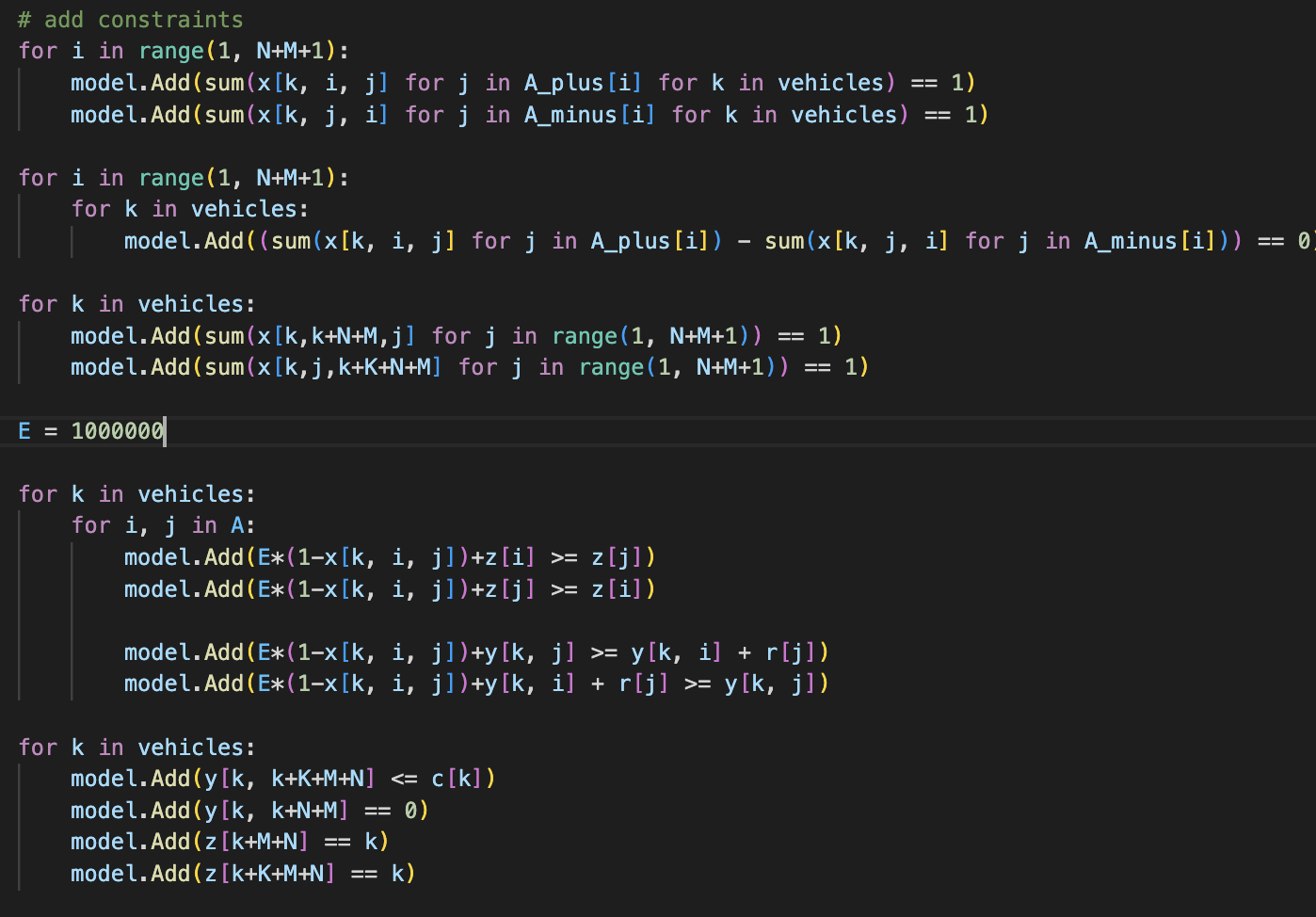
Total cost: 34

**2. Constraint programming**

In this method, we will use a very strong tool of Google. That’s Ortools.

We will implement our code based on the constraints we showed above.

At first we add constraints:



For the constraint “ each patient only go to some specific hositals”, we have 2 approaches:

* The first approach: we will set the indexes of the cars are prime numbers rather than the indexes from 1 to K.

Let j is the location of hospital in set H(i) of patient at point i. Then, we have the constraint:

**) % z(i) = 0**  (z(i) is the index of the car visiting pointi)

* The second approach: we also have j is the location of hospital in set H(i) of patient at point i. We use the constraint:

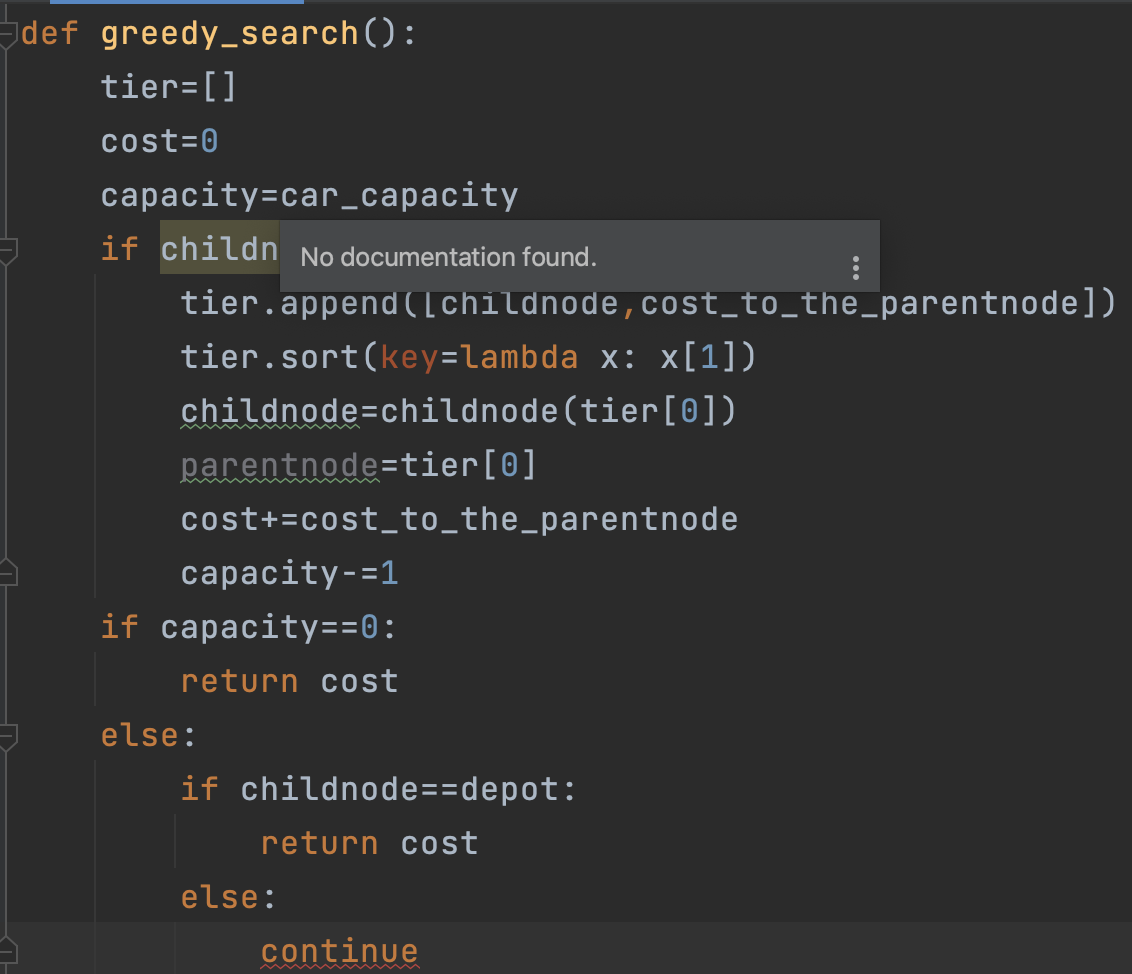
**) - z(i))= 0**  (z(i) is the index of the car visiting pointi)

But eventually, we realized that two approaches above are very difficult to add the constraints to the code. We have to admit that we have not figured out how to add the constraints to the code up to now, but we are working on it. Temporarily, the set H(i) only have 1 elements for all patient i.

**2. Greedy method**

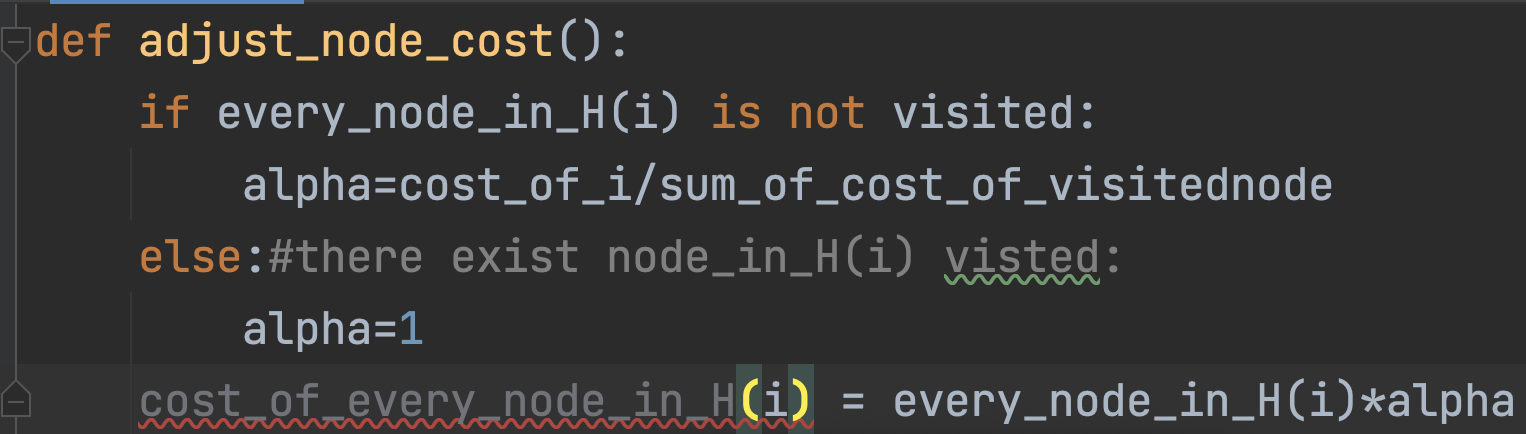
We have 2 conditions for greedy algorithm:

* Choose car with biggest capacity
* Then choose the node with lowest cost



Greedy algorithm has the same problem as the CP method: How to add the specific set of h(i) hospitals constraints?​

SOLUTION: We make the hospital in the set H(i) have higher priority by making their cost  JUST like “tier list” below (Because the greedy algorithm always start with lowest cost node), and make sure it doesn’t affect much to the optimal path. ​

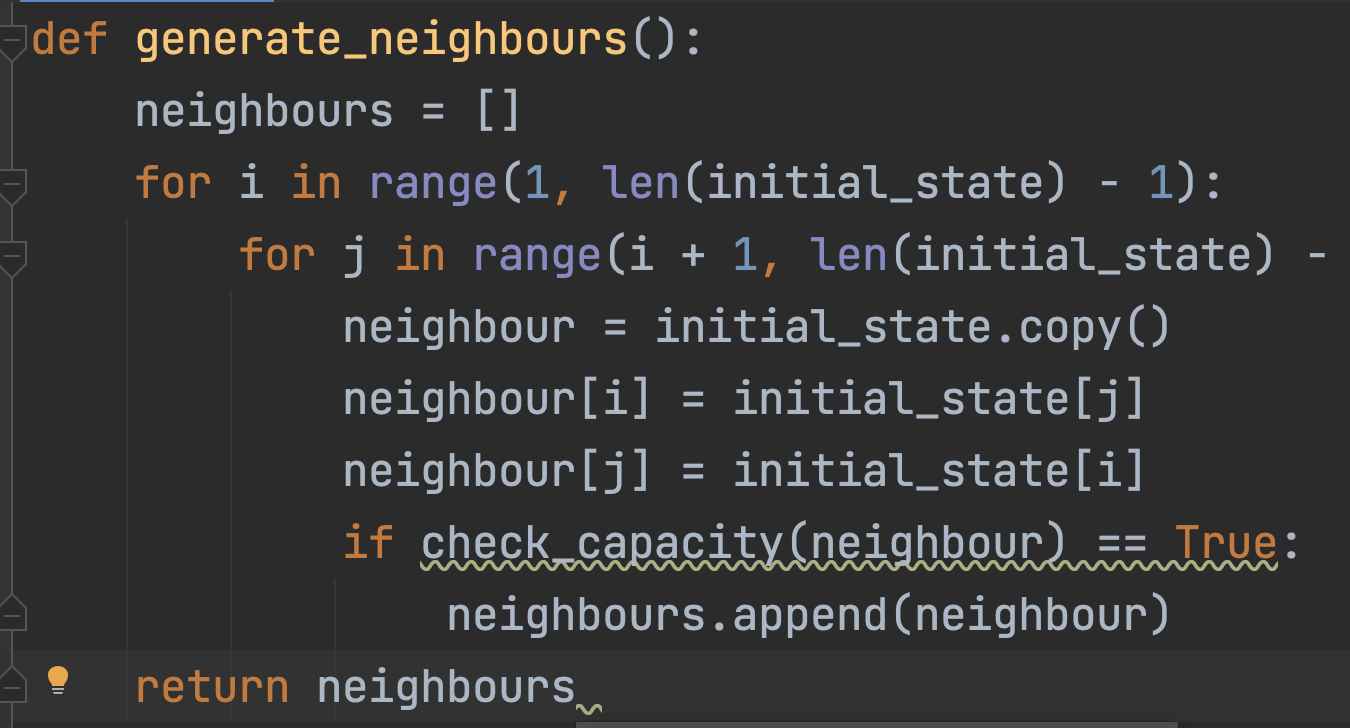
​

The Alpha make sure that the longer of ”not any node of H(i) chosen”, the smaller of their cost​

**4.Hill climbing**

It only evaluates the neighbor node state at a time and selects the first one whichoptimizes current cost and set it as a current state.​

Way to generate neighbours:​

****

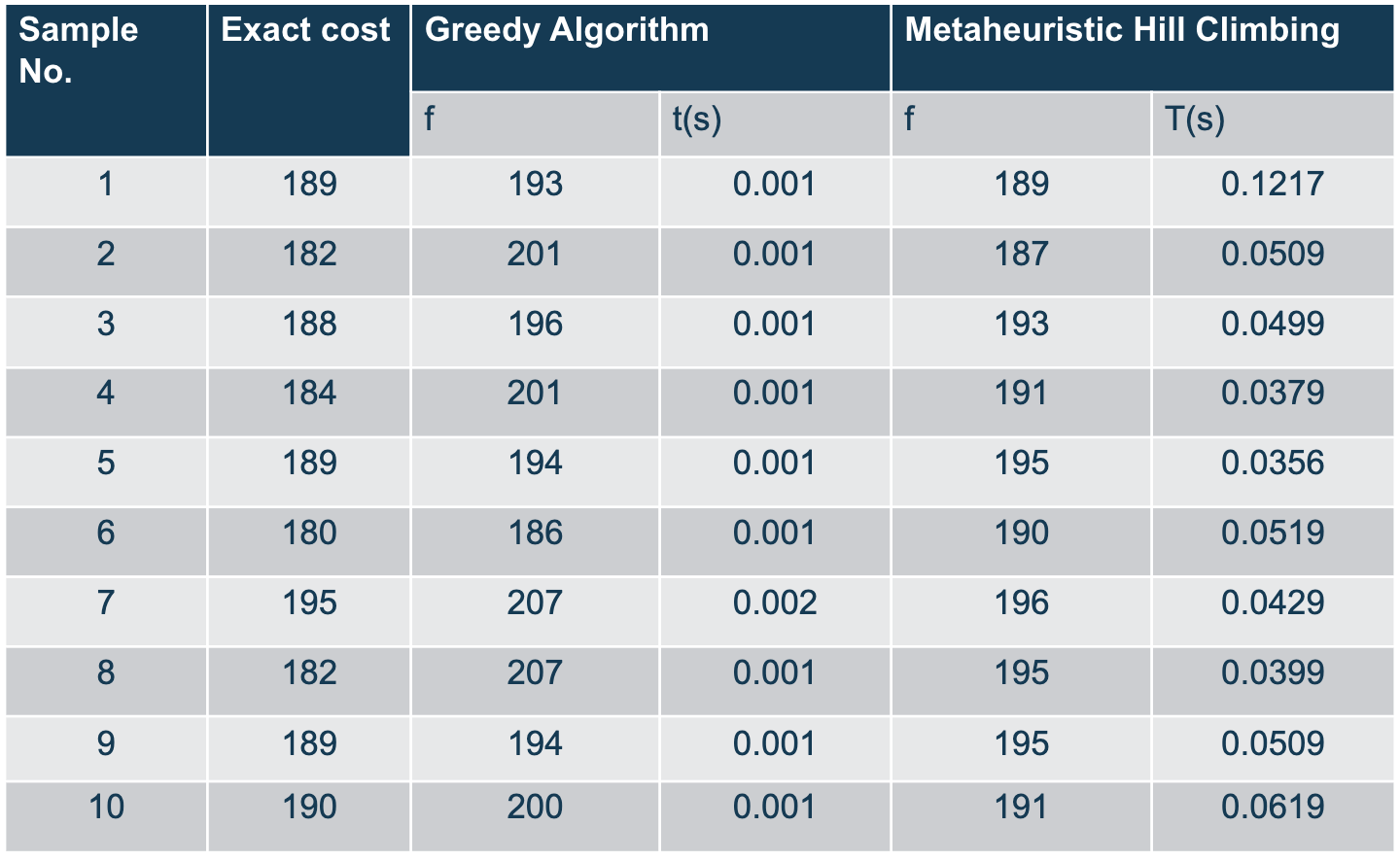
Exist problem: ​

 - This is the simple hill\_climbing which cannot fix the “local optimal” problem, we are still working for the backtracking hill-climbing. ​

 - Backtracking technique can be a solution of the local maximum in state space landscape. It creates a list of promising paths so that the algorithm can backtrack the search space and explore other paths as well. ​

**RESULT**

We have used the data as following: N= 4 (patients), M = 5 hospitals, K: random from 2 to 5 vehicles, 2 depot points in 11x11 matrix and the set of specific hospitals H(i) is random.



**Conclusion:**

Bactracking :​

* Always give exact result​
* Easier to code than other methods.​
* But takes at least 3 seconds to excute (That’s 200 times greater than the “hill-climbing” and 4000 times for “Greedy algorithm”)​

Metaheuristic Hill Climbing​

* Short execution time, often give the exact optimal result​
* Sometime the result is “local optimal result”, and distant from optimal result

Greedy Algorithm​

* Shortest execution time​
* But the result always close to optimal result, never equal​

**POSSIBLE EXTENTIONS**

In general, Patients Deliveries problem is a variant of Capacitated Vehicle Routing problem (CVRP) that requires finding the shortest time for the cars to transport the patients to the proper hospitals. In this mini-project, we have explored four different algorithms to solve this problem: backtracking, constraint programming (uncompleted), hill climbing and greedy algorithm.

Backtracking always gives exact results and it is the easiest algorithm to code compared to other methods. However, backtracking is the algorithm that runs in the longest time.

Meta heuristic hill climbing runs in short time, but sometimes the result is “local optimal result”.

Greedy algorithm runs in shortest time but never gives exact result.

The most challenging method is constraint programming that it is quite hard to add the constraint “each patient specific to a set of hospital H(i)” and we have come up with some ideas but none of them possible up to now.

So in the future, our team want to find a way to add that constraint to our code or try another method like integer programming,…

​