**Report for AI Project**

**1. Presentation of the subject**

If you are a business or organization that has delivery drivers or delivers goods or services to many clients, then this applies to you. Couriers, food delivery services, [field sales](https://blog.route4me.com/field-sales-apps/), florists, and many more have the same **route planning** problem: [make routes more profitable](https://blog.route4me.com/routing-optimization-software-profitable-business/), create a map and optimize routes to save time and also resources.

Our aims of this project are to try to approach the problem with our own algorithms and applying on the 63 cities of Vietnam.

**2. Description of the problem**

* Similar the route planning, the intelligent vehicle can only travel  
  from the current city to the city near it within a certain time, and also additional time if it goes at peak hour. The objective is to try to minimize the time cost to travel between the two cities.
* Formulation:

+ Initial state: In starting city

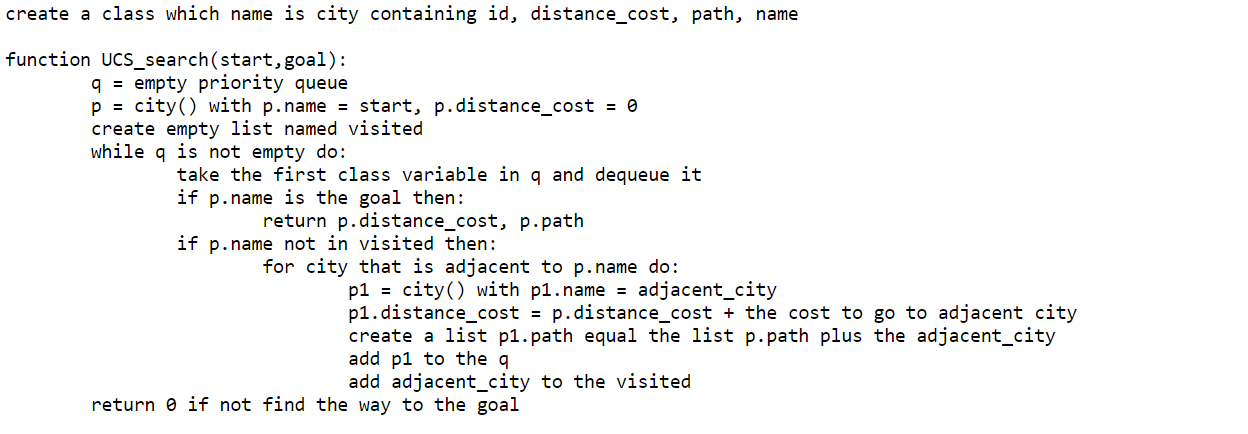
+ Action/Transition model: Action {In: starting city} = {Go: adjacent cities}

+ Goal test: {In: Goal city}

+ Path cost: Sum of time traveling

**3. Algorithms for the problem**

The UCS function here is a modified version of the normal UCS. Instead of using a normal queue, we use a priority queue. This queue will automatically compare the distance cost between variables when a new city class variable is added, and the smallest value will be placed first, no matter the current city it is, using the heap sort. This approach is for more efficiency.



We also apply UCS to solve the minimum time cost ( with both traffic or not). When we Pre code we find out the data for traffic jams is not very realistic as we can call directly from API of Google Maps. So we decide not to apply it to our presenting, only Normal Time we got from Goong-API which can’t give us traffic mode.

Text

Description automatically generated

The A\* star function is similar to the UCS but we will compare air distance cost plus the normal distance cost, as the heuristic variable.

Graphical user interface, text

Description automatically generatedWe multiply the air distance cost by 2 here because after some tries we find that this heuristic will dominated run faster but it’s some time it won’t give the best answer.

Reason: Viet Nam is

**NOTE**: In the topic, we mention the complete search but we don’t use it, after discussing and reading some materials about the problem, we just want to consider the efficient difference between the A\* and the UCS algorithms. Also, the complete search which we can present by using backtracking with maximum number of cities to visit. But we all agree that it’s much worse than UCS.

**4. Performance comparison between algorithms**

We would want to observe the efficiency of two algorithms in terms of the time cost to run the function. And there are two kinds of variables we think of:

* The distance needed to travel: This is not consistent because the distance path connecting city to city depends on the complexity of the path, something like it is curved or straight, so it is hard to determine if the algorithm is effective.
* The number of cities between the starting and the destination: This variable we think will present the complicated connection between cities and then show the difference more obviously.

We use a range instead of a number so that the variable can reach a larger number and also reduce the complexity of the chart.

The range variables we will use are: [6-8], [9-11], …, and will present 5 consecutive variables of this kind. The range from [1-5] is not necessary to mention because it is too small to affect the result. The time cost will be the average of all data satisfied in their range.

Need chart and analyze

**5. Problems occurring during working on the assignment and how we handled them**

* Input data takes too much time to make. Because of the approach in which we attempt to solve the problem, we need to motivate the team and try hard to input data by hand including creating a distance matrix between cities, and an additional time cost matrix with google support so it can be as accurate as possible.
* Additional time is hard to implement. Our based knowledge is not too much, so we struggle to learn or find a function that can predict traffic time. At last, we came to a simple code, trying to add time costs when traveling at a certain time. Not the best but the only solution we find.
* Call API of google is not available in Vietnam. At first, we don’t know what to do because we only know and are familiar with google maps. However, then we know and learn to use the apple map API but it is a bit annoying because you need the city coordinate to call it precisely.

**6. Conclusion and possible extensions**

**Conclusions:**

* Although the time complexity of the two algorithms is O(n2), the n is constant so we can use matrix input here
* The heuristic is not as far as good. It can give the optimal time and use less memory if the path connecting cities is straight but the more complex it is, the less effective the A\* function is, and in some cases, the better UCS performance is in time cost.
* The A\* and UCS can give optimal results in all starting and goal cities because the Vietnam city map is not too complex and also central Vietnam is at most two adjacent cities (not including the previous city) for each city in here which are much more simple for the algorithms.

**Possible extension:**

* Try to predict the traffic time by google support and machine learning instead of giving it a certain time. Because time and resources are limited, we can’t give the best way to implement this but only the simple one. In the future, a better prediction algorithm will give a more accurate time cost.
* Using a tree as input data rather than a matrix for less complicated and reduces the unnecessary loop. With the linked list data structure, the adjacent city with the leftmost city and right city (not necessarily to be in order), also includes the previous city that we travel from. The cycle will be prevented by adding the visited list. The space cost for data input will be less than the matrix and more efficient.
* Add the borderline between cities and some specific places in each city as the starting or destination point to achieve higher accuracy instead of just traveling from city to city.

**7. List of bibliographic references**

[**https://docs.goong.io/rest/**](https://docs.goong.io/rest/) **using: API key, distance matrix**

**https://support.route4me.com/faq/route-planning-glossary/what-is-route-planning/ using: introduction**