ASSIGNMENT 1: DISASTER RELIEF HELICOPTER ROUTING

Goal: The goal of this assignment is to take a complex new problem and formulate and solve it as search. Formulation as search is an integral skill of AI that will come in handy whenever you are faced with a new problem. Heuristic search will allow you to find optimal solutions. Local search may not find the optimal solution, but is usually able to find good solutions for really large problems.

Scenario

There are floods in a region. There is an urgent need to carry out relief operations from neighboring cities. There are a fixed set of relief packages to deliver to affected villages using a fleet of helicopters. Each village has a number of people stranded. The goal is to allocate relief deliveries to helicopters such that maximum aid is delivered, while respecting capacity and range limits of each helicopter, and while simultaneously minimizing a "logistical strain" cost caused by routing aid inefficiently.

For this we assume, we are provided a list of villages V that need relief. Each village v is associated with its 2D coordinates (x_v, y_v). Similarly, we have a few cities C, along with 2D coordinates for each city. The distance between any two locations is given by the straight line distance between their coordinates (in kms). For each village, we are also provided n_v: the number of people expected to be stranded in that village.

There are three types of packets T = {d, p, o}, which stand for dry food, wet (perishable) food, and other supplies. The goal is to send about 9 meals per stranded person, and about 1 unit of other supplies per stranded person. Each packet of type t weighs a fixed amount w(t). Each packet of type t has a fixed value v(t). The meals may be dry or perishable food, but wet food is preferable, i.e., v(p) > v(d).

There are H helicopters. Each helicopter h has a home city home(h). It also has a weight capacity wcap(h), and a distance capacity dcap(h) per trip. Each trip starts at the home city and ends at the home city, and package weight on the trip cannot exceed wcap(h) and total distance traveled cannot exceed dcap(h). Moreover, total distance traveled by any helicopter across trips cannot exceed a given DMax.

Each trip costs F + alpha*distance. Here F is the fixed cost per trip (for takeoff and landing). And alpha represents some notion of fuel efficiency.

Total value of solution = Total value achieved – total trip cost.

The goal of the assignment to produce a plan for each helicopter, such that the total value is maximized. How many trips they do. How many total packages of each type they start with per trip. Which villages do they visit and in what order. How many packages of each type do they drop per village.

Input:

The first line has total processing time available in minutes.

The second line has DMax: the max distance in kilometres

The third line has six numbers representing: w(d) v(d) w(p) v(p) w(o) and v(o)

The fourth line has C: the number of cities followed by 2C coordinates. Imagine each city is {1, …, C}. The 2C numbers represent x, y coordinates of each city, successively.

The fifth line has V: the number of villages, followed by 3V numbers. Imagine each village is {1, …, V}. The 3V numbers represent x, y, n – the coordinates of each village and number of people stranded.

The sixth line has H: the number of helicopters followed by 5H numbers representing home city id, wcap and dcap, F and alpha of the helicopter, successively.

Here is a sample input

1

100

0.01 1 0.1 2 0.005 0.1

2 0 0 10 10

2 0 5 1000 0 10 1000

2 1 100 25 10 1 2 100 50 10 1

This input suggests that there are two cities at (0, 0) and (10, 10) and two villages at (0, 5) and (0,10). 1000 villagers are stranded in each village. Each packet weighs 10 gms, 100 gms and 5 gms for d, p, and o. Similarly, value of dropping a dry packet is 1, for a perishable packet is 2 and other supplies packet is 0.1. There are two helicopters, one at each city. Their weight capacity is 100 kg each, but distance capacity is 25 km and 50 kms each. Both helicopters have F and alpha 10 and 1 respectively. The maximum distance covered by any helicopter cannot exceed 100 kms.

Output:

Your algorithm should return the trips undertaken by each helicopter. Make one row for each helicopter in the order 1 to H. You should first write helicopter number. Follow this with number of trips. Then for each trip, write the number of packages of type d, p and o to pick up. Then mention number of villages the helicopter will travel, and village id and the number of packages of each type it will drop per village. Make one new row per trip. In the end add a -1 to move to the next helicopter. For example, see the following output:

```
1 2

9000 0 2000 2 1 9000 0 1000 2 0 0 1000

8889 111 0 1 2 8889 111 0

-1

2 0

-1
```

The first row is read as: helicopter#1 makes two trips. In the first trip it picks up 9000 packets of dry food and 2000 packets of other supplies. It visits two villages in this trip: village 1 dropping all dry packets and 1000 of other supplies and then village 2 dropping 1000 other supplies. In trip 2 it picks up 8889 packets of dry food and 111 packets of perishable food and drops all of them in village 2. A -1 indicates that we are done with helicopter 1. Helicopter 2 does not undertake any trips.

The value of objective function for this solution is evaluated as:

Distances covered in both trips: 20 kms each.

Cost of both trips: 10 + 1*20 = 30

Value gained by village 1: 9000*1 + 1000*0.1 = 9100

Value gained by village 2: 8889*1 + 111*2 + 1000*0.1 = 9211

Total objective function = 9100 + 9211 – 60 = 18251

Note that total distance covered is 40 km by helicopter 1, which is under 100 kms. Note that each trip of helicopter was 20 km which is within dcap constraint of 25 kms. Note that trip1 and trip2 had weights of 100 Kg and 99.99, which are under 100Kg. Hence, it is a valid solution.

Note also that if at any place more than 9x#stranders amount of food is dropped, then no additional value is gained from it.

Basic Algorithms you can try:

1. Heuristic Search: Design a state space and transition function for this problem. Define a heuristic function for evaluating a state. Implement A* (or variants) and measure quality of solutions found (and scalability). If heuristic is admissible – quality is optimal but algorithm may be slower. Test a couple of heuristics if you can design them.
2. Branch and Bound: Design a state space and transition function for this problem. Optionally define a heuristic function for evaluating a partial walk. Also, optionally, define a ranking to pick the next successor. Implement Depth First Branch and Bound (or variants) and measure scalability.
3. Local search: Implement a neighbor function for this problem. Implement local search (or variants). Measure of quality of best solution found as a function of time or other model parameters.

Recommended: start with local search as your base algorithm.

Sample Code:

You are being provided sample code that can take in the input and generate the output in C++. You may choose to not use this code. For this assignment, use C++ as the programming language. Use **g++** 9.4.0 as your compiler.

The sample code (written in cpp) can be downloaded here. Please refer to FileStructure.md to understand the file structure of the starter files. This code reads in an input file and computes the cost of the allocation. You need to write the additional logic for allocation of resources. Here, we have allocated the resources sequentially. The allocation is then written in the output file. You may or may not use the sample code. Note that you need to implement logic to compute allocation in given time. You can compile and run the sample code as follows:

```
make
```

```
./main <input_filename> <output_filename>
```

We have even supplied format_checker.cpp which checks the format of the output file generated and gives the cost (if the format is valid). To run the format checker, use the following command:

```
make checker
```

```
./format_checker <input_filename> <output_filename>
```

We have provided three input files representing easy problems. We recommend you experiment with other problems as well.

What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip. Make sure that when we run "unzip yourfile.zip" the following files are produced in the working directory:
   .cpp/.h files
   Makefile
   writeup.txt

   You will be penalized for any submissions that do not conform to this requirement.

   Makefile should compile your code and create the executable "main".
   Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 20.04. You are already provided information on the compilers on those VMs. They have RAM of 8 GB

   Your main should take 2 inputs, someinputfile.txt and someoutputfile.txt. It should read the first input as input and output the answer in the output file.
   ./main input.txt output.txt   (please note any name could be given to the two files).

2. The writeup.txt should have two lines as follows
   First line should start with this honor code. "Even though I/we have taken help from the following students and LLMs in terms of discussing ideas and coding practices, all my/our code is written by me/us."
   Second line should mention names of all students and LLMs you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.
   After these first two lines you are welcome to write something about your code, though this is not necessary.

Code verification before submission: Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Failure to follow any instruction will incur significant penalty.

We shall be generating a log report for every submission within 12 hours of submission. This log will let you know if your submission followed the assignment instructions (format checker, scripts for compilation & execution, file naming conventions etc.). Hence, you will get an opportunity to resubmit the assignment within half a day of making an inappropriate submission. However, please note that the late penalty as specified on the course web page will still apply for resubmissions beyond the due date. Exact details of log report generation will be notified on Piazza soon.

Also, note that the log report is an additional utility in an experimental stage. In case the log report is not generated, or the sample cases fail to check for some other specification of the assignment, appropriate penalty for not adhering to the input/output specifications of the assignment will still apply at the time of evaluation on real test cases.

Evaluation Criteria: Performance, i.e., quality of the allocations output by your method on a variety of test inputs. If you return an infeasible solution, it will be treated as worst solution, without looking at objective function. Extra credit may be given to standout performers.

What is allowed? What is not?

1. You may work in teams of two or by yourself. We do not expect a different quality of assignment for 2 people teams. At the same time, please spare us the details in case your team cannot function smoothly. Recall, that you can't repeat a team in COL333, and repeat a time only once in other courses. If you are short of partners, our recommendation is that because this assignment is in C++, you may do it with a partner. Some future assignments will be in other programming languages.
2. You cannot use built-in libraries/implementations for search or scheduling or optimization algorithms. To request use of a library, use Piazza, and TAs will respond in 48 hours.
3. You must not discuss this assignment with anyone outside the class. You cannot ask LLMs to write code for you. However, you are allowed to give the LLM your code in case you are stuck in debugging, so that you can learn about your mistakes fast, and do not waste time in debugging. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
4. Please do not search the Web for solutions to the problem.
5. Please do not use ChatGPT or other large language models for creating direct solutions (code) to the problem. Our TAs will ask language models for solutions to this problem and add its generated code in plagiarism software. If plagiarism detection software can match with TA code, you will be caught.
6. Your code will be automatically evaluated. You get a *minimum* of 20% penalty if your output is not automatically parsable.
7. We will run plagiarism detection software. Any team found guilty of either (1) sharing code with another team, (2) copying code from another team, (3) using code found on the Web, will be awarded a suitable strict penalty as per IIT and course rules.