Advanced Programming and Data Structure

Project 2 of the first semester – Combinatorial Optimization

CatTheHobie – The Sequel

Engineering Department
La Salle - Universitat Ramon Llull
12 December, 2022

Index

1 Introduction

Very pleased with our previous work, CatTheHobie have contacted us again. This time, they want us to help them with a new project.

CatTheHobie is organizing a regatta, which will take place in June of next year. Having closed the inscription process, they sent us a list of sailors that will partake in it. CatTheHobie wants their boats to shine during the competition and have therefore requested that we find the participants that can make the most out of them.

Furthermore, they want to figure out the best location for the races, considering that they need to be in proximity to a set of centers that can provide them with boats of all types.

2 Data format

We have been provided with 2 types of datasets: some containing the sailors' information and the others containing the information of the boats.

Each dataset is a text file, and, for each type, we are given 5 datasets of different sizes. Consider that each boat dataset is linked to the participant dataset of the same size category.

2.1 Boats

Concerning the boats, each dataset has the following information:

```
15
713;Cassia bark Stella Baines;Patí Català;86.5;4.14;3;82010;new;28;Dimension C-137
4780369;Jump Pork;Laser;221.55;5.19;2;78327;new;16;Dimension C-137
6;Entropy Projection Juniper Berries;Ludic;202.81;4.47;8;84737;new;10;Earth
```

- **n_boats:** number of boats in the file, ending in '\n'
- **boat 1**, ending in '\n', its data is separated by ';' and consists of the following fields:
  - **ID:** integer to identify the boat. It's unique.
  - **name:** string containing the boat's name.
  - **type:** string containing the type of sailing boat. Possible values: Windsurf, Optimist, Laser, Patí Català, HobieDragoon, HobieCat.
  - **weight:** float that indicates the weight in kg.
  - **length:** float that indicates the length in meters.
  - **capacity:** integer that indicates the number of people that fit in the boat.
  - **n_competitions:** integer indicating the number of competitions won with the boat.
  - **state:** string containing the state of the boat, which can be new, restored, broken or unavailable.
  - **speed:** integer indicating the maximum speed in knots.
  - **center:** string containing the name of the center where the boat is stored. The number of unique values is bound depending on the dataset size.
- **boat 2**, ending in '\n'
- ...
- **boat N**, ending in '\n'

As you can see, the format is the same as in P1, meaning that you will be able to reuse some parts of your code to read the new datasets.

2.2 Sailors

For the sailors, each dataset has the following information:

```
120
8179933;Dusty Carr;92.85;5;0;6;8;3;2;5;7;4;12
5228;Anita Knapp;68.88;7;7;6;3;3;5;2;8;9;30
748651777;Chrissy El-Aurian;79.05;3;0;1;4;4;6;2;3;8;91
9481;Joe Kerr;50.08;2;8;6;6;7;1;6;6;9;49
334;Sid Down;41.33;6;5;10;8;5;8;0;7;9;22
8;Misty Shore;46.91;0;4;7;9;5;3;1;3;10;32
```

- **n_sailors:** number of sailors in the file, ending in `'\n'`
- **sailor 1**, ending in `'\n'`. Its data is separated by `';'` and consists of the following fields:
    - **n_affiliation:** unique integer to identify the sailor.
    - **name:** string containing the name of the sailor.
    - **weight:** float that indicates their weight in kg.
    - **skills:** array of 9 integers indicating the skill of the sailor for each type of boat. Each number will range from 0 to 10 and represent the following based on their position:
        1. Windsurf skill
        2. Optimist skill
        3. Laser skill
        4. Patí Català skill
        5. HobieDragoon skill
        6. HobieCat skill
    - **win_rate:** integer from 0 to 100 indicating the percentage of competitions won by the sailor (over the number of competitions they participated in).
- **sailor 2**, ending in `'\n'`
- ...
- **sailor N**, ending in `'\n'`

3 Problems

To plan the regatta, we want to solve two problems: First, we want to maximize the speed of partaking boats. Next, we want to find the minimum set of centers that can provide us with all boat types.

3.1 High speed sailing

In this case, the goal is to distribute the sailors in boats, looking for a combination that maximizes the sum of their real speeds. To get valid results, each boat must be full, meaning that there must be as many sailors assigned to it as its maximum capacity indicates. Some sailors may be left without a boat.

Generally, each boat has a maximum sailing speed, which is affected by a multiplying factor ranging from 0 to 1. This factor depends on the sailors that end up in the boat, specifically being the result of multiplying them all (each being a number from 0 to 1 as well).

$$speed_{real} = speed_{maximum} \cdot \prod_{i} impact_{sailor}$$

The individual impact of a sailor in the corresponding boat will be determined by the following formula:

$$impact_{sailor} = \frac{impact_{weight} + impact_{skill}}{2}$$

As can be appreciated, this impact is the average of two other more specific factors, each one of them also being a number from 0 to 1.

First, a sailor reduces the boat's speed according to their weight, which will fall between 40.00 and 99.99kg, and you can assume that it will be lower than the boat's weight.

$$impact_{weight} = \frac{100 - weight_{participant}}{weight_{boat}}$$

On the other hand, a boat's speed is also affected by the sailor's skill with boats of that kind, as well as their general skill in competitions. This factor will fall between 0 and 1, and will be calculated as the average of both values:

$$impact_{skill} = \frac{ability[type] + win_{rate}}{2}$$

Take into account that you'll have to normalise both the skill (given as an integer from 0 to 10) and the win rate (given as a percentage), obtaining numbers in the desired range (from 0 to 1).

3.2 Full fleet

To organize the regatta, we need to guarantee that at least one boat of each type will be available on site. However, boats are stored in different centers, meaning that it's possible that no single center satisfies this criterion.

Therefore, we want to figure out the smallest set of centers that, together, can satisfy this need, providing in total at least one boat of each type. Depending on the input, it's possible that no valid solution exists.

Remember that boats have attributes stating both the center they belong to and their type. Consider changing the representation of your data so that it helps you in more efficiently solving the problem.

4 Requirements

This section presents the project's main goals and describes the results expected from its realization.

4.1 Objectives

Even though there are alternatives, you're asked to solve **each** of the two problems with **at least two different strategies** based on combinatorial optimization (i.e., choosing between backtracking, branch and bound and greedy). Additionally, you must use each one of these three strategies **at least once**. The appropriateness of these choices will be evaluated.

The project's main objective is to compare the performance of the different proposed solutions, which means you should apply efficiency tweaks (marking, PBCBS...) whenever possible.

Similarly, to facilitate the result analysis, it's highly advisable to keep different versions of a solution to examine them in detail. Some **examples** of interesting comparisons include:

- **Brute force *vs* backtracking**, to see the effect of pruning.
- **Before *vs* after applying marking**, to appreciate the improvement.
- **Backtracking with PBCBS *vs* branch & bound**, to see the effect of changing the order in which we explore the solution space.
- **Use of different heuristics** in those algorithms that require them, to see their utility.

Naturally, there are many others, and you're free to choose the more relevant ones depending on the solutions you've decided to implement for each problem.

4.2 Code

The project that you submit must implement the different chosen strategies correctly, but there are no restrictions in terms of user interaction. You can also leave any mechanisms for performance evaluation that you may have used in your code (for instance, the measure of execution time).

Regardless, it's recommendable that you structure your code in a way that's easy to modify in terms of behavior (or the functionality being executed). This can be achieved in many ways, such as defining command line arguments, coding a small menu, using constants, commenting out code blocks in the main procedure…

You must use the provided datasets. Additionally, the solution to each problem must be logged to console in a clear and understandable format.

4.3 Report

The report should have the following sections, or an equivalent set that covers the same contents:

- **Cover** (with the group number and its members' full names and *logins*).
- **Numerated index**.
- Justification for the **chosen programming language**, with the advantages it brings and, if needed, possible issues.
- Explanation of the **design decisions behind the different algorithms** and strategies (pruning, heuristics, tweaks...).
- **Result analysis**, considering *inputs* of different sizes and explaining how the measurements were achieved. You should make any comparisons you consider necessary as well as provide charts supporting them.
- **Observed problems** and their solutions.
- Total **dedication** in hours. Also include a thematic breakdown into any categories you feel are necessary. Some examples include: Comprehension, investigation, design, implementation, result analysis, documentation...
- **Conclusions**, both on a personal but mostly on a technological level.
- **Bibliography** following ISO 690 or APA 7th standards.

The report must be written in **formal tone**. It has the same importance or more than the implementation of the project itself, as it reflects the knowledge acquired with the project and in the subject. Therefore, its content should **prioritize quality over quantity**.

The result analysis section is critically important to pass any project in this subject and will determine a substantial part of your mark. You must be capable of making correct performance measurements, as well as identifying relevant comparisons from them, and extracting substantial conclusions.

Using LaTeX to write the report is encouraged, but it won't affect the final mark.

5 Considerations

This section points out a series of details regarding the project and its development.

5.1 Programming language

The project can be implemented in any programming language, to be chosen by the group. If it's developed in C, it must compile, execute, and work properly on the university servers (*matagalls*).

5.2 Groups

The project is to be developed in pairs. By default, the same groups from last project will be maintained. If you want to make changes, you must inform the subject's professor by email (Pol Muñoz - pol.munoz@salle.url.edu) before December 21st, 2022 at 23:55h.

Students without a group won't be able to submit their project. To break groups up, refer to the subject's project rules.

5.3 Atlassian tools

Once this statement is presented, and after a small amount of time for possible group modifications, you will be assigned a repository in the university's Bitbucket server for your git-based version control. Independently of the chosen language and the group's size, **you must regularly use it during the project's development**.

If you want to use other Atlassian tools such as Jira or Confluence, you can ask the subject's professor for them via email.

5.4 Plagiarism detection

Cheating prevents any student from learning and denotes a lack of respect towards classmates who have spent time and efforts in their work.

A project is classified as a **highly important** academic activity. Plagiarism, whether partial or total, from a classmate or from the internet will be considered a **premeditated action**. Therefore, applying the university's copies regulation, it will be considered a **very serious misconduct**.

6 Delivery

This section describes everything having to do with the project's delivery, from the format to the deadline and retake mechanisms.

6.1 Format

Two different files must be submitted to the corresponding eStudy deliverable: The report in **PDF** format and a **ZIP** file containing:

- README: **TXT** or **MD** file with **ALL** the necessary steps to test the code are explained in the most detailed way possible (language, language version, used IDE, compilation/execution instructions, usage instructions…)
- Folder with the code and/or structure of the IDE-generated project. **This must match the version in your Bitbucket repository**.

Deliveries where the report is compressed inside the ZIP won't be accepted.

6.2 Deadline

The project will be published after the knowledge base to start implementing it is established and must be delivered before the **January 15th, 2023, at 23:55h** deadline. **No delivery will be accepted past this point**.

6.3 Retake

If the deadline is missed (NP) or the task is failed, the project can be submitted again before the **March 19th, 2023, at 23:55h** deadline, with a maximum grade of 8.

The last chance will be during the extraordinary call, before the **July 2nd, 2023, at 23:55h** deadline, and with a maximum grade of 5.