Algorithms for Computational Logic

Project1 Report - Group 19

José António Ribeiro da Silva Lopes
ist1103938

The Flying Tourist Problem

Imagine a traveler wants to plan a vacation trip where he already knows the amount of nights he's going to spend in each city and that he will only travel by plane. The order of the flights is not predefined, and the same city in which the trip starts is the same in which it ends.

The objective of this project is to develop a tool that given a list of flights computes the most efficient travel plan: the one with the lowest cost that visits all the cities for the specified number of nights.

How to run the project locally

First, make sure to have `python` and `pip` installed. On Ubuntu based linux distributions this can be done by running the following commands:

```
sudo apt-get install python3
sudo apt-get install pip
```

Then, to install the python-sat toolkit run the following command:

```
pip3 install python-sat
```

And finally to run the program:

```
python3 project1.py < <input_file> > <output_file>
```

If `output_file` is not specified, the program prints the result to `stdout`.

The format of the output is in compliance with the one represented in the project statement. Which means:

- One line with an integer defining the overall final cost;
- A sequence of n lines with the chronological sequence of flights defining the trip. Each line contains the date of the flight, the city of departure, the city of arrival, the departure time and the cost of the flight, separated by one white space. All lines terminate with an end of line character.

For all explanations below assume that n is the number of cities to be visited, m the number of flights, $\mathcal{F}$ the set of all the flights, *base* the beginning and end city and K the total number of nights spent traveling.

Variables used

- $\forall i \in \{1..n\} : c_i$ is true if the traveler as left city i.
- $\forall i \in \{1..n\} : c'_i$ is true if the traveler as arrived at city i.

- $\forall i \in \{1..n\}$: f_i is true if the flight i as been taken.

Hard Clauses

- $\forall i \in \{1..n\}$: $c_i \wedge c'i$, meaning that the traveler has to have left every city and arrived at every city.
- Assume that $i = 1$ is the index of the *base* city. Then, $\forall i \in \{2..n\} : (c_1 \vee \neg c_i)$, meaning that if the traveler hasn't left the *base* city then he can't have left any other city. This forces the trip to start at *base* city. Subsequently, $\forall i \in \{2..n\} : (\neg c'_1 \vee c'_i)$, meaning that if the traveler has arrived at the *base* city then he also has to have arrived at every other city. This forces the trip to end at *base* city.
- $\forall i \in \{1..m\} : \forall j \in \{1..n\}$: If $\mathcal{F}_i.\text{departure} = j$ then $\left(\neg f_i \vee c_j\right)$, meaning that if the traveler takes a flight from city j, he has certainly left that city.
- $\forall i \in \{1..m\} : \forall j \in \{1..n\}$: If $\mathcal{F}_i.\text{arrival} = j$ then $\left(\neg f_i \vee c'_j\right)$, meaning that if the traveler takes a flight to city j, he has certainly arrived at that city.
- Assume that *first_day* is the day of the first flight. Then, $\forall i \in \{1..m\}$: if $\mathcal{F}_i.\text{date} = first\_day$ and $\mathcal{F}_i.\text{departure} \neq base : (\neg f_i)$, meaning the traveler can't be leaving any city other that *base* on the first day.
- Assume that *first_day* is the day of the first flight. Then, $\forall i \in \{1..m\}$: if $\mathcal{F}_i.\text{date} - first\_day < K$ and $\mathcal{F}_i.\text{arrival} = base : (\neg f_i)$, meaning the traveler can't be going back to *base* if K nights haven't passed.
- Assume that *last_day* is the day of the last flight. Then, $\forall i \in \{1..m\}$: if $\mathcal{F}_i.\text{date} = last\_day$ and $\mathcal{F}_i.\text{departure} = base : (\neg f_i)$, meaning the traveler can't be leaving *base* on the last day.
- Assume that *last_day* is the day of the last flight and. Then, $\forall i \in \{1..m\}$: if $last\_day - \mathcal{F}_i.\text{date} < K$ and $\mathcal{F}_i.\text{departure} = base : (\neg f_i)$, meaning the traveler can't be leaving *base* if there aren't at least K nights left.
- $\forall i \in \{1..m\}, \forall j \in \{i+1..m\}$: If $\mathcal{F}_i.\text{depart} = \mathcal{F}_j.\text{arrival}$ and $\mathcal{F}_i.\text{depart} \neq base : \left(\neg f_i \vee \neg f_j\right)$, meaning that if the traveler leaves a city, no more flights to that city can be taken, except for *base*.
- $\forall i \in \{1..m\}, \forall j \in \{i+1..m\}$: If $\mathcal{F}_i.\text{arrival} = \mathcal{F}_j.\text{depart}$ and $\mathcal{F}_j.\text{date} - \mathcal{F}_i.\text{date} \neq \mathcal{F}_i.\text{arrival.city.nights} : \left(\neg f_i \vee \neg f_j\right)$, meaning that if the traveler takes a flight to a city c then if a flight leaving c doesn't do so in k_c nights, it cannot be taken.

Some of this clauses are not necessary to ensure the correctness of the algorithm, but are here to help the SAT solver reduce searching space, thus making it more efficient, as the drawback of creating more clauses is compensated by a better performance during the SAT solver processing.

Soft Clauses

- $\forall i \in \{1..m\} : (\neg f_i)$ with weight = $f_i.\text{price}$. Out of all the valid flights, only the ones the lowest price are chosen.

Cardinality Constraints

For all the cardinality constraints, the encoding used was the `totalizer` encoding.

- $\sum_{i=0}^{l} f_i = 1$ for all flights that depart from the same city, meaning that the traveler has to leave every city once.
- $\sum_{i=0}^{l} f_i = 1$ for all flights that arrive at the same city, meaning that the traveler has to arrive at every city once.

- $\sum_{i=0}^{l} f_i \leq 1$ for all flights that depart on the same day, meaning that no more that one flight can be taken on the same day.

Algorithm and configurations

The algorithm used was `RC2Stratified` with the underlying solver being `Glucose4`, as this was the combination that proved to be the fastest.

Variations of the problem

If the tourist was able to make layovers when travelling between cities, a hard clause needs to be added, saying that if a flight to a city that is not part of the route was taken then a flight departing from that same city on the same day would have to be taken, since there are not overnight layovers.

If the tourist could provide a minimum and maximum number of nights per city, then the clause that rejects a flight if it doesn't comply with the number of nights to spend in a city would need to be relaxed, to allow for the traveler to spend a number of nights between minimum and maximum on that city. So, this clause would only reject flights if they don't leave the city within the bounds given. The value of K would also need to be changed to the sum of all the maximum possible nights spent at each city, so that the clause that rejects a flight from *base* if there aren't at least K nights left would work properly.