INSTITUTO SUPERIOR TÉCNICO

Algorithms for Computational Logic
Project 1 Report

João Gouveia, 102611
October 7, 2024

1 Problem Description

A tourist plans to visit several cities by plane and aims to spend as little money as possible on flights.

- The trip must start and end in the tourist's home city.
- The tourist will stay a pre-defined number of nights in each city.
- The tourist will visit each city exactly once.
- The order in which the cities are visited is arbitrary.

2 Modeling the Problem

From this description, we can extract four conditions:

1. The tourist cannot take a flight before their departure from home.
2. After arriving in a city, the tourist must depart exactly X days later.
3. The tourist will arrive in each city exactly once.
4. The tourist must depart from home exactly once.

Condition 1 and 4 guarantee that the trip starts in the tourist's home city. Condition 2 guarantees that the tourist's stay in each city, except for the home city, adheres to the desired duration. It also guarantees that the trip ends at the home city, as the tourist must eventually leave every stop. Condition 3 guarantees that the tourist visits each city exactly once.

2.1 Encoding

The solution to this problem can be modeled as a MaxSAT problem.

Let V represent the set of cities the tourist will travel to, including their home city. Let $base$ denote the city in V corresponding to the tourist's home city. For each city $c \in V \setminus \{base\}$, we also define k_c, the number of nights the tourist will stay in c.

Let D_c represent the set of flights departing from city c, and A_c the set of flights arriving at c.

For each flight, define a variable f that represents whether a flight is to be taken. The variable f has two components: f_{cost}, which represents the cost of the flight, and f_d, which represents its date.

The hard clauses are defined as follows:

$$(\neg f \vee \neg f'),\ \forall f' \in D_c,\ \forall f \in D_{base}, \text{ where } c \in V \setminus \{base\}, \text{ and } {f'}_d < f_d \tag{2.1}$$

$$(\neg f \vee f' \vee \ldots \vee f^{(n)}),\ \forall f \in A_c, \text{ where } c \in V \setminus \{base\},\ \{f',\ \ldots,\ f^{(n)}\} \in D_c, \text{ and } f'_d,\ \ldots,\ f^{(n)}_d = f_d + k_c \tag{2.2}$$

$$\sum_{f \in A_c} f = 1, \text{ where } c \in V \tag{2.3}$$

$$\sum_{f \in D_c} f = 1, \text{ where } c \in V \tag{2.4}$$

The first three clauses correspond directly to the encoding of the first three aforementioned conditions. Clause 2.4 represents the encoding of Condition 4 when $c = base$. While the rest of the clause is not strictly necessary to maintain correctness, it improves the time required to solve the problem.

Next, we guarantee that the cheapest flight combination is selected by defining the following soft clauses:

$$(\neg f),\ \forall f \in D_c,\ \text{where } c \in V \text{ and with weight equal to } f_{cost} \tag{2.5}$$

3 Solving the Encoding

The solver used was the PySAT's RC2Stratified MaxSAT solver, with all heuristics enabled. The encoding used for the cardinality constraints was the bitwise encoding.

4 Variants

4.1 Allowing layovers

Currently, our modeling of the problem does not allow for layovers, we assume that the tourist only takes direct flights. If we wanted to include layovers in the planning, we could simply add a new f variable for each layover we want to potentially include.

This approach allows us to use the same clauses we used for direct flights, resulting in a straightforward encoding. It also provides us with great control over which layovers we want to allow. For instance, if we want to exclude layovers that result in wait times greater than 3 hours, we can choose not to create the f variables for those particular combinations of flights.

4.2 Allowing a variable number of nights per city

Currently, we enforce that the tourist spends exactly the predefined number of nights in each city. If we wanted to allow a variable number of nights — let's say the user provides a minimum, min_c, and maximum, max_c, number of nights — we would need to modify the clause used to ensure Condition 2.

Instead of including the negation of flight f and all flights that depart from c on the date $f_d + k_c$, we extend this clause to include all flights f' that depart from c, where $f_d + min_c \leq {f'}_d \leq f_d + max_c$.

5 Installing and Running

This project has only one dependency: PySAT, which can be installed by running:

```
pip install python-sat
```