INSTITUTO SUPERIOR TÉCNICO

Algorithms for Computational Logic
Project 1 Report

João Gouveia, 102611
October 8, 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.
- The tourist only takes direct flights, layovers are not allowed.

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. Finally, let $F = \{f_1, \ldots, f_n\}$ represent the set of available flights to the tourist. We define four functions associated with these flights: $o(f_i)$, which returns the city flight f_i departs from; $a(f_i)$, which returns the city flight f_i arrives at; $d(f_i)$, which returns the flight's date; and $c(f_i)$, which returns the cost associated with taking the flight.

We associate a variable x_i with each flight f_i, such that $x_i = 1$ iff the flight f_i is to be taken. We define D_c as the set of variables corresponding to flights departing from c, and A_c as the set of variables corresponding to flights arriving at c. Finally, we define D_{f_i} as the set of variables corresponding to flights that depart from $a(f_i)$ on the date $d(f_i) + k_{a(f_i)}$.

The hard clauses are defined as follows:

$$(\neg x_i \vee \neg x_j), \qquad \forall f_i, f_j \in F : \ o(f_i) = base, \ o(f_j) \in V \setminus \{base\}, \ d(f_j) \leq d(f_i) \tag{2.1}$$

$$(\neg x_i \vee \bigvee_{x_j \in D_{f_i}} x_j), \qquad \forall f_i \in F : \ a(f_i) \in V \setminus \{base\} \tag{2.2}$$

$$\sum_{x_i \in A_c} x_i = 1, \quad \forall c \in V \tag{2.3}$$

$$\sum_{x_i \in D_c} x_i = 1, \quad \forall c \in V \tag{2.4}$$

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:

$$(\neg x_i) \text{ with weight } c(f_i), \quad \forall f_i \in F \tag{2.5}$$

3 Solving the Encoding

The solver used was PySAT's RC2Stratified MaxSAT solver, with all heuristics enabled. The underlying SAT Solver used by RC2 was Glucose42. 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. If we wanted to include layovers in the planning, we could simply create a new flight f_i and the respective x_i 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 x_i variables corresponding to 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 want to allow a variable number of nights — with the user providing a minimum (min_c) and maximum (max_c) number of nights to spend in city c — we would need to modify the clause used to ensure Condition 2.

This can be achieved by redefining D_{f_i}, extending the set to include all variables corresponding to flights departing from $a(f_i)$ on a date d, such that:

$$d(f_i) + min_{a(f_i)} \leq d \leq d(f_i) + max_{a(f_i)}$$

5 Installing and Running

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

```
pip install python-sat
```