INSTITUTO SUPERIOR TÉCNICO

Algorithms for Computational Logic
Project 2 Report

João Gouveia, 102611
October 21, 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 has a minimum and maximum number of nights they want to spend 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. After taking a flight, the tourist's next flight must depart exactly X days later, with X being a value between the minimum and maximum number of nights the tourist wishes to spend in the city they arrived at.
2. The tourist will arrive in each city exactly once.
3. The tourist's first flight must depart from home.
4. The tourist's last flight must arrive at home.

Condition 1 guarantees that the tourist's stay in each city, except for the home city, adheres to the desired duration. Condition 2 ensures that the tourist does not visit the same city twice. Condition 3 and 4 guarantee that the trip starts and ends at the tourist's home city, respectively.

2.1 Encoding

The solution to this problem can be modeled as an SMT problem.

Let $V = \{v_0, \ldots, v_n\}$ represent the set of cities the tourist will travel to, including their home city. Let v_0 denote the city in V corresponding to the tourist's home city. For each city $v_i \in V \setminus \{v_0\}$, we also define min_i and max_i, the minimum and maximum number of nights the tourist will stay in v_i. Finally, let $F = \{f_0, \ldots, f_m\}$ represent the set of flights available to the tourist. We define four functions associated with these flights:

- $origin(f_i)$, which returns the ID of the city flight f_i departs from;
- $arrival(f_i)$, which returns the ID of the city flight f_i arrives at;
- $day(f_i)$, which returns how many days after the start of the vacation period the flight departs;
- $cost(f_i)$, which returns the cost associated with taking the flight.

We create the set of integer variables $X = \{x_0, \ldots, x_n\}$. These variables will hold the IDs of the chosen flights in order, that is, x_0 holds the ID of the first flight taken, x_n the ID of the last flight taken. We associate with X four sets of auxiliary integer variables:

- $A = \{a_0, \ldots, a_n\}$, where $a_i = arrival(f_{x_i})$;
- $O = \{o_0, \ldots, o_n\}$, where $o_i = origin(f_{x_i})$;
- $D = \{d_0, \ldots, d_n\}$, where $d_i = day(f_{x_i})$;
- $C = \{c_0, \ldots, c_n\}$, where $c_i = cost(f_{x_i})$.

The constraints are defined as follows:

$$(a_i = j) \implies (d_{(i+1)} - d_i \geq min_j), \qquad \forall a_i \in A \setminus \{a_n\},\ \forall f_j \in F \tag{2.1}$$

$$(a_i = j) \implies (d_{(i+1)} - d_i \leq max_j), \qquad \forall a_i \in A \setminus \{a_n\},\ \forall f_j \in F \tag{2.2}$$

$$(a_i \neq a_j), \qquad \forall a_i, a_j \in A\colon\ i \neq j \tag{2.3}$$

$$(o_0 = 0) \tag{2.4}$$

$$(a_n = 0) \tag{2.5}$$

$$(x_i = j) \implies (a_i = arrival(f_j)), \qquad \forall x_i \in X,\ \forall f_j \in F \tag{2.6}$$

$$(x_i = j) \implies (o_i = origin(f_j)), \qquad \forall x_i \in X,\ \forall f_j \in F \tag{2.7}$$

$$(x_i = j) \implies (d_i = day(f_j)), \qquad \forall x_i \in X,\ \forall f_j \in F \tag{2.8}$$

$$(x_i = j) \implies (c_i = cost(f_j)), \qquad \forall x_i \in X,\ \forall f_j \in F \tag{2.9}$$

$$(o_{(i+1)} = a_i), \qquad \forall a_i \in A \setminus \{a_n\} \tag{2.10}$$

$$(x_i \geq 0) \wedge (x_i \leq m), \qquad \forall x_i \in X \tag{2.11}$$

The first five constraints correspond directly to the encoding of the four aforementioned conditions. The following four clauses establish the association between the variables in X and their corresponding auxiliary variables. Clause 2.10 guarantees that the tourist can only depart from the city they are currently in. Finally, the last clause ensures that only valid flight IDs are selected.

Next, we ensure that the cheapest flight combination is chosen by minimizing the following function:

$$\sum_{c_i \in C} c_i \tag{2.12}$$

2.2 Alternative Encodings

The encoding described earlier is not the only solution to this problem. I also experimented with a Boolean encoding, which is similar to a MaxSAT approach, as well as an SMT encoding similar to the one previously discussed, with the main difference being that it did not use any auxiliary variables.

2.2.1 Boolean Encoding

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;
- $c(f_i)$, 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 a date d, such that:

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

The hard clauses are defined as follows:

$$(x_i \implies \neg x_j), \qquad \forall f_i, f_j \in F : \ o(f_i) = base, \ o(f_j) \in V \setminus \{base\}, \ d(f_j) \leq d(f_i) \tag{2.13}$$

$$(x_i \implies \bigvee_{x_j \in D_{f_i}} x_j), \qquad \forall f_i \in F : \ a(f_i) \in V \setminus \{base\} \tag{2.14}$$

$$\sum_{x_i \in A_c} x_i = 1, \ \ \forall c \in V \tag{2.15}$$

$$\sum_{x_i \in D_c} x_i = 1, \ \ \forall c \in V \tag{2.16}$$

The first clause guarantees that no flight is selected before the departure from the base and, together with clause 2.4, ensures that the trip starts at the tourist's hometown. Clause 2.2 ensures that the duration of the tourist's stay in each city respects the given interval and that all selected flights depart from the city where the tourist previously arrived. Additionally, together with clause 2.5, it guarantees that the trip ends at the tourist's hometown. The last two clauses also ensure that no destination is visited twice during the trip.

To guarantee that the cheapest flight combination is selected, we define the following soft clauses:

$$(\neg x_i) \text{ with weight } c(f_i), \ \ \forall f_i \in F \tag{2.17}$$

2.2.2 Alternative SMT Encoding

Let $V = \{v_0, \ldots, v_n\}$ 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 $v \in V \setminus \{v_0\}$, we also define k_v, the number of nights the tourist will stay in v. Finally, let $F = \{f_0, \ldots, f_m\}$ 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;
- $c(f_i)$, which returns the cost associated with taking the flight.

Let D_v define the set of flights departing from v, and A_v the set of flights arriving at v. Finally, let D_{f_i} denote the set of flights that depart from $a(f_i)$ on a date d, such that:

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

We create the set of integer variables $X = \{x_0, \ldots, x_n\}$. These variables will hold the IDs of the chosen flights in order, that is, x_0 holds the ID of the first flight taken, x_n the ID of the last flight taken. We also create the function c, that receives a flight ID and returns its cost, that is, $c(i) = cost(f_i)$.

The hard clauses are defined as follows:

$$\bigvee_{f_i \in D_{v_0}} x_0 = i \tag{2.18}$$

$$\bigvee_{f_i \in A_{v_0}} x_n = i \tag{2.19}$$

$$(x_i \geq 0) \wedge (x_i \leq m), \quad \forall x_i \in X \tag{2.20}$$

$$(\bigvee_{f_k \in A_v} x_i = k) \implies (\bigwedge_{f_k \in A_v} x_j \neq k), \quad \forall v \in V, \forall x_i, x_j \in X : \ i \neq j \tag{2.21}$$

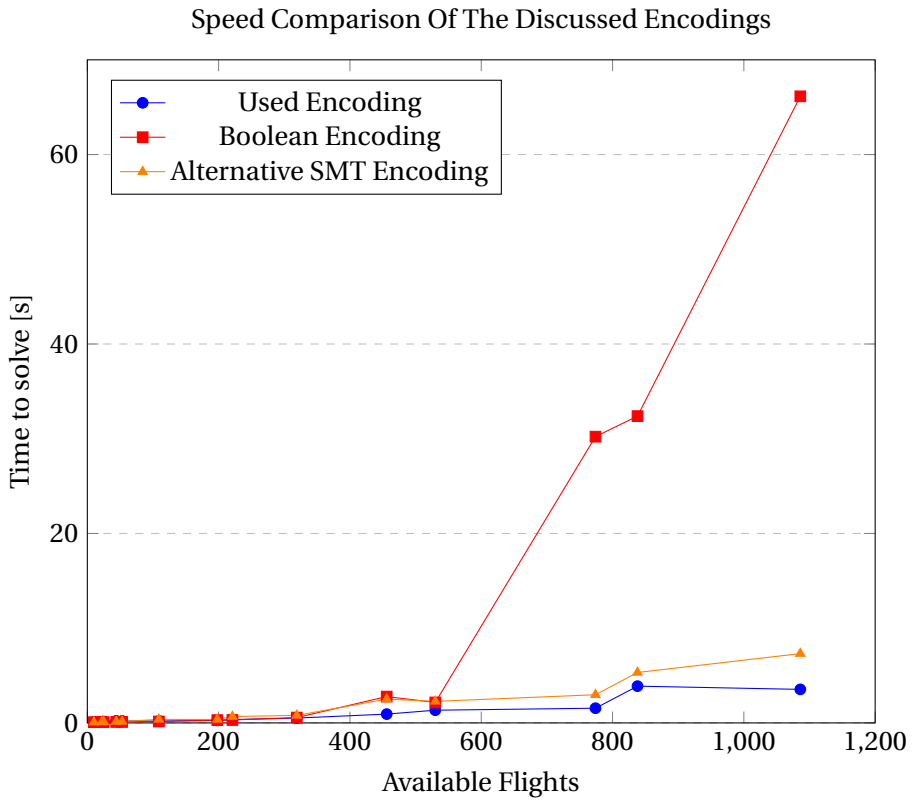
$$(x_i = j \implies \bigvee_{f_k \in D_{f_j}} x_{i+1} = k), \forall x_i \in X \setminus \{x_n\} \tag{2.22}$$

The first two clauses ensure that the trip starts and ends at the tourist's hometown. The next clause ensures that only valid flight IDs are selected. Clause 2.4 guarantees that no destination is visited twice during the trip. The final clause ensures that the duration of the tourist's stay in each city respects the provided interval and that all selected flights depart from the city where the tourist previously arrived.

2.3 Econding comparison

While the performance difference on smaller instances is fairly negligible, the encoding used significantly outperforms the other two discussed encodings when handling instances with a large number of flights.

This becomes particularly evident when handling instances containing 1200 flights. The encoding used allowed the solver to find the optimal answer in approximately 11 seconds, while the alternative SMT encoding took 30 seconds to solve. The boolean encoding could not be solved in a reasonable amount of time.



3 Solving the Encoding

The solver used was Microsoft's Z3 Theorem Prover (version 4.13.3.0). No tactics were used besides the ones already active by default.

4 Installing and Running

This project has only one dependency: z3py, which can be installed by running:

```
pip install z3-solver
```