ALC - Project1: Flying Tourist Problem

Group 6: Inês Ji 99238, Sebastião Carvalho 99326

3 October 2024

1 Problem Description

This project addresses a vacation planning problem for a European tourist who wishes to visit several cities across Europe. The tourist has a fixed number of vacation days and has a predetermined number of nights they intend to spend in each city they plan to visit.
The tourist will only travel between cities by plane. The sequence in which these cities are visited is not predefined, only that the trip begins and ends in their home city. The objective is to identify the most cost-effective travel itinerary, ensuring that travel requirements are met: the specific number of nights in each city, visiting each city exactly once, and starting and returning to the home city. Only direct flights between cities are considered, and no stopovers or layovers are allowed.

2 Install and run

To set up and run the project, follow the steps outlined below:

1. Prerequisites.
Ensure that Python and pip are installed on your system. Detailed instructions for installing Python and pip can be found in this installation guide.

2. Package Installation.
Once Python is installed, install the necessary Python packages by running the following command:

```
pip3 install pysat
```

3. Cloning the Repository.
Next, clone the project's GitLab repository to your local machine using the following command:

```
git clone git@gitlab.rnl.tecnico.ulisboa.pt:alc24/project1/6.git
```

4. Running the Project.
After cloning the repository, navigate to the project directory and execute the following command to run the program:

```
./proj1
```

3 Solution

3.1 Domain

Let $\mathbb{F}$ be the set of all possible flights, with cardinality M, and $\mathbb{C}$ be the set of all cities, with cardinality N. The tourist has a total of K vacation days. The set of cities $\mathbb{C}$ is defined as {1,...,N}.

3.2 Variables

The objective is to find the cheapest set of flights, subject to constraints. Considering the domain of the problem, we use the flights as our variables. A flight in $\mathbb{F}$ is represented as $f_{i,j,d}$, where i denotes the departure city, j denotes the arrival city, and d is the day of the flight. Additionally, let $p_{i,j,d}$ represent the price of flight $f_{i,j,d}$, and let k_c represent the number of nights to be spent in city c.

3.3 Encoding

3.3.1 Hard clauses

1. Depart from each city only once

Formal description:

$$\forall c \in \mathbb{C}, \sum_{d=1}^{K} \sum_{i=1}^{N} f_{c,i,d} = 1$$

Explanation: The tourist must depart from each city exactly once, therefore only one flight departing from each city can be taken.

2. Arrive at each city only once

Formal description:

$$\forall c \in \mathbb{C}, \sum_{d=1}^{K} \sum_{i=1}^{N} f_{i,c,d} = 1$$

Explanation: The tourist must arrive at each city exactly once, therefore only one flight arriving at each city can be taken.

3. Stay k_c days in city c

Formal description:

$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, h \in \mathbb{C}, \forall d, e \in \{1, ..., K\}, \neg f_{i,j,d} \vee (e = d + k_j) \vee \neg f_{j,h,e},$$
$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, h \in \mathbb{C}, \forall d, e \in \{1, ..., K\}, \neg f_{i,j,d} \vee (\sum f_{j,h,d+k_j} >= 1)$$

Explanation: The tourist must stay in each city c for exactly k_c nights. Therefore, the tourist must depart the city on a flight scheduled exactly k_c nights after arriving. If the tourist arrives in city c on day d, any flights departing from that city before or after the k_c nights are not valid options.

4. End in base city

Formal description:

$$\forall d, e \in \{1, .., K\}, \forall i, j, k \in \mathbb{C}, \neg f_{i,c,d} \vee (e < d) \vee \neg f_{j,k,e}, c \text{ is the base city}$$

Explanation: Once the tourist flies to the base city, no further flights can be taken. The last flight must be to the base city.

5. Start in base city

Formal description:

$$\forall d, e \in \{1, .., K\}, \forall i, j, k \in \mathbb{C}, \neg f_{c,i,d} \vee (e > d) \vee \neg f_{j,k,e}, c \text{ is the base city}$$

Explanation: The first flight of the trip must be from the base city. No flights that occur before this can be taken.

6. Remove flights with invalid departure date

Formal description:

$$\forall d \in \{1, .., K\}, \forall c, j \in \mathbb{C}, \forall i \in (\mathbb{C} \setminus \text{BaseCity}), \neg f_{i,j,d} \quad \text{if } \neg \exists f_{c,i,d-k_i} \quad \text{where } (d > k_i)$$
$$\forall d \in \{1, .., K\}, \forall c, j \in \mathbb{C}, \forall i \in (\mathbb{C} \setminus \text{BaseCity}), \neg f_{i,j,d} \quad \text{if } (d <= k_i)$$

Explanation: A flight $f_{i,j,d}$ (excluding those departing from the base city) is not valid if there is no flight arriving in city i from any city c on day $d - k_i$, where k_i is the number of nights to be spent in city i.

7. Remove invalid base city flights

Formal description:

$$\forall d \in \{1, ..., K\}, \forall i \in \mathbb{C}, (d >= \sum_{j=1}^{N} k_j) \vee \neg f_{icd}, \text{c is the base city.}$$

$$\forall d \in \{1, ..., K\}, \forall i \in \mathbb{C}, (d <= K - (\sum_{j=1}^{N} k_j)) \vee \neg f_{cid}, \text{c is the base city.}$$

Explanation: The tourist cannot arrive back at the base city before spending all K_I nights, where K_I is the sum of k_i nights for all cities i, except the base city. Therefore, flights arriving at the base city before K_I nights have passed are invalid. Similarly, the tourist cannot depart from the base city when fewer than K_I days remain, as it would be impossible to visit all planned cities. As a result, any flight departing from the base city with fewer than K_I remaining days is not valid.

3.3.2 Soft clauses

Flights as soft clauses
Formal description:

$$\forall f_{i,j,d} \in \mathbb{F}, \neg f_{i,j,d}, \text{with } weight = p_{i,j,d}$$

Explanation: For each flight, its negation is added as a soft clause with its price as the weight. The goal is to minimize the total cost of the flights taken while satisfying the hard constraints.

3.4 Algorithm

To solve the problem, we employ the Maximum Satisfiability (MaxSat) approach, utilizing the RC2 solver with default flags and WCNF as the formula, also with the default flags.
For encoding cardinality constraints into Conjunctive Normal Form (CNF), we utilize the Sequential Counter.

4 Extensions

4.1 Min and Max Nights

Suppose the problem is changed to allow the tourist to stay in each city for a minimum and maximum number of nights. To solve it, modifications to the encoding are required, specifically for the rules related to "Stay k_c days in city c", "Remove flights with invalid departure dates", and "Remove invalid base city flights".

The "Stay k_c days in city c" rule would be adjusted to have the following restrictions:

$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, h \in \mathbb{C}, \forall d, e \in \{1, ..., K\}, \neg f_{i,j,d} \vee (e <= d + k_{cmax}) \vee \neg f_{j,h,e}$$

$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, h \in \mathbb{C}, \forall d, e \in \{1, ..., K\}, \neg f_{i,j,d} \vee (e >= d + k_{cmin}) \vee \neg f_{j,h,e}$$

The "Remove flights with invalid departure dates" rule will remain fundamentally the same; however, k_i will no longer be a fixed integer but rather an integer belonging to the interval $[k_{imin}, k_{imax}]$. Thus, the formal description is given by:

$$\forall d \in \{1, .., K\}, \forall c, j \in \mathbb{C}, \forall i \in (\mathbb{C} \setminus \text{BaseCity}), \neg f_{i,j,d} \quad \text{if } \neg\exists f_{c,i,d-k_i} \quad \text{where } (d > k_i),\ k_i \in \mathbb{Z} \text{ and } k_i \in [k_{imin}, k_{imax}]$$

$$\forall d \in \{1, .., K\}, \forall c, j \in \mathbb{C}, \forall i \in (\mathbb{C} \setminus \text{BaseCity}), \neg f_{i,j,d} \quad \text{if } (d <= k_i) \quad \text{where } k_i \in \mathbb{Z} \text{ and } k_i \in [k_{imin}, k_{imax}]$$

The "Remove invalid base city flights" would also need to be changed to consider the sum of all k_{cmax} (instead of k_c) for the flights that arrive at the base city, and the sum of all k_{cmin} for the flights that depart from the base city.

$$\forall d \in 1, ..., K, \forall i \in \mathbb{C}, (d >= \sum_{j=1}^{N} k_{jmax}) \neg f_{i,c,d}, \text{where } c \text{ is the base city}$$

$$\forall d \in 1, ..., K, \forall i \in \mathbb{C}, (d <= K - (\sum_{j=1}^{N} k_{jmin})) \neg f_{c,i,d}, \text{where } c \text{ is the base city and } d > K_C - K_{min}$$

4.2 Allow Layovers

If layovers are permited within the same day, a new rule must be introduced to allow the creation of additional variables representing these layovers. The variable representation must also be adjusted to consider takeoff and landing times. Consequently, the variables will be represented as $f_{i,j,d,t,l}$, where i, j, and d maintain their previous meanings, while t represents the takeoff time and l denotes the landing time. Let $\mathbb{T}$ represent the set of valid timestamps in the format "HH:mm", specifically defined as {00:00, ..., 23:59}
We consider there is no time needed to switch between flights after landing. The following rule to create new variables would be used:

$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, k \in \mathbb{C}, \forall d \in \{1, ..., K\}, \forall t1, t2, l1, l2 \in \mathbb{T}, \neg f_{i,j,d,t1,l1} \vee \neg f_{j,k,d,t1,l2} \vee (l1 < t2) \vee f_{i,k,d,t1,l2}$$

This formulation ensures that if there is a flight from city i to city j and a subsequent flight from city j to city k on the same day d, and the landing time of the flight from city i to city j is earlier than the takeoff time of the flight from city j to city k, then a layover is considered to occur from city i to city k.