ALC 2024/2025

2^{st} Project – Flying Tourist Problem using SMT

17/10/2024, version 1.1

Overview

The 2nd ALC project is to develop a software tool for solving the Flying Tourist Problem problem. In order to solve this problem, students must use solvers for Satisfiability Modulo Theories (SMT).

Context

Consider an European tourist that is planning vacations. Suppose he has a given number of days for vacations and plans to visit several European cities.

The tourist will only travel between cities by plane and his trip starts and finishes in the city where he lives. For each city to visit, he has some friend that allows him to stay for free. Hence, additional days in a given city are not costly. However, for each city he defines a minimum and a maximum number of days to stay so that he can enjoy the visit. Note that the order he visits the cities is not pre-defined.

For example, suppose the tourist lives in Lisbon and wants to visit Rome, Paris and Stockholm. Moreover, the tourist wishes to spend a minimum of 2 nights in Rome, Paris and Stockholm. The maximum number of nights in Rome and Stockholm is 3, but he is willing to stay up to 4 nights in Paris.

Figure 1 shows an example of a possible trip. In this case, the order is Paris, Stockholm and Rome and the trip starts and ends in Lisbon. The goal is always to find the cheapest way to travel, as long as all the travel requirements are satisfied.

Example

Table 1 provides an example of flights considering the cities of Lisbon (LIS airport), Paris (CDG airport), Rome (FCO airport) and Stockholm (ARN airport). Consider that the tourist is located in Lisbon and wants to: (1) spend at least 2 nights in Rome, Paris and Stockholm and, (2) spend at most 3 nights in Rome and Stockholm and at most 4 nights in Paris.

Considering the flights in Table 1, the cheapest option is to start the trip on the 01/09 to Paris (CDG), then fly to Stockholm (ARN) on the 05/09, followed by a flight to Rome (FCO) on the 07/09 and arriving in Lisbon on the 09/09. The total cost of the trip would be 550

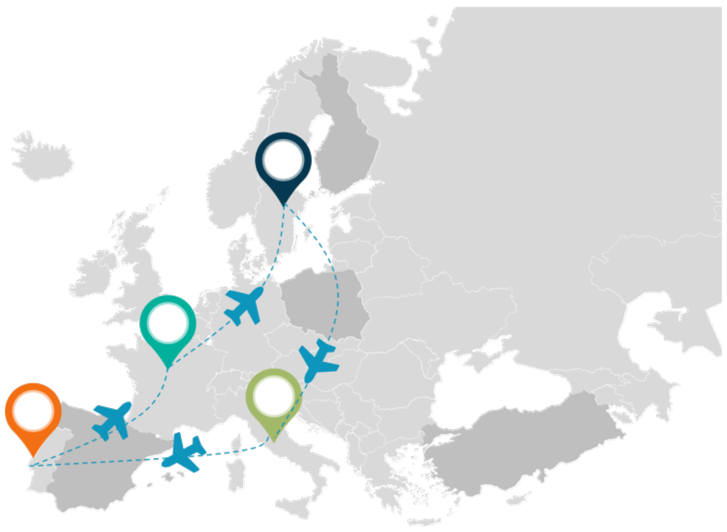
Figure 1: Example of a trip.

(80+100+200+130). In this case, he would spend 4 nights in Paris and 2 nights in both Rome and Stockholm.

Note that moving from a city to another can only be made with direct flights. The tourist does not want plane trips with layovers. Notice also that it might be the case that there are more than one option in each day to fly between two cities. For instance, on the 03/09 there are two flights between CDG and FCO. On the other hand, there might be some situations where there are no flights between two cities (e.g., there are no flights between LIS and ARN on the 02/09).

Problem Specification

Let $\mathscr{V}$ define the set of cities the tourist wants to visit and the origin city. Let n denote the cardinality of $\mathscr{V}$. Let *base* denote the city in $\mathscr{V}$ that defines the start and end city.

For each city c such that $c \in \mathscr{V} \setminus \{base\}$, k_m and k_M denote respectively the minimum and maximum number of nights the tourist wants to spend in city c. Let K denote the number of days available for travelling. The sum of all minimum number of nights is smaller than K (i.e., $\sum_{c \in \mathscr{V} \setminus \{base\}} k_m < K$). However, this is not guaranteed for the sum of all maximum number of nights.

Let $\mathscr{F}$ denote the set of flights in consideration. All departure and arrival cities of the flights in $\mathscr{F}$ belong to set $\mathscr{V}$. Moreover, all flights occur in a period of K consecutive days and they arrive the same day as they depart. There are no overnight flights.

Your goal is to help the tourist to define its travel plans such that:

Date (day/month)	Origin	Destination	Departure Time	Arrival Time	Price (EUR)
01/09	LIS	CDG	10:00	12:30	80
01/09	LIS	FCO	11:00	14:00	120
01/09	LIS	ARN	13:00	17:15	250
02/09	LIS	CDG	10:00	12:30	100
02/09	LIS	FCO	11:00	14:00	170
...	...	...	...	...	...
03/09	CDG	FCO	10:00	12:00	160
03/09	CDG	FCO	19:00	21:00	120
...	...	...	...	...	...
04/09	FCO	CDG	10:00	12:00	200
04/09	CDG	ARN	11:00	13:40	160
04/09	FCO	ARN	17:00	20:15	300
...	...	...	...	...	...
05/09	FCO	CDG	10:00	12:00	200
05/09	CDG	ARN	11:00	13:40	100
05/09	FCO	ARN	17:00	20:15	300
...	...	...	...	...	...
06/09	FCO	CDG	10:00	12:00	260
06/09	CDG	ARN	11:00	13:40	110
06/09	FCO	ARN	17:00	20:15	300
...	...	...	...	...	...
07/09	FCO	CDG	10:00	12:00	220
07/09	CDG	FCO	11:00	13:40	140
07/09	ARN	FCO	17:00	20:15	200
...	...	...	...	...	...
09/09	FCO	LIS	10:00	12:00	130
09/09	CDG	LIS	11:00	13:40	110
09/09	ARN	LIS	17:00	20:15	250
...	...	...	...	...	...
10/09	FCO	LIS	10:00	12:00	150
10/09	CDG	LIS	11:00	13:40	110
10/09	ARN	LIS	17:00	20:15	300

Table 1: Sample of flights considering the cities in Figure 1.

- The trip start and ends in the city defined as *base*.
- The tourist arrives and departs from each city only once.
- For each city $c \in \mathcal{V} \setminus \{base\}$, the tourist stays there at least k_m nights and at most k_M nights.
- The total cost of plane tickets is minimized.

Project Details

You are to implement a tool, or optionally a set of tools, invoked with command `proj2`. Your implementation must use a SMT tool to solve the travelling tourist problem as previously specified.

Your tool does not take any command-line arguments. The problem instance is to be read from the standard input.

Consider an instance file named `instance.ttp`. The tool is expected to be executed as follows:

```
./proj2 < instance.ttp > solution.out
```

The tool must write the solution to the standard output, which can then be redirected to a file (e.g., `solution.out`).

The programming languages to be used are only C/C++, Java or Python. The formats of the files used by the tool are described below.

File Formats

You can assume that all input files follow the description provided in this document. There is no need to check if the input is correct. Additionally, all lines (input or output) must terminate with the end-of-line character.

Input Format

The input file representing a problem instance is a text file that follows the following format:

- One line with an integer n $(n > 1)$ defining the number of cities to visit;
- One line with the name of a city and a three letter code (airport code) defining the *base* city;
- A sequence of $n-1$ lines where each line has the name of a city, a three letter code defining the airport code for the city and two integers defining respectively the minimum and maximum number of nights to be spent in the city;
- One line with an integer m $(m \geq 1)$ defining the number of flights;
- A sequence of m lines where each line contains:
  - flight date in the format DD/MM;
  - two airport codes denoting the departure and arrival airports;
  - two timestamps in the format HH:MM denoting the departure and arrival time;
  - one integer denoting the flight's cost in Euros

There is always one white space between names and integers in the same line. Moreover, all lines end with an end-of-line character.

Output Format

The output representing an optimal solution to the problem instance must comply with the following format:

- 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.

Important: The final version to be submitted for evaluation must comply with the described output. Project submissions that do not comply will be severely penalized, since each incorrect output will be considered as a wrong answer. An application that verifies if the output complies with the description will be available on the course's website.

Example 1

```
3
Madrid MAD
London LHR 2 4
Berlin BER 2 3
22
01/09 MAD LHR 10:00 12:30 150
01/09 MAD BER 10:00 12:30 130
02/09 MAD BER 12:00 14:30 150
02/09 MAD LHR 13:00 15:30 300
03/09 LHR BER 09:00 11:00 200
03/09 MAD LHR 10:00 12:30 250
03/09 BER LHR 10:00 12:00 100
03/09 MAD BER 11:00 13:30 60
04/09 LHR BER 09:00 11:00 200
04/09 BER LHR 10:00 12:00 100
05/09 LHR BER 09:00 11:00 300
05/09 BER LHR 10:00 12:00 300
05/09 BER MAD 15:00 17:30 20
05/09 LHR MAD 18:00 20:30 150
06/09 LHR BER 09:00 11:00 300
06/09 BER LHR 10:00 12:00 300
06/09 BER MAD 15:00 17:30 250
06/09 LHR MAD 18:00 20:30 150
```

```
07/09 LHR BER 09:00 11:00 300
07/09 BER LHR 10:00 12:00 300
07/09 BER MAD 15:00 17:30 400
07/09 LHR MAD 18:00 20:30 120
```

The optimal solution would be:

```
350
01/09 Madrid Berlin 10:00 130
03/09 Berlin London 10:00 100
07/09 London Madrid 18:00 120
```

Example 2

```
4
Lisbon LIS
Paris CDG 2 4
Rome FCO 2 3
Stockholm ARN 2 3
25
01/09 LIS CDG 10:00 12:30 80
01/09 LIS FCO 11:00 14:00 120
01/09 LIS ARN 13:00 17:15 250
02/09 LIS CDG 10:00 12:30 100
02/09 LIS FCO 11:00 14:00 170
03/09 CDG FCO 10:00 12:00 160
03/09 CDG FCO 19:00 21:00 120
04/09 FCO CDG 10:00 12:00 200
04/09 CDG ARN 11:00 13:40 160
04/09 FCO ARN 17:00 20:15 300
05/09 FCO CDG 10:00 12:00 200
05/09 CDG ARN 11:00 13:40 100
05/09 FCO ARN 17:00 20:15 300
06/09 FCO CDG 10:00 12:00 260
06/09 CDG ARN 11:00 13:40 110
06/09 FCO ARN 17:00 20:15 300
07/09 FCO CDG 10:00 12:00 220
07/09 CDG FCO 11:00 13:40 140
07/09 ARN FCO 17:00 20:15 200
09/09 FCO LIS 10:00 12:00 130
```

```
09/09 CDG LIS 11:00 13:40 110
09/09 ARN LIS 17:00 20:15 250
10/09 FCO LIS 10:00 12:00 150
10/09 CDG LIS 11:00 13:40 110
10/09 ARN LIS 17:00 20:15 300
```

The optimal solution would be:

```
510
01/09 Lisbon Paris 10:00 80
05/09 Paris Stockholm 11:00 100
07/09 Stockholm Rome 17:00 200
09/09 Rome Lisbon 10:00 130
```

Additional Information

The project is to be implemented in groups of one or two students.

The project is to be submitted through the git repository of the group. The faculty will create the group's repositories with a sample of problem instances. Jointly with your code, the repository must also contain a short report describing the main features of your project.

The report to be submitted must describe: (1) the problem to be solved, (2) how to install and run your project, (3) a description of the encodings you tryed, (4) the algorithm and configurations used to solve it and (5) an experimental evaluation and analysis of the SMT encodings you tryed. To compare the efficiency of the encodings and algorithm configurations, please report an experimental evaluation. The experimental evaluation can be made using the set of public instances.

The evaluation will be made taking into account correctness and efficiency given a reasonable amount of CPU time (80%) and the report (20%).

The input and output formats described in this document must be strictly followed.

Project Dates

- Project published: 10/10/2024.
- Project due: 23/10/2024 at 23:59.

Omissions & Errors

Any detected omissions or errors will be added to future versions of this document. Any required clarifications will be made available through the course's official website.

Versions

10/10/2024,	version 1.0:	Original version.
17/10/2024,	version 1.1:	Clarifications to the structure of the report.