TÉCNICO LISBOA

Project 1: Flying Tourist Problem

Students Eemi Kärkkäinen 113095
Senior Lecturer Vasco Manquinho

Overview

This project tries to construct an efficient solution to Flying Tourist Problem. Furthermore, solvers for Satisfiability (SAT), Maximum Satisfiability (MaxSAT), Pseudo-Boolean Satisfiability (PBS) and Pseudo-Boolean Optimization (PBO) were considered. From these Maximum Satisfiability (MaxSAT) was chosen because the constraints were the simplest to construct.

The Flying Tourist Problem is as described in the project instructions: The tourist has a given number of days for vacations and plans to visit several European cities. For each city he has fixed a number of nights to spend in. Traveling is done by direct flights and the trip starts and end to the city he lives in. Nights for each city is fixed but the order of the visits is free. The goal for the project is to always find the cheapest way to travel complying the given constraints.

How to install and run the project

For the project to work accordingly pysat-library has to be installed with *python3 -m pip install python-sat* command. After cloning the project it is also necessary to run *unzip ttp-checker.zip; chmod +x ttp-checker* command in order to unzip the ttp-checker script and give it execution permissions. After these steps the project can be run for single test, for example, test t01.txt with command *python3 project1.py < public-tests/t01.ttp > public-tests/t01.myout* . The result can be checked to be correct with command *./ttp-checker public-tests/t01.ttp public-tests/t01.myout* . These have to be ran from the main folder of the project.

It is also possible to run all of the 16 tests with command *./run.sh* but it should be noted that the script also needs execution permissions so running *chmod +x run.sh* command first is necessary.

Description of the encoding

Let $\mathbb{V}$ denote the set of cities to be visited including the base city. Let $\mathbb{F}$ denote the set of all flights in consideration. Let $\mathbb{D}$ denote the set of all dates there are flights considering all $f_i \in \mathbb{F}$. Let *base* denote the base city. For each city c such that $c \in \mathbb{V} \setminus \{base\}$, k_c denotes the number of nights to be spent there. Let the total amount of nights spent in all of the cities denoted by k. All flights $f_i \in \mathbb{F}$ have a origin city o_i, arrival city a_i and date d_i such that $o_i \in \mathbb{V}$, $a_i \in \mathbb{V}$ and $d_i \in \mathbb{D}$. Every flight $f_i \in \mathbb{F}$ additionally has a cost p_i. Flights never depart and arrive in different days. Furthermore, let $\mathbb{F}^{from}_{c,d} \subseteq \mathbb{F}$ be the subset of flights departing from city $c \in \mathbb{V}$, on date $d \in \mathbb{D}$. Let $\mathbb{F}^{to}_{c,d} \subseteq \mathbb{F}$ be the subset of flights arriving in city $c \in \mathbb{V}$, on date $d \in \mathbb{D}$.

First constraint mentioned in the project instructions is that the trip must start and end in the city defined as *base*. Lets encode it accordingly:

$$\forall d_i \in \mathbb{D}, \forall f_i \in \mathbb{F}^{from}_{base,d_i} \left(\neg f_i \vee \sum_{r \in \mathbb{F}^{to}_{base,d_i+k}} r \geq 1 \right)$$

The next constraint is that the tourist arrives and departs each city once. This can be encoded as follows:

$$\forall c \in \mathbb{V}, \sum_{d \in \mathbb{D}} \sum_{f_i \in \mathbb{F}^{to}_{c,d}} f_i = 1, \sum_{d \in \mathbb{D}} \sum_{f_i \in \mathbb{F}^{from}_{c,d}} f_i = 1$$

.

Furthermore, there is a constraint that for each city $c \in \mathbb{V} \setminus \{base\}$, the tourist stays there exactly k_c nights:

$$\forall d_i \in \mathbb{D}, \forall c \in \mathbb{V} \setminus \{base\}, \forall f_i \in \mathbb{F}^{to}_{c,d} \left(\neg f \vee \sum_{t \in \mathbb{F}^{from}_{c,d+k_c}} t \geq 1 \right)$$

Last constraint makes sure that the total cost of plane tickets is minimized:

$$\min F(f, p) = \forall f_i \in \mathbb{F} \sum_{i=1}^{m} f_i p_i$$

Algorithm and configurations

The project was constructed with Python and pysat-library that offer different SAT-solvers. From the solvers RC2 (*relaxable cardinality constraints*) MaxSAT solver was used for this project. It was configured to use Glucose4 SAT solver with core exhaustion, core minimization and At-Most1 adaptation enabled. Core exhaustion works by increasing the bound as much as possible and core minimization by over-approximating an MUS using simple deletion-based MUS extraction algoritm. AtMost1 adaptation detects and adapts intrinsic AtMost1 constraints. [3]

For creating the clauses for the solver, weighted CNF formula was used. That enabled the addition of hard clauses with no weights and soft clauses with weights. Furthermore, this enabled the solver to try to minimize the price of the plane tickets.[1]

Cardinality constraints were used to ensure that from every city the tourist departs from and arrives to only once. Bitwise encoding was used for the cardinality constraints since minimizing the amount of clauses seemed to make the solver perform faster. [2]

Additionally, some filters were used to lower the amount of variables the solver has to consider. First, flights originating from *base* were filtered if there were no flights to *base* after the duration of the overall trip. The second filtration process removed the flights where the departure date from the considered city exceeded the last date of flights.

The Python code was optimized to used function calls only if necessary. For example, dictionaries were used to access IDs efficiently. Further optimization with binary trees and recursive functions were considered. The implementation of these methods did not succeed so they did not make it to the project.

Future work

If the problem definition would allow layovers, one should also consider more than just the flights from and to city. Additional clauses should state that with the distinct flight chosen, there must

be a set of flights that are connected and end up to the desired destination. Soft clauses with weights (prices) would make sure that the cheaper option between the set or a direct flight is to be chosen.

Implementation using minimum and maximum amount of stays over the fixed amount would add some complexity to the solution. Where now the solution only implements cardinality constraints over the fixed amount of time after arrival to a city, with flexible timeframe, one should introduce more constraints and maybe even more variables.

References

[1] Antonio Morgado Alexey Ignatiev Joao Marques-Silva. *Boolean formula manipulation*. 2024. URL: https://pysathq.github.io/docs/html/api/formula.html?highlight=wcnf#pysat.formula.WCNF (visited on 10/09/2024).

[2] Antonio Morgado Alexey Ignatiev Joao Marques-Silva. *Cardinality encodings*. 2024. URL: https://pysathq.github.io/docs/html/api/card.html?highlight=cardenc#pysat.card.CardEnc (visited on 10/09/2024).

[3] Joao Marques-Silva António Morgado Carmine Dodaro. 2024. URL: https://pysathq.github.io/docs/html/api/examples/rc2.html (visited on 10/09/2024).