University of Glasgow | School of Computing Science

The Traveller's Problem

Iva Babukova (2030458b)

January 25, 2017

1. INTRODUCTION

The Traveller's Problem (TP) is a combinatorial optimisation problem whose extensions and variations are often encountered by travellers around the world. Given a set of airports, a set of flights, a set of destinations that is a subset of the airports, and a special airport A_0, a solution to TP is a travel schedule that starts and finishes at A_0, visits all destinations and is in accordance with additional constraints specified by the traveller. For instance, the traveller may wish to spend a certain amount of days in each destination, to take a minimum number of connection flights, or to give minimum amount of money for flights.

1.1 Problem Formulation

Each instance of TP consists of:

1. A set of airports $A = \{A_0, ..., A_n\}$ for $n > 0$. Each airport $A_i \in A$ represents a location the traveller can begin their commute in, visit as a desired destination, or connect in on the way to their destination.
2. The trip starts and ends at the same airport A_0, which is referred to as the *home point*.
3. The total travel time T, within which the traveller must have visited all destinations and returned to the home point. The first day is day 0.
4. A set of flights $F = \{f_0, ..., f_m\}$. Each flight f_j has:
   - departure airport A^d_j,
   - arrival airport A^a_j,
   - date t_j,
   - duration Δ_j,
   - cost c_j,

   for some non-negative integer j less than or equal to n. The date t_j is a positive rational number less than or equal to T that shows at which day f_j leaves its departure airport. The duration Δj is a positive fraction that shows the amount of time that takes for flight f_j to go from A^d_j to A^a_j. The cost c_j is a positive number that denotes the number of units of some currency ϵ that the traveller pays in order to be able to board flight f_j.
5. Each airport A_i has a *connection time* C_{A_i}, that is the time that takes to switch from any selected flight f_p with $A^a_p = A_i$ to any selected flight f_q with $A^d_q = A_i$, where f_q is immediately after f_p in a solution.
6. A set of *destinations* $D = \{D_1, ..., D_l\}$, $D \subseteq A$, $l \leq n$.

A solution to any instance of TP is a sequence s of k valid flights, $\langle f_{i_1}, f_{i_2}, ..., f_{i_k}\rangle \subseteq F$, also called a *trip*. We say that s is valid if the flights in s have the following properties:

1. $A^d_{i_1} = A^a_{i_k} = A_0$
2. $A^a_{i_j} = A^d_{i_{j+1}}, \quad 0 < j < k$
3. $t_{i_j} + \Delta_{i_j} + C_r \leq t_{i_{j+1}}, \quad 0 < j \leq k, \quad r = A^d_{i_{j+1}}$
4. $t_{i_k} + \Delta_{i_k} \leq T$
5. $\forall D_p \in D,\ \exists\ f_{i_j} \in s$, such that $A^a_{i_j} = D_p$

In this work, we refer to these properties as *trip properties.* Note that a valid sequence of flights may contain one or more flights to and from airports that are not destinations. Such airports are called *connections.*

TP is NP-hard The *optimization* version of TP (TPO) asks for an *optimal* solution s which minimizes the total sum of the prices of the flights in s, denoted by $c(s)$.

The *decision* version of TP (TPD) asks whether there exists a valid sequence of flights s, such that $c(s)$ is less than or equal to some given integer B. The solution of this problem is a 'yes' or 'no' answer.

There exist a variety of additional constraints and extensions that can be added to TP. Our problem formulation has only presented the hard constraints which every valid solution to a TP instance must satisfy. In real-world problems, travellers may have additional preferences (soft constraints) and requirements (hard constraints) with regards to their travel. These are discussed in the next two sections.

1.1.1 Hard Constraints

This section presents some additional constraints that might be imposed on a TP instance. If any of them is required, then a solution that does not satisfy the requirement is considered as invalid.

1. Travellers may wish to spend a certain amount of days at a given destination, specified by both upper and lower bounds. The days may additionally be constrained to be consecutive or not.
2. Travellers may require to spend a given date at a given destination, for example due to an event occurring on that date in this destination.
3. Travellers may require not to fly through a given airport more than once.

1.1.2 Soft Constraints

It may be desirable to search for a solution that satisfies some of the following requirements:

1. Travellers may wish to spend a certain amount δ_i of days in each destination D_i, where δ_i may be specified as a lower or an upper bound.

2. Travellers may wish to avoid taking connection flights. In such requirement, we wish to maximise the number of flights to and from destinations.

3. Travellers may want to spend as little time on flying as possible. In such case, we wish to find a solution that minimises the sum of the durations of all flights.

Note that we may have an instance for which all soft constraints can not be satisfied simultaneously. In such case, the traveller may be required to rank his requirements in an order of preference. The instance becomes a lexicographic optimisation problem, where we first optimise the highest ranked objective, and subject to this we optimise the second ranked objective and so on. If all requirements are equally important for the traveller, we have to solve a multiobjective optimisation problem, where the objectives are all constraints required by the traveller. Each of the objectives is given a weight of importance. The problem is then to optimize the objective function, composed by the constraints, each of them multiplied by its weight.

Note that most of the aforementioned constraints can be viewed as either hard or soft, depending on the user requirements. It is therefore suggested that any attempt at an investigation of TP assumes as an additional non-functional requirement that any proposed model to solve TP is flexible and can be easily extended by adding, removing and modifying the aforementioned constraints.

2. TP SOLUTIONS

We have implemented two solutions for TP using Integer Programming (IP) and Constraints Programming (CP). This section presents the implemented models and our choice of IP and CP solvers.

2.1 CP Solution

2.1.1 CP Model

Let $m = |F|$. We introduce an array $\mathcal{S}$ of size $m+1$ (indexed $0, 1, ..., m$) that represents the TP tour and a variable z with domain $dom_z = \{1, ..., m\}$ to denote the number of flights in the trip. If $\mathcal{S}[i] = j$, then flight $f_j \in F$ is the $(i+1)^{th}$ flight in the tour, for $(0 \leq i < m)$ and $(1 \leq j \leq m)$. To denote the end of the tour, we set $\mathcal{S}[z] = 0$. All subsequent variables in $\mathcal{S}$ will then have to be 0. Each variable v in $\mathcal{S}$ is either 0, if no flight is taken at that step, or it is equal to some flight number, that is $dom_v = \{0, ..., m\}$.

TP can then be formulated as the problem of minimising the objective function:

$$\sum_{i=0}^{m-1} c_{\mathcal{S}[i]} \tag{1}$$

subject to all constraints presented below.

The following constraint restricts that once the end of the trip is reached at some position z, no flights are further added to $\mathcal{S}$.

$$\mathcal{S}[i] > 0 \wedge \mathcal{S}[i+1] = 0, \ (0 \leq i \leq z-1) \tag{2}$$

The all-different constraint below enforces that every flight is taken only once [**?**].

$$\text{allDiff}(\mathcal{S}[0], ..., \mathcal{S}[z-1]) \tag{3}$$

The trip properties are enforced by the following five constraints, where constraint 1 corresponds to trip property (1), constraint 2 to trip property (2), etc:

1. $dom_{\mathcal{S}[0]} = \{j \in \{1, ..., m\} : A_j^d = A_0\}$

   $dom_{\mathcal{S}[z-1]} = \{j \in \{1, ..., m\} : A_j^a = A_0\}$
2. $dom_{\mathcal{S}[i]} = \{j \in \{1, ..., m\} : A_j^d = A_p^a, \ p = \mathcal{S}[i-1]\}$,

   $\forall i (1 \leq i < z)$
3. $t_p + \Delta_p + C_r \leq t_q, \ p = \mathcal{S}[i], q = \mathcal{S}[i+1], r = A_q^a$,

   $\forall i (1 \leq i < z)$
4. $t_q + \Delta_q \leq T, \quad$ where $q = \mathcal{S}[z-1]$
5. $\forall A_k \in D, \ |\{i : (0 \leq i < z) \wedge A^a_{\mathcal{S}[i]} = A_k\}| > 0$

Constraint 1 restricts the domains of the first and the $(z-1)^{th}$ variable in $\mathcal{S}$ to contain only flights that depart from/arrive at the home point. For constraint 2, the domain of each variable in $\mathcal{S}$ is set to include only flights that depart from the arrival airport of the previous flight. Constraint 5 restricts that the number of the flights that arrive at every destination in the trip is positive.

2.1.2 Choice of CP Solver

We use Choco 4.

2.1.3 Hard Constraint 2

Hard constraint 2 (HC2) in Section 1.1.1 is implemented in the CP model as follows. Let $P = \{p_1, ..., p_h\}$ denote the list of every HC2 for a given TP instance π. Each HC2 is represented as a pair $p_r = \langle A_{k_r}, t_{i_r} \rangle \in P$ for $0 \leq r \leq h \leq |D|$, where the traveller must be at destination $A_{k_r} \in D$ at date $t_{i_r} < T$.

We introduce an array $\mathcal{D}$ of size h. The domain of each variable in $\mathcal{D}$ is $\{1, ..., m\}$. If $\mathcal{D}[r] = i$, then the flight at position $\mathcal{S}[i]$ departs from A_{k_r} at least one day after date t_i [1] and the flight $\mathcal{S}[i-1]$ arrives at A_k before date t_{i_r}, satisfying p_r. This is achieved by the following constraints:

$$\forall \, p_r \in P, \, \exists \, i \, (0 < i < m) : \tag{4}$$

$$\mathcal{D}[r] = i \wedge dom_{\mathcal{S}[i-1]} = S_1 \wedge dom_{\mathcal{S}[i]} = S_2, \text{ where:}$$

$$S_1 = \{j : f_j \in F \wedge A_j^a = A_{k_r} \wedge t_j + \Delta_j < t_{i_r}\},$$

$$S_2 = \{j' : f_{j'} \in F \wedge A^d{}_{j'} = A_{k_r} \wedge t_{j'} > t_{i_r} + 1\}$$

$$\text{allDiff}(\mathcal{D}[1], ..., \mathcal{D}[h]) \tag{5}$$

2.2 Objective Functions

3. TP DATASETS

This section is dedicated to the process of obtaining all TP instances used later during the evaluation.

3.1 Data Format

Each TP instance is represented as JSON formatted file, as shown in Figure 1. We have chosen JSON over other possible representations for its ease of use, flexibility and wide support for most programming languages.

[1]It is assumed that the traveller wants to stay at least for a day after the required date. Ideally, the solver should allow for taking this as an input parameter.

```
{
  "airports" : [
    {
      "name" : "Glasgow",
      "connection_time" : 0.1,
      "purpose" : "home_point"
    },
    ...
  ],
  "flights": [
    {
      "id" : 1,
      "dep_airport" : "Glasgow",
      "arr_airport" : "London",
      "date" : 1.34,
      "duration" : 1.12,
      "price" : 74
    },
        ...
  ],
  "holiday_time" : 20,
}
```

Figure 1: The format of TP instances.

3.2 Random Datasets

In order to generate random TP instances, we have written a custom data generator G. This section discusses its implementation.

Program G is written in Python. Given a number of flights m, a number of airports n, a number of destinations d, a holiday time T and a unique id i, it returns a TP instance formatted as described in the previous section.

TP instances are generated using the following steps. First, G chooses n airports, each with randomly assigned connection time between 0.01 and 1 and a unique string as name. From these airports, one is chosen as a home point and d are chosen as destinations.

Next, G generates a list l consisting of the names of all destinations and the home point. The first and the last element in l is the name of the home point. The other elements are all destinations, ordered randomly.

As a third step, G creates a list of flights l_f that forms a valid trip and visit all airports in the specified order in l.

The fourth step consists of dividing each flight in l_f in two flights until either the size of l_f reaches m or no flight can be further divided. For each flight $f_j \in l_f$ we create two flights f_j1 and f_j2, such that:

- $A^d_{j1} = A^d_j$,
- $A^a_{j2} = A^a_j$,
- $A^a_{j1} = A^d_{j2} = A_k$, where A_k is chosen randomly from A,
- $t_{j1} + \Delta_{j1} + C_{A_k} \leq t_{j2}$,
- $t_{j1} \geq t_j$, and
- $t_{j2} + \Delta_{j2} \leq t_j + \Delta_j$

Step five consists of generating flights with random arrival and departure airports, random date that is less than T and random duration. It is executed only if the number of flights in l_f is less than m and until it reaches m.

Without loss of generality, the cost of each flight generated after either of the aforementioned steps is configured to be a random number from 1 to 200.

3.3 Skyscanner Flights Data

4. REFERENCES

[1] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Bell Telephone Laboratories, Incorporated, United States of America, 1979.

[2] W. J. van Hoeve. The alldifferent constraint: A survey. *CoRR*, cs.PL/0105015, 2001.

APPENDIX

A. INVESTIGATED PROBLEMS

1. **Travelling Salesman Problem * (TSP)** [2]

   Instance. Set A of n cities, distance $d(A_i, A_j)$ between each pair of cities A_i, $A_j \in A$, positive integer B.

   Question. Is there a tour of A having length B or less, i.e., a permutation of cities $\gamma = \langle A_{\pi_1}, ..., A_{\pi_n} \rangle$ of A such that the total travel distance L_γ:

$$L_\gamma = \left(\sum_{i=1}^{n-1} d(A_{\pi_i}, A_{\pi_{i+1}}) \right) + d(A_{\pi_n}, A_{\pi_1}) \leq B \quad ?$$

B. COMPLEXITY OF TP

Theorem 1. *TPD is NP-complete.*

Proof. This proof first shows the membership of TPD in the NP class of problems. Second, we prove the NP-hardness of TP by constructing a polynomial-time reduction from a known NP-complete problem Π to TPD, where Π is chosen to be TSP, defined in Section A. Its NP-hardness follows by a reduction from the Hamiltonian Cycle problem. The proof is presented by [**?**].

Given an instance of TPD and s, which is a sequence of flights from F, we can write an algorithm that checks in polynomial time whether s is a solution. To accept or reject validity, the algorithm only needs to traverse s and check that it satisfies all required properties. Therefore, TP is in NP.

Let π be an instance of TSP. Let π' be an instance of TPD with the following properties:

- The set of airports in π' is identical to the set of cities in π and it is similarly denoted as A (a city in π is called an airport in π'). Airport A_1 is the home point.
- Each airport in A is also a destination.
- The connection time C_{A_i} for each airport A_i is equal to 0.
- T is equal to n.
- Let C be the Cartesian product of the airports in A with itself, that is $C = A \times A = \{(A_i, A_j) : A_i \in A,\ A_j \in A,\ i \neq j\}$. Then F is a set of flights, such that for every $(A_i, A_j) \in C$, there exists a flight f_k in F, such that $A^d_k = A_i$ and $A^a_k = A_j$ for every date $0 \leq t < T$.
- For every $f_k \in F$, c_k is equal to $d(A^d_k, A^a_k)$ in π. Therefore, the flight costs also satisfy the triangle inequality.
- For every $f_k \in F$, $\Delta_k = 1$.
- B is the upper bound on the allowed total cost.

[2]Note that TSP is a special case of VRP when only one vehicle is allowed.

Suppose that $\gamma = \langle A_{i_1}, A_{i_2}, ..., A_{i_n} \rangle$ is a solution to π, where $\langle i_1, ..., i_n \rangle$ is a permutation of $\langle 1, ..., n \rangle$ and the total travel distance $L_\gamma \leq B$. Without loss of generality, assume that $i_1 = 1$. In π', γ is equivalent to the order of visited airports by some sequence of flights $s = \langle f_{j_1}, f_{j_2}, ...f_{j_n} \rangle$, such that for each p $(1 \leq p \leq n)$ there exists q $(1 \leq q \leq n)$ such that $A^d_{j_q} = A_{i_p}$ and $A^a_{j_q} = A_{i_{p+1}}$, where subscripts are taken modulo n. Therefore, s satisfies property (1) of a valid solution. For each q $(1 \leq q \leq n)$, $A^a_{j_q} = A^d_{j_{q+1}}$ and $t_{j_q} = q - 1$. We know that such flights exist in F by the construction of the set F.

From the construction of s, it follows that property (2) also holds. Properties (3) and (4) also hold, since we have chosen flights from F such that for every $f_{j_q} \in s$, $t_{j_q} = q - 1$ $((1 \leq q \leq n))$. Property (5) is satisfied, since all airports in A are destinations.

Since the cost of every flight in F is equal to the distance between the two cities in π that correspond to its departure and arrival airport, it follows that $c(s) = L_\gamma \leq B$.

The sequence s satisfies all requirements for a valid solution to π'. Therefore, a solution of π is also a solution to π'.

Conversely, suppose that $s = \langle f_{j_1}, ..., f_{j_k} \rangle$ is a solution to π', where the flights in s visit destinations in the sequence $\gamma' = \langle A_{i_1}, A_{i_2}, ..., A_{i_m} \rangle$. We will prove that γ' is a solution of π.

By construction of π', all airports in A are also destinations. Therefore, γ' contains all cities in A, that is $m \geq n$. Suppose that $m > n$ and an arbitrary airport A_p $(1 \leq p < n)$ is included more than once in γ'. Then s must contain more than one flight with arrival airport equal to A_p. The duration of each flight in π' is one day. Therefore, for every q $(1 \leq q < n - 1)$, $t_{j_{q+1}} = q$. Since the traveller has only n days of total travel time, and s is restricted to contain exactly n flights, that is $k = n$. The only way to visit n distinct destinations, given n flights is that all flights in s have unique arrival airports. Assuming that A_p is visited more than once means that there is more than one flight with arrival airport equal to A_p, which is a contradiction. Therefore, $m = n$ and each airport in A is visited exactly once.

From the properties of s it follows that $A^d_{j_1} = A^a_{j_n} = A_1$. Therefore, γ' is a cycle of size n. We know that $c(s) \leq B$. We assigned a cost of each flight f_k in F to be equal to $d(A^d_k, A^a_k)$ in π. Therefore, the total travel distance $L_{\gamma'} = c(s)$ which is less than or equal to B.

According to the specifications of π, γ' is a solution to π. Therefore, a solution to π' is also a solution to π.

The transformation from a TSP instance to an instance of TPD can be done in polynomial time. For each of the $n(n\text{-}1)/2$ distances $\mathrm{d}(A_i, A_j)$ that must be specified in π, it is sufficient to check that the same cost is assigned to the flights from A_i to A_j for all dates.

Therefore, TP is in NP and the decision version of TSP can be reduced to TPD in polynomial time, from which it follows that TPD is NP-complete. $\square$