The Traveller's Problem

Iva Stefanova Babukova

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Masters project proposal

18 December 2016

Contents

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: _____ Signature: _____

1 Introduction

In this work we present the Traveller's Problem (TP), a computational task whose extensions and variations are often encountered by travellers around the world. The task is concerned with creating a valid travel schedule, using airplanes as a means of transportation and in accordance with certain constraints specified by the traveller.

2 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_j^d,
   - arrival airport A_j^a,
   - 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_j^d to A_j^a. 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 $A_p^a = A_i$ to any selected flight f_q with $A_q^d = A_i$, 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 *tour*. We say that s is valid if the flights in s have the following properties:

$$A^d_{i_1} = A^a_{i_k} = A_0 \tag{1}$$

$$A^a_{i_j} = A^d_{i_{j+1}}, \quad 0 < j < k \quad j \tag{2}$$

$$t_{i_j} + \Delta_{i_j} + C_r \leq t_{i_{j+1}}, \quad 0 < j \leq k, \quad \text{where } r = A^d_{i_{j+1}} \tag{3}$$

$$t_{i_k} + \Delta_{i_k} \leq T \tag{4}$$

$$\forall D_p \in D, \; \exists \, f_{i_j} \in s, \text{ such that } A^a_{i_j} = D_p \tag{5}$$

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*.

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.

2.1 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.

- 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.
- 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.
- Travellers may require not to fly through a given airport more than once.

2.2 Soft Constraints

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

- 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.
- Travellers may wish to avoid taking connection flights. In such requirement, we wish to maximise the number of flights to and from destinations.
- 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. Multiobjective and lexicographic optimisation problems are discussed later in this work.

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.

3 Worked Examples

We present an example instance of TP and comment on some of its solutions.

Example 1. A traveller wishes to visit 4 airports from a set of 7 airports available to travel to and from:

Glasgow (G), Berlin (B), Milan (M), Amsterdam (A), Paris (P), Frankfurt (F), London (L).

Airport G is the home point, F and L are connections, and B, M, A and P are the destinations. The travel time of the traveller is 15 days. All available flights are listed on Table 1. For simplicity, the duration of each flight is assumed to be 1 day. This means that if the traveller gets a flight at date x, they will reach the arrival airport at day $x + 1$.

Solution. A valid solution of the TP instance in the example above is the sequence of flights s, where the each flight is represented by its flight number, specified in the first column of Table 1:

$$s = \langle GA1, AP4, PM6, MF9, FB11, BG13 \rangle$$

Flight No	Departs	Arrives	Date	Price	
1	GA1	G	A	1	74
2	GF1	G	F	1	86
3	FB2	F	B	2	156
4	GL3	G	L	3	25
5	MF3	M	F	3	78
6	BP4	B	P	4	67
7	AP4	A	P	4	58
8	PM6	P	M	6	71
9	FM8	F	M	8	234
10	MF9	M	F	9	39
11	FA10	F	A	10	220
12	FB11	F	B	11	122
13	FM12	F	M	12	250
14	PL12	P	L	12	45
15	BG13	B	G	13	335
16	BL13	B	L	13	102
17	AG13	A	G	13	90
18	LG14	L	G	14	24

Table 1: List of flights with departure and arrival airports, flight date and price.

The total flights cost $c(s)$ is 699.

A valid solution with lower cost is the following sequence:

$$s' = \langle GA1, AP4, PM6, MF9, FB11, BL13, LG14 \rangle$$

Here $c(s')$ is 483 and hence s is not optimal.

Example 2. Given the same problem instance as in Example 1, suppose that the traveller has booked a ticket for a concert in B on day 3. The traveller requires to attend the concert.

Solution. In such case, neither s, nor s' from Example 1 are solutions, because both of them assign the traveller to be at a different location (airport A) at day 3. The following sequence is a solution:

$$s'' = \langle GF1, FB2, BP4, PM6, MF9, FA10, AG13 \rangle$$

The total cost $c(s'')$ is equal to 729, which is more expensive than s and s'.

4 List of Problems

This Section gives a list of known problems, referred to in this work when proving the NP-hardness of TP (Section 5) and when reviewing the existing work (Section 6). The problems, marked with an asterisk ($*$) next to their title, are known to be NP-complete [9].

1. **Travelling Salesman Problem** * **(TSP)** [1]
   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 $\gamma = \langle A_{\pi(1)}, A_{\pi(2)}, ..., A_{\pi(n)} \rangle$ of A such that the total travel distance L_γ:

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

2. **Travelling Salesman Problem Under the Triangle Inequality*** **(TSP-Δ)** [2]
   Instance. Set A of n cities, distance $d(A_i, A_j)$ between each pair of cities A_i, $A_j \in A$ that satisfies the triangle inequality, positive integer B.

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

Question. Is there a tour of A having length B or less, i.e., a permutation of cities $\gamma = \langle A_{\pi(1)}, A_{\pi(2)}, ..., A_{\pi(n)} \rangle$ of A such that the total travel distance L_γ:

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

3. **Time-Constrained TSP* (TCTSP)** [3]

   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, lower and upper bounds l_i and u_i respectively for each city A_i that specify its time window.

   Question. Is there a permutation of cities $\gamma = \langle A_{\pi(1)}, A_{\pi(2)}, ..., A_{\pi(n)} \rangle$ of A, such that each city A_{π_j} is visited at time t_j, where $l_j \leq t_j \leq u_j$, $t_j < t_{j+1}$ for $(1 \leq j \leq n-1)$ and

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

4. **The Assignment Problem (AP)**

   Instance. Set A and set B with equal size, cost $c(a, b)$ of matching $a \in A$ to $b \in B$.

   Question. Find a bijection $f : A \leftarrow B$ such that $\sum_{a \in A} c(a, f(a))$ is minimised.

5 Complexity of TP

We state a theorem about the complexity of TP and prove it.

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 4. Its NP-hardness follows by a reduction from the Hamiltonian Cycle problem. The proof is presented by Garey and Johnson [9].

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.

[3] Note that this problem is a generalisation of TSP.

- 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 $A_k^d = A_i$ and $A_k^a = A_j$ for every date $0 \leq t < T$.
- For every $f_k \in F$, c_k is equal to $d(A_k^d, A_k^a)$ 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.

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 $A_{j_q}^d = A_{i_p}$ and $A_{j_q}^a = A_{i_{p+1}}$, where subscripts are taken modulo n. Therefore, s satisfies property (1) of a valid solution. For each q $(1 \leq q \leq n)$, $A_{j_q}^a = A_{j_{q+1}}^d$ 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 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.

□

6 Background Survey

We present some existing approaches to dealing with NP-hard problems, namely branch and bound (Section 6.3), constraint programming (Section 6.2) and integer programming (Section 6.1). The way each of these methods tackles NP-hardness is illustrated with a specific problem as an example, defined in Section 4.

6.1 Linear and Integer Programming

Linear Programming (LP) is a way to solve problems by modeling their requirements as a system of linear inequalities and equations and subject to them finding an optimal solution to the problem, expressed as a function that has to be either minimised or maximised. This function is called the *objective function*.

The field of LP is well-studied [12, 6] and it has many applications in the industry for resource optimisation [11, 10]. However, Dantzig [5] shows that there are many important optimisation problems that can not be modeled as a linear program, because their variables can take only integer values. Such problems can be modeled with the tools of integer programming (IP). Integer programming (IP) is an extension of linear programming (LP). The difference between LP and IP is that in the IP model variables are restricted to take only integer values.

The rest of this section presents different methods to solve problems with the tools offered by IP and comments on their performance by using TSP as an example problem.

6.1.1 Integer Programming Formulation of TSP

Dantzig et al. [7] formulate TSP as the following IP problem. Let $x(i, j)$ be a variable that denotes whether city j succeeds city i in a TSP tour. The value of $x(i, j)$ is equal to 1 if this is true, or 0 if it is false. If $x(i, j)$ is 1, then i is called the *outgoing* city and j - the *incoming* city. The objective function of TSP is to minimise the total length of the tour, that is:

$$\min \sum_{i \in A} \sum_{j \in A} d(i, j) x(i, j) \tag{6}$$

In each TSP tour every city is visited once. Moreover, each city has to be incoming and outgoing exactly once. Therefore, the value of $x(i, j)$ for every i will be 1 for only one j and 0 for the rest. This can be enforced with the following constraint:

$$\begin{aligned} \sum_{j \in A} x(i, j) &= 1, \qquad i \in A, \\ \sum_{i \in A} x(i, j) &= 1, \qquad j \in A, \end{aligned} \tag{7}$$

Constraint (7) ensures that each city in A is picked exactly once as an incoming and an outgoing city. However, it allows for the existence of one or more cycles of $n_1 < n$ cities, called *subtours*. For instance, consider Figure 1 and Figure 2. Both of them satisfy constraint (7). However, Figure 2 does not represent a valid tour, because it contains two subtours, each of size 3. To tackle this problem, Dantzig et al. [7] introduce the *subtour elimination* constraint, that is:

$$\sum_{S \bar{S}} x(i, j) \geq 2, \quad (i, j) \in (S \times \bar{S}) \cup (\bar{S} \times S) \text{ for all } S \subset A, \bar{S} = A \setminus S, \quad \bar{S}, S \neq \emptyset \tag{8}$$

Here, S is a subset of A and $\bar{S}$ is the set of all cities that are in A and not in S. Constraint (8) enforces that at least two cities in S are connected with cities from $\bar{S}$. We give Figure 1 and Figure 2 as an example. Let $S = \{A_1, A_2, A_3\}$, then $\bar{S} = \{A_4, A_5, A_6\}$. Constraint (8), would detect the tour on Figure 2 as invalid, as there is no city in S that is connected to a city in $\bar{S}$. Figure 1 will be accepted, since A_1 and A_3 are connected with A_6 and A_4 respectively.

The formulation of the general TSP problem is given by the objective function (6), which has to be minimised, subject to constraints (7) and (8).

6.1.2 The Assignment Problem Relaxation

The assignment problem (AP) relaxation consists of removing the subtour elimination constraint from the TSP IP model and minimising the objective function (6), only subject to constraint (7).

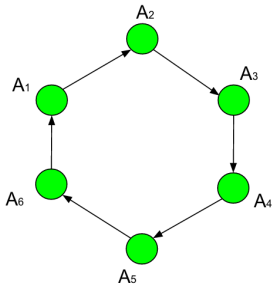

Figure 1: A valid TSP tour

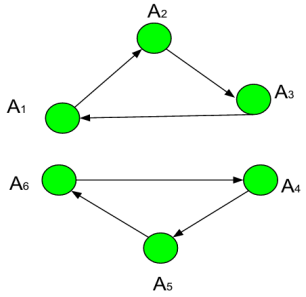

Figure 2: Two subtours of size 3

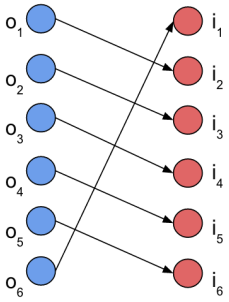

Figure 3: AP solution equivalent to Figure 1

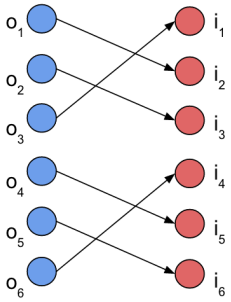

Figure 4: AP solution equivalent to Figure 2

It is well known and extensively used relaxation [15, 8, 2, 1, 13]. In this section we explain how this method is used for solving NP-hard problems, using TSP as an example.

Let π' be the problem that results after performing AP relaxation on some instance π of TSP. We will show that π' is equivalent to the Assignment Problem (AP).

The model for π' can be expressed as a bipartite graph G like the graphs shown on Figure 4 and Figure 3. G has two sets of vertices: $O = o_1, .., o_n$ and $I = i_1, ..., i_n$ and a set of edges $E = \{(o_p, i_q) \,|\, a_p, a_q \in A,\ p \neq q\}$ for $a_p, a_q \in A$, where I and O are both identical to A. Each edge $e = (o_p, i_q)$ in E has weight w_e equal to $d(a_p, a_q)$ and it represents a path from some city $o_p \in O$ to some city $i_q \in I$. Finding a solution to π' is equivalent to finding a perfect matching in G of minimum value of the sum of weight of the edge between every two matched vertices. Each vertex in O then represents an outgoing city and each vertex in I represents an incoming city.

Problem π' asks for assigning each vertex in O to a vertex in I, which is equivalent to an instance of AP with $A = O$, $B = I$ and $w_e = c(a_p, b_q)$, $e = (o_p, i_q)$, $p \neq q$ or $w_e = \infty$ if $p = q$.

Note that a solution to π' can be either a tour or a collection of subtours (since we relaxed the subtour elimination constraint). Figure 4 and Figure 3 are both solutions to AP obtained after performing an AP relaxation to a TSP instance with 6 cities. They are examples of how not all solutions to π' are solutions to π. Let the optimal solution to π' $\pi'^{\star}$ have total edge weight equal to α. Suppose that it is an invalid tour in π. Therefore, the tour in $\pi^{\star}$ is of length bigger than α. Conversely, suppose that $\pi'^{\star}$ is a valid tour in π. Therefore, the length of the tour in $\pi^{\star}$ is equal to α. Hence, the length of an optimal solution to $\pi^{\star}$ is bigger than or equal to the value of $\pi'^{\star}$. In the existing work, this feature is used to derive a lower bound on the cost of the most optimal

tour [15, 7, 1]. This approach is frequently combined with a branch and bound technique [14]. We discuss this approach more in the next section.

Dantzig et al. [7] solve the constructed TSP model using a novel method, which they call the "cutting-plane method". This work skips to present the procedure. We point the interested reader back to [7] and [4], where in the latter the cutting-plane method is explained with more detail using a 10 city TSP instance as an example.

6.2 Constraint Programming

Constraint Programming (CP) is a widely used method for solving optimisation problems [3, 16, 17]. Given an instance π of some optimisation problem, we first model it as a constraint programming problem π_M. Problem π_M consists of a set of variables V, each with a set of possible values, called its *domain*, and a set of rules, called *constraints* about variables assignment. A solution to π is an assignment of all variables in V, such that all constraints are satisfied.

Whenever the domain d_v of a variable v is empty in some temporal assignment of variables, we say that we have a *domain wipeout* and this variables assignment is regarded as invalid.

search tree?

maybe explain the all-different constraint, because it is used in the CP model.

6.2.1 The Vehicle-Routing Problem with Constraint Programming

explain how VRP is modeled and solved with CP. Comment on the difference between CP and IP.

6.2.2 Heuristics

common heuristics for TPS, TCTSP, VRP?

6.3 Branch and Bound Algorithms

1. Bounds based on the Assignment Problem
2. Lin-Kernigham heuristic (2-opt, k-opt, the Concorde code) very briefly

7 Proposed Approach

We model TP as an integer and as a constraints programming problem. The constraint programming approach is explained in Section 7.1 and the integer programming model is presented in Section 7.2.

7.1 CP Model

Let m be equal to $|F|$. We introduce an array $\mathcal{S}$ of size m that represents the TP tour. The domain of each variable in $\mathcal{S}$ is $\{0, ..., m\}$. If $\mathcal{S}[i] = j$, then flight $f_j \in F$ is the ith flight in the tour. To denote the end of the tour at some position $z + 1 < m$, we set $\mathcal{S}[z + 1] = 0$. All subsequent variables in $\mathcal{S}$ will then have to be 0 and we say that the tour is of size z.

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

$$\sum_{j=1}^{m} c_k k, \quad \text{where } k = \mathcal{S}[j] \tag{9}$$

subject to the following constraints:

$$\mathcal{S}[j] = 0 \Rightarrow \mathcal{S}[j+1] = 0, \quad \forall\, j,\ 0 \leq j < m \tag{10}$$

$$\text{allDiff}(\mathcal{S}[0], ..., \mathcal{S}[z]), \quad \forall\, z,\ 0 \leq z < m \text{ s.t } \mathcal{S}[z] \neq 0 \tag{11}$$

Constraint (10) restricts that once the end of the tour is reached at some position j, no flights are further added to the tour. Constraint (11) enforces that every flight in the tour is taken only once, using an all-different constraint [18].

The five properties of a valid tour are added to the model as the following constraints:

$$\text{d}_{\mathcal{S}[0]} = \{j \in \{1, ..., m\} : A^d_j = A_0\} \tag{12}$$

$$(\mathcal{S}[z] \neq 0 \wedge \mathcal{S}[z+1] = 0) \Rightarrow (\text{d}_{\mathcal{S}[z]} = \{j \in \{1, ..., m\} : A^a_j = A_0\})$$

$$\forall\, i > 1,\ \text{d}_{\mathcal{S}[i]} = \{0, j \in \{1, ..., m\} : A^d_j = A^a_p,\ p = \mathcal{S}[i-1]\} \tag{13}$$

$$t_p + \Delta_p + C_r \leq t_q, \text{ where } p = \mathcal{S}[j],\ q = \mathcal{S}[j+1],\ r = A^a_q, \quad \forall\, j,\ 0 \leq j < m - 1 \tag{14}$$

$$(\mathcal{S}[z] \neq 0 \wedge \mathcal{S}[z+1] = 0) \Rightarrow (t_q + \Delta_q \leq T), \text{ where } q = \mathcal{S}[z] \tag{15}$$

$$\forall A_k \in A,\ J_k = \{j \in \mathcal{S}[i] \mid A^a_j = A_k \text{ for } 0 \leq i < m\}.\ \forall k,\ A_k \in D \Rightarrow |J_k| > 0 \tag{16}$$

Constraint (12) is equivalent to property (1). Each of the two equations restricts the domain of the corresponding variable in $\mathcal{S}$ to contain only flights that depart from/arrive at the home point. Property (2) is enforced by constraint (13). The domain of each variable in $\mathcal{S}$ is set to include only flights that depart from the arrival airport of the previous flight. Constraints (14) and (15) did not undergo substantial changes from properties (3) and (4). To enforce that every destination airport is visited at least once (constraint (16)), we create a set J, where J_k is equal to the number of flights that arrive at A_k in the tour. If A_k is a destination, then the size of J_k can only be positive.

7.2 IP Model

Most of the constraints for the TP CP model need modification in order to be applicable to a IP model. In particular, IP does not allow for "if-then", "all-different" and other non-integer constraints. The model described in this section is a modification of the TP CP model that complies with the rules of integer programming.

Let m be equal to $|F|$. We introduce a variable $x_{i,j}$ for every i, j between 0 and m, such that $x_{i,j} = 1$ if i^{th} flight is f_j or 0 otherwise. This variable is somewhat similar to $\mathcal{S}$, used for the CP model, where $\mathcal{S}[i] = p$ is equivalent to $x_{i,p} = 1$. The objective function of TP is to minimise:

$$\sum_{j=1}^{m} \sum_{i=1}^{m} c_j x_{i,j} \tag{17}$$

subject to:

$$\forall\, i, \sum_{j=1}^{m} x_{i,j} \leq 1 \tag{18}$$

$$\forall\, j, \sum_{i=1}^{m} x_{i,j} \leq 1 \tag{19}$$

Constraint (18) restricts that only one flight is taken at each step i. Constraint (19) is equivalent to the all-different constraint (11): it enforces every flight to be taken at most once.

The five properties of valid tour are expressed as follows:

$$\sum_{j \in S_1} x_{1,j} = 1, \quad \text{where } S_1 = \{j \in \{1, ..., m\} : A_j^d = A_0\} \tag{20}$$

$$\sum_{j \in S_2} x_{z,j} = 1 \wedge \sum_{j=1}^{m} x_{z+1,j} = 0, \quad \text{where } S_2 = \{j \in \{1, ..., m\} : A_j^a = A_0\} \tag{21}$$

Constraint (20) enforces that the flight taken at step 1 must depart from the home point. Constraint (21) enforces an end to the sequence for some integer z and makes the last flight arrive at the home point. Property (1) of a valid tour is expressed by enforcing both (20) and (21).

$$\sum_{j \in S_1} x_{i-1,j} = \sum_{j' \in S_2} x_{i,j'}, \quad \forall 1 < i \leq z \wedge \forall y \in A, \text{ where} \tag{22}$$

$$S_1 = \{j \in \{1, ..., m\} : A^a_j = y\} \text{ and } S_2 = \{j' \in \{1, ..., m\} : A^d_{j'} = y\}$$

To add property (2) to the IP model, we define the sets of flights S_1 and S_2, such that all flights in S_2 depart from the arrival airport of S_1. Constraint (22) restricts that every flight always departs from the arrival airport of the previous taken flight, assuming that is it non-zero, where z denotes the number of flights in a tour.

$$\sum_{j=1}^{m} x_{i-1,j} = \sum_{j' \in S} x_{i,j'}, \quad \text{where } S = \{j' \in \{1, ..., m\} : t_j + \Delta_j + C_r \leq t_{j'},\ r = A^d_j\} \tag{23}$$

$$\sum_{j \in S} x_{z,j} = 1, \quad \text{where } S = \{j \in \{1, ..., m\} : t_j + \Delta_j \leq T\} \tag{24}$$

Properties (3) and (4) are expressed similarly to (2) by defining the set of flights which the j of every $x_{i,j}$ can be chosen from. The properties are enforced via constraints (23) and (24) respectively.

To enforce property (5), we introduce a new variable $\mathcal{D}_{i,j}$. Let $k = |D|$, that is the number of destinations in the TP instance. For every i, $\mathcal{D}_{i,j} = 1$ if there is a flight f_j in the tour such that $A^a_j = D_i$. Since each destination has to be visited in the tour, the sum of all i, there must be a j such that $\mathcal{D}_{i,j} = 1$. This can be expressed as the following IP constraint:

$$\forall\, i, \sum_{j \in J} \mathcal{D}_{i,j} \geq 1, \quad J = \{j \in \{1, ..., m\} : x_{i,j} = 1 \wedge A^a_j = D_i\} \tag{25}$$

7.3 Experimental Datasets

7.3.1 Skyscanner Flights API

7.3.2 Randomly Generated TP instances

8 Work Plan

show how you plan to organize your work, identifying intermediate deliverables and dates.

References

[1] Edward K. Baker. Technical note - an exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):938–945, 1983. doi: 10.1287/opre.31.5.938. URL http://dx.doi.org/10.1287/opre.31.5.938.

[2] Mandell Bellmore and John C. Malone. Pathology of traveling-salesman subtour-elimination algorithms. *Oper. Res.*, 19(2):278–307, 1971. ISSN 0030-364X. doi: 10.1287/opre.19.2.278. URL http://dx.doi.org/10.1287/opre.19.2.278.

[3] Yves Caseau and Franois Laburthe. Solving small tsps with constraints, 1997.

[4] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. On a linear-programming, combinatorial approach to the traveling-salesman problem. *Oper. Res.*, 7(1):58–66, February 1959. ISSN 0030-364X. doi: 10.1287/opre.7.1.58. URL http://dx.doi.org/10.1287/opre.7.1.58.

[5] George B. Dantzig. On the significance of solving linear programming problems with some integer variables. *Econometrica*, 28(1):30–44, 1960. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1905292.

[6] George B. Dantzig. *Linear programming and extensions*. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ, 1963. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+180926950&sourceid=fbw_bibsonomy.

[7] George B. Dantzig, D. Ray Fulkerson, and Selmer M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2(4):393–410, 1954. doi: 10.1287/opre.2.4.393. URL http://dx.doi.org/10.1287/opre.2.4.393.

[8] W.L. Eastman. *Linear Programming with Pattern Constraints*. PhD thesis, Harvard University, Cambridge, MA, 1958.

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

[10] W. W. Garvin, H. W. Crandall, J. B. John, and R. A. Spellman. Applications of linear programming in the oil industry. *Manage. Sci.*, 3(4):407–430, July 1957. ISSN 0025-1909. doi: 10.1287/mnsc.3.4.407. URL http://dx.doi.org/10.1287/mnsc.3.4.407.

[11] P. M. Jacovkis, H. Gradowczyk, A. M. Freisztav, and E. G. Tabak. A linear programming approach to water-resources optimization. *Zeitschrift für Operations Research*, 33(5):341–362, 1989. ISSN 1432-5217. doi: 10.1007/BF01416081. URL http://dx.doi.org/10.1007/BF01416081.

[12] L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, pages 366–422, 1960.

[13] G. Laporte, H. Mercure, and Y. Nobert. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1):33–46, 1986.

[14] Eugene L. Lawler, David B. Shmoys, Alexander H. G. Rinnooy Kan, and Jan K. Lenstra. *The Traveling salesman problem : a guided tour of combinatorial optimization*. Wiley interscience series in discrete mathematics and optimization. J. Wiley and sons, Chichester, New York, Brisbane, 1987. ISBN 0-471-90413-9. URL `http://opac.inria.fr/record=b1117783`.

[15] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Oper. Res.*, 11(6):972–989, December 1963. ISSN 0030-364X. doi: 10.1287/opre.11.6.972. URL `http://dx.doi.org/10.1287/opre.11.6.972`.

[16] David Manlove, Gregg O'Malley, Patrick Prosser, and Chris Unsworth. A constraint programming approach to the hospitals / residents problem. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, pages 155–170, 2007. doi: 10.1007/978-3-540-72397-4_12. URL `http://dx.doi.org/10.1007/978-3-540-72397-4_12`.

[17] Kevin Mcdonald and Patrick Prosser. A case study of constraint programming for configuration problems, 2002.

[18] Willem Jan van Hoeve. The alldifferent constraint: A survey. *CoRR*, cs.PL/0105015, 2001. URL `http://arxiv.org/abs/cs.PL/0105015`.