Flying Tourist Problem: An Integer Linear Programming Approach

Francisco Madaleno dos Santos
franciscomadaleno@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

November 2019

Abstract

This work addresses the Flying Tourist Problem (FTP), which aims to find the best schedule, route, and set of flights for a given unconstrained multi-city flight request. The developed work proposes an Integer Linear Programming formulation with the intent of finding optimal solutions. This formulation was integrated with CPLEX and evaluated comparing its results and performance to other similar systems. The obtained results show that, contrary to the existing systems, this optimization system invariably finds the optimal solution. Nonetheless, to improve the computational performance for large instances, this system is itegrated with a metaheuristic optimization algorithm. Furthermore, the FTP was adapted to a similar problem and the Generalized Flying Tourist Problem (GFTP) arised. This problem is a generalization of the FTP whereby it is required to find the best route in a graph which visits all specified subsets of cities. Considering similar size instances, this variation presents a cost decrease of 46% when compared to the FTP. Finally, multi-objective variations of both problems, the Multi-objective Flying Tourist Problem (MO-FTP) and the Multi-objective Generalized Flying Tourist Problem (MO-GFTP), were characterized and formulated. Both present advantages to the FTP concerning the travel time. In fact, compared to the FTP, the MO-FTP and the MO-GFTP presented, for same sized instances, decreases of 52% and 80%, respectively, in the traveling time.
Keywords: Flight Search, Traveling Salesman Problem, Integer Linear Programming, Combinatorial Optimization, Multi-objective Combinatorial Optimization

1. Introduction

In the last decades, flying has become a common and affordable means of transportation around the globe. In fact, the International Air Transport Association (IATA) states that there were 3.8 billion air travelers in 2016 and predicts an increase to 8.2 billion by 2037 [1]. This also means that the variety of flight options is getting larger, and nowadays the traveler is often faced with multiple decisions when planning a trip. To respond to this variety of options, online travel agencies have surged in an endeavor to ease the process of buying flight tickets. With this in mind, several new opportunities to develop new products targeting different groups are surging.

To embrace one of those opportunities, the Flying Tourist Problem (FTP), an unconstrained multi-city problem, was defined as a special case of the well known Traveling Salesman Problem (TSP). More specifically, it is related to the Time-Dependent Traveling Salesman Problem variation. Considering a tourist wishing to visit n cities in no particular order, by plane, in a given time period, starting and ending at a given city, the FTP aims to minimize the cost or the traveling time. To the best of our knowledge, this problem is only addressed by one online travel agency, Kiwi [2]. Moreover, recently, R. Marques [3] suggested a web application to assess this problem. Nonetheless, both web services solve this problem in an incomplete manner and hence, do not guarantee the user the best possible solution. In accordance, this work aims to extend previous solutions to this particular problem, the FTP, (mainly based on stochastic methods) by exploiting other optimization strategies (complete methods).

Furthermore, other functionalities shall be offered to the end user (e.g., route solutions comprising only a subset of cities - generalized FTP).

With this work, we propose to solve the FTP to optimality, i.e to find the best solution for a certain user request. Therefore, an Integer Linear Programming ILP formulation for this problem shall be defined. We aim to integrate this formulation in a commercial solver. Next, upon understanding that the optimal solution might still not suit every

end user, we propose to formulate variations of the FTP. Three variations of the FTP should be hypothesized. These formulations aim end users that pretend, for instance, a cheaper trip. Finally, the work developed shall be integrated in a web application prototype.

The remaining of this article is structured as follows: Section 2 presents a review on the most relevant literature. Section 3 introduces a formal definition, as well as linear programming formulations, of the problems. This is followed by a brief review of the system prototype in Section 4. Section 5 presents the obtained results with this work and finally, Section 6 describes the main conclusions of this work, and addresses future work directions.

2. Literature Review

This work addresses Combinatorial Optimization Problems (COPs), more specifically variations of the classical formulation of the Traveling Salesman Problem (TSP) [4]. In graph theory, this problem is addressed as the search for the Hamiltonian Cycle[1] that minimizes the cost of the graph. Since determining the existence of an Hamiltonian Path is a NP-complete problem, then the TSP is also NP-complete [5].

In this work, a special variation of the TSP is revisited, the Time Dependent Traveling Salesman Problem (TDTSP) [6], due to the time dependency inherent to the problem herein presented. Other variations, such as the Generalized TSP (GTSP) [7] and the Multi-objective TSP (MO-TSP) [8], are also important to take into consideration when formulating the vaiations of the FTP.

Several optimization algorithms that have been proposed for the TSP can be grouped into complete, heuristic and metaheuristic approaches. In this work we focus on complete techniques, relying on Integer Linear Programming (ILP) formulations. During this work, the Dantzig-Fulkerson-Johnson Assymetric TSP ILP formulation [4] is given special attention. Nonetheless, since this formulation integrates Subtour[2] Elimination Constraints (SECs)}, which present an exponential growth with the number of nodes, we revisit polynomial size formulations for these constraints [9].

In the present work a commercial ILP solver is used, CPLEX [10]. This solver explores various methodologies. It is based on Branch and Cut (B&C) algorithms [11], which in turn are the integration of Branch and Bound (B&B) and cutting planes algorithms together.

[1] An Hamiltonian Path is a directed or undirected path that visits every vertex of a graph exactly once. As a cycle it repeats the first and last vertex.

[2] Subtours are solutions consisting of several disconnected tours.

3. Problem Formulation and Optimization Models

In this Section, four different ILP models that formalize the Flying Tourist Problem (FTP) and its considered variations are presented.

3.1. Flying Tourist Problem

The FTP formulates the problem of a tourist that desires to visit several cities by plane in a specified time-window. The main objective is to minimize the cost or traveling time of the tourist. The solution is then a set of flights that the tourist should buy in order to minimize one of these objectives. This set of flights is a Hamiltonian Cycle. Moreover, this problem is very close to the TDTSP, as the tourist is to specify the stop-time at each city.

Considering a tourist that wants to visit a set V of n cities connected by the set of arcs A, the FTP minimizes the weight function c_{ijk}, the weight of the arc connecting nodes i and j on time instance k, of a tour that starts on origin city 0, visits exactly once all n cities and finishes at arrival city $n+1$. The weight function is defined by the user itself and it can be the travel cost or the traveling time. The user is also the provider of the stop time s_i at each intermediate city i and the time horizon during which the travel may start $T_0 = [T_{0m}, T_{0M}]$. Even though in this work the stop time will always be considered as an integer, it may be a range of values instead. With T_0 and s_i, it is then possible to define a time window during which each city may be visited, denoted by TW. The set S containing all valid solutions is hereinafter defined and the purpose of this model is to find the global minimum, by considering the objective function of this set.

The constructed ILP model is hence composed by an objective function and a few constraint equations and inequations. The total weight of the tour, and thus the function aimed to be minimized, is:

$$\sum_{i=0}^{n+1} \sum_{j=0}^{n+1} \sum_{k=1}^{I_{ij}} c_{ijk} x_{ijk} \tag{1}$$

where x_{ijk} is a binary decision variable that carries the value 1 if flight k from city i to city j is taken, c_{ijk} is the cost value of flight from city i to city j using flight k and I_{ij} is the set of flight alternatives from city i to city j. In what concerns the constraints, they are divided in 4 different sections (degree, time, subtour elimination and integrality). Degree constraints ensure that each city is visited exactly once. With this in mind, there are out-degree and in-degree constraints. The former imply that there is exactly one flight leaving from each city, while the latter imply that there is exactly one

flight arriving to each city. Respectively:

$$\begin{aligned} \sum_{j=1}^{n+1}\sum_{k=1}^{I_{ij}} x_{ijk} = 1, \quad i = 0, 1, \ldots, n, \\ \sum_{i=0}^{n}\sum_{k=1}^{I_{ij}} x_{ijk} = 1, \quad j = 1, 2 \ldots, n+1, \end{aligned} \tag{2}$$

It is also important to bear in mind that there should not be any trip arriving at the origin city or departing from the arrival city. These are called the source and the sink respectively.

$$\begin{aligned} \sum_{i=1}^{n+1}\sum_{k=1}^{I_{ij}} x_{i0k} = 0, \\ \sum_{j=0}^{n}\sum_{k=1}^{I_{ij}} x_{(n+1)jk} = 0, \end{aligned} \tag{3}$$

Next, time constraints assure that the start time T_{start}, stop times s_i and end time T_{end} provided by the user are respected. The following notation is used: d_{ijk} and a_{ijk} are the departure date and time from city i to city j and the arrival date and time at city i from city j using flight k, respectively. For now, s_i is an integer. Nonetheless, in future work, it may be implemented as a time window. Constraints (4) (represented below) state that the departure from the origin city must not happen before T_{start}. On its counterpart, constraints (6) force that the arrival at the final destination city must not happen after T_{end}. It is important to note that since s_i are fixed and as T_{end} is defined by the latest T_{start} plus the sum of s_i then, the latest start date T_{start} is also indirectly forced. Lastly, constraints (5) impose that the departure and arrival from an intermediate city must be spaced s_i days.

$$\sum_{j=1}^{n+1}\sum_{k=1}^{I_{ij}} d_{0jk}x_{0jk} \geq T_{start}, \tag{4}$$

$$\sum_{j=1}^{n+1}\sum_{k=1}^{I_{ij}} d_{ijk}x_{ijk} - \sum_{j=0}^{n+1}\sum_{k=1}^{I_{ij}} a_{jik}x_{jik} = s_i, \quad i = 1, 2, \ldots, n, \tag{5}$$

$$\sum_{i=0}^{n+1}\sum_{k=1}^{I_{ij}} a_{i(n+1)k}x_{i(n+1)k} \leq T_{end}, \tag{6}$$

Assuming now that the user may provide a trip where the number of intermediate cities n is greater than 1, there is the need of eliminating any subtours that exist, as these are not part of the desired solution space. To guarantee a single tour, the model is given a set of subtour elimination constraints based on the polynomial formulations reported in Section 2:

$$\begin{aligned} u_i - u_j + (n+1)x_{ij} \leq n, \\ i, j = 1, 2, \ldots, n+1, \end{aligned} \tag{7}$$

where u_i is an integer decision variable that carries the non-negative value associated to the position of city i in the tour. Lastly, in order to ensure that all decision variables are binary, integrality constraints are applied:

$$\begin{aligned} x_{ijk} \in \{0, 1\} \quad i, j = 0, \ldots, n+1, \\ k = 1, 2, \ldots I_{ij}. \end{aligned} \tag{8}$$

Bearing all of the above, the complete model is presented in Figure 1(a).

3.2. Generalized Flying Tourist Problem

This problem is an adaptation of the FTP to the GTSP presented in Section 2. The user defines n groups of cities and wishes to visit each group exactly once. Considering an example in which the traveller wishes to visit three different cities in one trip, if these three cities are, for instance, {(London), (Barcelona), (Rome)} then the FTP model that was presented in the previous subsection will return the optimal solution for a specific start date window and stop time at each of the cities. On the other hand, if the main purpose of the traveller is, for example, to experience distinct cultural backgrounds, a few adaptations to improve the objective can be made as there are other cities where the user will face identical experiences. If we add cities to each of the groups in a way that we have a group of clusters in which at least one of them has more than one city, then the FTP model will no longer be able to return a feasible solution. In this case, consider the example {(London, Liverpool), (Barcelona), (Rome, Florence, Venice)}. This new problem has necessarily a better or at least equal optimal solution than the previous one, since the solution of the former is also a solution of the latter. On the other hand, the solution space is at least as large as the one of the FTP and is generally larger. With the purpose of enabling the user to postulate such problems, the GFTP is introduced.

Let $G = (V, A)$ be a graph where V is the set of nodes and A is the set of arcs connecting these nodes. The weight (usually the cost or distance) of each arc performed by a specific flight, k, is represented as c_{ijk}. Moreover, the set V is composed by the disjoint subsets $V_0, V_1, \ldots, V_{n+1}$ of cities. Then, the size of $\mid M \mid = m$ represents the total number of cities and $\mid V \mid = n$ the number of clusters. Since this GFTP formulation (as represented in Figure 1(b)) is very similar to the FTP model, only the

$$\min \quad \sum_{i=0}^{n+1}\sum_{j=0}^{n+1}\sum_{k=1}^{I_{ij}} c_{ijk}x_{ijk}$$

$$\text{s.t.} \quad \sum_{j=1}^{n+1}\sum_{k=1}^{I_{ij}} d_{0jk}x_{0jk} \geq T_{start},$$

$$\sum_{j=1}^{n+1}\sum_{k=1}^{I_{ij}} d_{ijk}x_{ijk} - \sum_{j=0}^{n+1}\sum_{k=1}^{I_{ij}} a_{jik}x_{jik} = s_i, \qquad i = 1, 2, \ldots, n$$

$$\sum_{i=0}^{n}\sum_{k=1}^{I_{ij}} a_{i(n+1)k}x_{i(n+1)k} \leq T_{end},$$

$$\sum_{j=1}^{n+1}\sum_{k=1}^{I_{ij}} x_{ijk} = 1, \qquad i = 0, 1, \ldots, n$$

$$\sum_{i=0}^{n}\sum_{k=1}^{I_{ij}} x_{ijk} = 1, \qquad j = 1, 2, \ldots, n+1$$

$$\sum_{i=1}^{n+1}\sum_{k=1}^{I_{ij}} x_{i0k} = 0,$$

$$\sum_{j=0}^{n}\sum_{k=1}^{I_{ij}} x_{(n+1)jk} = 0,$$

$$u_i - u_j + (n+1)\sum_{k=1}^{I_{ij}} x_{ijk} \leq n, \qquad i \neq j; \quad i, j = 1, 2, \ldots, n+1$$

$$x_{ijk} \in \{0, 1\}, \qquad i, j = 0, 1, \ldots, n+1; \quad k = 1, 2, \ldots, I_{ij}$$

$$u_i \geq 0. \qquad i = 1, 2, \ldots, n+1$$

(a) ILP formulation of the FTP

$$\min \quad \sum_{i\in V}\sum_{j\in V\setminus\{i\}}\sum_{k=1}^{I_{ij}} c_{ijk}x_{ijk}$$

$$\text{s.t.} \quad \sum_{j\in V\setminus V_0}\sum_{k=1}^{I_{ij}} d_{ijk}x_{ijk} \geq T_{start}, \qquad i \in V_0$$

$$\sum_{j\in V\setminus V_0}\sum_{k=1}^{I_{ij}} d_{ijk}x_{ijk} - \sum_{j\in V\setminus V_{n+1}}\sum_{k=1}^{I_{ji}} a_{jik}x_{jik} = s_i, \qquad i \in V \setminus \{V_0, V_{n+1}\}$$

$$\sum_{i\in V_p}\sum_{j\in V\setminus V_p}\sum_{k=1}^{I_{ij}} x_{ijk} = 1, \qquad p = 0, 1, \ldots, n,$$

$$\sum_{i\in V\setminus V_p}\sum_{j\in V_p}\sum_{k=1}^{I_{ij}} x_{ijk} = 1, \qquad p = 1, 2, \ldots, n+1,$$

$$\sum_{i\in V\setminus V_0}\sum_{k=1}^{I_{ij}} x_{ijk} = 0, \qquad j \in V_0,$$

$$\sum_{j\in V\setminus V_{n+1}}\sum_{k=1}^{I_{ij}} x_{ijk} = 0, \qquad i \in V_{n+1},$$

$$\sum_{i\in V_p}\sum_{j\in V_q}\sum_{k=1}^{I_{ij}}(-x_{ijk} + \sum_{r\in V\setminus\{V_0,V_p,V_q\}} x_{jr(a_{ijk}+s_j)}) \geq 0 \qquad p = 0, 1, \ldots, n, \quad q = 1, 2, \ldots, n,$$

$$w_{pq} = \sum_{i\in V_p}\sum_{j\in V_q}\sum_{k=1}^{I_{ij}} x_{ijk}, \qquad p \neq q; \quad p, q = 0, 1, \ldots, n+1,$$

$$u_p - u_q + (n+1)w_{pq} \leq n, \qquad p \neq q; \quad p, q = 1, 2, \ldots, n+1,$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall (i, j) \in A, \quad k = 1, 2, \ldots I_{ij},$$

$$u_p \geq 0, \qquad p = 1, 2, \ldots, n+1,$$

$$w_{pq} \in \{0, 1\}. \qquad p \neq q; \quad p, q = 0, 1, \ldots, n+1.$$

(b) ILP formulation of the GFTP

Figure 1: ILP formulations of the proposed problems.

different constraints are addressed here. It is important to mention that V_0 and V_{n+1} are usually of unit size as the user starts and finishes in a specific city.

Several different constraints should be taken into consideration in this model. Firstly, the constraints that already existed on the FTP model are now relative to clusters instead of cities. Then, a new auxiliary binary variable is needed and is described by the following constraints:

$$w_{pq} = \sum_{i \in V_p} \sum_{j \in V_q} \sum_{k=1}^{I_{ij}} x_{ijk}, \quad p \neq q, \qquad p, q = 0, 1, \ldots, n+1, \tag{9}$$

This variable, w_{pq} takes the value 1 if the traveler goes from cluster p to cluster q and the value 0 otherwise. Lastly, we created a new set of constraints that equalizes the inflow and outflow of each cluster as defined below:

$$\sum_{i \in V_p} \sum_{j \in V_q} \sum_{k=1}^{I_{ij}} (-x_{ijk} + \sum_{r \in V \setminus zV_{0,p,q}} x_{jr(a_{ijk}+s_j)}) \geq 0 \qquad p = 0, 1, \ldots, n, \quad q = 1, 2, \ldots, n, \tag{10}$$

3.3. Multi-objective Flying Tourist Problem

More often than not, the cheapest flights that are available usually imply several connections in their route, usually known as layovers. Hence, this low flight costs usually imply a consequent large traveling time, due to additional layovers. In this particular context, the problem that is herein formulated is an adaptation of the FTP to the MO-TSP, presented in Section 2 and focuses on minimizing both the cost of the trip and the implicit traveling time. This has inevitably a downside: the cost of the trip is generally higher and at most as good as the cost from the solution of the FTP, since now the time is also being minimized. On the other hand, the advantage of such model is that the traveling time is at most as large as the one from the FTP but generally lower. With the intent of giving the user the possibility of deciding on this trade-off, the MO-FTP is now introduced. Despite its similarities with the formulation depicted in Figure 1(a), its difference lays on the number of objective functions to be considered. In this problem, a set of 2 objectives will be taken in consideration: $F(x) = (f_1(x), f_2(x))$, where the overall objective is defined as min $F(x)$ and:

$$\begin{aligned} f_1(x) &= \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{I_{ij}} c_{ijk} x_{ijk} \\ f_2(x) &= \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{I_{ij}} t_{ijk} x_{ijk} \end{aligned} \tag{11}$$

In the context of the MO-FTP, the cost of the flights is represented as c_{ijk} and the traveling time is defined as t_{ijk}. In addition, the concept of Key Performance Indicator (KPI) is important to the understanding of the optimization process on this model. KPIs are performance measurements that evaluate the success of a certain activity. In this specific problem, the lower the KPIs of $f_1(x)$ and $f_2(x)$ are, the better, since this is a minimization problem. Defining different priorities determines the order in which KPIs are approached. If various KPIs have the same priority, then these are blended together and are treated as a single objective. Moreover, when this is the case, blending the objectives with different weights can be useful. In this particular case, if the priorities of $f_1(x)$ and $f_2(x)$ are the same and if we denote w_1 and w_2 as each respective weight, then the objective function $F(x)$ wished to be minimized will be $F(x) = w_1 f_1(x) + w_2 f_2(x)$.

3.4. Multi-objective Generalized Flying Tourist Problem

In Subsection 3.2, we claim that the GFTP has the advantage of returning lower optimal values when compared to the FTP. Moreover, in Subsection 3.3 we suggest that the MO-FTP has the advantage of improving the traveling time at the expense of increasing the cost of the trip. The model suggested in this subsection is an adaptation of the FTP to both the GTSP and the MO-TSP. By joining the previously presented models, the capacity of the generalized problem in achieving better optimal values can diminish, negate or even reverse the increase in the cost of the trip, caused by the multi-objective variation. With this objective in mind, the Multi-objective Generalized Flying Tourist Problem (MO-GFTP) is introduced.

By following the same methodology as in the last model, the MO-GFTP is based on Section 2, applied this time on the model depicted in Figure 1(b). In this problem, a set of 2 objectives will be taken in consideration: $F(x) = (f1(x), f2(x))$, where the overall objective is defined as min $F(x)$ and:

$$\begin{aligned} f_1(x) &= \sum_{i \in V} \sum_{j \in V \setminus \{i\}} \sum_{k=1}^{I_{ij}} c_{ijk} x_{ijk} \\ f_2(x) &= \sum_{i \in V} \sum_{j \in V \setminus \{i\}} \sum_{k=1}^{I_{ij}} t_{ijk} x_{ijk} \end{aligned} \tag{12}$$

Where, again, c_{ijk} represents the cost of the flights whilst t_{ijk} depicts the traveling time. Key Performance Indicators (KPIs) are also used to evaluate the these objectives.

Finally, the defined models shall now be used to find solutions for complex real world cases. For such purpose, Section 4 presents the implementation details of these formulations on a commercial solver and the integration of this optimization module in a web application system.

4. Prototype Implementation

The considered infrastructure consists in a web-based system that enables the end user to use the work developed during this work. The prototype system is composed of two different applications: the Client Side Application (CSA) and the Server Side Application (SSA).

4.1. Client Side Application

The web application was designed solely with the purpose of interacting with the end user in two different steps: First, the user defines the flight requests. Then, the CSA presents the final solutions to the end user. In this environment, the user defines the departure and arrival cities, the list of intermediate cities, the waiting periods associated to each city and the start time window of the trip. The user's input is processed and sent to the SSA. Later, the CSA receives a response from the SSA, containing the output information to be shown to the end user. The response contains at least one multi-city tour, containing hyperlinks where these can be bought, a map of the trajectories taken by each of the group of flights and some information such as duration, cost, departure/arrival dates of the flights and number of layovers.

4.2. Server Side Application

The SSA is divided in two different modules: the Data Management module and the Optimization module.

Data Management Module: The SSA receives the input information that was introduced by the user from the CSA. Next, it creates a graph with all the possible arcs linking the chosen cities, collects the data corresponding to all the flights connecting all the nodes during that time-window from Kiwi API [2] and creates the objective matrices. Through the study of this module it was possible to understand how it could still be improved. To overcome unnecessary requests, Figure 4.2 was taken into consideration. Consider different sets of arcs: initial, intermediate and final arcs that together form the solution. In the case of the FTP, initial arcs correspond to the arcs connecting node V_0 with the rest of the nodes during the start time window, $k \in T_0 = [T_{0m}, T_{0M}]$. The final arcs are the arcs connecting the arrival city, node V_{n+1} during the end time window. This end time window starts on the day which corresponds to the sum of stop times with the first possible start day and ends on the day that corresponds to the sum of stop times with the last possible start day, hence $k \in [T_{fm}, T_{fM}]$ where $T_{fm} = T_{0m} + \sum(s_i)$ and $T_{fM} = T_{0M} + \sum(s_i)$. Finally, intermediate arcs connect every intermediate node $V_1, \ldots, V_n$ from the day corresponding to the minimum stop time of every city summed with the first possible start day, until the day corresponding to the minimum stop time of every city subtracted from the last possible end day. This means that these arcs occur from $t_1 = T_{0m} + min(s_i)$ to $t_2 = T_{fM} - min(s_i)$.

In the GFTP, initial arcs are the same as in the FTP. On the other hand, the end time window starts in the sum of the minimum stop times at each cluster with the first possible start day and ends in the sum of the maximum stop times at each cluster with the last possible start day. This is: $T_{fm} = T_{0m} + \sum_{p=1}^{n} min(s_i, i \in V_p)$ and $T_{fM} = T_{0M} + \sum_{p=1}^{n} max(s_i, i \in V_p)$ respectively. Lastly, intermediate arcs are the same as in the FTP.

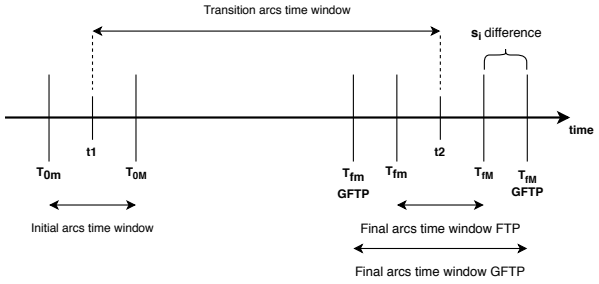


Figure 2: Timeframe of initial, intermediate and final arcs of both the FTP and the GFTP.

Optimization Module: The system is built upon a modular approach, so that different optimization techniques can be used. The Multi-city flight request is illustrated in Figure 3. If a FTP request is received, the problem is first solved using a Random Search, then utilizing the Nearest Neighbour, proceeding to one of the Metaheuristic methods, either Ant Colony Optimization or Simulated Annealing, and finally utilizing the models presented in Section 3. It is important to notice that both Metaheuristic approaches are dependent on the solution derived from the Nearest Neighbour algorithm. The CPLEX solver was used without loss of generalization, as there are diverse solvers[3] able to solve the proposed models.

This solver was integrated through the Python framework, IBM Decision Optimization CPLEX

[3]E.g. Gurobi, Lpsolve, Glpk, Scip

Modeling for Python, also known as DOcplex. This is an object oriented Application Programming Interface and integrated library that offers mathematical programming modeling.

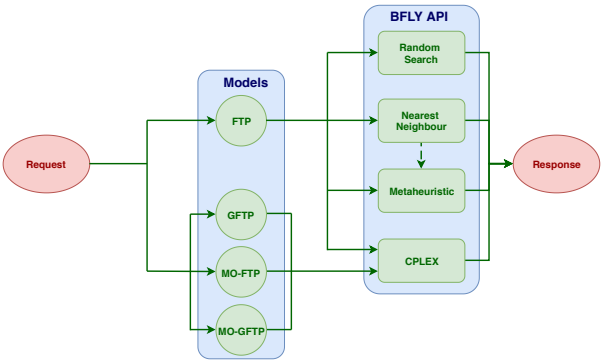


Figure 3: Simplified optimization system. Depending on the type of problem, different optimization systems are used. The FTP is solved using 4 different methods while the remaining problems are solved only by CPLEX.

5. Evaluation and Experimental Results

In this section some experimental results regarding the optimization models described in Section 3 are presented. For this evaluation, 50 different cities were considered as intermediate nodes and Lisbon was considered as the departure and arrival city in all considered trips.

On the 1st of August of 2019, the prototype described in Chapter 4 was used to construct a matrix containing the cheapest flights between these cities for each day during a period of 65 days. This period starts on the 1st of October of 2019 and ends on the 5th of December of 2019. The aforementioned analysis does not take into account the time for the data acquisition since we want to evaluate the optimization module. For this reason, this analysis considers the hypothesis where the prototype has a preemptive database with all necessary flights. Moreover, the evaluation does not take into account any special event or holiday, even though these could have impact on the results. All the results were obtained on a 2.3 GHz AMD Opteron(TM) 6276 Processor and the code was developed using Python programming language.

5.1. Flying Tourist Problem Evaluation

Having the full weight matrix properly filled in, it is possible to evaluate the performance of the developed FTP model. For such purpose, a script choosing random requests was created and respecting the following rules: the requested trip must start and end in Lisbon, the tourist wishes to visit between two and ten cities, at each city, the user wishes a stop time between two and five days and finally, the trip must start during the first fifteen days of October. This script was used to create 9000 different requests with a uniform distribution of cities to be optimized relative to the total flight cost. All the instances relative to the FTP achieved optimality, i.e the solution that was found is the global optimum of each specific problem. Since this problem was previously approached in [12, 3], we solved the same instance set with the SA algorithm described in both of these works and compared the results of the two optimization techniques. The parameters evaluated were the total cost of flights, the elapsed computation time and the elapsed wall clock time. Table 1 illustrates the entire set of results with both optimization techniques.

	Method	**Entire Set**
Cost per Flight in €	***SA***	62.37
	FTP ILP	60.27 (-3.35%)
Elapsed CPU time [s]	***SA***	2.46
	FTP ILP	8.23 (+2.35 times)
Elapsed Wall time [s]	***SA***	2.46
	FTP ILP	0.70 (-71.80%)

Table 1: Comparison between the SA algorithm used by R.Marques [3] and the FTP ILP formulation (solved with CPLEX) implemented in this work. The results indicate the median of the mean cost, elapsed CPU time and elapsed Wall time of each request according to the experiments described throughout this section.

All the results in the table represent the median value of the mean of each parameter. Three parameters are compared since we want to evaluate the response in two ways. First, the main objective of each approach should be analyzed in order to understand if there are significant improvements in using a complete method in this problem. Secondly, the feasibility of applying these methods in a real-time web application needs to be assessed. For such purpose, both the Elapsed CPU time and the Elapsed Wall time are evaluated. It is possible to observe that the achieved improvements in the cost come at the expense of a large increase in the Elapsed CPU time. However, since CPLEX makes use of coarse grained parallel computing, the real optimization time in this machine largely decreases in normal conditions, outperforming the SA algorithm in most cases of requests with less than seven cities.

These results emphasize that the SA algorithm is faster in achieving a good solution but cannot, in most cases, find the optimal solution. On the other hand, using CPLEX, optimality was achieved at all cases. Furthermore, considering the parallel computing capability of CPLEX, one concludes

that this method provides great advantages. With this in mind, the idea of using the result from the SA algorithm to build an additional constraint for the ILP formulation, arised. The new constraint is hence written as:

$$c_{sa}(x) = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{I_{ij}} c_{ijk} x_{ijk} \leq SA_{solution} \quad (13)$$

This constraint will act an aditional cut, during the B&C algorithm, and has the purpose of accelerating the process. The whole data set of 9000 requests was analyzed. With this new ILP formulation, all requests were solved to optimality. Nevertheless, this new formulation has the disadvantage of requiring the SA solution beforehand.

It is concluded that, by using the SA solution as an additional constraint to the ILP formulation, the computational time required to solve the requests to optimality can be greatly reduced. More specifically, the median of the global CPU time was enhanced in 70.80%. With this change, instead of an increase of 2.35 times in the CPU time (vs the SA algorithm), the FTP presents an increase of only 19.35%. However, because of the non-parallel implementation of the SA algorithm, the wall clock time only presents improvements in requests of more than 8 cities. Hence, considering the fact that the prototype presented in Chapter 4 is able to progressively update the responses to the end user, it is concluded that a good solution for large requests is presenting the SA solution first. Then, use it as a constraint to the ILP formulation to search for optimality to retrieve the final and optimal response.

Next, the results of the remaining variations of this model are presented, commented and compared with the results shown in this section.

5.2. Generalized Flying Tourist Problem Evaluation

Similarly to what was described in Section 5.1, a script to construct 9000 random requests for the analysis of this model was also implemented. The script respects the following rules: the requested trip must start and end in Lisbon, the tourist wishes to visit between two and nine clusters, each containing between two and five cities, at each city, the user wishes a stop time between two and five days and finally, the trip must start during the first fifteen days of October.

An additional restriction was considered by limiting the number of visited cities to 9, since at this point some requests to visit 9 clusters could not be solved in less than CPU seconds. More specifically, in 11.26% of the requests with 9 clusters, no solution was found in this time span. This corresponds to 1.32% of the total data set. Moreover, some other requests were also not solved to optimality in this time span, as seen in Table 2. All the requests not shown in the table, with less than 6 clusters, were solved to optimality in all cases.

In Section 3 it was postulated that the GFTP would necessarily have a better or at least equal optimal solution when compared to the FTP. Figure 4(a) indicates that this is verified. Moreover, it illustrates that the gap of the objective increases with the number of visited cities, which might imply that this model is better suited for users wishing to visit a large quantity of cities. Not only is it possible to observe that the cost decreased as expected, but the traveling time also increased accordingly.

Nevertheless, this gain occurs at the expense of having a solution space generally larger than that of the FTP. For this reason, requesting large quantities of cities to be visited might lead to a computationally heavy request. Figure 4(c) depicts that the exponential growth of the CPU time associated to this new model has a larger growth ratio than that of the FTP ILP formulation.

5.3. Multi-Objective Flying Tourist Problem Evaluation

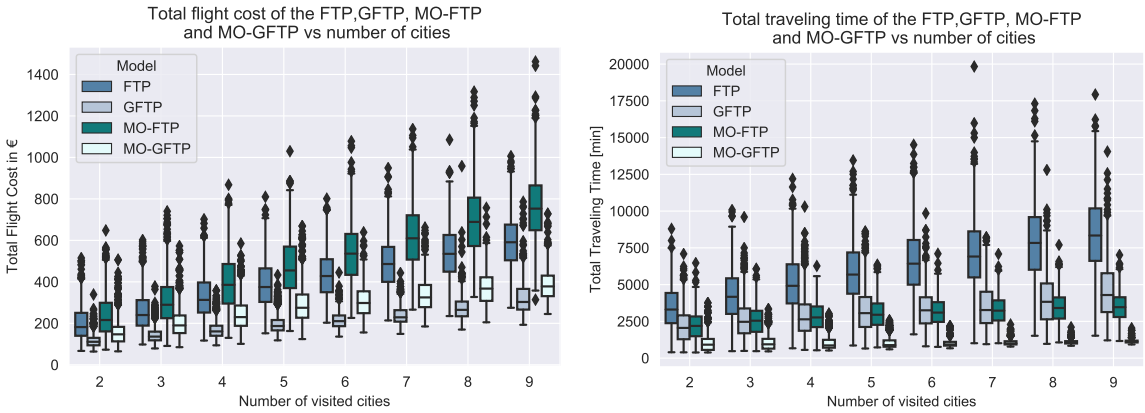
This formulation was analyzed using the same request method as in Section 5.1.

CPLEX solver, has two options to approach the multi-objective optimization. The first option is a lexicographic multi-objective optimization, where some objective functions are incomparably more important than others and so, this optimization is processed in a sequential level of importance. The second option reduces the multi-objective problem to a single-objective optimization through the merge of both objective functions with the associated weights. Since the former implies a substantially greater optimization effort, the computational time would substantially increase as two single-objective optimizations are performed. For this reason, only the latter method is used, both in this section as well as in 5.4.

Consider a user that does not prioritize any of the objectives, i.e considers the cost and the traveling time to be of the same importance when deciding on a trip. The weights of the objective functions were not normalized since the trade-off between the cost and the traveling time is strictly subjective to each user's personal preference. Nonetheless, since the

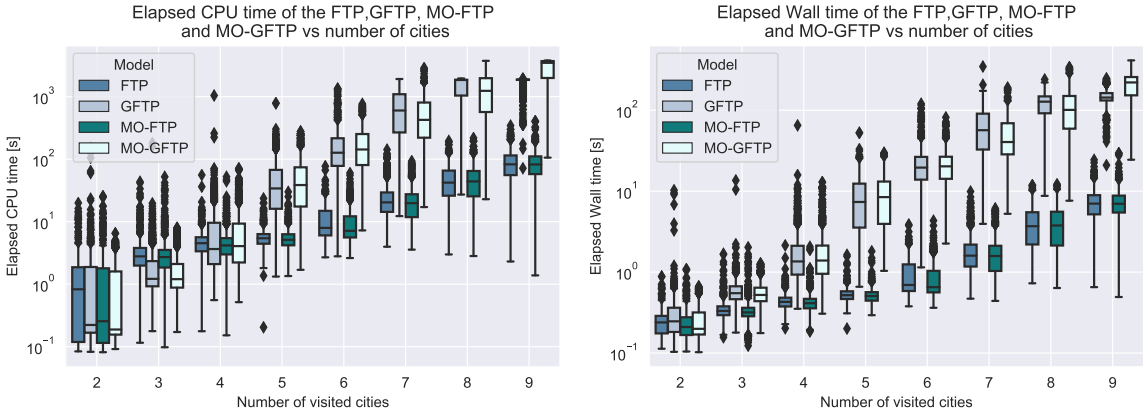
number of clusters	**7**	**8**	**9**
requests solved	100%	96.80%	83.98%
requests solved to optimality	90.38%	42.77%	12.65%

Table 2: Requests of the GFTP that did not achieve optimality in less than CPU seconds.



(a) Total cost of flight increase with number of visited cities.

(b) Total traveling time increase with number of visited cities.



(c) Total optimization elapsed CPU time increase with number of visited cities.

(d) Total optimization wall CPU time increase with number of visited cities.

Figure 4: Evaluation and comparison of the results obtained with the formulations of the FTP, GFTP, MO-FTP and MO-GFTP.

main objective of this formulation is to present the improvements in terms of the traveling time when compared to the FTP, the used weights were both set to the value 1. However, since the magnitude of the traveling time of a flight (in minutes) is usually larger than its cost (in euros), the traveling time presents as a larger term in the single-objective function.

Figures 4(a) and 4(b) illustrate the comparison of the results between the FTP and the MO-FTP, when considering both the cost and the traveling time objectives to have the same weight on the latter formulation.

In section 3, it was assumed that the MO-FTP would certainly achieve a better (or at least equal) traveling time, when compared to the FTP. However, this improvement is achieved at the expense of penalizing the flight cost. Figure 4(a) verifies these assumptions, and the average increase of 33.59% in cost results, on average, on the reduction in 44.74% of the traveling time. Furthermore, since this optimization treated both objective functions as a singular objective function, then the computational time remains, approximately, the same as in the FTP, as clarified in Figure 4(c).

5.4. Generalized Multi-Objective Flying Tourist Problem Evaluation

This formulation was analyzed using the same request method as in Section 5.2. Moreover, considering both the cost and the traveling time objective functions to be of the same importance, they are given the same priority and hence, merged together.

Figures 4(a) and 4(b) depicts the comparison of the MO-GFTP with both the MO-FTP and the GFTP when considering both objective functions to have the same weight. In Section 3, we theorized that the MO-GFTP would attain the advantages of both the MO-FTO and the GFTP, i.e achieve better traveling times through the multi-objective optimization, although not penalizing the flight cost as much by introducing more arcs as a generalized, or set, problem. Furthermore, the addition of more arcs, by using the MO-GFTP, allowed even

better traveling times. However, the results concerning the MO-GFTP take longer to process, since this is heavier computational problem. Considering a timeframe of 3600 CPU seconds, a solution for 2.48% of the requests was not found and 5.08% of the results are not proved to be optimal.

6. Conclusions

The increase on flights over the last decade has posed several new challenges. A vast variety of online travel agencies have started to surge with the attempt of making flight choices easier for the users in many distinct ways. Throughout this work, we developed different models that shape peculiar situations we believe to be of the user's interest. Nonetheless, when accounting for complex problems, for instance an unconstrained multi-city search, these search engines aim to present solutions in a small time frame. Notwithstanding, this implies the use of incomplete methods and therefore there is no guarantee that these solutions are optimal. With this in mind, the main contributions of this work were the formulations of complete methods to assure the quality of four different scenarios: the FTP, GFTP, MO-FTP and MO-GFTP.

After the problem statements, Integer Linear Programming (ILP) formulations for each of the problems were proposed and implemented using a commercial solver, CPLEX, as an optimization module. This module was later integrated in a web application previously developed by R.Marques [12, 3] in order to solve real world complex routing problems. This application was subject to several design modifications to adapt it to the new proposed models.

The results of the ILP formulation of the FTP were compared to an incomplete method, the SA algorithm and outperformed it, in what concerns the solution, since it consistently found an optimal solution in less than 1800 seconds for a maximum of ten cities. Moreover, even though the ILP formulation involved larger computational optimization times, this method outperformed the SA algorithm in what concerns the wall clock optimization time for less than eight cities. Comparing the other formulations with the obtained results of the FTP, we concluded that the GFTP managed to consistently reduce the total flight cost in half, the MO-FTP achieved a large reduction in the traveling time at the expense of the total flight cost and finally, the MO-GFTP attained both benefits.

Considering that the work developed proposes to resolve complex routing problems in near real-time, there is the need to consistently improve and find innovative techniques for the foregoing system. Regarding the multi-objective formulations, we recommend exploring the multi-objective evolutionary algorithms in order to perceive the quality of the solutions here presented. With this in mind we suggest using framework MOEA/D and the NSGA II Algorithm.

References

[1] International Air Transport Association Annual Review 2019. 75th Annual General Meeting, Seoul, June 2019. Accessed October 2019.

[2] Kiwi.com API documentation. https://docs.kiwi.com/. Accessed October 2019.

[3] Nuno Roma Rafael Marques, Luís M. S. Russo. Flying tourist problem: Flight time and cost minimization in complex routes. *Expert Systems with Applications*, 130:172 – 187, September 2019.

[4] George B. Dantzig, Delbert Ray Fulkerson, and Selmer M. Johnson. Solution of a large-scale traveling-salesman problem. In *50 Years of Integer Programming*, 1954.

[5] Alexander Schrijver. *Algorithms and Combinatorics*. Springer Science+Business Media, 2008.

[6] Jean Claude Picard and Maurice Queyranne. Time-Dependent Traveling Salesman Problem and Its Application To the Tardiness Problem in One-Machine Scheduling. *Operations Research*, 26(1):86–110, 1978.

[7] Charles E. Noon and James C. Bean. A Lagrangian Based Approach for the Asymmetric Generalized Traveling Salesman Problem. *Operations Research*, 39(4):623–632, 1991.

[8] E. L. Ulungu and J. Teghem. Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3(2):83–104, 1994.

[9] C. E. Miller, A. W. Tucker, and R. A Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, 7:326–329, 1960.

[10] Copyright © 2019 ILOG IBM CPLEX. https://www.ibm.com/analytics/cplex-optimizer. Accessed October 2019.

[11] E. Mitchell John. Integer programming: Branch and cut algorithms. In *Encyclopedia of Optimization 2nd Edition*, pages 1643–1649. Springer, 2009.

[12] Rafael Marques. Flight Time and Cost Minimization in Complex Routes. Master's thesis, Universidade de Lisboa, Instituto Superior Técnico, October 2018.