Solving the family traveling salesman problem

Raquel Bernardino, Ana Paias

PII: S0377-2217(17)31074-3
DOI: 10.1016/j.ejor.2017.11.063
Reference: EOR 14851

To appear in: *European Journal of Operational Research*

Received date: 26 April 2017
Revised date: 17 October 2017
Accepted date: 28 November 2017

Please cite this article as: Raquel Bernardino, Ana Paias, Solving the family traveling salesman problem, *European Journal of Operational Research* (2017), doi: 10.1016/j.ejor.2017.11.063

Highlights

- We propose exact and heuristic procedures for the family traveling salesman problem.
- Theoretical and practical comparison of compact and non-compact models is provided.
- We solve efficiently benchmark instances with up to 127 nodes.
- Improved upper bounds were provided for the higher dimensioned benchmark instances.

Solving the family traveling salesman problem

Raquel Bernardino[1,2,3], Ana Paias[1,2,4]

[1] DEIO, Faculdade de Ciências, Universidade de Lisboa,
C6, Piso 4, 1749-016 Lisboa, Portugal.

[2] Centro de Matemática, Aplicações Fundamentais e
Investigação Operacional, Faculdade de Ciências,
Universidade de Lisboa.

[3] E-mail: raquelbernardino@live.com.pt

[4] E-mail: ampaias@fc.ul.pt

December 5, 2017

Abstract

In this paper we address the family traveling salesman problem (FTSP), an NP-hard problem in which the set of nodes of a graph is partitioned into several subsets, which are called families. The objective is to visit a predefined number of nodes in each family at a minimum cost. We present several compact and non-compact models for the FTSP. Computational experiments with benchmark instances show that the non-compact models outperform the compact ones. One of the non-compact models is able to solve instances with 127 nodes, in less than 70 seconds, and one of the instances with 280 nodes in 3615 seconds. The optimal values of these instances were not known. For the higher dimensioned instances, the ones whose optimal value remains unknown, we propose an iterated local search algorithm that is able to improve the best known upper bounds from the literature.

Keywords: Combinatorial optimization, traveling salesman problem, multicommodity flows, branch-and-cut, metaheuristics.

1 Introduction

In this work we address the family traveling salesman problem (FTSP), which is a variant of the traveling salesman problem (TSP). Given a depot and a set of cities, in the TSP the traveling salesman must find a minimum cost route that visits all the cities, whereas in the FTSP the traveling salesman must also find a minimum cost route but is only required to visit a predefined number of cities. Hence, in the FTSP we have an additional level of decision which consists of choosing the cities to be visited.

More formally, in the FTSP the set of cities is partitioned into several subsets which are called families. The cost of traveling between each pair of cities and between the depot and each city is known. The objective is to determine a minimum cost route that: i) begins and ends at the depot; and ii) visits a given number of cities in each family.

The FTSP was introduced by Morán-Mirabal et al. (2014) and, as far as we know, this work is the only one that addresses this problem. Morán-Mirabal et al. (2014) motivated the FTSP by the order picking problem in warehouses where products of the same type are stored in different warehouses or in different places in the same warehouse. Note that due to technological advances, it is possible to locate a product very easily and thus there is no need to store products from the same type in the same place. If we consider that each product is a family and

the number of family members that we wish to visit is the demand of the product associated with that family, then the order picking problem in warehouses may be seen as the FTSP.

The FTSP may be modeled by using a complete directed graph $G = (\{0\} \cup N, A)$, where 0 represents the depot and N is the set of nodes, previously called cities, that is partitioned into the several families.We will refer to the singleton subset $\{0\}$ as 0 to simplify the notation further on. The cost of using arc $(i, j) \in A$ is denoted by c_{ij}. There are L disjoint families, represented by F_l, with $l = 1, \ldots, L$, such that $N = \cup_{l \in L} F_l$. The number of members in family l is n_l and $\sum_{l=1}^{L} n_l = |N|$. We consider, without loss of generality, that the nodes that belong to family 1 are nodes 1, 2, ..., n_1, nodes that belong to family 2 are $n_1 + 1$, ..., $n_1 + n_2$, etc. In each family we are required to visit v_l nodes and the total number of visits that we are required to make is defined as $V = \sum_{l=1}^{L} v_l$. We say that family l is *complete* if there are v_l nodes from family l in the route. Clearly, in order to have a feasible solution all families must be complete. Figure 1 shows a feasible solution for an FTSP instance with two families. Family 1, which is represented with the light gray color, has two family members (nodes 1 and 2), and family 2, which is represented with the dark gray color, has three (nodes 3, 4 and 5), and we are required to visit one node from family 1 and two nodes from family 2.

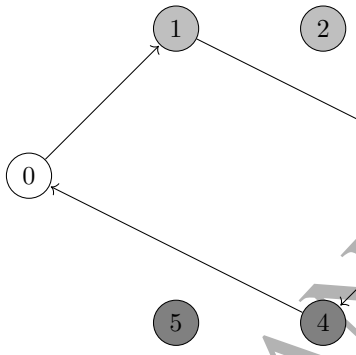


Figure 1: An example of a feasible solution for an FTSP instance

Assuming that the traveling salesman wants to visit every node from every family, the feasible solutions for the FTSP will be hamiltonian cycles. Hence, the TSP is a particular case of the FTSP and, for that reason, we can conclude that the FTSP is NP-hard. The FTSP was created as being an extension of the generalized traveling salesman problem (GTSP). In the GTSP the set of nodes is partitioned into clusters (families) and one wants to find the route with minimum cost that visits each cluster (family) at least once and every node no more than once (see, e.g., Srivastava et al., 1969; Gutin and Punnen, 2002; Pop, Petrica C, 2007). If in the FTSP one only wants to visit one node per family, that is, $v_l = 1$, with $l = 1, \ldots, L$, we will obtain a particular case of the GTSP which is called equality GTSP (see, e.g., Srivastava et al., 1969; Cacchiani et al., 2011).

The FTSP may also be seen as a variant of the generalized covering salesman problem (GCSP), that was presented by Golden et al. (2012). In the GCSP each node $i \in N$ can cover a subset of nodes D_i and it has a predefined covering demand of k_i. The objective of the GCSP is to determine a route in which each node i is covered at least k_i times by the nodes in the route. Let us consider that we have $|N|$ families. Family i, which is associated with node i, will have as family members nodes j such that $i \in D_j$ and v_i is equal to k_i. Since in one of the variants of the GCSP it is possible to visit the same node more than once, the feasible solutions for the FTSP presented previously will also be feasible for the GCSP.

Another problem that can be transformed into the FTSP is the capacitated traveling purchaser problem (CTPP). In the CTPP (see, e.g., Boctor et al., 2003) one wishes to purchase several copies of items that belong to a list and each item is available in a subset of markets (nodes). Consider that we have L distinct items, item l is available in the markets (nodes) that belong to the set F_l and one wishes to purchase v_l units of product l. Assuming that each market only sells one unit of product and that the cost of each item is the same in every market that sells it, solving this CTPP is equivalent to solving an FTSP.

As it was mentioned above the FTSP is not widely studied. Morán-Mirabal et al. (2014) proposed two meta-heuristics, a biased random key genetic algorithm and a hybridization of a GRASP method with an evolutionary path-relinking procedure, and presented an integer linear programming (ILP) model. The ILP model is similar to the one proposed by Dantzig et al. (1954) for the TSP with an additional set of constraints to ensure that v_l nodes are visited in each family l, with $l = 1, \ldots, L$. With this ILP model, Morán-Mirabal et al. (2014) were able to solve up to optimality nine out of 21 benchmark instances of the FTSP, which they created by adapting TSPLIB instances, and provided the only known upper bounds for the remaining benchmark instances with their heuristics.

Even though there is not much literature related with the FTSP, this problem seems to be a natural extension of problems that have a wide variety of applications (see, e.g., Laporte et al., 1996), hence the importance of studying it. In this paper we present compact and non-compact models to solve the existing benchmark instances, and an iterated local search (ILS) algorithm to provide upper bounds for the instances that the exact methods are not able to solve.

In Section 2 we present the several models and the branch-and-cut procedure developed to solve the non-compact models. Section 3 is devoted to the ILS algorithm. In Section 4 the computational experiment is presented and, finally, in Section 5 we draw the main conclusions from this work.

2 Modeling the family traveling salesman problem

In this section we present several models for the FTSP. These models only differ on how the subtours, which are routes that do not involve the depot, are eliminated. Therefore, we will start by presenting a generic model where the subtour elimination constraints are modeled in an implicit way and, afterwards, we will present the different ways of modeling them explicitly. We will conclude this section with a theoretical comparison between the models.

2.1 A generic model

Let x_{ij} be a binary variable stating whether the arc $(i,j) \in A$ is used in the route or not. Let us also define another binary variable y_i that indicates whether the node $i \in N$ is visited or not. The FTSP can be formulated by using the following ILP model:

$$\textit{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

Subject to:

$$\sum_{j\in N} x_{0j} = 1 \tag{2}$$

$$\sum_{j\in 0\cup N} x_{ij} = y_i, \qquad \forall i \in N \tag{3}$$

$$\sum_{j\in 0\cup N} x_{ji} - \sum_{j\in 0\cup N} x_{ij} = 0, \qquad \forall i \in 0\cup N \tag{4}$$

$$\sum_{i\in F_l} y_i = v_l, \qquad \forall l = 1,\ldots,L \tag{5}$$

$$\{(i,j)\in A : x_{ij} = 1\} \textit{ does not contain subtours} \tag{6}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{7}$$

$$y_i \in \{0,1\}, \qquad \forall i \in N \tag{8}$$

The objective of the FTSP is to minimize the route cost which is represented in (1). Constraint (2) ensures that one and only one arc leaves the depot. The set of constraints (3) are the linking constraints between the x and the y variables and guarantee that if node i is visited, then there has to be an arc leaving node i. These constraints also show that variables y are auxiliary, since we could formulate the problem using only the x variables. Constraints (4) ensure that the in-degree and the out-degree of a node are equal. Constraints (5) make sure that we visit the number of nodes that we are supposed to in each family. Constraints (7) and (8) define the variables' domain.

A solution that satisfies the equation system (2)–(5), (7)–(8) will visit the required number of nodes per family and guarantee that the in-degree and out-degree of every node that belongs to the route are the same. However, this solution may contain subtours, which is equivalent to saying that the solution may not be a single connected route. Constraints (6), which are written in a generic way, are supposed to prevent that. Throughout the next section we will present several ways of modeling the set of constraints (6) in order to obtain several valid formulations for the FTSP. Before doing so let us introduce some notation:

$$x(S_1,S_2) = \sum_{i\in S_1, j\in S_2} x_{ij}$$

2.2 Modeling subtour elimination

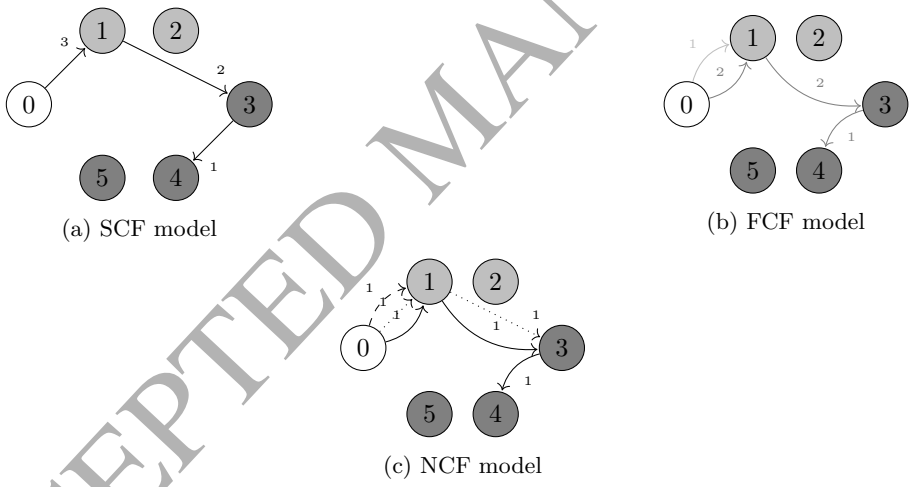
We have developed compact and non-compact models. The compact models use flow variables to ensure that the solutions obtained are a single connected route while the non-compact models are based on cut inequalities. Firstly, we will present the compact models and then the non-compact ones.

2.2.1 Compact models

Some of the proposed models with flow variables for the FTSP are similar to the ones presented by Gavish and Graves (1978) and Wong (1980) for the TSP. The main difference is related with the fact that in the TSP the traveling salesman must visit all the nodes.

Figure 2 shows a graphical representation of the flow systems associated with the flow models to be presented next, using as example the feasible solution for the FTSP instance presented in figure 1. We created three different flow models: a single-commodity flow model (SCF) where we send one single flow from the depot with V units (see figure 2a); a family-commodity flow model (FCF) where, as the name suggests, we send L different flows with v_l units each, from the depot to each one of the families (see figure 2b); and a node-commodity flow model (NCF) where we send V different flows, with one unit, from the depot to each one of the nodes that will be visited (see figure 2c). Models SCF and NCF for the FTSP are a straightforward adaptation of the SCF and the multi-commodity flow models for the TSP, respectively, and may be applied, with slight modifications, to several routing problems, while the FCF model is specific for the FTSP.

Figure 2 highlights the differences between the several models introduced in the previous paragraph in terms of the number of different flows and the amount of flow. Figures 2b and 2c have different arcs to represent the several flows associated with the corresponding models. The values above each arc represent the amount of flow that traverses that arc. Observing the nodes that are not the depot it is possible to verify how the flow conservation works for each model. The mathematical formulation for these models will be presented in the following subsections.



(a) SCF model

(b) FCF model

(c) NCF model

Figure 2: Representations of the several flow systems

2.2.1.1 The single-commodity flow model

In order to model the single-commodity flow let us define variables f_{ij} that represent the amount of flow that traverses arc $(i,j) \in A$, which corresponds to the number of nodes that will still be visited in the route. These variables are non-negative since their integrality is ensured by the other constraints of the model. If we replace generic constraints (6) by constraints

$$\sum_{j \in N} f_{0j} = V \tag{9}$$

$$\sum_{j \in 0 \cup N} f_{ji} = \sum_{j \in 0 \cup N} f_{ij} + y_i, \qquad \forall i \in N \tag{10}$$

$$f_{ij} \leq V x_{ij}, \qquad \forall (i,j) \in A \tag{11}$$

$$f_{ij} \geq 0, \qquad \forall (i,j) \in A \tag{12}$$

we will obtain a valid model for the FTSP denoted by SCF. Constraint (9) ensures that one flow with V units leaves the depot. The set of constraints (10) are the flow conservation constraints. These constraints are similar to the ones presented in Gavish and Graves (1978) although, since we do not wish to visit every node, we will only leave one unit of flow in the nodes that are visited, which is indicated by the value of the variable y (see figure 2a). Finally, in (11) we have the relation between the x and the f variables which states that flow can only traverse an arc that is chosen to be in the solution.

The flow system (9)–(11) ensures that the solution obtained is a single connected route since V units of flow leave the depot and are sent through a path that goes by each one of the visited nodes. If there were to be a subtour with visited nodes, then there would be no path between the depot and the nodes on that subtour.

2.2.1.2 The family-commodity flow model

To obtain the FCF model we need to disaggregate the f variables presented in the SCF model by family. Let t^l_{ij} be the amount of flow for family $l \in \{1, \ldots, L\}$ that traverses arc $(i, j) \in A$, which is the same as the number of nodes from family $l \in \{1, \ldots, L\}$ that will still be visited in the route when we traverse arc $(i, j) \in A$. Once again, these variables are non-negative since their integrality is guaranteed by the other constraints present in the model. A valid model for the FTSP is obtained if we replace the set (6) in the generic model with the following constraints:

$$\sum_{j \in N} t^l_{0j} = v_l, \qquad \forall l \in \{1, \ldots, L\} \tag{13}$$

$$\sum_{j \in 0 \cup N} t^l_{ji} = \sum_{j \in 0 \cup N} t^l_{ij} + y_i, \qquad \forall i \in F_l \qquad \forall l \in \{1, \ldots, L\} \tag{14}$$

$$\sum_{j \in 0 \cup N} t^l_{ji} = \sum_{j \in 0 \cup N} t^l_{ij}, \qquad \forall i \in N \backslash F_l \qquad \forall l \in \{1, \ldots, L\} \tag{15}$$

$$t^l_{ij} \leq v_l x_{ij}, \qquad \forall (i, j) \in A \qquad \forall l \in \{1, \ldots, L\} \tag{16}$$

$$t^l_{ij} \geq 0. \qquad \forall (i, j) \in A \qquad \forall l \in \{1, \ldots, L\} \tag{17}$$

Constraints (13) ensure that we have L different flows leaving the depot each one with v_l units. Constraints (14) and (15) are the flow conservation constraints which are divided in two cases: either the node belongs to the same family as the flow variable or it does not. In the former, constraints (14) are similar to constraints (10) presented in the SCF model. In the latter, the amount of flow that enters and leaves a node remains the same (see figure 2b). Constraints (16), which model the relation between the x and the t variables, guarantee that flow can only traverse an arc that was chosen to be in the solution.

2.2.1.3 The node-commodity flow model

The NCF model is obtained by disaggregating variables f per node. Let z^k_{ij} be a binary variable that has value 1 if arc $(i, j) \in A$ is used to send one unit of flow from the depot to $k \in N$ and value 0 otherwise. Even though variables z are binary, we only need to add non-negativity constraints since their integrality and bounds are ensured by the other constraints of the model. In order to obtain a valid model for the FTSP we must substitute (6) in the generic model by the following constraints:

$$\sum_{j \in N} z^k_{0j} = y_k, \qquad \forall k \in N \tag{18}$$

$$\sum_{j \in N} z^k_{jk} = y_k, \qquad \forall k \in N \tag{19}$$

$$\sum_{j \in 0 \cup N} z^k_{ji} = \sum_{j \in N} z^k_{ij}, \qquad \forall i, k \in N : i \neq k \tag{20}$$

$$z^k_{ij} \leq x_{ij}, \qquad \forall (i, j) \in A, k \in N \tag{21}$$

$$z^k_{ij} \geq 0. \qquad \forall (i, j) \in A, k \in N \tag{22}$$

The first three sets of constraints are similar to the ones presented by Wong (1980) for the TSP. The main difference is that, in constraints (18) and (19), we will only use arcs to send flow from the depot to a node if that node is visited. Constraints (20) are the flow conservation constraints for all the nodes that are not the destination

node of the flow (see figure 2c). Constraints (21), which represent the relation between the z and the x variables, guarantee, once again, that flow can only be sent through an arc if that arc was chosen to be in the route.

This flow system prevents subtours because it ensures that there exists a path from the depot (node 0) to every node $k \in N$ such that $y_k = 1$. Therefore, if there were to be a subtour with visited nodes (nodes k such that $y_k = 1$) then there would be no path from the depot to the nodes in the subtour.

2.2.2 Non-compact models

The state-of-the-art regarding exact approaches for routing problems is composed mainly by models with an exponential number of constraints which are solved using branch-and-cut procedures.

We propose three different non-compact models. The connectivity cuts (CC) model which is an adaptation of the well-known TSP connectivity cuts model (see, e.g., Öncan et al., 2009; Roberti and Toth, 2012); the rounded visits (RV) model which can be obtained through the SCF model; and, finally, the rounded family visits (RFV) model which can be derived from the FCF model.

2.2.2.1 The connectivity cuts model

The connectivity cuts for the TSP ensure that we must choose at least one arc from each cut-set of every possible cut in order to obtain a single connected route. Since in the FTSP we do not wish to visit every node, instead of ensuring that a solution is a single connected route that visits every node, we only have to guarantee that the nodes which are chosen to be visited form a single connected route. This variation has already been proposed for different problems (see, e.g., Ljubić et al., 2006).

For $S \subset N$ define $S' = (0 \cup N) \setminus S$. The CC model for the FTSP is obtained by replacing (6) in the generic model with

$$x(S', S) \geq y_k, \qquad \forall S \subset N, \qquad \forall k \in S \tag{23}$$

Since we do not wish to visit all nodes we must replace the 1 in the right-hand side of the usual connectivity cuts for the TSP with the variable y. This ensures that we only need to have an arc crossing the cut-set $[S', S]$ if there is at least a node in S that is visited.

Constraints (23) are clearly valid for the FTSP. When $y_k = 0$ the constraints that we obtain are redundant due to the x variables being non-negative. When $y_k = 1$ there is a node in S that is visited, therefore, using a similar argument as in the TSP, in order to obtain a connected route there must be at least one arc in the cut-set that separates S' from S that is used.

Henceforth, constraints (23) will be designated as CC inequalities.

2.2.2.2 The rounded visits model

The CC model is a straightforward adaptation of a TSP model, hence it does not take into account the specificities of the FTSP. In the FTSP instance presented in figure 1 the total number of visits is three, that is, $V = 3$. Let us consider the sets S and $S' = (0 \cup N) \setminus S$ presented in figure 3. The number of nodes in S', besides the depot, is less than three thus, in order to obtain a feasible solution there is at least one node in S that has to be visited which implies that there is at least an arc in the cut-set $[S', S]$ that has to be used in the route (regardless of its family).

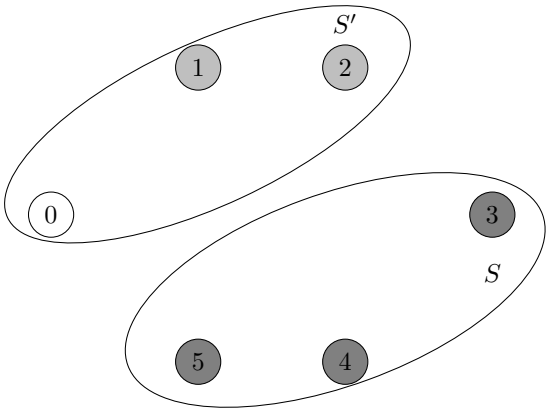


Figure 3: RV model motivation

For the reasons stated in the previous paragraph, the constraints obtained by replacing the right-hand-side of constraints (23) with 1 are valid for the FTSP as long as the number of nodes is S', besides the depot, is less or equal than $V-1$ or, equivalently, the number of nodes in S is at least $|N|-V+1$. Consequently, the RV model for the FTSP is obtained by replacing (6) with

$$x(S', S) \geq 1, \forall S \subset N : |S| \geq |N| - V + 1 \tag{24}$$

Proposition 1. *The RV model is a valid model for the FTSP in which constraints (24) are the subtour elimination constraints.*

Proof. Let (x^*, y^*) be an unfeasible solution that satisfies the equation system (2)–(5), (7)–(8). As we already mentioned, the only way in which solution (x^*, y^*) can be unfeasible is if it contains subtours. Let $\bar{S} = \{i \in N : y_i^* = 0\}$ be the set of nodes that were not chosen to be in the solution and $\mathbb{S} = \{i_1, \ldots, i_n, i_1\}$ be a subtour (which we recall is a route that does not contain the depot). Let $S = \bar{S} \cup \mathbb{S}$ and $S' = (0 \cup N) \setminus S$. Since: i) S' is only composed by visited nodes and the depot; ii) S' does not contain all the visited nodes due to subtour $\mathbb{S}$; and, iii) solution (x^*, y^*) visits V nodes in N due to constraints (5); we can conclude that $|S' \setminus 0| \leq V-1$. Equivalently, $|S| \geq |N| - V + 1$ therefore S satisfies the conditions in which constraints (24) are valid. Note that there is no arc between S' and S, since S is composed by non-visited nodes and a subtour. Hence, there is a violated inequality (24). □

The proof of proposition 1 not only shows that if an integer solution contains subtours it is always possible to find a violated inequality (24) but also shows how to construct the set S in order to obtain a violated inequality.

Henceforward, constraints (24) will be called RV inequalities.

2.2.2.3 The rounded family visits model

In the previous subsection we used the fact that we need to visit a total of V nodes. However, these V nodes are divided by the several families thus we can disaggregate the number of visits per family. Consider then the sets S' and S presented in figure 4. Note that set S' does not satisfies the conditions in which the RV inequalities are valid. Nevertheless, if we only consider the nodes in S' we could never obtain a feasible solution for the FTSP instance presented in figure 1 as the number of nodes that we are required to visit in family 2 is two and the set S' only contains one node from family 2. So, in order to complete family 2 and, consequently, obtain a feasible solution, we must visit the set S, which implies that there is at least one arc in the cut-set $[S', S]$ that has to be used in the route.

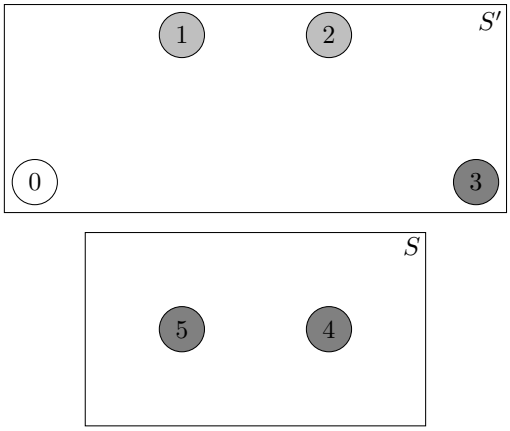


Figure 4: RFV model motivation

If the set S is in the conditions stated previously, that is, if there is at least a family l such that the number of nodes from family l in S' is less or equal than $v_l - 1$, which implies that the number of nodes from family l in S is at least $n_l - v_l + 1$, then replacing the right-hand side of constraints (23) by 1 originates valid constraints for the FTSP.

The RFV model is obtained by replacing constraints (6) in the generic model with

$$x(S', S) \geq 1, \forall S \subset N : \exists l \in \{1, \ldots, L\} : |S \cap F_l| \geq n_l - v_l + 1 \tag{25}$$

Proposition 2. *The RFV model is a valid model for the FTSP in which constraints (25) are the subtour elimination constraints.*

Proof. Let (x^*, y^*) be an unfeasible solution that satisfies the equation system (2)–(5), (7)–(8) and consider $\bar{S}$ and $\mathbb{S}$ defined as in the proof of proposition 1. Let $S = \bar{S} \cup \mathbb{S}$ and $S' = (0 \cup N) \setminus S$. Since solution (x^*, y^*) satisfies the visit requirements per family and S' does not contain all the nodes that were chosen to be in the solution, we know that there is at least a family l such that $F_l \cap S \neq \emptyset$ (due to subtour $\mathbb{S}$). Consequently, for the same family l, $|S' \cap F_l| \leq v_l - 1$, which implies that $|S \cap F_l| \geq n_l - v_l + 1$. Thus, S satisfies the conditions in which constraints (25) are valid. As there is no arc in the cut-set $[S', S]$ used in the solution, constraint (25) is violated. □

The above proof shows that if an integer solution contains subtours it is always possible to find a violated inequality (24). Additionally, it shows how to construct the set S in order to obtain a violated inequality.

Henceforth, constraints (25) will be called RFV inequalities.

2.3 Model comparison

In this section we compare the models presented previously. To ease the reading of this paper we will omit the detailed proofs of some results as the literature on model comparison for routing problems is vast (see, e.g., Öncan et al., 2009; Godinho et al., 2011; Roberti and Toth, 2012).

Before beginning with the model comparison let us denote by $LP(M)$ the linear programming relaxation value obtained using model M.

From the flow variables' definition the following relations hold (see figure 2):

$$f_{ij} = \sum_{l=1}^{L} t_{ij}^l \qquad \text{and} \qquad t_{ij}^l = \sum_{k \in F_l} z_{ij}^k$$

The first result relates the several compact models.

Proposition 3. $LP(SCF) \leq LP(FCF) \leq LP(NCF)$

The SCF model can be obtained from the FCF model by aggregating constraints (13)-(16) per family and replacing variables f by variables t according to the relation stated previously. The same applies to models FCF and NCF but now constraints (18)-(21) must be aggregated per F_l.

In order to establish relations regarding the linear programming relaxation (LP) value between the compact and the non-compact models we will use the max-flow/min-cut theorem. For more information on the max-flow/min-cut theorem see Ahuja et al. (1993). Before presenting the results let us define

$$\mathbb{S}^V = \{S \subset N : |S| \geq |N| - V + 1\}$$
$$\mathbb{S}^{FV} = \{S \subset N : \exists l \in \{1, \ldots, L\} : |S \cap F_l| \geq n_l - v_l + 1\}$$

Proposition 4. $LP(SCF) \leq LP(RV)$

Proof. Let (x^*, y^*) be a fractional feasible solution for the LP of the RV model. In order to prove the relation stated in proposition 4 we need to show that we can build a solution (x^*, y^*, f^*) which is feasible for the LP of the SCF model.

As (x^*, y^*) is a feasible solution for the LP of the RV model it satisfies constraints (24), in particular it satisfies them when $S = N$. Clearly, $N \in \mathbb{S}^V$. This means that any cut, including the minimum one, that separates the depot (recall that $0 \in S'$) from all the other nodes has a value greater or equal than 1.

Let us consider a capacitated network such that the capacity of arc $(i, j) \in A$ is $V \times x^*_{ij}$. Obviously, in this network the minimum cut has a value greater or equal than V. Therefore, from the max-flow/min-cut theorem, the maximum flow from the depot to N is greater or equal than V. Consequently, it is possible to obtain a feasible flow f^* with value V from the depot to the other nodes such that $f^*_{ij} \leq V \times x^*_{ij}$, for all $(i, j) \in A$.

□

Proposition 5. $LP(FCF) \leq LP(RFV)$

Proof. Let (x^*, y^*) be a fractional feasible solution to the LP of the RFV model. This proof will use the same reasoning as the proof of proposition 4.

Since (x^*, y^*) is a feasible solution for the LP of the RFV model it satisfies all constraints (25). Let us consider, for a given $l \in \{1, \ldots, L\}$, that $F_l \subset S$. Obviously, $S \in \mathbb{S}^{FV}$ and, therefore, $x^*(S', S) \geq 1$. This means that any cut that separates the depot (recall that $0 \in S'$) from family l has a value greater or equal than 1, including the minimum cut.

Consider now a network in which the capacity of arc $(i, j) \in A$ is $v_l \times x^*_{ij}$. In this network the minimum-cut has a value greater or equal than v_l. From the max-flow/min-cut theorem, the maximum flow from the depot to family l is greater or equal than v_l. Therefore, it is possible to obtain a feasible flow t^* with value v_l from the depot to family l such that $t^{l*}_{ij} \leq v_l \times x^*_{ij}$, for all $(i, j) \in A$.

□

Proposition 6. $LP(NCF) = LP(CC)$

Proof. Constraints (18)–(20) presented in the NCF model may be seen as $|N|$ different flows from the depot (node 0) to each node $k \in N$, each one with value of y_k. The amount of flow intended to each node k that traverses arc $(i, j) \in A$ is limited by x_{ij}, which is expressed by constraints (21). Hence, the max-flow/min-cut theorem states that for each node $k \in N$, y_k units of flow are sent from the depot to k if and only if every cut separating the depot from k has capacity of at least y_k, which is mathematically expressed by constraints (23) of the CC model.

□

All there is left now is to compare the non-compact models.

Proposition 7. $LP(RV) \leq LP(RFV)$

Proof. Proving that $LP(RV) \leq LP(RFV)$ is equivalent to proving that $\mathbb{S}^V \subseteq \mathbb{S}^{FV}$.

Consider $S \in \mathbb{S}^V$. Suppose that $S \notin \mathbb{S}^{FV}$, thus $\forall l \in \{1, \ldots L\}$, $|S \cap F_l| < n_l - v_l + 1 \iff |S \cap F_l| \leq n_l - v_l$. Since $\cup_{l=1}^{L} F_l$ is a partition:

$$|S| = \sum_{l=1}^{L} |S \cap F_l| \leq \sum_{l=1}^{L} (n_l - v_l) = \sum_{l=1}^{L} n_l - \sum_{l=1}^{L} v_l = |N| - V$$

Which is a contradiction since $S \in \mathbb{S}^V$. □

For some FTSP instances the set $\mathbb{S}^V$ is strictly contained in $\mathbb{S}^{FV}$ as there are sets in $\mathbb{S}^{FV}$ that do not belong to $\mathbb{S}^V$, an example is the set S presented in figure 4. Additionally, a consequence of proposition 7 is that constraints (24) are a particular case of constraints (25), thus we will not pursue the study of the RV model.

Regarding models CC and RFV, it is not possible to establish a comparison between their linear programming relaxation values. It is obvious that when a set S belongs to $\mathbb{S}^{FV}$ the RFV inequalities dominate the CC inequalities. However, there are many subsets of nodes that do not belong to $\mathbb{S}^{FV}$. This implies that there are several cases where inequalities RFV may not be applied while the CC inequalities may. Figure 5 shows an example of a fractional solution that does not violate an RFV inequality but violates a CC inequality. Consider a new FTSP instance in which family 1, which is represented by the light gray color, is only composed by one node (node 1) and family 2, which is represented by the dark gray color, has four family members (nodes 2, 3, 4 and 5). The number of nodes that we are required to visit in family 1 is one and in family 2 is two. The fractional solution satisfies constraints (2)–(5). The filled arcs correspond to the ones in which the x variables have value $\frac{1}{3}$ and the dashed ones correspond to the x values equal to $\frac{2}{3}$. The values on the top (or bottom) of each node corresponds to the value of the y variable associated with that node.

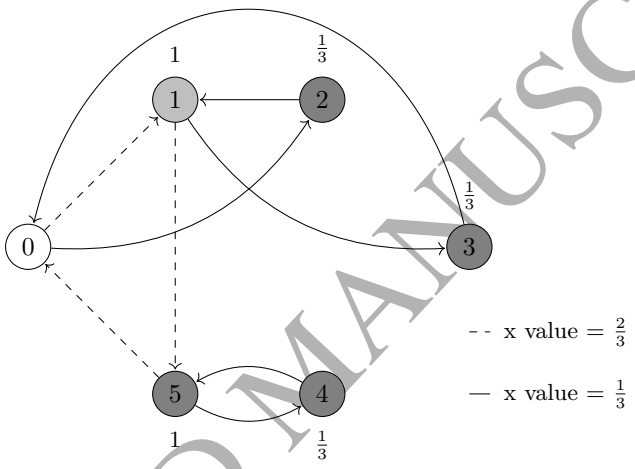


Figure 5: Feasible fractional solution for the CC model

Let us consider $S = \{4, 5\}$. In order to belong to $\mathbb{S}^{FV}$, S should have either $n_1 - v_1 + 1 = 1$ nodes from family 1 or $n_2 - v_2 + 1 = 4 - 2 + 1 = 3$ from family 2, which does not happen. But if we consider the cut-set $[S', S]$, being $S' = (0 \cup N) \setminus S$, we verify that it has value $\frac{2}{3}$ while $y_5 = 1$, hence there is a violated CC inequality which proves that the models RFV and CC are not comparable in terms of linear programming relaxation value.

Due to proposition 7 we can also state that the models CC and RV are not comparable in terms of linear programming relaxation value. Proposition 8 summarizes these findings.

Proposition 8. *Model CC is not comparable with models RV and RFV in terms of linear programming relaxation value.*

Figure 6 shows a summary of the results stated throughout this section.

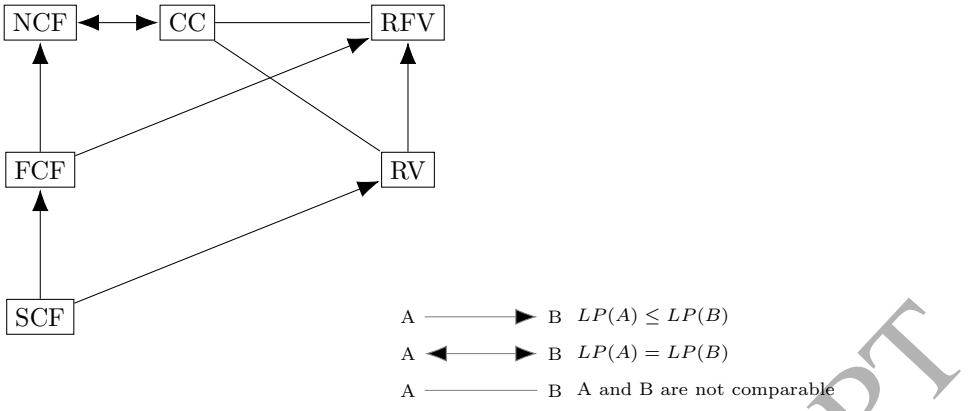


Figure 6: Known relationships between the proposed formulations

2.4 The separation algorithms

Since there are exponentially many CC and RFV inequalities we have to resort to a branch-and-cut scheme. In order to compute violated inequalities we need to use a separation algorithm. We will start by presenting the separation algorithm for the CC inequalities and then the one used for the RFV inequalities.

2.4.1 Separating the CC inequalities

Similarly to the separation algorithm for the TSP connectivity cuts, the separation algorithm for the CC inequalities consists in doing maximum flow computations on a capacitated network in which the capacity of arc (i, j) is x^*_{ij}, where (x^*, y^*) is a non-negative fractional solution that satisfies constraints (2)–(5). The main differences between the separation of the usual TSP connectivity cuts and the separation of the CC inequalities are: i) we only need to compute the maximum flow from the depot to a node k if $y^*_k > 0$; and ii) instead of sending 1 unit of flow for every node we only need to send y^*_k units to each node k.

When we use this separation algorithm we usually find more than one violated inequality. Hence, we set a limit to the number of violated inequalities found before we resolve the LP. Then, the algorithm stops either when it computes a maximum flow for every node k such that $y^*_k > 0$ or when the maximum number of added violated inequalities was reached. The computational results presented in section 4 were obtained using 20 as the maximum number of added violated inequalities. Due to the limit in the number of added violated inequalities, instead of computing the maximum flow from the depot to each node using the lexicographic ordering, we use a permutation of the nodes generated randomly every time we execute the separation algorithm. When the solution (x^*, y^*) is integer, we use the same separation algorithm but we only add one violated inequality.

In order to accelerate the separation of the CC inequalities in the case of fractional solutions, we developed a heuristic separation based on a standard heuristic separation for the TSP (see, e.g., Grötschel and Holland, 1991; Fischetti et al., 1997). Using this heuristic separation we were able to obtain a significant decrease in the computational time to obtain the LP value (see table 9 in Appendix versus table 2).

The main idea behind this heuristic separation is to add CC inequalities that are violated by the connected components induced by the fractional solution (x^*, y^*). We consider that nodes i and j belong to the same connected component if $x^*_{ij} > 0$. Let $\{C_0, C_1, \ldots, C_p\}$ be the set of connected components associated with solution (x^*, y^*) and $\bar{S} = \{i \in N : y^*_i = 0\}$ the set of non-visited nodes. We assume that the connected component C_0 is the one that contains the depot (node 0). With these $p + 1$ components, we will add a maximum of $p + 1$ violated inequalities determined heuristically. Recall that set S' always includes the depot, therefore, in the first violated inequality we define $S' = C_0$ and $S = C_1 \cup \ldots \cup C_p \cup \bar{S}$ and for the remaining p violated inequalities sets S' and S are defined as follows: for the kth-inequality $S = C_k$ and $S' = (0 \cup N) \setminus S$, $k = 1, \ldots, p$. All there is left now is to choose the right-hand side of these constraints. We will choose the node k in S such that $y^*_k \geq y^*_i, \forall i \in S, i \neq k$.

Instead of adding all $p + 1$ violated inequalities we decided, once again, to add at most 20 of the $p + 1$ violated inequalities. At the end, when the heuristic procedure is not able to provide new violated inequalities, we will use the exact separation algorithm, described previously, either to find more violated inequalities or to conclude that there are no more violated inequalities. Using this procedure we ensure that the solution obtained is the optimal solution for the linear programming relaxation of the CC model.

2.4.2 Separating the RFV inequalities

The separation of the RFV inequalities is similar to the separation of the CC inequalities, the main difference is related with the cardinality of S. In order to obtain a violated inequality the cardinality of $S \cap F_l$, for some family l, has to be at least $n_l - v_l + 1$. Thus, if we fix $n_l - v_l + 1$ nodes from family l to S and determine the minimum cut $[S', S]$ we know that the number of nodes in S will be, at least, $n_l - v_l + 1$ and, consequently, $S \in \mathbb{S}^{FV}$. Hence, in order to determine all violated inequalities we need to compute, for all families l, all the sets of nodes from that family with cardinality $n_l - v_l + 1$.

Let (x^*, y^*) be a fractional solution that satisfies the equation system (2)–(5) and let us define

$$\mathbb{S}_l^{=} = \{S \subset F_l : |S| = n_l - v_l + 1\}$$

Since the *RFV* inequalities are cut inequalities they can be computed using maximum flow computations on a graph where the capacity of arc (i, j) is x^*_{ij} as long as we ensure that $S \in \mathbb{S}^{FV}$. This is done by creating a fictitious node t and arcs from the nodes that we want to fix in S to t with infinite capacity. To obtain the minimum cut we just have to compute the maximum flow from the depot to t.

When (x^*, y^*) is an integer unfeasible solution the separation algorithm is different. Instead of searching the set $\mathbb{S}_l^{=}$ we will take into account the fact that the solution contains subtours. We will determine all the subtours that are induced by the solution (x^*, y^*). Let $\mathbb{T}^*$ be the set of all subtours in (x^*, y^*) and $\bar{S} = \{i \in N : y^*_i = 0\}$. Using the same argument as in the proof of proposition 2 it is possible to construct several sets S that violate RFV inequalities by ensuring that S contains the nodes in $\bar{S}$ and the nodes from, at least, one subtour in $\mathbb{T}^*$. Note that the nodes that belong to the subtour that contains the depot are always in S'.

A heuristic algorithm to accelerate the separation of the RFV inequalities can be devised using a procedure similar to the heuristic algorithm for the CC inequalities. The only difference is that, after determining sets S and S', it still has to be verified whether $S \in \mathbb{S}^{FV}$ or not. We will only add a violated inequality if $S \in \mathbb{S}^{FV}$. Once again, we set a limit to the number of added violated inequalities before resolving the LP. When it is not possible to find more violated inequalities using the heuristic procedure, we will use the exact separation to ensure that the solution obtained is the optimal solution for the linear programming relaxation of the RFV model.

Even though the calculation of the set $\mathbb{S}_l^{=}$ is polynomial it is very time consuming, as the computational results show. The importance of the exact separation is purely theoretical, since it could never be used to obtain the optimal values within a reasonable computational time. Nonetheless, we can use the RFV inequalities as valid inequalities for the linear programming relaxation and use other constraints to ensure the route connectivity.

We experimented adding the RFV inequalities as valid inequalities to the CC model. In order to do so we developed a new separation algorithm which is also based on min-cut/max-flow computations. We start by computing the maximum flow from the depot to every node $k \in N$. Let β^* be the value of the maximum flow from the depot to node k. If $\beta^* > 1$ there is neither a violated RFV nor a violated CC. Otherwise, we start by verifying if set $S \in \mathbb{S}^{FV}$ and if it is we will add an RFV inequality. If $S \notin \mathbb{S}^{FV}$ we will check if there is a violated CC inequality, that is, if $\beta^* < y^*_k$. Once again, if that is the case we will add the violated CC to the model. With this separation we do not ensure that there are no more violated RFV inequalities to separate. Nevertheless, this is ensured for the CC inequalities, hence, this separation algorithm is exact for the CC inequalities and, therefore, it can be used in a branch-and-cut algorithm to solve the FTSP. This procedure originates the CC+RFV model. The heuristic separation for the CC+RFV model is similar to the heuristic separation for the CC inequalities but every time S is determined we will verify if $S \in \mathbb{S}^{FV}$ and if that is the case we will add an RFV inequality, otherwise we will add a CC inequality.

To summarize, throughout this section we presented separation algorithms for: i) the CC model; ii) the RFV model; and iii) the CC+RFV model, which was described in the previous paragraph. We also presented, for each of the models referred previously, heuristic algorithms design to accelerate the separation process. All computational results were obtained using the heuristic separation as a starting point for the separation process.

3 The iterative local search algorithm

The main idea behind the ILS metaheuristic is to find a local optimum, using a local search procedure, and then apply a perturbation method in order to escape from that local optimum and continue the search of the solution space, which is done iteratively. For more information on the ILS see, for instance, Boussaïd et al. (2013); Lourenço et al. (2003).

Even though several local search procedures and several perturbation methods were experimented, for the sake of simplicity, we will only present the ones that provided the best results, which are the ones shown in section 4.

The first step of the ILS is to determine an initial feasible solution for the FTSP. The constructive heuristic proposed is an adaptation of the nearest neighbor heuristic for the TSP. The nearest node i will only be added to the route if its family is not complete, otherwise node i is ignored and another node has to be chosen to be inserted in the route instead. The process ends when all the families are complete. Let s be the feasible solution obtained.

Afterwards we will apply a local search procedure to s in order to obtain a local optimum. The local search procedure consists in searching the neighborhoods $\mathcal{N}_I$ and $\mathcal{N}_O$ that are defined as follows:

- $\mathcal{N}_I(s) = \{s' \text{ feasible} : s' \text{can be obtained from } s \text{ by switching two nodes in the route}\}$
- $\mathcal{N}_O(s) = \{s' \text{ feasible} : s' \text{can be obtained from } s \text{ by switching two nodes, one that belongs to the route and another that does not, from the same family}\}$

The neighborhoods will be searched using the procedure, which can be applied to any generic neighborhood $\mathcal{N}$, that follows. We start by computing the cost of every solution s' that belongs to $\mathcal{N}(s)$. This process is not very time consuming since, in both neighborhoods, the cost of s' can easily be calculated through the cost of s and both neighborhoods, $\mathcal{N}_I$ and $\mathcal{N}_O$, are relatively small neighborhoods. In fact, the size of neighborhood $\mathcal{N}_I$ is $\frac{(V+1)\times V}{2}$ and the size of neighborhood $\mathcal{N}_O$ is $\sum_{l=1}^{L} v_l \times (n_l - v_l)$. To continue the search, we will choose the solution $s^* \in \mathcal{N}(s)$ such that $Cost_{s^*} \leq Cost_{s'}$, $\forall s' \in \mathcal{N}(s)$. Now, we will repeat the search for $\mathcal{N}(s^*)$. This process is repeated until we cannot find a solution in $\mathcal{N}(s^*)$ that has a lower cost than the cost of s^*.

The pseudo-code for the neighborhood search is presented in algorithm 1 and the pseudo-code for the local search procedure is given in algorithm 2.

Algorithm 1 The neighborhood search procedure

Require: A neighborhood $\mathcal{N}$ and a feasible solution s for the FTSP. Let z be the cost of s.

1: **while** There is a solution in $\mathcal{N}(s)$ with lower cost than z **do**
2: Compute the cost of every solution $s' \in \mathcal{N}(s)$.
3: Select the solution s' with the lowest cost. Let s^* be that solution.
4: $s = s^*$.
5: **end while**

Return: Solution with a cost lower or equal than the cost of s.

Algorithm 2 The local search procedure

Require: A feasible solution s for the FTSP

1: Search $\mathcal{N}_I(s)$ and obtain s^*.
2: Search $\mathcal{N}_O(s^*)$ and obtain s^{**}.
3: Search $\mathcal{N}_I(s^{**})$ and obtain s^{***}.

Return: Solution s^{***} such that $Cost_{s^{***}} \leq Cost_s$

After obtaining a local optimum using the local search procedure presented, we will apply the perturbation method to escape from that local optimum. The perturbation method that provided the best results consists in choosing at random v_l nodes in n_l possible nodes from family l, with $l = 1, \ldots, L$. These randomly chosen nodes will be inserted in the route in the best possible position, that is, in the position that leads to the lowest increase in the route cost. Note that if we choose a node that already belongs to the route we do not need to insert it. After the insertion we will, most likely, obtain an unfeasible solution since we visit more than v_l nodes per family l. In order to restore the feasibility we will remove the extra nodes according to one of the following criteria:

- *Greedy*: The nodes that lead to the highest decrease in the solution value;
- *Random*: Nodes chosen randomly.

The choice of the removal criterion is a parameter of the ILS algorithm. Once we restore the solution's feasibility we will apply the local search procedure to obtain a local optimum and then we will perturb it again. The pseudo-code for the perturbation method is given in algorithm 3 and the pseudo-code for the ILS algorithm is given in algorithm 4.

Algorithm 3 The perturbation method

Require: A feasible solution s for the FTSP and the choice of a removal criterion.

for Every family $l \in \{1, \ldots, L\}$ **do**
  Chose randomly v_l nodes in F_l.
  Insert the randomly chosen nodes in the route in the best possible position.
  Remove the nodes accordingly to the chosen criterion.
end for

Algorithm 4 The ILS algorithm

Determine a feasible solution s using the adaptation of Nearest Neighbor.
Apply the local search procedure to s and obtain a new solution s^*.
$i = 1$
while $i \leq Maximum\_number\_iterations$ **do**
  Apply the perturbation procedure to s^* and obtain solution p.
  Apply the local search procedure to p and obtain a new solution p^*.
  if Cost p^* < Cost s^* **then**
    Update the best solution to p^*.
  end if
  $s^* = p^*$.
  $i = i + 1$
end while

Return: The best solution found

Before creating the metaheuristic ILS we developed a genetic algorithm combined with a local search procedure since the state-of-the-art regarding metaheuristics to solve the GTSP is composed mainly by evolutionary algorithms (see, e.g., Tasgetiren et al., 2010; Snyder and Daskin, 2006; Pop, Petrica and Oliviu, Matei and Sabo, Cosmin, 2017). However, the results were not as good as expected and the algorithms were very time consuming. This can also be seen in the results obtained by Morán-Mirabal et al. (2014). The biased random-key genetic algorithm was the slowest one and, for the highest dimension instances, is the algorithm that provided the worst results.

4 Computational results

Computational experiments were carried out using the benchmark instances presented in Morán-Mirabal et al. (2014). These instances are based on instances from the TSPLIB (see Reinelt (1991)) where the families as well as the number of visits for each family were generated. In order to simplify the notation, the instance whose name is $instancename|N| + 1\_F\_1001\_100i\_2$ will be designated by $instancename\_i$ henceforth. A complete description of the benchmark instances is available in Appendix, Table 8.

The models were implemented in C++ and they were solved using the Concert Technology from CPLEX 12.6.1. The ILS procedure was also implemented in C++. All computational experiments were carried out in an Intel Core i7, 3.60GHz, 8GB RAM.

We will start by presenting the results obtained with the several ILP models proposed and compare them with each other and then, for the instances that the exact methods were unable to solve, we will present the upper bounds obtained using the ILS metaheuristic.

4.1 Results for the exact methods

We will start by presenting the results concerning the linear programming relaxation. These results were obtained without using the cuts generated by CPLEX to allow us to establish a fair comparison between the several models.

Table 1 shows the LP results obtained using the several compact models. It is divided into three parts, each one devoted to a different model. Each of those parts has three columns, one with the LP value (LP), other with the percentage of gap between the LP value and the optimal value ($gap = 100 \times (\text{optimal value} - \text{LP value})/\text{optimal value}$) and the final one with the time, in seconds, to obtain the LP value (t_s).

	SCF			FCF			NCF		
Instance	LP	gap	t_s	LP	gap	t_s	LP	gap	t_s
burma_1	10.00	28.19%	0	11.36	18.46%	0	12.07	13.34%	0
burma_2	23.25	9.38%	0	24.33	5.18%	0	25.66	0.00%	0
burma_3	7.92	33.40%	0	10.22	13.98%	0	10.43	12.25%	0
bayg_1	4767.12	10.83%	0	4982.57	6.80%	0	5273.32	1.36%	3
bayg_2	4837.19	16.47%	0	5100.93	11.92%	0	5754.64	0.63%	3
bayg_3	4945.35	10.80%	0	5127.44	7.52%	0	5544.33	0.00%	2
att_1	18224.40	23.06%	0	18957.50	19.96%	2	23686.00	0.00%	85
att_2	14288.90	30.67%	0	14862.20	27.89%	2	20609.10	0.00%	101
att_3	7262.57	19.52%	0	7925.97	12.17%	2	8742.08	3.13%	135
bier_1	20475.00	39.26%	10	27104.10	19.60%	1242	33227.80	1.43%	257646
bier_2	67984.50	23.39%	9	72781.20	17.98%	1270	88308.90	0.48%	222840
bier_3	25423.90	46.73%	9	32278.00	32.37%	1145	47162.50	1.18%	259977
average		24.31%	2		16.15%	305		2.81%	61733

Table 1: Linear programming relaxation compact models

The LP values obtained are consistent with the results stated in section 2.3, the LP value obtained using the SCF model is the lowest one followed by the one obtained with the FCF model and the highest LP value was obtained with the NCF model. Note that in every instance the relation stated in proposition 3 is satisfied with the strict inequality, which shows that, for these instances, the polyhedron associated with these models are strictly contained in each other. The computational time increases with the quality of the LP values. For instances *bier*, the SCF model provides the LP values in less than 10 seconds with a gap of, at least, 23.39% while the NCF model provides LP values with a maximum gap of 1.43%, in more than 200000 seconds.

Table 2 shows the LP results of the several non-compact models and it is organized in a similar manner as table 1 but with the additional information of the number of added violated inequalities (#name_of_inequality) per model.

	CC				RFV				CC + RFV				
Instance	LP	gap	t_s	#CC	LP	gap	t_s	#RFV	LP	gap	t_s	#CC	#RFV
burma_1	12.07	13.34%	0	34	13.93	0.00%	1	97	13.93	0.00%	0	19	40
burma_2	25.66	0.00%	0	90	25.66	0.00%	0	32	25.66	0.00%	0	0	35
burma_3	10.43	16.47%	0	55	11.89	0.00%	0	52	11.89	0.00%	0	11	36
bayg_1	5273.32	1.36%	0	249	5316.85	0.54%	0	515	5330.17	0.29%	0	39	201
bayg_2	5754.64	0.63%	0	283	5791.01	0.00%	0	289	5791.01	0.00%	1	13	224
bayg_3	5544.33	0.00%	0	150	5544.33	0.00%	0	490	5544.33	0.00%	0	24	172
att_1	23686.00	0.00%	1	501	23580.50	0.45%	1	1218	23686.00	0.00%	0	18	502
att_2	20609.10	0.00%	0	795	20609.10	0.00%	4	2515	20609.10	0.00%	0	110	769
att_3	8742.08	3.13%	1	419	8760.03	2.93%	3	2309	9024.58	0.00%	1	111	263
bier_1	33227.80	1.43%	9	1221	33314.70	1.17%	5842	21103	33446.00	0.78%	8	358	883
bier_2	88308.90	0.48%	19	1389	87336.20	1.58%	7973	29031	88448.10	0.32%	20	361	1008
bier_3	47162.50	1.18%	84	2289	46830.70	1.88%	4199	19932	47392.70	0.70%	77	680	1445
average		2.82%	9	623		0.71%	1502	6465		0.17%	9	145	465

Table 2: Linear programming relaxation non-compact models

As it was expected the LP value obtained using the NCF model is equal to the one provided by the CC model. However, if we focus on the computational time we verify that the CC model is much faster. In the case of instances *bier* the NCF model took more than 200000 seconds to obtain the LP value while the CC model took, in the most time consuming instance, 84 seconds which is a very significant difference. As we know from proposition 5 the RFV model provides an LP value greater or equal than the FCF model. In fact, for these instances, the RFV model always provides a significantly larger LP value than the FCF model (see average of gap). Nonetheless, regarding the computational time, the FCF model is more efficient than the RFV model, contrary to what happened with the NCF and the CC models.

In some instances the CC model provides higher LP values than the RFV model, while in others is the contrary. To be more precise the CC model is better than the RFV in three cases whereas the RFV model provides a higher LP value in seven instances. These results emphasize the fact that the referred models are not comparable. Regarding the computational efficiency, the CC model is much faster than the RFV model, which is a consequence of the cardinality of sets $\mathbb{S}_l^=$. Considering instance *bier*_3, the number of added CC inequalities is 2289 while the number of added RFV is almost nine times more.

As we already mentioned, the study of the RFV model is purely theoretical since due to the number of added violated inequalities being huge, as we verified in table 2, it is impossible to use it to solve up to optimality instances of average size.

The CC+RFV model outperforms both the CC and the RFV model in terms of quality of the LP value. The CC+RFV provides a higher LP value than the other non-compact models in five instances and it is able to reach the optimal value while solving the LP in eight out of 12 instances. Focusing on the computational time, we verify that the CC+RFV model is competitive with the most efficient model so far, the CC model. When we compare it with the compact models the only one that provides LP values in less time is the SCF model however, the quality of the LP values provided by the SCF model is much worse.

Regarding the number of added violated inequalities in the CC+RFV model, we observe that the total number of violated inequalities is not significantly different than the number of violated inequalities in the CC model but the number of added RFV is much higher than the number of added CC, hence the better LP values.

Table 9 in Appendix shows the LP results of the several non-compact models obtained without using the heuristic separation as a starting point of the separation procedure. As it was already mentioned, the usage of the heuristic separation lead to significantly decreases in the computational time to obtain the LP value. In fact, comparing tables 2 and 9, we can conclude that the heuristic separation reduces the average computational times in 90%, 86% and 82%, respectively for models CC, RFV and CC+RFV.

We also tried to compute the LP values of the higher dimensioned instances. Regarding instances *gr* and *pr*, which have 666 and 1002 nodes, respectively, it was impossible to obtain any results due to lack of computational memory. However, it was possible to obtain the LP values of instances *a* which have 280 nodes. We will only present the results obtained using models CC and CC+RFV since these are the most efficient models. Table 3 is divided into two parts, the first part is dedicated to the CC model and the second one to the CC+RFV model. In each part it is possible to see the LP values obtained (LP), the percentage of gap between the LP and the optimal value ($gap = 100 \times$ (optimal value − LP value)/optimal value), the time, in seconds, to obtain the LP values (t_s) and the number of added violated inequalities (#name_of_inequality). For the instances whose optimal value is not known, which are marked with *, the percentage of gap is calculated replacing the optimal value by the best known upper bound.

	CC				CC + RFV				
Instance	LP	gap	t_s	#CC	LP	gap	t_s	#CC	#RFV
a_1	1652.46	12.61%*	729	3011	1677.52	11.30%*	2015	1359	3206
a_2	1478.70	12.41%*	1206	3299	1509.88	10.56%*	2227	1651	3263
a_3	1365.56	3.02%	1136	4047	1387.04	1.50%	1226	1948	2693
average		9.35%	1024	3452		7.79%	1823	1653	3054

Table 3: Linear programming relaxation instances *a*

Observing table 3 we see that the CC+RFV model is the best one concerning the quality of the LP value while the CC model is the most efficient one concerning the computational time. In fact, the CC model took an average of 1024 seconds to provide the LP values of instances' *a* while the CC+RFV took 1653. The best results, in terms of computational time, provided by the CC model are a consequence of the number of added violated inequalities.

To obtain the optimal values we set as an upper cut off the results obtained using the ILS algorithm rounded up.

These results are available in Appendix, table 10. We also allowed CPLEX cuts and set as time limit 10800 seconds. Table 4 shows the instance name (Instance), the optimal value (v^*) and the time, in seconds, to obtain the optimal value (model designation) using the different models. The reported computational times already incorporate the computational time to obtain the upper bounds with the ILS algorithm. The last column of table 4 (MM) shows the computational time, in seconds, needed to obtain the optimal value with the model presented by Morán-Mirabal et al. (2014), reported in their work. Note that the times on this column were obtained in a computer with different characteristics, and so they cannot be directly compared to the ones of the proposed models.

Instance	v^*	SCF	FCF	NCF	CC	CC+RFV	MM
burma_1	13.93	0	0	1	0	0	0
burma_2	25.66	1	0	0	0	0	0
burma_3	11.89	0	0	1	0	0	0
bayg_1	5345.86	1	1	6	1	0	5
bayg_2	5791.01	3	2	3	1	1	28
bayg_3	5544.33	0	1	5	0	0	5
att_1	23686.00	9	75	117	0	1	3033
att_2	20609.10	131	364	143	0	1	3224
att_3	9024.58	5	11	181	0	1	1131
bier_1	33709.70	741	10800	10800	30	20	-
bier_2	88736.40	10800	10800	10800	147	32	-
bier_3	47726.30	10800	10800	10800	65	70	-
average∗		17	50	51	0	0	815

*Only considering the first nine instances

Table 4: Computational times for optimal values

The non-compact models are much faster than the compact ones, which is even more noticeable as the instance dimension increases. Regarding the compact models, even though the SCF model was the one that provided the worst LP value it was the only one able to solve one instance *bier* within the time limit. Concerning the computational time, there is no relation that we can establish between the FCF and the NCF model. On the one hand, disaggregated models provide better LP values and on the other hand they have more variables which make its resolution more time consuming. With respect to the non-compact models, in the instances in which the computational time is not negligible, the CC+RFV model is the most efficient one on average (41 seconds versus 81 seconds from the CC model). In fact, the CC+RFV model provides the optimal value of instances with 127 nodes in less than 70 seconds.

The integer programming model proposed by Morán-Mirabal et al. (2014) was able to solve instances up to 48 nodes, in an average of 815 seconds. Therefore, for the instances that had never been solved, namely instances *bier*, the CC+RFV model was able to obtain the optimal values within a very reasonable computational time.

Since the CC+RFV model is the most efficient one, on average, to solve the FTSP to optimality, we tried to obtain the optimal values of instances *a* using the referred model. Table 5 shows the lower bound (LB), upper bound (UB), the gap between the lower and upper bound (*gap*) provided by CPLEX and the correspondent computational time, in seconds (t_s). Again, the reported computational times already incorporate the computational time of the ILS algorithm.

Instance	LB	UB	*gap*	t_s
a_1	1688.74	1692.92	0.25%	10838
a_2	1514.38	1624.86	6.80%	10837
a_3	1408.14	1408.14	0.00%	4990
average			2.35%	8430

Table 5: Instances' *a* results

We were only able to obtain the optimal value of instance a_3 within the time limit. We believe that this happens because instance a_3 is the one with the lowest number of required visits. In fact, in instance a_3 the total number of required visits is 141 while in instances a_1 and a_2 is, respectively, 179 and 156.

4.2 Results for the ILS procedure

Since the CC+RFV model provides the optimal value of instances *bier* and instance *a_3* within a very reasonable computational time we will only present the results obtained with the ILS procedure for the instances whose optimal value remains unknown (results for the smaller instances available in Appendix, table 11). We performed 5000 iterations of the ILS procedure and the perturbation method was applied using the criterion *Greedy* but once in every 100 iterations the criterion adopted was the criterion *Random*. The quality of the solutions obtained will be measured in terms of percentage gap between the best known upper bound and the value of the solution provided by the ILS procedure ($\%gap = 100 \times$ (heuristic value − best known upper bound)/best known upper bound). The best known upper bounds used are the ones obtained by Morán-Mirabal et al. (2014).

Due to the fact that the perturbation method used is random we did five runs with different seeds for each instance. Table 6 is divided into two parts. In the first part we report the results obtained by Morán-Mirabal et al. (2014), namely the best known upper bound (*best*) and the average time, in seconds, to obtain the best known upper bound (tb_s). In the second part it is possible to see the minimum percentage of gap (*min*), the average percentage of gap (*average*), the maximum percentage of gap (*max*), the difference between the maximum and the minimum percentage of gap (*range*) and the average time, in seconds (t_s) considering the five independent runs.

	Results Morán-Mirabal et al. (2014)		Results ILS				
Instance	*best*	tb_s	*min*	*average*	*max*	*range*	t_s
a_1	1891.16	218	2.93%	4.29%	5.45%	2.53%	37
a_2	1697.48	2701	-0.55%	0.00%	1.17%	1.72%	36
gr_1	1817.06	6610	-9.61%	-7.22%	-5.64%	3.97%	532
gr_2	1443.05	4005	-10.33%	-9.54%	-7.31%	3.03%	535
gr_3	1384.18	7200	-4.63%	-3.38%	-2.02%	2.61%	562
pr_1	163461.79	21	-11.11%	-9.38%	-7.85%	3.26%	2156
pr_2	182144.13	9	-12.19%	-11.28%	-10.16%	2.03%	2008
pr_3	149456.63	228	-8.08%	-5.15%	-3.50%	4.58%	1666
average		2624	-6.70%	-5.21%	-3.73%	2.97%	941

Table 6: Results obtained using the ILS algorithm

Instance	Best known upper bound	New best known upper bound
a_2	1697.48	1688.13
gr_1	1817.06	1642.35
gr_2	1443.05	1293.96
gr_3	1384.18	1320.12
pr_1	163 461.79	145 303.34
pr_2	182 144.13	159 933.64
pr_3	149 456.63	137 385.12

Table 7: Best upper bounds

Table 6 shows some negative % of gap values which means that the value of the solution obtained by the ILS procedure is lower than the best known upper bound. In fact, we were able to improve the best known upper bound in seven out of eight instances. The new best known upper bounds are available in table 7.

There are several ranges higher than 2%. Nonetheless, if we consider the worst solution obtained in the five runs we still obtain better upper bounds in six of the eight instances with unknown optimal value.

Observing table 5 we verify that, for instance *a_2*, the upper bound provided by CPLEX after 10837 seconds of computational time is 1624.86 which is lower than the new best known upper bound obtained by the ILS procedure. However, this upper bound was obtained after 10000 seconds of computational time while the new best known upper bound was obtained after 36 seconds.

The ILS procedure not only provides feasible solutions for instances *gr* and *pr*, which are impossible to solve using the exact approaches due to lack of computational memory, in less than 2156 seconds but also improves their best known upper bounds presented in the literature.

A summary with the current best known upper bound for each benchmark instance is available in Appendix, table 12.

Regarding the computational times, it is not possible to establish a fair comparison with the metaheuristics proposed by Morán-Mirabal et al. (2014) for the FTSP due to the fact that the stopping criterion used is different from ours. The referred metaheuristics stop either when they reach the time limit, which is 7200 seconds, or when they find a solution with a cost less or equal than the target solution. The target solution is the best solution found, by the referred metaheuristics, after 36000 seconds.

5 Conclusions

We proposed several formulations and a metaheuristic for the FTSP, which is a generalization of the TSP. The FTSP has not been widely studied in the literature, but we believe it is an important problem nonetheless.

The formulations may be divided into compact formulations and non-compact ones. The compact formulations use flow variables to model the route connectivity. There are three compact models each one with a different type of flow variable. Even though some of the non-compact models are not solvable in practice, due to the time consuming separation processes, the subtour elimination constraints of those models may be used as valid inequalities for the FTSP. To the best of our knowledge, this is the first work that presents several formulations for the FTSP and establishes a theoretical comparison between them. The computational results show that the non-compact models are the most effective ones. Using the CC+RFV model we were able to obtain the optimal values of FTSP benchmark instances with 127 nodes, which had never been solved, in less than 70 seconds and of one instance with 280 nodes in 3615 seconds, thus we advise the usage of this model.

Even though the non-compact models are very efficient, they were not able to provide solutions for instances with more than 280 nodes due to lack of computational memory. We believe that in order to solve the higher dimensioned instances new valid and effective inequalities must be derived to further improve the quality of the lower bounds obtained during the branch-and-cut procedure.

In order to solve the instances whose optimal value remains unknown we propose an ILS metaheuristic. For the benchmark instances, this procedure was able to provide very good solutions, within a very reasonable computational time, better than the existing ones for seven out of eight instances that the exact methods were not able to solve.

Acknowledgments

This research was supported by Portuguese National Funding from FCT - Fundação para a Ciência e a Tecnologia, under projects UID/MAT/04561/2013 and PTDC/MAT-NAN/2196/2014.

The authors would like to thank the anonymous reviewers for the helpful comments.

References

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, 1993.

F. Boctor, G. Laporte, and J. Renaud. Heuristics for the traveling purchaser problem. *Computers & Operations Research*, 30(4):491–504, 2003.

I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

V. Cacchiani, A. Muritiba, M. Negreiros, and P. Toth. A multistart heuristic for the equality generalized traveling salesman problem. *Networks*, 57:231–239, 2011.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

M. Fischetti, J. J. Salazar González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

B. Gavish and S. Graves. The travelling salesman problem and related problems. 1978.

M. T. Godinho, L. Gouveia, and P. Pesneau. On a time-dependent formulation and an updated classification of atsp formulations. In *Progress in Combinatorial Optimization*, pages 223–254. ISTE-Wiley, 2011.

B. Golden, Z. Naji-Azimi, S. Raghavan, M. Salari, and P. Toth. The generalized covering salesman problem. *INFORMS Journal on Computing*, 24(4):534–553, 2012.

M. Grötschel and O. Holland. Solution of large-scale symmetric travelling salesman problems. *Mathematical Programming*, 51(1):141–202, 1991.

G. Gutin and A. Punnen. *The traveling salesman problem and its variations*. Springer Science & Business Media, 2002.

G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah. Some applications of the generalized travelling salesman problem. *Journal of the Operational Research Society*, 47(12):1461–1467, 1996.

I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. *Mathematical programming*, 105:427–449, 2006.

H. Lourenço, O. Martin, and T. Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

L. Morán-Mirabal, J. González-Velarde, and M. Resende. Randomized heuristics for the family traveling salesperson problem. *International Transactions in Operational Research*, 21:41–57, 2014.

T. Öncan, İ. K. Altınel, and G. Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009.

G. Reinelt. Tsplib-a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

R. Roberti and P. Toth. Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison. *EURO Journal on Transportation and Logistics*, 1(1-2):113–133, 2012.

L. V. Snyder and M. S. Daskin. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, 174(1):38 – 53, 2006.

S. Srivastava, S. Kumar, R. Garg, and P. Sen. Generalized traveling salesman problem through n sets of nodes. *Journal of the Canadian Operational Research Society*, 7:97–101, 1969.

M. F. Tasgetiren, P. Suganthan, and Q.-K. Pan. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation*, 215(9):3356 – 3368, 2010.

Pop, Petrica and Oliviu, Matei and Sabo, Cosmin. A Hybrid Diploid Genetic Based Algorithm for Solving the Generalized Traveling Salesman Problem. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 149–160. Springer, 2017.

Pop, Petrica C. New integer programming formulations of the generalized traveling salesman problem. *American Journal of Applied Sciences*, 4(11):932–937, 2007.

R. Wong. Integer programming formulations of the traveling salesman problem. In *Proceedings of the IEEE international conference of circuits and computers*, pages 149–152, 1980.

A Instance description

Instance name	$\|N\|+1$	L	V	n_l	v_l
bruma14_3_1001_1001_2	14	3	6	[4, 5, 4]	[2, 2, 2]
bruma14_3_1001_1002_2			10		[4, 2, 4]
bruma14_3_1001_1003_2			4		[2, 1, 1]
bayg29_4_1001_1001_2	29	4	16	[7, 9, 6, 6]	[6, 4, 5, 1]
bayg29_4_1001_1002_2			17		[2, 9, 1, 5]
bayg29_4_1001_1003_2			18		[6, 6, 1, 5]
att48_5_1001_1001_2	48	5	34	[12, 9, 9, 7, 10]	[10, 4, 9, 7, 4]
att48_5_1001_1002_2			25		[8, 2, 9, 1, 5]
att48_5_1001_1003_2			15		[6, 1, 3, 3, 2]
bier127_10_1001_1001_2	127	10	62	[12, 12, 14, 8, 13, 16, 13, 8, 17, 13]	[10, 4, 13, 1, 12, 4, 6, 1, 5, 6]
bier127_10_1001_1002_2			85		[8, 2, 12, 7, 9, 9, 5, 5, 17, 11]
bier127_10_1001_1003_2			60		[6, 1, 13, 3, 3, 13, 13, 2, 2, 4]
a280_20_1001_1001_2	280	20	179	[15, 14, 16, 11, 19, 15, 18, 10, 17, 16, 16, 8, 7, 15, 24, 8, 11, 13, 15, 11]	[14, 10, 14, 4, 13, 9, 15, 4, 5, 14, 6, 7, 6, 8, 13, 7, 9, 13, 6, 2]
a280_20_1001_1002_2			156		[8, 2, 12, 9, 9, 5, 17, 6, 3, 9, 7, 2, 6, 11, 4, 6, 11, 7, 11, 11]
a280_20_1001_1003_2			141		[14, 14, 6, 1, 13, 3, 18, 3, 2, 4, 10, 4, 4, 8, 5, 4, 9, 4, 14, 1]
gr666_30_1001_1001_2	666	30	357	[27, 24, 24, 17, 29, 19, 20, 17, 27, 24, 26, 15, 15, 30, 40, 11, 19, 28, 27, 20 28, 22, 24, 14, 23, 15, 17, 18, 20, 25]	[14, 10, 15, 4, 13, 9, 15, 4, 22, 5, 14, 6, 15, 30, 24, 7, 2, 1, 19, 5, 6, 13, 18, 9, 21, 10, 15, 2, 10, 19]
gr666_30_1001_1002_2			328		[8, 2, 15, 9, 21, 17, 14, 3, 9, 7, 10, 6, 11, 4, 39, 11, 11, 26, 7, 8 1, 8, 14, 7, 19, 5, 6, 9, 9, 12]
gr666_30_1001_1003_2			328		[6, 17, 13, 2, 4, 12, 4, 5, 12, 14, 15, 9, 4, 14, 33, 10, 17, 27, 17, 8, 6, 8, 2, 5, 8, 9, 17, 15, 6, 9]
pr1002_40_1001_1001_2	1002	40	486	[22, 28, 27, 30, 32, 24, 21, 22, 29, 30, 27, 16, 20, 30, 38, 16, 21, 23, 27, 28 23, 25, 26, 26, 21, 24, 20, 30, 18, 25 25, 27, 27, 21, 26, 24, 28, 28, 25, 21]	[14, 10, 15, 4, 13, 9, 15, 4, 22, 25, 5, 14, 6, 30, 24, 14, 13, 7, 25, 22, 2, 1, 19, 5, 6, 13, 18, 9, 15, 2, 22, 10, 19, 11, 1, 8, 3, 8, 6, 17]
pr1002_40_1001_1002_2			538		[8, 2, 15, 25, 9, 21, 17, 14, 22, 22, 3, 9, 7, 10, 6, 11, 4, 22, 27, 7 11, 7, 8, 1, 8, 14, 19, 21, 6, 9 9, 12, 26, 8, 23, 21, 8, 28, 18, 20]
pr1002_40_1001_1003_2			463		[6, 17, 13, 19, 19, 18, 19, 2, 4, 26, 12, 4, 5, 12, 15, 9, 4, 14, 1, 15, 17, 17, 8, 6, 8, 2, 5, 8, 17, 15, 6, 9, 3, 20, 15, 5, 14, 26, 18, 10]

Table 8: Complete description of the benchmark instances

B Results obtained with the non-compact models without using heuristic separation

Table 9 shows the LP results obtained using the several non-compact models without using the heuristic separation to accelerate the separation process. It is divided into three parts, each one devoted to a different non-compact model. Each of those parts has a column with the LP value (LP), other with the percentage of gap between the LP value and the optimal value ($gap = 100 \times$ (optimal value − LP value)/optimal value), other one with the time, in seconds, to obtain the LP value (t_s) and, the final one, with the number of added violated inequalities (#name_of_inequality).

	CC				RFV				CC + RFV				
Instance	LP	gap	t_s	#CC	LP	gap	t_s	#RFV	LP	gap	t_s	#CC	#RFV
burma_1	12.07	13.34%	0	76	13.93	0.00%	0	92	13.93	0.00%	0	16	105
burma_2	25.66	0.00%	0	68	25.66	0.00%	0	33	25.66	0.00%	0	0	85
burma_3	10.43	16.47%	0	48	11.89	0.00%	0	46	11.89	0.00%	0	3	83
bayg_1	5273.32	1.36%	0	286	5316.85	0.54%	0	683	5330.17	0.29%	0	41	384
bayg_2	5754.64	0.63%	0	352	5791.01	0.00%	0	428	5791.01	0.00%	0	13	364
bayg_3	5544.33	0.00%	0	369	5544.33	0.00%	1	525	5544.33	0.00%	0	76	335
att_1	23686.00	0.00%	1	1092	23580.50	0.45%	1	1537	23686.00	0.00%	0	18	859
att_2	20609.10	0.00%	1	1100	20609.10	0.00%	15	4240	20609.10	0.00%	1	134	1052
att_3	8742.08	3.13%	0	740	8760.03	2.93%	5	2619	9024.58	0.00%	0	277	443
bier_1	33227.80	1.43%	90	3420	33314.70	1.17%	50190	53900	33446.00	0.78%	44	845	2237
bier_2	88308.90	0.48%	117	3538	87336.20	1.58%	45094	61278	88445.70	0.33%	153	601	3073
bier_3	47162.50	1.18%	870	5890	46830.70	1.88%	37905	61550	47439.20	0.60%	393	1761	4155
average		2.82%	90	1423		0.71%	11101	15578		0.17%	49	315	1098

Table 9: Linear programming relaxation for non-compact models without using heuristic separation

C Upper bounds used in the branch-and-cut procedure obtained with the ILS heuristic

Table 10 has two columns, one with the instance's name and another one with the upper bound, obtained using the ILS procedure, used as an upper cut-off in the branch-and-cut procedure.

Instance name	Upper bound
burma_1	13.93
burma_2	25.66
burma_3	11.89
bayg_1	5345.86
bayg_2	5791.01
bayg_3	5544.33
att_1	24556.14
att_2	20609.08
att_3	9024.58
bier_1	35038.96
bier_2	94055.91
bier_3	48698.64
a_1	1963.31
a_2	1688.13
a_3	1589.37

Table 10: Values used as an upper cut off in the branch-and-cut procedure

D Results obtained with the ILS procedure for the smaller instances

In the first two columns of table 11 it is possible to see the instance's name (*Instance*) and the optimal solution value (v^*). The rest of the table is divided in two parts. In the first part we report the results obtained by the heuristics proposed by Morán-Mirabal et al. (2014), namely the gap (gap_M) between the best solution obtained and the optimal value ($\%gap = 100 \times$ (heuristic value − optimal value)/optimal value) and the time ($time_M$), in seconds, to obtain the best solution. In the second part we present the results obtained using the ILS metaheuristic. For the reasons stated in section 4.2 we did five independent runs with the ILS procedure. Therefore, in table 11 it is possible to see the minimum percentage of gap (*min*), the average percentage of gap (*average*), the maximum percentage of gap (*max*) and the average time, in seconds (*time*), considering those five independent runs.

		Results Morán-Mirabal et al. (2014)		Results ILS			
Instance	v^*	gap_M	$time_M$	*minimum*	*average*	*maximum*	*time*
burma_1	13.93	0.00%	0	0.00%	0.00%	0.00%	0
burma_2	25.66	0.00%	0	0.00%	0.00%	0.00%	0
burma_3	11.89	0.00%	0	0.00%	0.00%	0.00%	0
bayg_1	5345.86	0.00%	3	0.00%	0.00%	0.00%	0
bayg_2	5791.01	0.00%	1	0.00%	0.01%	0.02%	0
bayg_3	5544.33	0.00%	0	0.00%	1.05%	2.17%	0
att_1	23686.00	0.00%	143	3.67%	3.82%	3.92%	0
att_2	20609.10	0.00%	63	0.00%	0.69%	1.66%	0
att_3	9024.58	0.00%	0	0.00%	0.00%	0.00%	0
bier_1	33709.70	9.17%	4	3.94%	6.30%	7.62%	4
bier_2	88736.40	10.01%	2966	5.99%	6.38%	6.57%	4
bier_3	47726.30	5.84%	1049	2.04%	2.53%	3.72%	3
average		2.08%	352	1.30%	1.73%	2.14%	1

Table 11: Results obtained with the ILS procedure for the smaller instances

E Current best known upper bounds for the existing benchmark instances

Table 12 contains two columns, one with the instance's name and other with the best known upper bound. The instances which the best known upper corresponds to the optimal value are marked with the symbol *.

Instance	Best known upper bound
burma_1	13.93*
burma_2	25.66*
burma_3	11.89*
bayg_1	5345.86*
bayg_2	5791.01*
bayg_3	5544.33*
att_1	23686.00*
att_2	20609.10*
att_3	9024.58*
bier_1	33709.70*
bier_2	88736.40*
bier_3	47726.30*
a_1	1891.16
a_2	1688.13
a_3	1408.14*
gr_1	1642.35
gr_2	1293.96
gr_3	1320.12
pr_1	145303.34
pr_2	159933.64
pr_3	137385.12

Table 12: Summary of the best known upper bounds