European Journal of Operational Research 261 (2017) 475–485

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Exact algorithms for the Equitable Traveling Salesman Problem☆

Joris Kinable[a,b], Bart Smeulders[c,1,*], Eline Delcour[d], Frits C. R. Spieksma[e]

[a] Robotics Institute & Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave Pittsburgh, PA 15213, USA
[b] KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
[c] University of Liège, HEC Management School, QuantOM, Rue Louvrex 14, 4000 Liège, Belgium
[d] Barry Callebaut, Aalstersestraat 122, 9280 Lebbeke, Belgium
[e] KU Leuven, Faculty of Business and Economics, ORSTAT, Naamsestraat 69, 3000 Leuven, Belgium

ARTICLE INFO

Article history:
Received 30 March 2016
Accepted 15 February 2017
Available online 24 February 2017

Keywords:
Combinatorial optimization
Traveling Salesman Problem
Branch-and-bound
Branch-and-price
Exact algorithms

ABSTRACT

Given a weighted graph $G = (V, E)$, the Equitable Traveling Salesman Problem (ETSP) asks for two perfect matchings in G such that (1) the two matchings together form a Hamiltonian cycle in G and (2) the absolute difference in costs between the two matchings is minimized. The problem is shown to be NP-Hard, even when the graph G is complete. We present two integer programming models to solve the ETSP problem and compare the strength of these formulations. One model is solved through branch-and-cut, whereas the other model is solved through a branch-and-price framework. A simple local search heuristic is also implemented. We conduct computational experiments on different types of instances, often derived from the TSPLib. It turns out that the behavior of the different approaches varies with the type of instances. For small and medium sized instances, branch-and-bound and branch-and-price produce comparable results. However, for larger instances branch-and-bound outperforms branch-and-price.

1. Introduction

We consider the following variation of the Traveling Salesman Problem (TSP). Given is an edge-weighted graph $G = (V, E)$, with $|V|$ even, and with edge-costs d_e for each $e \in E$. The cost of a matching in G is defined as the sum of edge-costs of the edges in the matching. The problem is to find two perfect matchings in G such that (i) the two matchings form a Hamiltonian cycle in G, and (ii) the absolute difference of the costs of these two matchings is minimum. Notice that a feasible solution need not exist. We call this problem the Equitable Traveling Salesman Problem, or ETSP for short.

This name is motivated by the following, more frivolous, description of our problem: two friends, in possession of a single bike, have agreed to jointly visit all given cities, i.e., to construct a tour. In addition, they have agreed to use the bike as follows: one friend rides (pedals) the bike, while the other sits on the bike's back. Directly after having visited a city, the two friends interchange roles. The objective in this problem is to find a tour such that the difference between the distances pedalled by each of the two friends, is minimum. (Clearly, one can also imagine two coach-drivers who swap driving at each stop.) An example of an instance of the ETSP is given in Fig. 1.

The ETSP belongs to a broader class of combinatorial problems, known as *balanced* optimization problems. Balanced optimization problems differ from regular optimization problems in the sense that, informally speaking, costs of different "parts" of the solution should be close to each other; this happens in situations where an equal or fair distribution of resources/costs is pursued. One consequence is that 0 is a lowerbound for any optimum value (which is not necessarily true in a regular optimization problem). The general mathematical form of balanced optimization problems can be stated as:

$$\underset{S \in \mathcal{F}}{\text{minimize}} \{\max_{s \in S} w(s) - \min_{s \in S} w(s)\} \quad (1)$$

where $\mathcal{F}$ is a family of feasible subsets (solutions) of some superset $\mathcal{S}$, and $w(s)$ a cost (weight) function which assesses the cost (weight) of element $s \in \mathcal{S}$. For instance, in the context of ETSP, superset $\mathcal{S}$ encompasses all possible perfect matchings in the graph G, and $\mathcal{F}$ comprises of all subsets $S \subseteq \mathcal{S}$ such that $|S| = 2$ and the two matchings in S form a Hamiltonian cycle. The objective

☆ This research has been partially funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. The research was carried out when the first author was at KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC - Belgium and the second author was at KU Leuven, Faculty of Business and Economics, ORSTAT.

* Corresponding author.
E-mail addresses: jkinable@cs.cmu.edu (J. Kinable), bart.smeulders@ulg.ac.be (B. Smeulders), frits.spieksma@kuleuven.be (F.C.R. Spieksma).

[1] The author is a post-doctoral fellow of the F.R.S.-FNRS.

http://dx.doi.org/10.1016/j.ejor.2017.02.017
0377-2217/

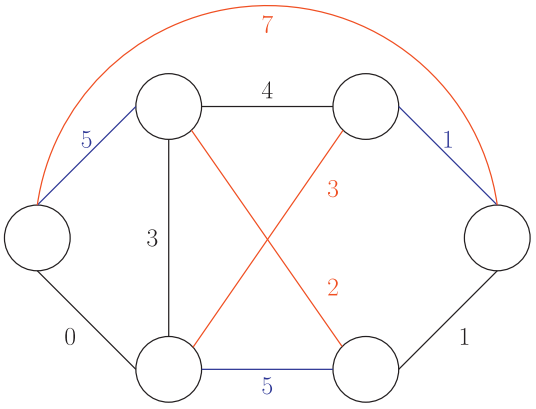

Fig. 1. Example of an ETSP instance and a solution. The perfect matching consisting of the blue edges has a cost of 12 and the perfect matching consisting of the red edges has a cost of 11. The value of the resulting solution equals 1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

function in (1) minimizes the imbalance: the absolute difference in costs between the most expensive and least expensive element in the solution.

A general class of balanced optimization problems as defined by Eq. (1) has been characterized first by Martello, Pulleyblank, Toth, and de Werra (1984). Their focus is on balanced optimization problems that are solvable in polynomial time. Many optimization problems identified in literature can be reduced to the aforementioned structure. A number of examples can be found in Becker (2010); Berežný and Lacko (2005); Camerini, Maffioli, Martello, and Toth (1986); Cappanera and Scutellà (2005); Delcour (2012); Ficker, Spieksma, and Woeginger (2016); Katoh and Iwano (1994); Martello et al. (1984); Zeitlin (1981) - most of these results either involve polynomial-time algorithms for specific balanced optimization problems, or heuristics for balanced optimization problems that are (NP-)hard. In this paper, we describe exact approaches for a particular balanced optimization problem that is NP-hard; we see the outcome of this study as a first step towards understanding the behavior of different types of approaches that can be applied to NP-hard balanced optimization problems.

A problem that is closely related to (but different from) the ETSP is the so-called *balanced* TSP, studied by Larusic and Punnen (2011). In the balanced TSP, one seeks to minimize the difference between the largest edge-cost and the smallest edge-cost of edges used in a tour. Larusic and Punnen (2011) develop several heuristics, mainly relying on lower and upper bounding procedures, to solve the balanced TSP. They mention that their algorithms can be used to solve an optimization problem originating in the maintenance of aircraft engines. It is not difficult to define a problem that is a generalization of both the ETSP and the balanced TSP. Indeed, given an edge-weighted graph G, and an integer k, consider the question: do there exist k disjoint matchings $M_1, M_2, \ldots, M_k$ with $M_i \subset E, |M_i| = \frac{|V|}{k}$ for $i = 1, \ldots, k$ such that $\cup_i M_i$ forms a Hamiltonian cycle in G? Observe that for $k = 2$ the ETSP arises, while for $k = |V|$ the balanced TSP arises (see Kinable (2014)).

Bassetto and Mason (2011) explore a periodic routing problem, where two tours of minimal total length through a number of nodes (customers) need to be found. Some customers must occur in both tours, others must occur in exactly one tour. The absolute difference between the number of customers in each tour is restricted from above, thereby obtaining two balanced tours. Notice that Bassetto and Mason (2011) do not consider reducing the imbalance between the two tours as part of their objective function; instead, the maximum allowed imbalance is part of the problem input and enforced through a constraint in the model. Using a similar approach to Bassetto and Mason (2011), Gouveia, Riera-Ledesma, and Salazar-González (2013) investigate a vehicle routing problem where the number of customers serviced by a vehicle is bounded from below and from above. Solutions to this problem are balanced in an indirect way, because each vehicle has approximately the same workload.

Another problem, which structurally bears strong resemblance to the ETSP problem, is the Market-Split problem (Aardal, Bixby, Hurkens, Lenstra, & Smeltink, 2000; Cornuéjols & Dawande, 1998). The Market-Split problem attempts to minimize the total amount of slack (positive and negative) which has to be added to a set of diophantine equations to make the system feasible. The problem is often introduced with the example of a company having two sales divisions responsible for supplying retailers with products. The objective is to allocate each retailer to either one of the divisions such that each division controls a predefined fraction of the market for a given product; deviation of these predefined fractions should be minimized. A similar structure is present in the ETSP.

The goal of this work is to investigate how to solve instances of ETSP to optimality. For that purpose, we study and compare exact solution methods based on two integer programming formulations for the ETSP. We will detail a branch-and-price approach for our problem, and compare it with a more traditional branch-and-bound method. In addition, we describe a local search method, and show how all these methods fare on different classes of instances of the ETSP.

The paper is organized as follows. Section 2 analyzes the complexity of the ETSP, showing that the problem, including some special cases, is NP-Hard. Section 3 introduces two integer programming formulations for the ETSP problem. A branch-and-price framework to solve one of these formulations is outlined in Section 4, and the local search algorithm is described in Section 5. Outcomes of computational experiments for the ETSP are reported in Section 6. Section 7 offers the conclusion.

2. Complexity analysis

In this section we analyze the computational complexity of the ETSP. We show that deciding whether a feasible solution exists to an instance of ETSP is NP-complete, and we show that, even for a complete graph, it is NP-hard to find an optimum solution. Let us formally state the decision version of ETSP:

Input: an undirected graph $G = (V, E)$, with $|V|$ even.
Goal: do there exist two disjoint perfect matchings M_1 and M_2 with $M_1 \subset E$ and $M_2 \subset E$ such that $M_1 \cup M_2$ is an Hamiltonian cycle in G?

It is not difficult to verify that the decision version of the ETSP is at least as hard as deciding whether a graph with an even number of nodes has a Hamiltonian cycle. Hence we can state the following theorem:

Theorem 1. *The decision version of ETSP is NP-complete.*

The optimization version of ETSP, simply referred to as ETSP, involves a cost d_e for each edge $e \in E$. Again, let us formally state the problem, where, for a subset of the edges $Q \subseteq E$, we use the following notation: $c(Q) = \sum_{e \in Q} d_e$.

Input: an undirected, weighted graph $G = (V, E)$, with edge costs d_e for all $e \in E$, and with $|V|$ even.
Goal: find two disjoint perfect matchings $M_1, M_2 \subset E$ such that $M_1 \cup M_2$ forms a Hamiltonian cycle in G, and such that $|c(M_1) - c(M_2)|$ is minimum.

Even for complete graphs, ETSP is a difficult problem:

Theorem 2. *ETSP is NP-hard for complete graphs.*

Proof. We use a reduction from Hamiltonicity: given an undirected graph $H = (W, F)$, does H contain a Hamiltonian cycle? We assume (wlog) that $|W|$ is even. We now build a complete, edge-weighted graph $G = (V, E)$ that forms the instance of ETSP. Let $V = W$, and for each edge $e \in F$, we introduce an edge $e \in E$, with edge cost $d_e = 0$. Consider now the edges not in F; we (arbitrarily) denote them by $\{e_1, e_2, \ldots, e_p\}$, with $p = |E \setminus F|$. Each of these edges is also present in E; we set $d_{e_j} = 2^j,\ j = 1, \ldots, p$. This completes the instance of ETSP.

We now argue that the existence of a Hamiltonian cycle is equivalent to the instance of ETSP having an optimum solution with value 0. Clearly, if there exists a Hamiltonian cycle in H then there is a unique decomposition of this cycle into two disjoint perfect matchings, each with cost 0. This leads to a solution of ETSP with value 0. On the other hand, suppose that the instance of ETSP has a solution with value 0. Thus, the difference between the costs of the two matchings forming a Hamiltonian cycle equals 0. Consider the most expensive edge in this pair of matchings, and denote its cost by d_{max}. If $d_{max} > 0$, then, by choice of the edge-costs, the sum of the edge-costs in the other matching will be less than d_{max}. Hence, no solution with value 0 exists. It follows that $d_{max} = 0$, which implies that all edges in the two matchings that form a Hamiltonian cycle have cost 0, leading to a Hamiltonian cycle in the graph H. □

We end this section by defining the concept of an *equitable* tour.

Definition 3. A tour is called *equitable* if the two matchings composing the tour have equal cost, i.e., if the value of the objective function of an instance of ETSP equals 0.

3. Formulations for the ETSP

In this section, we introduce two integer programming formulations for the ETSP. Model FBB (Section 3.1) is a direct adaptation of the well-known model proposed by Dantzig, Fulkerson, and Johnson (1954). We introduce a new formulation based on matchings in Section 3.2.

The integer programming formulations will treat a problem slightly more general than the ETSP, by allowing the edge-sets, and the edge-costs, to be identical for the two matchings. We use E_B and E_R to denote the two edge-sets; E_B refers to the 'blue' edges that can be used for one matching, while E_R refers to the 'red' edges to be used for the other matching. More precisely, we are given a graph $G = (V, E_B \cup E_R)$, and, let $\mathcal{M}_B$ ($\mathcal{M}_R$) refer to the set of perfect matchings in (V, E_B) ((V, E_R)). Each edge e in E_B (E_R) has a cost d_e^b (d_e^r). The cost of a matching $M \in \mathcal{M}_B$ is defined as $c^b(M) = \sum_{e \in M} d_e^b$. Analogously, the cost of a matching $M \in \mathcal{M}_R$ is $c^r(M) \equiv \sum_{e:e \in M} d_e^r$. Note that this problem definition does not require that $E_B \cap E_R = \emptyset$. Furthermore, a single edge $e \in E_B \cap E_R$ may have different weights in the red or the blue matching (i.e., $d_e^b = d_e^r$ does not necessarily hold). Finally, we define $\delta(S)$, $S \subseteq V$ as the set of edges having exactly one endpoint in S. Additionally, $\delta(v), v \in V$ is used as shorthand notation for $\delta(\{v\})$.

3.1. Formulation FBB

The first formulation uses a binary variable for each edge $e \in E_B$:

$$x_e^b := \begin{cases} 1 & \text{if edge } e \text{ is selected in the blue matching,} \\ 0 & \text{otherwise,} \end{cases}$$

as well as a binary variable for each edge $e \in E_R$:

$$x_e^r := \begin{cases} 1 & \text{if edge } e \text{ is selected in the red matching,} \\ 0 & \text{otherwise.} \end{cases}$$

$$FBB: \min \quad |\sum_{e \in E_B} d_e^b x_e^b - \sum_{e \in E_R} d_e^r x_e^r| \tag{2}$$

$$\text{s.t.} \quad \sum_{e \in \delta(v) \cap E_B} x_e^b = 1 \qquad \forall v \in V \tag{3}$$

$$\sum_{e \in \delta(v) \cap E_R} x_e^r = 1 \qquad \forall v \in V \tag{4}$$

$$x_e^b + x_e^r \leq 1 \qquad \forall e \in E_B \cap E_R \tag{5}$$

$$\sum_{e \in \delta(S) \cap E_B} x_e^b + \sum_{e \in \delta(S) \cap E_R} x_e^r \geq 2 \qquad \forall S \subset V, |S| \geq 3 \tag{6}$$

$$x_e^b \in \{0, 1\} \qquad \forall e \in E_B \tag{7}$$

$$x_e^r \in \{0, 1\} \qquad \forall e \in E_R \tag{8}$$

The formulation above is not linear due to the absolute value present in the objective function. A standard trick exists to make the formulation linear: using an additional variable, say w, adding two constraints of the form $w \geq \sum_{e \in E_B} d_e^b x_e^b - \sum_{e \in E_R} d_e^r x_e^r$ and $w \geq \sum_{e \in E_R} d_e^r x_e^r - \sum_{e \in E_B} d_e^b x_e^b$, and replacing the objective function by 'min w' makes the formulation linear. For reasons of compactness we use formulation (2)–(8). Constraints (3) and (4) ensure that each vertex is incident to exactly one edge from the blue matching and one edge from the red matching. Constraints (5) imply that each edge can be used by at most one matching. Constraints (6) model the subtour elimination constraints (notice that there is an exponential number of them). Observe that model FBB remains a correct formulation of the ETSP when all subtour constraints with $|S|$ odd are removed from the formulation; this is a consequence of the fact that any integral subtour must have an even number of edges due to the alternation of blue and red edges. We chose to keep the redundant subtour constraints with $|S|$ odd in the model, because they strengthen the LP-relaxation. Finally, constraints (7) and (8) are the integrality constraints.

When we speak of the solution of the linear relaxation of FBB (which we denote by LFBB), we refer to a solution (x_e^b, x_e^r) satisfying (3)–(6), $x_e^b, x_e^r \geq 0$, for which the value $|\sum_{e \in E_B} d_e^b x_e^b - \sum_{e \in E_R} d_e^r x_e^r|$ (which we denote by z_{BB}) is minimum.

Notice that other models for the TSP can be adapted in a similar way. In particular, time (position) indexed formulations used to model the time-dependent TSP (Godinho, Gouveia, and Pesneau (2014); Gouveia and Voß (1995)) only need minimal adaptations to model the ETSP problem. Conventional time-dependent models use variables $x_{e,t}$, with t an index indicating the position of the edge in the tour. The color of an arc can thus be determined by checking whether t is odd or even and only the objective function needs to be changed to account for the different objective function.

3.2. Formulation FBP

Our second formulation has a variable for each perfect matching in the graph. More precisely, we define a binary variable for each perfect matching $M \in \mathcal{M}_B$ in (V, E_B):

$$y_M^b := \begin{cases} 1 & \text{if perfect matching } M \text{ is selected as the blue matching,} \\ 0 & \text{otherwise,} \end{cases}$$

as well as a binary variable for each perfect matching $M \in \mathcal{M}_R$ in (V, E_R):

$$y^r_M := \begin{cases} 1 & \text{if perfect matching } M \text{ is selected as the red matching,} \\ 0 & \text{otherwise.} \end{cases}$$

$$FBP: \min \quad |\sum_{M\in\mathcal{M}_B} c^b(M)y^b_M - \sum_{M\in\mathcal{M}_R} c^r(M)y^r_M| \tag{9}$$

$$\text{s.t.} \quad \sum_{M\in\mathcal{M}_B} y^b_M = 1 \tag{10}$$

$$\sum_{M\in\mathcal{M}_R} y^r_M = 1 \tag{11}$$

$$\sum_{M\in\mathcal{M}_B:\ e\in M} y^b_M + \sum_{M\in\mathcal{M}_R:\ e\in M} y^r_M \le 1 \qquad \forall e \in E_B \cap E_R \tag{12}$$

$$\sum_{e\in\delta(S)} \Big(\sum_{M\in\mathcal{M}_B: e\in M} y^b_M + \sum_{M\in\mathcal{M}_R: e\in M} y^r_M \Big) \ge 2 \qquad \forall S \subset V, |S| \ge 3 \tag{13}$$

$$y^b_M \in \{0,1\} \qquad \forall M \in \mathcal{M}_B \tag{14}$$

$$y^r_M \in \{0,1\} \qquad \forall M \in \mathcal{M}_R \tag{15}$$

The same comment we made with respect to the linearity of formulation FBB applies to formulation FBP as well. Formulation FBP selects two perfect matchings by constraints (10) and (11). Constraints (12) ensure that an edge in the intersection of E_B and E_R can be used by at most one of the two matchings. Constraints (13) express the subtour elimination constraints, and constraints (14) and (15) are the integrality constraints. Observe that this formulation not only has an exponential number of constraints, it also has an exponential number of variables. We describe in Section 4 how we deal with this feature when solving this formulation.

When we speak of the solution of the linear relaxation of FBP (which we denote by LFBP), we refer to a solution (y^b_M, y^r_M) satisfying (10)–(13), $y^b_M, y^r_M \ge 0$, for which the value $|\sum_{M\in\mathcal{M}_B} c^b(M)y^b_M - \sum_{M\in\mathcal{M}_R} c^r(M)y^r_M|$ (which we denote by z_{BP}) is minimum.

3.3. Comparing formulations FBB and FBP

Let us compare the linear relaxations of formulations FBB and FBP. Recall that, when given an instance I of ETSP, $z_{BB}(I)$ ($z_{BP}(I)$) denotes the value of LFBB (LFBP) when applied to instance I. Following standard terminology (see Vielma (2015) and references contained therein), we say that the linear relaxation of FBP is *stronger* than the relaxation of FBB when the two following conditions are fulfilled:

C1: for each instance I of ETSP, $z_{BP}(I) \ge z_{BB}(I)$,
C2: there exists an instance I of ETSP for which $z_{BP}(I) > z_{BB}(I)$.

Theorem 4. *The linear relaxation of FBP (i.e., LFBP) is stronger than the linear relaxation of FBB (i.e., LFBB).*

Proof. First, to prove C1, we show that any feasible solution to LFBP corresponding to some instance I can be transformed into a feasible solution of LFBB, while the costs of these two solutions are equal.

Consider a feasible solution (y^b_M, y^r_M) of LFBP. Construct a solution to LFBB as follows:

$$\text{for each edge } e \in E_B, x^b_e := \sum_{M\in\mathcal{M}_B: e\in M} y^b_M, \tag{16}$$

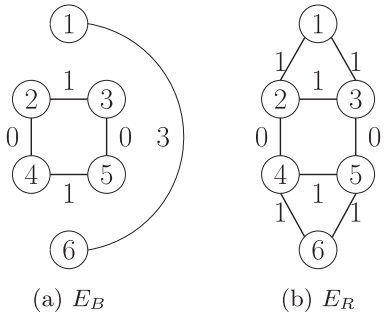

Fig. 2. Edge sets.

$$\text{for each edge } e \in E_R, x^r_e := \sum_{M\in\mathcal{M}_R: e\in M} y^r_M. \tag{17}$$

We need to show that (x^b_e, x^r_e) satisfies constraints (3)–(6), and that the solutions to LFBB and LFBP have an equal objective value. Let us consider constraints (3). Since all matchings are perfect matchings, it follows that each matching $M \in \mathcal{M}_B$ includes a single edge incident to each v. Consider some node $v \in V$. We have:

$$\sum_{e\in\delta(v)\cap E_B} x^b_e = \sum_{e\in\delta(v)\cap E_B} \sum_{M\in\mathcal{M}_B: e\in M} y^b_M = \sum_{M\in\mathcal{M}_B} y^b_M = 1 \tag{18}$$

The first equality follows from (16), and the second equality follows from the fact that the matchings in $\mathcal{M}_B$ that contain an edge e incident to node v are pairwise distinct matchings (since, by definition, no perfect matching can have two edges incident to node v). Finally, the last equality follows from (10). Thus, we conclude from the validity of (18) that constraint (3) is satisfied, and by a similar argument so is constraint (4).

Next, it is easily checked that constraint (5) is satisfied by (x^b_e, x^r_e), since constraint (12) is satisfied by (y^b_M, y^r_M); indeed, their left hand sides are equal by construction ((16) and (17)).

We now turn to the subtour elimination constraints. Observe that for each $S \subset V$, with $|S|$ even, and $4 \le |S| \le |V| - 4$:

$$\sum_{e\in\delta(S)\cap E_B} x^b_e = \sum_{e\in\delta(S)} \sum_{M\in\mathcal{M}_B:\ e\in M} y^b_M \text{ and } \sum_{e\in\delta(S)\cap E_R} x^r_e = \sum_{e\in\delta(S)} \sum_{M\in\mathcal{M}_R: e\in M} y^r_M. \tag{19}$$

It follows that

$$\sum_{e\in\delta(S)\cap E_B} x^b_e + \sum_{e\in\delta(S)\cap E_R} x^r_e = \sum_{e\in\delta(S)} \Big(\sum_{M\in\mathcal{M}_B: e\in M} y^b_M + \sum_{M\in\mathcal{M}_R: e\in M} y^r_M \Big) \ge 2.$$

Thus, when given a feasible solution (y^b_M, y^r_M), the solution (x^b_e, x^r_e) constructed using (16) and (17) is a feasible solution to LFBB.

It remains to show that both solutions have an equal objective value. This is true by construction:

$$\sum_{e\in E_B} d^b_e x^b_e = \sum_{e\in E_B} d^b_e \sum_{M\in\mathcal{M}_B: e\in M} y^b_M = \sum_{M\in\mathcal{M}_B} \Big(\sum_{e\in E_B} d^b_e \Big) y^b_M = \sum_{M\in\mathcal{M}_B} c^b(M) y^b_M.$$

We have now shown that if there exists a feasible solution (y^b_M, y^r_M) to LFBP, we can construct a feasible solution (x^b_e, x^r_e) with an equal value. It follows that, for each instance I, we have $z_{BP}(I) \ge z_{BB}(I)$.

Let us now turn to condition C2. We exhibit an instance I of ETSP, for which $z_{BP}(I) > z_{BB}(I)$. Let $|V| = 6$, $E_B = \{(v_1, v_6), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5)\}$ and $E_R = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5), (v_4, v_6), (v_5, v_6)\}$. We refer to Fig. 2 for an illustration of the instance where the edge-costs are depicted near the corresponding edges.

It can be checked $x^b_{16} = x^b_{24} = x^b_{35} = 1$ and $x^r_{12} = x^r_{13} = x^r_{23} = x^r_{45} = x^r_{46} = x^r_{56} = 0.5$ constitutes a feasible solution for LFBB, with $z_{BB}(I) = 0$ (see Fig. 3 showing this solution).

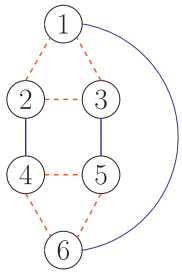

Fig. 3. Solution to LFBB. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

It is also true that there does not exist a solution feasible to LFBP with $z_{BP}(I) = 0$. To see this, notice that both $\mathcal{M}_B$ and $\mathcal{M}_R$ contain only two perfect matchings. Both matchings in $\mathcal{M}_B$ contain the edge (v_1, v_6). Thus, for each $M \in \mathcal{M}_B$, we have $c^b(M) \geq 3$ and $\sum_{M\in\mathcal{M}_B} c^b(M)y^b_M \geq 3$. For $\mathcal{M}_R$, both matchings have two edges with weight 1 and one edge with weight 0. Summing these, we find $\sum_{M\in\mathcal{M}_R} c^r(M)y^r_M = 2$. Thus, for each solution to this instance of LFBP, we have $z_{BP}(I) \geq 1$.

We have now shown that for each instance of ETSP, the value of an optimal solution of LFBB is at least as low as the value of an optimal solution to LFBP and that there exist instances for which the value of an optimal solution of LFBB is lower than for LFBP. We conclude that the linear relaxation of formulation FBP is stronger than the linear relaxation of formulation FBB. □

Although Theorem 4 suggests to prefer Formulation FBP over FBB, there is a relevant set of instances for which the value of the linear programming relaxations of the two models coincide:

Theorem 5. *If $E_B = E_R$ and if $d^b_e = d^r_e$ for each $e \in E$, then $v \equiv z_{BP}(I) = z_{BB}(I)$, with $v \in \{0, \infty\}$ for each instance I.*

Proof. Let I be an instance of ETSP, with $E_B = E_R$ and $d^b_e = d^r_e$ for each $e \in E$, for which there exists a feasible solution (x^b_e, x^r_e) to LFBB. Consider now the following solution $(\bar{x}^b_e, \bar{x}^r_e)$, with $\bar{x}^b_e = \bar{x}^r_e = (x^b_e + x^r_e)/2$. Observe that $(\bar{x}^b_e, \bar{x}^r_e)$ is both feasible (as $E_B = E_R$) and has value 0 (as $d^b_e = d^r_e$). We now claim that there exists a feasible solution to LFBP with value 0, as both $\bar{x}^b_e$ and $\bar{x}^r_e$ lie within the perfect matching polytope. Indeed, consider the following set of linear inequalities describing the perfect matching polytope (Edmonds (1965); Schrijver (2003)).

$$x_e \geq 0 \quad \forall e \in E. \tag{20}$$

$$\sum_{e\in\delta(v)\cap E} x_e = 1 \quad \forall v \in V \tag{21}$$

$$\sum_{e\in\delta(S)\cap E} x_e \geq 1 \quad \forall S \subset V \text{ with } |S| \text{ odd}. \tag{22}$$

Inequalities (20) and inequalities (21) are obviously satisfied. Indeed any feasible solution to LFBB must satisfy these, as they are equivalent to respectively (the linear relaxation of) constraints (7) and (4) for blue, and constraints (8) and (5) for red. Furthermore, since $E_b = E_r$, and $\bar{x}^b_e = \bar{x}^r_e$ for each edge, constraints (6) imply that

$$\sum_{e\in\delta(S)\cap E} 2\bar{x}^b_e \geq 2 \qquad \forall |S| \geq 3,$$

$$\sum_{e\in\delta(S)\cap E} 2\bar{x}^r_e \geq 2 \qquad \forall |S| \geq 3.$$

As a consequence, inequalities (22) are also satisfied. Since both $\bar{x}^b_e$ and $\bar{x}^r_e$ lie within the perfect matching polytope, they can both be described by a convex combination of perfect matchings (Edmonds (1965); Schrijver (2003)), represented by y^b_M, y^r_M respectively. These convex combinations form a solution to LFBP, which is equivalent to (x^b_e, x^r_e).

To conclude, we have shown (in the proof of Theorem 4) that the existence of a feasible solution to LFBP implies the existence of a feasible solution to LFBB with the same objective value. We have now shown that if $E_B = E_R$ and if $d^b_e = d^r_e$ for each $e \in E$, a feasible solution to LFBB implies the existence of a feasible solution with value 0 for both LFBB and LFBP. Thus, when $E_B = E_R$ and $d^b_e = d^r_e$ for all $e \in E$, there are only two possibilities: either both linear relaxations have a solution with value 0, or both linear relaxations are infeasible. □

Notice that there are two assumptions in Theorem 5. Each of these assumptions is necessary to achieve equality of $z_{BP}(I)$ and $z_{BB}(I)$: the example in the proof of Theorem 4 shows that $E_B = E_R$ is needed, and the instance described below implies that $d^b_e = d^r_e$ for each e is needed as well. For example, consider the following instance. Let

$$|V| = 6 \text{ and } E_b = E_r = \{(v_1, v_2), (v_1, v_3), (v_1, v_6), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5), (v_4, v_6), (v_5, v_6)\}.$$

The weight of all edges, for both red and blue is one, except for $d^r_{16} = d^r_{24} = d^r_{35} = 0$. It can be checked that a solution exists with $z_{BB}(I) = 0$. For LFBP, each blue matching has weight 3, and each red matching has a weight ≤ 2, thus $z_{BP}(I) > 0$.

Furthermore, we point out that, in case the instance I admits a feasible solution to ETSP, and assuming that $E_B = E_R$ and if $d^b_e = d^r_e$ for each e, the two values $z_{BP}(I)$ and $z_{BP}(I)$ not only coincide as predicted by Theorem 5, but are in fact equal to 0.

Finally, we note that the proof of Theorem 5 suggest a way of strengthening formulation FBB in such a way that it becomes equivalent to FBP. Imposing conditions (22) on both x^b_e and x^r_e implies all solutions can be described as convex combinations of perfect matchings, and thus that for any solution there exists a solution to LFBP with equal objective value.

4. A branch-and-price approach for solving model FBP

In this section, we describe how we solve model FBP using a branch-and-price approach. For a general description of this methodology, we refer to Desrosiers and Lübbecke (2011). There are a number of key ingredients in this approach: how to solve the pricing problem (Section 4.1), which branching rule to use (Section 4.2) and how to start (Section 4.3). We now give a detailed description of these.

4.1. Pricing problem

The pricing problem for the linear programming relaxation of formulation FBP amounts to establishing whether there exists a variable y^b_M (or y^r_M) with negative reduced costs. When we associate dual variables h, w, u_e and a_S to constraints (10), (11), (12), and (13) respectively, linear programming theory tells us that the reduced costs of variable y^b_M ($M \in \mathcal{M}_B$) equal:

$$c^b_M - h - \sum_{e\in M} u_e - \sum_{S\subset V, |S|\geq 3} |\delta(S) \cap M| a_S. \tag{23}$$

A similar expression can be written down for the reduced costs of variable y^r_M.

We claim that, given the dual variables, the existence of a variable with negative reduced costs can be detected by computing a minimum-weight perfect matching problem.

Lemma 6. *The pricing problem corresponding to formulation LFBP can be solved by computing a minimum-weight perfect matching.*

Proof. Consider the graph (V, E_B), and for each edge $e \in E_B$, introduce edge costs γ_e defined as follows:

$$\gamma_e = d^b_e - u_e - \sum_{S \subset V : e \in \delta(S), |S| \geq 3} a_S. \tag{24}$$

Notice that for an edge $e \in E_B \backslash E_R$, no u_e exists; in that case, this term disappears from the expression above. Suppose that we have a minimum weight perfect matching M^* in (V, E_1) with respect to the edge costs γ_e. We have, using (24):

$$\begin{aligned}\sum_{e \in M^*} \gamma_e &= \sum_{e \in M^*} (d^b_e - u_e - \sum_{S \subset V : e \in \delta(S), |S| \geq 3} a_S) \\ &= c^b(M^*) - \sum_{e \in M^*} u_e - \sum_{S \subset V : e \in \delta(S) \cap M^*, |S| \geq 3} a_S.\end{aligned}$$

Thus, if $\sum_{e \in M^*} \gamma_e < h$, it follows that $c^b(M^*) - \sum_{e \in M^*} u_e - \sum_{S \subset V : e \in \delta(S) \cap M^*, |S| \geq 3} a_S < h$ and, hence (23) implies that there is a variable with negative reduced costs. In addition, if $\sum_{e \in M^*} \gamma_e \geq h$, it is a fact that no variable y^b_M with negative reduced costs exists. □

We solve a pricing problem for each of the two edge sets. Strictly speaking it suffices to halt the pricing problem after a variable y^b_M with negative reduced costs is discovered; in practice, however, better results are obtained when multiple columns are returned simultaneously. Thus, in a single iteration, we add at most two variables to the model, at most one per color.

4.2. Branching

After solving the linear relaxation of FBP to optimality, the resulting solution may be fractional, that is, there may exist matching(s) $M \in \mathcal{M}_B$ with $0 < y^b_M < 1$ or matching(s) $M \in \mathcal{M}_R$ with $0 < y^r_M < 1$. Hence a branching rule is required. If there is a fractional solution, then there is an edge $e^* = (i, j)$ such that either $0 < \sum_{M \in \mathcal{M}_B : e^* \in M} y^b_M < 1$ or $0 < \sum_{M \in \mathcal{M}_R : e^* \in M} y^r_M < 1$ holds, or both. Indeed, one easily verifies that integrality of the y^b_M, y^r_M variables implies that each edge is either part of the blue matching, or part of the red matching, or not used at all. This allows us to specify the following branching rule. Suppose that, for some edge $e^* \in E_B$: $0 < \sum_{M \in \mathcal{M}_B : e^* \in M} y^b_M < 1$ holds. There exist two possibilities: either, in an incumbent solution edge $e^* = (i, j)$ is part of the blue matching, or it is not. In the former case all edges in E_B that are incident to vertices i or j (except edge $e^* = (i, j)$) are removed from E_B, and edge (i, j) (if present in E_R) is removed from set E_R. In the latter case, edge e^* is removed from set E_B. The case where $0 < \sum_{M \in \mathcal{M}_R : e^* \in M} y^r_M < 1$ works completely analogously.

This branching rule has the property that it solely affects the edge sets E_B and E_R. Thus, each node in the search tree corresponds to an instance solely defined by a specific graph. In our implementation, we select edge e^* based on the proximity to 0.5: the edge for which the value $\sum_{M \in \mathcal{M}_B : e^* \in M} y^b_M$ is closest to 0.5 is selected for branching.

The branching scheme is implemented using Depth First Search. We note that common branch-and-price techniques such as "Best Bound" and "Strong Branching" are not used in our implementation. Particular features of our problem make these much less effective in our application. In particular, (i) solving each node requires only a few iterations (usually less than 10, see tables in Section 6), and (ii) the lower bound at virtually every node equals 0; only nodes very deep in the search tree may yield a larger bound. Consequently, Best Bound and Strong Branching have little chance of guiding the branch-and-price. Best Bound is ineffective since the lower bounds are equal for all nodes. Strong Branching on the other hand requires performing a few column generation iterations and measuring the improvement. However, in our application performing a few iterations is equivalent to solving the entire node.

4.3. Initialization

Each node of the search tree must be initialized with a set of variables allowing a feasible solution to the linear relaxation of FBP, or one must show that, given the edge sets E_b and E_r no feasible solution exists, rendering the node infeasible.

Due to the presence of the subtour elimination constraints (13), and due to the fact that the graphs may be incomplete, it may be difficult to generate a set of variables (matchings) that admit a feasible solution satisfying constraints (10)–(13). And even if this would be possible, then this would require to solve a separate subproblem at each node of the search tree to establish a feasible initial solution. We avoid these issues by solving the linear relaxation of a slightly more general formulation of model FBP.

Indeed, by introducing a slack variable λ in the constraints (10), (11), (13), a model is obtained for which a trivial initial solution exists. Model FBP' is defined as follows:

$$FBP' : \min \quad |\sum_{M \in \mathcal{M}_B} c^b(M) y^b_M - \sum_{M \in \mathcal{M}_R} c^r(M) y^r_M| \quad + \lambda U \tag{25}$$

$$\text{s.t.} \quad \sum_{M \in \mathcal{M}_B} y^b_M + \lambda = 1 \tag{26}$$

$$\sum_{M \in \mathcal{M}_R} y^r_M + \lambda = 1 \tag{27}$$

$$\sum_{M \in \mathcal{M}_B : e \in M} y^b_M + \sum_{M \in \mathcal{M}_R : e \in M} y^r_M \leq 1 \qquad \forall e \in E_B \cap E_R \tag{28}$$

$$\sum_{e \in \delta(S)} (\sum_{M \in \mathcal{M}_B : e \in M} y^b_M + \sum_{M \in \mathcal{M}_R : e \in M} y^r_M) + 2\lambda \geq 2 \quad \forall S \subset V, |S| \geq 3 \tag{29}$$

$$y^b_M \in \{0, 1\} \qquad \forall M \in \mathcal{M}_B \tag{30}$$

$$y^r_M \in \{0, 1\} \qquad \forall M \in \mathcal{M}_R \tag{31}$$

In this model, original constraints (10), (11), (13) may be violated, but any violation is penalized in the objective, where $\lambda U > 0$ is the penalty incurred when $\lambda > 0$, with U being a fixed constant.

From the construction of model FBP' it is apparent that there is always a feasible solution having objective value U: simply set $y^r_M = y^b_M = 0$ for all $M \in \mathcal{M}_1 \cup \mathcal{M}_2$, and $\lambda = 1$. Furthermore, it is clear that the pricing problem can be solved through the implementation sketched in Section 4.1. Finally, observe that by setting $\lambda = 0$, FBP' becomes identical to the linear relaxation of FBP. In fact, FBP' only yields a feasible solution to the linear relaxation of FBP iff $\lambda = 0$. To prove infeasibility of FBP', we must guarantee that any solution in FBP' with $\lambda > 0$ is more expensive in terms of the objective value than any solution to FBP' with $\lambda = 0$. This can be achieved by setting U to a large value. However, it is well known that this reduces the stability of the column generation procedure and potentially introduces numerical issues while implementing the model. Recall that a node in the BPC tree can be pruned if either of the following conditions holds:

1. the node is infeasible
2. the lower bound on the node exceeds the upper bound, i.e., the best incumbent integer solution.

Hence it suffices to ensure that there does not exist a solution to FBP' with $\lambda > 0$ having an objective value less than W, where W is the value of the best incumbent integer solution or any other valid upper bound on the optimal objective value (see (32)). The latter can be achieved by the following procedure:

When the procedure terminates, either a solution to FBP' with $\lambda = 0$ is discovered, or a solution with $\lambda > 0$, $\omega \geq W$ is obtained in which case the node can be pruned.

Our models do not presume the existence of a feasible solution. In fact, we find that "interesting" instances are those that are in some sense close to the threshold between feasibility and infeasibility (See Section 6.3). However, it is clear that the existence of a perfect matching in (V, E_B) as well as the existence of a perfect matching in (V, E_R) are necessary conditions for the existence of a feasible solution. We do assume that each instance satisfies these necessary conditions. Moreover, we use this assumption to establish an upper bound W on the value of a feasible solution (if one exists); and if no (heuristically obtained) feasible solution is available, we use this upper bound at the root node of the search tree. The upper bound W is computed as follows:

$$W = \max\{\max_{M\in\mathcal{M}_B} c^b(M) - \min_{M\in\mathcal{M}_R} c^r(M),\ \max_{M\in\mathcal{M}_R} c^r(M) - \min_{M\in\mathcal{M}_B} c^b(M)\} + 1 \tag{32}$$

This bound is easy to calculate by computing both a min cost and a max cost perfect matching on graphs $G(V, E_B)$ and $G(V, E_R)$.

5. Local search

In order to asses the potential of simple heuristics for ETSP, we implemented a greedy constructive heuristic, as well as a local search algorithm. We use the outcomes of this algorithm to better understand the difficulty of different classes of instances.

The following heuristic is used to build a feasible solution from scratch. We say that two edges $e_1 \in E_B$, $e_2 \in E_R$ are adjacent if both e_1 and e_2 are incident to some node $v \in V$. First we identify the pair of adjacent edges (e_1, e_2), whose costs are as equal as possible, i.e., for which $|d^b_{e_1} - d^r_{e_2}|$ is minimum; we see this pair as a partial tour. In an iterative fashion, we extend this partial tour by adding, in each iteration, a pair of adjacent edges to the partial tour. In each iteration (apart from the final iteration), we select that pair of adjacent edges which results in a minimal solution value of the resulting partial tour, but which does not form a subtour. In the final iteration we select the pair of adjacent edges resulting in a complete tour with the minimal solution solution value. Notice that this procedure is not guaranteed to find a feasible solution even if one exists; however for a complete graph it will always find a feasible solution. We come back to this issue at the end of the section.

The local search algorithm is based on the well-known 2-opt neighborhood for the TSP, which goes back to Croes (1958). Since, in ETSP, the color of the edges must alternate, we distinguish two versions of this neighborhood which depend on whether the two removed edges have the same color, or not. Consider a feasible tour containing blue edges $(1, 2), (3, 4), \ldots, (n-1, n)$ and red edges $(2, 3), (4, 5), \ldots, (n, 1)$. In mono-color 2-opt, we remove, given a feasible solution the ETSP, two identically colored edges from the tour, say (i, j) and (k, l). A new tour is then formed by adding two edges of the same color, say (i, k) and (j, l).

In bi-color 2-Opt, we remove two distinctly colored edges from the tour. Observe that simply reconnecting the edges as in mono-color 2-Opt is not possible, as no combination of red and blue colorings of the new edges will satisfy the alternating color property. Consider a move where one blue edge (c, d) and one red edge (k, l) are removed from the tour. The remaining edges form paths from d to k and from l to c. There are two ways to reconnect the tour, either by adding a blue edge (c, k) and a red edge (d, l) or a red edge (c, k) and a blue edge (d, l). In the first case, the color of all edges $(d, e), \ldots, (j, k)$ must be switched, in the second case this is true for all edges $(l, m), \ldots, (b, c)$.

We use these neighborhoods in a simple greedy local search algorithm. For a given solution, these neighborhoods are searched until a better solution is found. This better solution then becomes the incumbent solution and in turn, its neighborhoods are searched for a better solution.

It is natural to investigate a generalization of mono-color 2-Opt in the following sense: while fixing the blue matching, consider all red matchings that, together with the blue matching, yield a feasible solution to ETSP. This is a larger neighborhood than mono-color 2-opt, and hence potentially more interesting. Unfortunately, deciding whether a given matching can be extended to a tour is NP-complete, see Bienkowski and Zalewski (2013). It follows that finding the best matching in this neighborhood is hard, and hence, searching through this neighborhood does not seem to be an attractive option.

To ensure we can use the greedy and local search algorithms for incomplete graphs, we use the following procedure. Let H be the weight of the heaviest blue or red edge in absolute value ($H = \max_{e\in E}\{|d^b_e|, |d^r_e|\}$).

For each $e \notin E_b$, set $d^b_e = nH$.
For each $e \notin E_r$, set $d^r_e = -nH$.

This choice of costs ensures that if the local search algorithm terminates with a solution whose value is smaller than nH, a feasible solution to the original instance is found, and vice versa.

6. Computational experiments

In this section we first give some details concerning the implementation of the two integer programming models (Section 6.1). Then, in Section 6.2, we describe the instances that we use for our experiments whose outcomes are reported in Section 6.3.

All experiments are conducted on an Intel Core i7-4790 (3.6 gigahertz) machine. ILOG Cplex version 12.6.3 has been used to solve the Linear and Integer Programming problems. For Cplex, we used the default settings and a maximum of 2 solver threads. The Branch-and-Price approach is implemented through the Branch-and-Price framework provided by the COIN-OR Java Operations Research Library [JORLIB] version 1.1.1. Again, we used the default settings and 2 solver threads. The pricing problem of the BPC approach (min cost perfect matching problem) has been solved using the implementation in the COIN-OR LEMON library [LEMON]. Separation of subtours in both LFBB and LFBP is implemented through the min-cut routines in JgraphT version 1.0 [JGRAPH]; more details about the separation routines are provided in the next section.

6.1. Separation of valid inequalities

The presence of an exponentially large number of constraints in both models FBB (Eq. (6)) and FBP (Eq. (13)) render these models impractical to solve. Instead of adding all these constraints a-priori to the models, separation routines are used to identify violated inequalities which are added to the models during the the search. Recall that any separation routine for model FBB can also be applied to model FBP, since a solution (z^b_M, z^r_M) to FBP can be translated to an x-solution using (16) and (17).

To separate the subtour inequalities, a min-cut procedure is used. Experiments have been conducted to determine the optimal frequency at which subtour cuts are separated: at every node, only at nodes where the solution is integral, or both at the root node

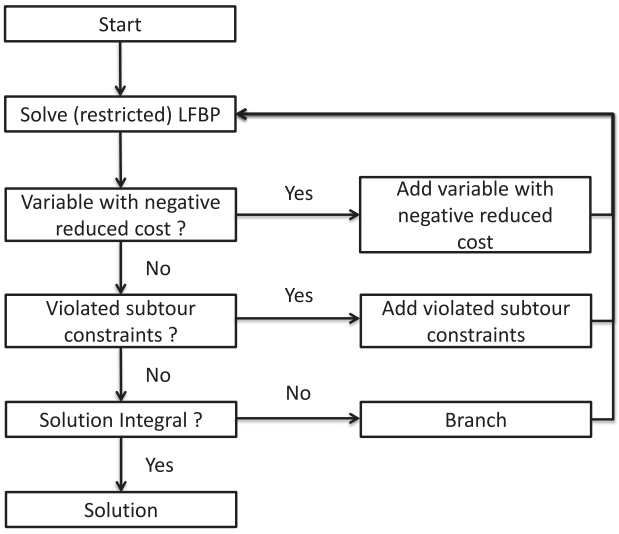

Fig. 4. A schematic overview of the branch-and-price algorithm.

and integral nodes. Similarly, experiments were conducted to determine how many cuts should be separated: a single, most violated cut, or more than one cut. To separate the most violated cut, a global min-cut problem is solved, using the Stoer–Wagner min-cut procedure. To separate multiple violated cuts, a separation procedure described in Hong (1972) is used, requiring $|V| - 1$ minimum s-t cut problems to be solved. More precisely, a minimum s-t cut problem is solved for a given source vertex $s \in V$ and all sink vertices $t \in V \backslash \{s\}$. The following variations were tested:

- Separate all violated cuts, but add at most 20 of the most violated cuts to the model.
- Terminate the separation procedure early, that is, terminate the separation routine as soon as 20 violated cuts have been found.
- Separating both the most violated cut using a Stoer–Wagner Minimum cut procedure, as well as up to 19 additional violated cuts.

Computational results for FBB and FBP revealed that only separating the most violated cut yielded the best results. Determining the 20 strongest cuts simply takes too much time; separating only a limited number of violated cuts resulted in too many weak cuts which drastically increases the size of the models. A future implementation could consider more advanced separation schemes for the DFJ subtour elimination constraints, for details, see Applegate, Bixby, Chvátal, and Cook (2003). For FBB, we obtained the best results when subtour elimination constraints were only separated at integral nodes, thereby minimizing the computational overhead induced by the separation routines. Note that these results deviate from common solution approaches for the traditional TSP, where multiple cuts are generated at each node of the branch-and-bound tree. Finally, for the FBP, the best solutions were obtained when subtour inequalities are separated each time the pricing problem fails to identify new columns with negative reduced cost. A schematic overview of the branch-and-price procedure can be found in Fig. 4.

6.2. Instances

In the computational experiments, we use four sets of instances, named instances of Type 1, Type 2, Type 3, and Type 4. The instances of Types 1, 2, and 3 are based on TSPLib instances. Recall that the number of vertices in an instance of the ETSP needs to be even. If this is not the case for a particular TSPLib instance, we remove the last vertex.

Table 1
Results for instances of Type 1.

Name of instance	obj	$t(s)$	# Optimal random tours
burma 14	0	0.00	3
ulysses 16	0	0.00	1
ulysses 22	0	0.00	2
berlin 52	0	0.01	1
eil 76	0	0.00	35
gr 96	0	0.02	0
gil 262	0	0.01	6
pr 264	0	0.03	1
lin 318	0	0.03	2

Type 1: For instances of Type 1, we use the nine smallest instances from TSPLib having between 14 and 318 nodes. Since the instances from TSPLib are geometric, the instances feature a complete graph with $E \equiv E_B = E_R$, and $d^b_e = d^r_e$ for each $e \in E$.

Type 2: In order to study the behavior of our models on more interesting instances, we create the instances Type 2 by removing edges from instances of Type 1. More in particular, for an instance of Type 1, we remove a certain number of the most expensive edges, thereby obtaining an incomplete graph $G(V, E)$. An edge (i, j) can only be removed if the remaining graph remains Hamiltonian; iteratively removing a single edge from the graph produces a new instance in each iteration. We only include instances (i) with a non-zero optimum value, and (ii) for which the optimal solution changed when compared to the solution found in the previous iteration. In total, this procedure gives us 29 instances of Type 2. Recall that these instances still have $E_B = E_R$, and $d^b_e = d^r_e$ for each $e \in E$.

Type 3: Instances of Type 3 are constructed using the asymmetric TSP instances from TSPLib. The original TSPLib instances are defined on complete graphs. To construct our instances, we give each edge a 20% chance of only being in E_B, a 20% chance of only being in E_R, and a 10% chance of being a member of both. Furthermore, for an edge (i, j) we set $d^b_{(i,j)} = d(i, j)$ and $d^r_{(i,j)} = d(j, i)$. In this way, we construct 19 instances for which $E_B \neq E_R$, and for which there exist e with $d^b_e \neq d^r_e$.

Type 4: Finally, instances of Type 4 are generated using complete graphs, with $E = E_B = E_R$. For each edge $e \in E$, the blue costs d^b_e are randomly generated from a uniform distribution between 201 and 300, while the red costs d^r_e are randomly generated from a uniform distribution between 1 and 100. As such, these instances have the property that for each edge $e \in E$, $d^b_e \neq d^r_e$ and the value of an optimum solution (as well as the value of the linear relaxations) can not be equal to 0. We use multiples of 100 for our instance size, up to 1600 vertices, and we generate 5 graphs for each value of $|V|$, leading to 80 instances in total.

All instances are available online at http://www.econ.kuleuven.ac.be/public/NDBAE03/ETSP_instances.zip

6.3. Experimental results

Let us first consider the instances of Type 1. The results of the local search algorithm are given in Table 1, where the column called 'obj' denotes the value of the solution found, where the column '$t(s)$' stands for the computation time needed (in seconds), and where the final column gives the number of optimum solutions among a set of 10.000 randomly generated tours.

From Table 1 it follows that instances of Type 1 are not difficult to solve. Indeed, the (simple) local search algorithm finds an optimum solution for each instance in negligible computing times. Perhaps even more telling, simply randomly generating a moderate number of tours will produce an optimum solution. Therefore, we chose not to run the exact approaches on instances of Type 1.

Table 2
Results for instances of Type 2.

Name of	Number of		Local search		Model FBB		Model FBP				
instance	remaining edges	OPT	obj	$t(s)$	$t(s)$	nodes	$t(s)$	t_{master}	$t_{pricing}$	nodes	cols
burma14_r50.rtsp	41	1	18	0.00	0.16	995	3.92	1.23	1.35	198	344
burma14_r54.rtsp	37	22	INF	0.00	0.13	56	0.71	0.24	0.29	32	52
burma14_r55.rtsp	36	35	35	0.00	0.10	11	0.29	0.11	0.11	11	17
burma14_r56.rtsp	35	97	202	0.00	0.11	0	0.16	0.08	0.04	5	9
burma14_r57.rtsp	34	234	234	0.00	0.06	0	0.10	0.03	0.04	2	3
burma14_r63.rtsp	31	258	INF	0.00	0.05	0	0.07	0.03	0.02	2	2
burma14_r71.rtsp	25	305	INF	0.00	0.03	0	0.06	0.02	0.02	3	2
ulysses16_r81.rtsp	38	1	12	0.00	0.11	266	2.76	1.09	0.92	180	241
ulysses16_r90.rtsp	32	10	INF	0.00	0.12	47	0.68	0.32	0.21	19	26
ulysses16_r103.rtsp	23	14	INF	0.00	0.02	0	0.17	0.10	0.04	1	2
ulysses22_r186.rtsp	50	1	INF	0.00	0.10	156	1.82	0.69	0.75	72	87
ulysses22_r191.rtsp	45	97	INF	0.00	0.08	22	0.17	0.05	0.10	1	2
ulysses22_r204.rtsp	36	101	INF	0.00	0.05	0	0.09	0.04	0.03	2	2
ulysses22_r216.rtsp	28	157	INF	0.00	0.04	0	0.06	0.02	0.03	2	2
berlin52_r1277.rtsp	82	2	INF	0.00	0.19	162	5.65	2.91	2.08	92	94
berlin52_r1285.rtsp	77	12	INF	0.00	0.15	5	0.52	0.31	0.14	3	4
berlin52_r1293.rtsp	70	28	INF	0.00	0.19	3	0.31	0.16	0.09	2	2
berlin52_r1299.rtsp	64	37	INF	0.00	0.18	0	0.24	0.14	0.06	1	2
eil76_r2744.rtsp	119	3	INF	0.01	0.25	5	1.48	0.75	0.55	8	6
eil76_r2751.rtsp	116	10	INF	0.01	0.18	3	1.39	0.67	0.57	12	6
gr96_r4415.rtsp	168	29	INF	0.01	0.46	226	1.15	0.62	0.39	1	2
gr96_r4421.rtsp	165	341	INF	0.02	0.37	5	1.15	0.63	0.44	2	2
gr96_r4439.rtsp	156	387	INF	0.01	0.42	7	1.03	0.67	0.28	4	5
gr96_r4447.rtsp	149	432	INF	0.02	0.38	5	2.14	1.14	0.78	8	12
gr96_r4456.rtsp	146	688	INF	0.01	0.49	5	0.45	0.28	0.12	1	2
gr96_r4460.rtsp	144	994	INF	0.01	0.43	7	0.53	0.30	0.17	1	2
gr96_r4479.rtsp	130	1013	INF	0.01	0.36	0	0.42	0.24	0.13	2	2
gr96_r4495.rtsp	123	1016	INF	0.02	0.28	0	0.43	0.27	0.09	2	2
gr96_r4518.rtsp	113	1199	INF	0.02	0.12	0	0.23	0.15	0.04	2	2

Consider now the instances of Type 2. Outcomes of the local search algorithm, model FBB, and model FBP are reported in Table 2, where the second column stands for the number of edges present in the instance, where the column called 'OPT' stands for the optimum value, where a column called 'nodes' stands for the number of nodes in the search tree corresponding to that instance, where a column called 'cols' stands for the number of columns generated in the branch-and-price, and where the column called 't_{master}' ('$t_{pricing}$') stands for the time spent solving the master problem (the pricing problem).

From this table, we conclude that the local search algorithm is no longer effective: for the majority of instances it fails to find even a feasible solution (while there exists one). The two exact approaches are very efficient on these instances, and find an optimum solution usually within a second, and always within four seconds. On most instances, model FBB is slightly faster, even though the number of nodes in the search tree of model FBP is usually less than the corresponding number for model FBB. Notice that Theorem 4 tells us that the values of the linear programming relaxations of these models equal 0 for instances of Type 2.

Let us now turn to instances of Type 3, whose outcomes are reported in Table 3. In addition to the information specified in Table 2, we now also give z_{BB} and z_{BP}, the LP-relaxations of the two models, for these instances. First, the local search algorithm works reasonably well, finding an optimum solution for the majority of instances, including one not solved to optimality by the exact approaches. Second, model FBB solves all instances much faster than model FBP. In fact, since we allotted each model 1800 seconds for each instance, there are eight instances that model FBP cannot solve within this time-limit (denoted by "TL" in the time column), while only one instance is not solved by model FBB. Interestingly, the values of the linear programming relaxations are equal except for one particular instance (although Theorem 4 does not apply for instances of Type 3).

Finally, we look at instances of Type 4 in Table 4. (Results for the individual instances can be found in the online appendix). For the exact methods, the reported numbers are averages over the instances that are solved to optimality. For the heuristic, computation times are reported over all instances. Optimality gaps are reported based on the optimal solution, or the best lower bound found from the execution of models FBB and FBP if no optimal solution was found. The simple local search method works reasonable well, finding solutions about 3% above the optimum value for the smallest instances. For the larges instances, the gap closes to below 0.5%. Model FBP is capable of solving all instances up to 800 vertices, Model FBB up to 1200 instances. Computations times are comparable for smaller instances, but as instance sizes grow, model FBB becomes noticeably faster. Model FBB is also more often capable of finding an optimal solution within the time limit. Notice that in most cases the Cplex implementation of model FBB solves the problem without any branching. For instances for which the optimal solution is known, the value of the LP relaxation of model FBP is equal to the optimal value in all but a few cases. If the LP relaxations deviates, the gap is typically in the order of 0.001%. The linear relaxation of model FBB is similarly strong, although there are 11 instances for which the FBB LP relaxation is weaker than the LP relaxation of FBP.

When studying the performance of the methods we implemented, we conclude as follows:

- The simple local search algorithm works when the instances are dense enough. Indeed, instances of Type 2 that are on the border of feasibility and infeasibility are not handled well by the local search, whereas instances of the other types are solved to optimality (Type 1 and Type 3), or within a reasonable deviation (Type 4).
- It is apparent that model FBB is faster than model FBP on instances of Type 3 and larger instances of Type 4; however, for instances of Types 2 and smaller instances of Type 4, the performance is comparable. We note that the reported averages for instances of Type 4 hide significant variance in the computation

Table 3
Results for instances of Type 3.

		Local search		Model FBB			Model FBP					
Name	OPT	obj	$t(s)$	z_{BB}	$t(s)$	nodes	z_{BP}	$t(s)$	t_{master}	$t_{pricing}$	nodes	cols
br17	0	INF	0.00	0	0.12	30	0	0.89	0.34	0.32	109	179
ftv33	0	INF	0.00	0	0.42	1623	0	6.53	3.67	1.64	639	1097
ftv35	0	0	0.00	0	0.73	1100	0	7.31	4.32	1.84	673	1629
ftv38	0	1	0.00	0	0.83	1517	0	2.65	1.75	0.36	247	245
p43	7611	7910	0.00	7606	0.17	0	7611	0.22	0.12	0.06	3	12
ftv44	0	0	0.00	0	0.52	512	0	6.47	4.30	1.30	501	1077
ftv47	0	7	0.00	0	0.95	750	0	12.60	8.54	2.26	995	1780
ry48p	0	17	0.00	0	0.99	723	0	6.48	4.69	0.81	591	740
ft53	0	2	0.00	0	0.77	552	0	45.71	29.49	10.74	3223	9291
ftv55	0	2	0.00	0	0.63	383	0	9.69	7.77	0.80	809	792
ftv64	0	2	0.00	0	1.35	460	0	46.86	37.64	5.91	1771	3762
ft70	0	2	0.00	0	2.97	1713	0	92.22	67.37	13.20	4737	6907
ftv70	0	0	0.00	0	1.53	246	0	23.22	19.39	1.91	1175	1401
kro124p	0	0	0.01	0	7.34	2204	0	248.4	165.0	73.40	3101	22315
ftv170	0	1	0.05	0	12.13	1855	0	328.7	302.2	13.33	2075	1965
rbg323	0	0	0.04	0	16.14	0	0	TL	1739	33.14	649	1166
rbg358	0	0	0.04	0	106.31	799	0	542.1	524.0	9.69	291	292
rbg403	0	0	0.06	0	132.1	526	0	TL	1752	28.21	309	615
rbg443	0	0	0.08	0	301.6	1889	0	TL	1757	25.88	247	487
dc563	0	0	0.09	0	641.6	2848	0	TL	1718	62.64	171	348
atex8(600)	0	0	0.30	0	976.8	6384	0	TL	1751	34.73	128	262
dc849	0	0	0.41	0	305.0	43	0	TL	1775	28.48	51	108
dc895	0	0	0.48	0	891.4	1936	0	TL	1694	112.8	44	94
dc932	0	0	0.50	0	TL	2312	0	TL	1712	75.58	40	94

Table 4
Results for instances of Type 4 (averages).

	Local search		Model FBB			Model FBP					
Instance size	t(s)	Opt.Gap	# Solved	t(s)	nodes	# Solved	t(s)	t_master	t_pricing	Cols	nodes
100	0.01	0.026	5	1.38	6.8	5	1.92	1.53	0.35	36.6	7.0
200	0.09	0.019	5	5.56	3.8	5	6.38	5.49	0.76	24.6	7.4
300	0.27	0.014	5	15.93	0.0	5	24.36	22.30	1.74	24.4	10.2
400	0.74	0.012	5	31.11	4.0	5	67.75	63.63	3.48	27.8	12.2
500	1.97	0.008	5	37.02	1.8	5	108.06	101.85	5.23	0.0	0.0
600	3.43	0.007	5	193.51	79.0	5	471.66	455.51	13.11	39.0	25.8
700	7.07	0.006	5	212.39	12.6	5	704.61	682.24	19.06	35.8	21.8
800	8.76	0.005	5	146.77	1.0	5	275.53	262.34	12.04	13.2	6.2
900	14.9	0.005	5	793.17	98.2	3	768.69	749.77	16.20	15.7	11.0
1000	24.3	0.004	5	859.54	0.0	3	162.47	152.70	9.53	6.3	1.7
1100	34.2	0.003	5	610.36	129.2	2	800.32	785.68	12.50	7.0	6.0
1200	74.1	0.003	5	1205.63	0.0	1	234.68	221.16	13.51	6.0	1.0
1300	83.6	0.003	3	751.26	0.0	0					
1400	104	0.003	2	962.67	0.0	2	1027.30	994.24	31.68	11.5	3.0
1500	162	0.002	3	955.23	18.9	1	543.69	514.28	29.39	11.0	1.0
1600	186	0.002	1	408.74	0.0	3	951.36	918.84	30.69	10.3	2.3

times for model FBP. There are several larger instances which model FBP can solve within the time limit, whereas model FBB can not, or where model FBP is faster.

- The lack of strong bounds severely hampers model FBP when solving instances of Type 3 (and to a lesser extent Type 2). At each non-leaf node, a lower bound of zero is encountered. Only at the leaves, non-zero integer solutions are discovered. Consequently it is very difficult to guide the search or to cut off branches due to the weak lower bounds. Indeed, theoretically, the main advantage of model FBP is the stronger linear relaxation, and model FBB only obtains better results on instances where the linear relaxation for both formulations is 0. For instances where the linear relaxation does come into play, results for both approaches are comparable.
- The majority of time, when using model FBP, is spent on solving the master problems and separating the subtour inequalities.
- A branching decision is made as soon as the master problem reaches a feasible solution equal to the lower bound of the node in the search tree corresponding to model FBP. Typically, this often happens after only a few iterations. Hence, the number of iterations, as well as the number of generated columns per node in the search tree is very low.
- The penalty update scheme (Algorithm 1) works well: the number of penalty updates is significantly lower than the number of nodes in the search tree corresponding to model FBP, meaning that for the majority of nodes no updates are required.

Algorithm 1: Penalty update procedure for FBP'.

```
1 U = W;
2 repeat
3     solve FBP'. Let ω be the resulting objective value;
4     if λ = 0 then
5         Feasible solution for the linear relaxation of FBP has
          been found with objective value ω;
6     else                                   /* 0 < λ! ≤ 1 */
7         U := U/λ ;                         /* Increase penalty */
8 until λ = 0 ∨ ω ≥ W + 1;
```

This approach is computationally much cheaper than using a dedicated method to generate a feasible initial solution at each node of the tree (or to prove that such a solution does not exist).

7. Conclusion

We have introduced and analyzed the Equitable Traveling Salesman Problem (ETSP). We have shown that the problem is NP-Hard, even when the graph is complete. Two integer programming models are presented for the ETSP, and we compare their linear programming relaxations. One model can be solved through a traditional branch-and-cut approach, whereas the other model is embedded in a branch-and-price framework. The pricing problem in the branch-and-price approach amounts to finding a minimum weight matching. Computational experiments on adapted TSPLib instances show that the best results are obtained with the branch-and-cut approach.

Acknowledgment

We thank the referees for their constructive and helpful remarks.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ejor.2017.02.017

References

Aardal, K., Bixby, R. E., Hurkens, C. A. J., Lenstra, A. K., & Smeltink, J. W. (2000). Market split and basis reduction: Towards a solution of the Cornuéjols-Dawande instances. *INFORMS Journal on Computing, 12*(3), 192–202.

Applegate, D., Bixby, R., Chvátal, V., & Cook, W. (2003). Implementing the Dantzig–Fulkerson–Johnson algorithm for large traveling salesman problems. *Mathematical Programming, 97*(1), 91–153.

Bassetto, T., & Mason, F. (2011). Heuristic algorithms for the 2-period balanced traveling salesman problem in euclidean graphs. *European Journal of Operational Research, 208*(3), 253–262.

Becker, K. H. (2010). *Twin-constrained hamiltonian paths on threshold graphs*. London School of Economics Ph.D. thesis.

Berežný, Š., & Lacko, V. (2005). The color-balanced spanning tree problem. *Kybernetika, 41*(4), 539–546.

Bienkowski, M., & Zalewski, P. (2013). (1, 2)-hamiltonian completion on a matching. *International Journal of Foundations of Computer Science, 24*(01), 95–108.

Camerini, P. M., Maffioli, F., Martello, S., & Toth, P. (1986). Most and least uniform spanning trees. *Discrete Applied Mathematics, 15*(23), 181–197.

Cappanera, P., & Scutellà, M. G. (2005). Balanced paths in acyclic networks: Tractable cases and related approaches. *Networks, 45*(2), 104–111.

Cornuéjols, G., & Dawande, M. (1998). A class of hard small 0–1 programs. In R. Bixby, E. Boyd, & R. Rıos-Mercado (Eds.), *Integer programming and combinatorial optimization: (Vol.1412* (pp. 284–293)). Springer Berlin Heidelberg.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research, 6*(6), 791–812.

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

Delcour, E. (2012). *Two salesmen and a bike*. Faculty of Business and Economics, KU Leuven Master's thesis.

Desrosiers, J., & Lübbecke, M. E. (2011). *Branch-price-and-cut algorithms*. Wiley encyclopedia of operations research and management science.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0, l-vertices. *Journal of Research of the National Bureau of Standards B, 69*, 125–130.

Ficker, A., Spieksma, F. C., & Woeginger, G. J. (2016). *Balanced vector optimization*. KUL. Research Report.

Godinho, M. T., Gouveia, L., & Pesneau, P. (2014). Natural and extended formulations for the time-dependent traveling salesman problem. *Discrete Applied Mathematics, 164*, 138–153.

Gouveia, L., Riera-Ledesma, J., & Salazar-González, J.-J. (2013). Reverse multistar inequalities and vehicle routing problems with a lower bound on the number of customers per route. *Networks, 61*(4), 309–321.

Gouveia, L., & Voß, S. (1995). A classification of formulations for the (time-dependent) traveling salesman problem. *European Journal of Operational Research, 83*(1), 69–82.

Hong, S. (1972). *A linear programming approach for the traveling salesman problem*. Johns Hopkins University.

JgraphT JgraphT. http://jgrapht.org.

JORLIB Java OR Library v1.1. http://www.coin-or.org/projects/jORLib.xml.

Katoh, N., & Iwano, K. (1994). Efficient algorithms for minimum range cut problems. *Networks, 24*(7), 395–407.

Kinable, J. (2014). *Decomposition approaches for optimization problems*. Science, Engineering and Technology Group, Campus Kulak Kortrijk, Computer Science, Campus Kulak Kortrijk, Faculty of Engineering Science Ph.D. thesis.

Larusic, J., & Punnen, A. P. (2011). The balanced traveling salesmanproblem. *Computers and Operations Research, 38*(5), 868–875.

LEMON COIN-OR Lemon Graph Library v1.3.1. https://lemon.cs.elte.hu/trac/lemon.

Martello, S., Pulleyblank, W., Toth, P., & de Werra, D. (1984). Balanced optimization problems. *Operations Research Letters, 3*(5), 275–278.

Schrijver, A. (2003). *Combinatorial optimization: Polyhedra and efficiency*: vol. 24. Springer Science & Business Media.

Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. *SIAM Review, 57*(1), 3–57.

Zeitlin, Z. (1981). Minimization of maximum absolute deviation in integers. *Discrete Applied Mathematics, 3*(3), 203–220.