The robust traveling salesman problem with interval data

R. Montemanni*†, J. Barta , L.M. Gambardella

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH-6928 Lugano-Manno, Switzerland

Technical report IDSIA-20-05

Abstract

The traveling salesman problem is one of the most famous combinatorial optimization problems, and has been intensively studied in the last decades. Many extensions to the basic problem have been also proposed, with the aim of making the resulting mathematical models as much realistic as possible.

We present a new extension to the basic problem, where travel times are specified as a range of possible values. This model reflects the intrinsic difficulties to estimate travel times exactly in reality. We apply the robust deviation criterion to drive optimization over the interval data problem so obtained. Some interesting theoretical properties of the new optimization problems are identified and presented, together a new mathematical formulation and some exact algorithms. The exact methods are compared from an experimental point of view.

The methodology we propose can be used to attack the robust counterpart of other NP-hard combinatorial optimization problems.

Keywords: Traveling salesman problem, Robust optimization, Interval data, Exact algorithms.

1 Introduction

Given the cost of travel between each pair of a finite number of cities, the *traveling salesman problem* (TSP) is to find the cheapest tour passing through all of the cities and returning to the point of departure. Many variants and generalizations of the problems have been deeply studied along the years, starting from the pioneer work presented by Dantzig et al. [5] in 1954. Here we consider a variation - motivated by the intrinsic difficulty in estimating travel times exactly in the reality - of the most classic, and most studied, version of the problem, namely the symmetric traveling salesman problem, where the cost

*Partially supported by the Swiss National Science Foundation through project 200020-109854/1.

†Corresponding author. Tel: +41 91 610 8667. Fax: +41 91 610 8661. Email: roberto@idsia.ch.

of travel between two cities is independent from the traveling direction. The simplicity of this model, coupled with its apparent intractability, makes it an ideal platform for the study of both exact and heuristic new methods in discrete optimization. Surveys on the results achieved over the years for the TSP can be found in Lawler et al. [11] and Reinelt [17]. A vast collection of resources and information concerning the traveling salesman problem can be found at the Traveling Salesman Problem web-page (http://www.tsp.gatech.edu) maintained by William Cook.

Estimating travel times (i.e. edge costs) exactly is typically a difficult task, since they depend on many factors that are difficult to predict, such as traffic conditions, accidents, traffic jams or weather conditions. For this reason, the simple model previously introduced may be inadequate. To overcome this problem, more complex models, that take into account uncertainty, have to be considered.

Approaches that deal with uncertainty have been recently developed for other optimization problems. For example, the *scenario model* represents uncertainty through a finite set of equally possible alternative graphs, that are considered at the same time (see Yu and Yang [19]). Alternatively, a model where an interval of possible values is associated with each edge, the so-called *interval data model* (see, for example, Karaşan et al. [8]) has been studied.

In this paper we apply for the first time - as far as we are aware - the interval data model to a problem that is NP-hard problem already in its classic formulation: the traveling salesman problem. As it will soon be clear, this introduces an extra layer of complexity to the original problem, since a set of classic traveling salesman problems has typically to be solved in order to compute the optimal solution of a problem with interval data (see Section 4). However, the recent advances in heuristic and exact algorithms for the classic symmetric traveling salesman problem (see, for example, Helsgaun [7] and Applegate et al. [1]), lead, in our opinion, to the opportunity of studying more realistic, although more complex, models of the problem, like the one we propose in this paper. The same methodological approach we discuss in this paper can be used to attach the robust counterpart of many other NP-hard combinatorial optimization problems.

Once the interval data model has been chosen, optimization criteria have to be selected. Robustness criteria are discussed in Kouvelis and Yu [9], a book entirely devoted to robust discrete optimization. The most popular criteria are the *absolute robustness criterion* and the *robust deviation criterion*. In the past few years, they have been applied to many combinatorial optimization problems in Averbakh [2], Montemanni [12] and Yaman et al. [18], although the list is by no means exhaustive. In this paper we will adopt the robust deviation criterion, since the absolute robustness would lead to an overconservative approach (see Section 2 for more details). A *robust deviation tour* is, intuitively, a tour which minimizes the maximum deviation from the optimal tour over all realizations of edge costs.

The robust deviation traveling salesman problem is formally described in Section 2. A mathematical formulation of the problem, based on an interesting theoretical result, is presented in Section 3, while three exact algorithms are described in Section 4. Computational results are presented in Section 5 and conclusions are given in Section 6.

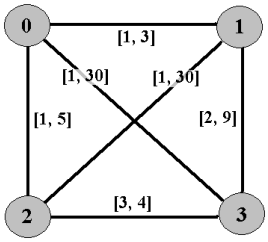

Figure 1: Example of undirected graph with interval costs.

2 Problem description

The robust deviation symmetric traveling salesman problem with interval data is defined on an undirected graph $G = \{V, E\}$, where V is a set of vertices, with vertex 0 associated with the depot, and vertices $1, \ldots, |V|$ representing the cities to be visited, and E is the set of edges of the graph. All the problems considered in this paper are based on complete graphs: $\forall i, j \in V \; \exists \{i, j\} \in E$. In the paper we will refer to an edge either as e or as $\{i, j\}$, depending on the the context, choosing the second notation when the two nodes of the edge have a role within the description. An interval $[l_{ij}, u_{ij}]$, with $0 \leq l_{ij} \leq u_{ij}$, is associated with each edge $\{i, j\} \in E$. An example of undirected graph with interval costs is given in Figure 1.

The objective of the optimization is to find a Hamiltonian circuit (indicated as *tour* in the remainder of the paper) with the minimum cost, according to the cost function associated with the chosen notion of robustness. Two main notions of robustness are considered by Kouvelis and Yu [9], each of them defining a different robust optimization problem. The application of the first of them, the *absolute robustness criterion*, would bring to a classic traveling salesman problem where each cost c_{ij} is given by u_{ij}, the upper bound of the respective interval. Therefore we do not consider this criterion here. It is however easy to see that this notion of robustness is heavily overconservative. In this paper we adopt the second criterion discussed in Kouvelis and Yu [9], namely the *robust deviation* criterion.

In order to formally describe the *robust deviation traveling salesman problem*, we need the following definitions.

Definition 1. *A* scenario *R is a realization of the edge costs, i.e. a cost $c_{ij}^R \in [l_{ij}, u_{ij}]$ is chosen for each edge of the graph.*

Definition 2. *The* robust deviation *of a tour t in scenario R is the difference between the cost of t in scenario R and the cost of a shortest tour in R.*

In the remainder of the paper we will refer to the robust deviation of tour t on scenario R as $dev(R, t)$, and to the shortest tour in scenario R as $St(R)$.

Definition 3. *A tour t is said to be a* robust tour *if it has the smallest (among all possible tours) maximum (among all possible scenarios) robust deviation.*

A scenario can be seen as a snapshot representing the real network situation, and a robust deviation tour can be seen as a path that should guarantee rea-

sonably good performance (compared to optimal solutions) under any possible edge costs configuration.

3 Mathematical formulation

In this section we introduce a mathematical formulation for the robust traveling salesman problem. This formulation is based on an important theoretical result that we will first introduce. The formulation, like all the exact algorithms we will describe in Section 4, is heavily based on this theoretical result.

The following theorem establishes a criterion that makes the robust traveling salesman problem much more tractable from a combinatorial point of view. It is inspired by analogous results for the robust counterpart of other combinatorial optimization problems (see, for example, Daniels and Kouvelis [4], Karaşan et al. [8] and Yaman et al. [18]).

Theorem 1. *Given a tour t, the scenario R that maximizes the robust deviation for t is the one where all the edges of tour t have the highest possible cost, and the costs of the remaining edges are at their lowest possible value, i.e. $c_{ij}^R = u_{ij}\ \forall\{i,j\} \in t$ and $c_{ij}^R = l_{ij}\ \forall\{i,j\} \notin t$.*

Proof. Here and in the remainder of the paper we will refer to the *scenario induced by tour t* described above as $Ind(t)$. Let R be a generic scenario. We want to prove that $dev(R,t) \leq dev(Ind(t),t)$, i.e. scenario $Ind(t)$ produces the maximal deviation of tour t over all possible scenarios R. According to the definitions of robust deviation we have that

$$dev(R,t) = \sum_{\{i,j\}\in t} c_{ij}^R - \sum_{\{i,j\}\in St(R)} c_{ij}^R = \sum_{\{i,j\}\in t\setminus St(R)} c_{ij}^R - \sum_{\{i,j\}\in St(R)\setminus t} c_{ij}^R \leq$$

$$\leq \sum_{\{i,j\}\in t\setminus St(R)} c_{ij}^{Ind(t)} - \sum_{\{i,j\}\in St(R)\setminus t} c_{ij}^{Ind(t)} \tag{1}$$

Since, by definition of $Ind(t)$, we have that $c_{ij}^{Ind(t)} \geq c_{ij}^R\ \forall\{i,j\} \in t \setminus St(R)$ and $c_{ij}^{Ind(t)} \leq c_{ij}^R\ \ \forall\{i,j\} \in St(R) \setminus t$.

We can observe that

$$\sum_{\{i,j\}\in t\setminus St(R)} c_{ij}^{Ind(t)} = \sum_{\{i,j\}\in t} c_{ij}^{Ind(t)} - \sum_{\{i,j\}\in t\cap St(R)} c_{ij}^{Ind(t)} \tag{2}$$

Consequently

$$dev(R,t) \leq \sum_{\{i,j\}\in t} c_{ij}^{Ind(t)} - \Big(\sum_{\{i,j\}\in t\cap St(R)} c_{ij}^{Ind(t)} + \sum_{\{i,j\}\in St(R)\setminus t} c_{ij}^{Ind(t)}\Big) =$$

$$= \sum_{\{i,j\}\in t} c_{ij}^{Ind(t)} - \sum_{\{i,j\}\in St(R)} c_{ij}^{Ind(t)} \leq$$

$$\leq \sum_{\{i,j\}\in t} c_{ij}^{Ind(t)} - \sum_{\{i,j\}\in St(Ind(t))} c_{ij}^{Ind(t)} = dev(Ind(t),t) \tag{3}$$

□

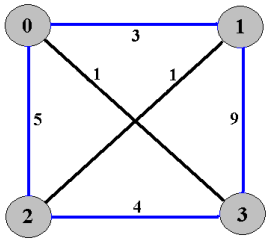

Figure 2: Scenario induced by the tour $t = \{\{0,1\},\{1,3\},\{3,2\},\{2,0\}\}$ on the graph with interval costs of Figure 1.

Theorem 1 states that the induced scenario $Ind(t)$ produces the maximal robust deviation of tour t. The very important implication is that, given a tour t, we can compute its robustness cost by solving a classic traveling salesman problem on its induced scenario.

Figure 2 depicts the scenario induced by the tour $t = \{\{0,1\},\{1,3\},\{3,2\},\{2,0\}\}$ on the graph of Figure 1. Since the tour with minimum cost on this scenario is $t' = \{\{0,1\},\{1,2\},\{2,3\},\{3,0\}\}$, the robustness cost of t is $\underbrace{(3+9+4+5)}_{\text{cost of } t} - \underbrace{(3+1+4+1)}_{\text{cost of } t'} = 21 - 9 = 12$. It is also easy to see that tour t is the robust tour of the example of Figure 1.

We are ready to give a mathematical formulation of the robust traveling salesman problem. Variables x are $\{0,1\}$ variables that identify the edges of the robust deviation tour: $x_{ij} = 1$ if edge $\{i,j\}$ is on the robust tour; 0 otherwise. Variables y define the shortest tour of the scenario induced by the tour defined by x variables: $y_{ij} = 1$ if edge $\{i,j\}$ is on the shortest tour; 0 otherwise. In the formulation we will also refer to the set of all possible tours (Hamiltonian circuits) on graph G as $\mathcal{TSP}$. The problem can be formalized in a compact way as follows:

$$\min_{x \in \mathcal{TSP}} \left(\sum_{\{i,j\} \in E} u_{ij} x_{ij} - \min_{y \in \mathcal{TSP}} \left(l_{ij} + (u_{ij} - l_{ij}) x_{ij} \right) y_{ij} \right) \tag{4}$$

The result of Theorem 1 has been used to give a closed definition of the shortest tour in the scenario defined by x variables.

The formulation is not yet expressed in terms of mixed integer linear programming, because of the presence of the nested min operator and because of the non linear interaction between x and y variables. We can however easily reduce it to a mixed integer linear program by adding a free variable r, that will contain the regret term now expressed through the nested min operator, and by inserting a new set of linear constraints. What we obtain, after a reorganization of terms, is the following formulation:

$$(RTSP) \quad \min \sum_{\{i,j\}\in E} u_{ij}x_{ij} - r \tag{5}$$

$$\text{s.t. } r \leq \sum_{\{i,j\}\in E} y_{ij}l_{ij} + \sum_{\{i,j\}\in E} y_{ij}(u_{ij} - l_{ij})x_{ij} \qquad \forall\ y \in \mathcal{TSP} \tag{6}$$

$$x \in \mathcal{TSP} \tag{7}$$

$$r \in \mathcal{R} \tag{8}$$

Constraint (7) states that x variables must be constrained to form a Hamiltonian circuit. Any feasible mixed integer programming formulation for the classic traveling salesman problem can be used for this purpose. Surveys on these formulations can be found, for example, in Langevin et al [10] and Orman and Williams [16]). Constraints (6) substitute the nested min operator and incorporate the result of Theorem 1 for the calculation of the maximum robust deviation of the tour defined by x variables. Notice that, in these inequalities, y plays the role of a constant vector representing a tour.

Independently from the formulation chosen to make constraint (7) explicit, the bottleneck of formulation $RTSP$ is represented by constraints (6), that are in a huge number (in case graph G is complete, all possible permutations of the nodes in $V\backslash\{0\}$ are feasible tours). The formulation is consequently not suitable to be directly handled as it is, in case of realistic size problems, and dedicated exact algorithms have to be developed.

4 Exact algorithms

Three different exact approaches for the robust traveling salesman problem are presented. One of them is a classic branch and bound algorithm. It does not use formulation $RTSP$, introduced in Section 3. A second exact method is a branch and cut algorithm, similar to the branch and bound, but based on formulation $RTSP$. Finally a decomposition approach based on formulation $RTSP$ is presented. The following sections are devoted to the description of them.

4.1 Branch and bound algorithm BB

As a first approach to solve the robust traveling salesman problem, we implemented a branch and bound (BB) method (inspired by those presented in Montemanni and Gambardella [13] and Montemanni et al. [15] for the robust counterparts of the minimum spanning tree and shortest path problems).

Method BB builds and visits a search-tree, and each search-tree node d can be identified by the following elements:

- $in(d)$: list of all edges that any tour associated with the search-subtree routed at d must contain.
- $out(d)$: list of all edges forbidden for any tour associated with the search-subtree routed at d.

- $t(d)$: shortest tour in scenario U, defined as the scenario where all costs are at their highest possible value, i.e. $c_{ij}^U = u_{ij}\ \forall\{i,j\} \in E$, that respects restrictions defined by sets $in(d)$ and $out(d)$. The technique we adopt for the calculation of $t(d)$ will be described in Section 4.1.2.

- $lb(d)$: lower bound for the robustness cost of all tours associated with the nodes of the search-subtree routed at d. The calculation of $lb(d)$ will be described in Section 4.1.3.

Algorithm BB can be summarized as follows, where ub is the cost of the best solution (in terms of robustness) encountered so far, Q is the set of search-tree nodes to be visited, and $RobCost(t)$ is used to indicate the *robustness cost*, i.e. the maximum robust deviation (according to Theorem 1) of a given tour t:

- **Step 0:**

  Set $ub := +\infty$.

  Generate the root of the search-tree and insert it into the set of search-tree nodes to be visited: $in(r) := \emptyset$; $out(r) := \emptyset$; $lb(r) = 0$ and $Q := \{r\}$.

- **Step 1:**

  If $Q = \emptyset$, the optimal solution is the one associated with the upper bound ub. **Stop**.

  Select the search-tree node d with the smallest lower bound: $d := \arg\min_{g \in Q}\{lb(g)\}$.

  Calculate the tour $t(d)$ as described in Section 4.1.2, and in case the lower bound $lb(d)$ as described in Section 4.1.3.

  If no feasible $t(d)$ exists, or $lb(d) \geq ub$, the search-subtree routed at node d is dominated (or infeasible). Set $Q := Q \setminus \{d\}$ and **goto step 1**.

  If $t(d) = in(d)$ a leaf of the search-tree has been reached. Calculate the robustness cost $RobCost(t(d))$ of $t(d)$ (using Theorem 1). If $RobCost(t(d)) < ub$ update the upper bound and delete dominated nodes from Q: $ub := RobCost(t(d)); \forall g \in Q$ such that $lb(g) \geq ub$, $Q := Q \setminus \{g\}$. **Goto step 1**.

- **Step 2:**

  Branching phase.

  Select edge e according to the branching strategy described in Section 4.1.1 and generate, from d, two new search-tree nodes d' and d''.

  Set $in(d') := in(d) \cup \{e\}; out(d') := out(d); in(d'') := in(d); out(d'') := out(d) \cup \{e\}$.

  Update search-tree nodes set Q: $Q := Q \cup \{d', d''\} \setminus \{d\}$.

  Goto step 1.

Notice that any exact algorithm for the classic symmetric traveling salesman problem can be used to solve the problems faced during the calculation of upper and lower bounds (see Section 4.1.3) and, with some expedient (see Section 4.1.2), for the calculation of tour $t(d)$, associated with a given search-tree node d.

4.1.1 The branching strategy

Each time an edge has to be selected in order to split the search-tree node under investigation into two (more specified) new nodes, we select the edge $e = \{i, j\}$ such that $e \in t(d) \setminus in(d)$, and e has a common vertex with an edge of $in(d)$ (if $in(d) \neq \emptyset$). Edge e is, among the (maximum two) edges with these characteristics, the one that maximizes the difference $u_e - l_e$.

The simple branching strategy we propose is based on the idea that $in(d)$ builds up as a chain of connected edges. This strategy is suitable for the application of the following rule, that considerably reduces the number of visited nodes in the search-tree.

Proposition 1. *Every edge $f \notin in(d)$ incident to a node that is common to two edges of the set $in(d)$ (i.e. f is incident to an internal node of the chain defined by the set $in(d)$) can be added to the set $out(d)$.*

Proof. The result holds since each internal node of the chain defined by the set $in(d)$ has already two incident edges, and a node has exactly two incident edges in any feasible tour. □

The result of Proposition 1 is used in our implementation of algorithm BB. It contributes to improve the quality of the lower bound $lb(d)$ (see Section 4.1.3)

4.1.2 Efficient calculation of tour $t(d)$

To efficiently compute the tour $t(d)$, i.e. the tour with minimum cost on scenario U that respects the limitations imposed by sets $in(d)$ and $out(d)$, we choose to create an artificial (classic) traveling salesman problem, in order to be in a position to run the extremely efficient tools available for the classic TSP solvers on it.

The following notation will be used. Given a scenario R and a tour t, we will refer to the cost (in terms of classic TSP) of t in R as $ClCost(R, t)$. Given a search-tree node d, the set of all tours in G that respect the restriction imposed by sets $in(d)$ and $out(d)$ will be referred to as $T(d)$. Set $T(d)$ contains then all tours associated with the search-subtree routed at d.

The artificial problem is created as follows, by means of the following artificial scenario $W(d)$: the cost of the edges contained in set $out(d)$ are set to the arbitrarily large value L, such that each unfeasible tour at search-tree node d will have a cost larger than L. The cost of all the edges not contained in sets $in(d)$ and $out(d)$ is raised by an appropriate constant l, able to guarantee that all the edges contained in set $in(d)$ are forced to be in the tour with minimum cost in the artificial scenario $W(d)$. The cost of each remaining edge $\{i, j\}$ is set to the highest possible value. Formally we have: $c_{ij}^{W(d)} := L \ \forall \{i, j\} \in out(d)$, $c_{ij}^{W(d)} := u_{ij} + l \ \forall \{i, j\} \in E \setminus \{in(d) \cup out(d)\}$, $c_{ij}^{W(d)} := u_{ij} \ \forall \{i, j\} \in in(d)$, where the values of constants l and L are defined according to the following considerations.

Constants l and L must guarantee the following condition:

$$ClCost(W(d), t_1) < ClCost(W(d), t_2) \ \ \forall t_1 \in T(d) \ \ \forall t_2 \notin T(d) \qquad (9)$$

The worst feasible tour on $W(d)$ must be shorter than the best unfeasible tour (in terms of sets $in(d)$ and $out(d)$). In this way, the shortest tour $St(W(d))$

will automatically be also a shortest tour on scenario U satisfying also the restrictions imposed by set $in(d)$. An appropriate value for parameter l can be derived using the following considerations. Let m and M be the minimal and maximal edge costs on scenario U, respectively. Since any feasible tour has $(|V| - |in(d)|)$ edges not contained in $in(d)$ and therefore with costs augmented by l, we have:

$$ClCost(W(d), t_1) \leq |V| \cdot M + (|V| - |in(d)|) \cdot l \quad \forall t_1 \in T(d) \tag{10}$$

On the other hand, any tour t_2 that does not satisfy the conditions imposed by $in(d)$ contains at least $|V| - (|in(d)| - 1)$ edges not contained in set $in(d)$, and consequently with a cost augmented by l. Therefore we have:

$$ClCost(W(d), t_2) \geq |V| \cdot m + (|V| - |in(d)| + 1) \cdot l \quad \forall t_2 \notin T(d) \tag{11}$$

By imposing the following condition, we can obtain a suitable value for constant l:

$$|V| \cdot M + (|V| - |in(d)|) \cdot l < |V| \cdot m + (|V| - in(d) + 1) \cdot l \tag{12}$$

Consequently, if we choose l such that $l = |V| \cdot (M - m) + 1$, condition (9) holds and the shortest tour $St(W(d))$ (if such a tour exists) will satisfy the constraints imposed by set $in(d)$. As already mentioned, if L is large enough, e.g. if $L = |V| \cdot (M + l) + 1$, we are able to classify the optimal tour $t(d) := St(W(d))$ in one of the following three categories:

- $ClCost(W(d), t(d)) \leq |V| \cdot M + (|V| - |in(d)|) \cdot l \Rightarrow t(d)$ is feasible for the search-tree node d.
- $|V| \cdot M + (|V| - |in(d)|) \cdot l < ClCost(W(d), t(d)) < L \Rightarrow$ tour $t(d)$ violates constraints imposed by set $in(d)$ and consequently no feasible tour exists at search-tree node d.
- $ClCost(W(d), t) \geq L \Rightarrow$ tour $t(d)$ violates constraints imposed by set $out(d)$ and consequently no feasible tour exists at search-tree node d.

An example of artificial scenario can be obtained by considering the graph with interval costs depicted in Figure 1. The shortest tour on scenario U is $St(U) = \{\{0, 1\}, \{1, 3\}, \{3, 2\}, \{2, 0\}\}$. Imagine now we have a search-tree node d such that $in(d) = \{\{0, 3\}\}$ and $out(d) = \{\{0, 2\}\}$. We create the artificial scenario $W(d)$ as follows. First we calculate the value of parameters l and L: $l = |V| \cdot (M - m) + 1 = 4 \cdot (30 - 3) + 1 = 109$, $L = |V| \cdot (M + l) + 1 = 4 \cdot (30 + 109) + 1 = 557$. Then scenario $W(d)$ is obtained by augmenting by l the cost of all edges not contained in $in(d)$ and $out(d)$, and by setting to L the cost of edge $\{0, 2\}$, that is contained in $out(d)$. Figure 3 shows the edge costs on scenario $W(d)$. The shortest tour on scenario U passing through edge $\{0, 3\}$ and avoiding edge $\{0, 2\}$ is obtained by solving a classical TSP on scenario $W(d)$: $St(W(d)) = \{\{0, 3\}, \{3, 2\}, \{2, 1\}, \{1, 0\}\}$. Since the cost of this solution in the artificial scenario is 394 and $394 \leq |V| \cdot M + (|V| - |in(d)|) \cdot l = 4 \cdot 30 + (4 - 1) \cdot 109 = 447$, we can conclude that $St(W(d))$ is a feasible solution with respect to sets $in(d)$ and $out(d)$.

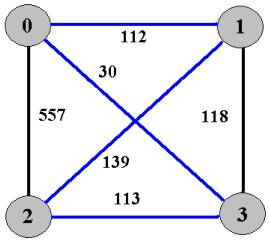

Figure 3: Example of artificial scenario for the efficient calculation of $t(d)$.

4.1.3 Calculation of lower bound $lb(d)$

In order to formally describe the lower bound $lb(d)$, we need to introduce the following notation. For a given search-tree node d, let $S(d)$ be the scenario where the costs of the edges contained in $out(d)$ are set to the lower possible value, and the costs of the remaining edges are set to the highest possible value: $c_{ij}^{S(d)} = l_{ij}\ \forall\{i,j\} \in out(d)$ and $c_{ij}^{S(d)} = u_{ij}\ \forall\{i,j\} \in E \setminus out(d)$.

We are now ready to prove the following preliminary result:

Lemma 1. *The following inequality holds:*

$$ClCost(S(d), St(S(d))) \geq ClCost(Ind(t), t) \quad \forall t \in T(d) \tag{13}$$

Proof. By definition of $Ind(t)$ and $S(d)$, and since $t \in T(d)$ and $t \subseteq E \setminus out(t)$, we know that

$$c_{ij}^{Ind(t)} \leq c_{ij}^{S(d)}\ \forall\{i,j\} \in E, \forall t \in T(d) \tag{14}$$

The thesis is directly implied by inequality (14). □

Lemma 1 allows us to define a lower bound for the robustness cost of the tours associated with the search-subtree $T(d)$.

Theorem 2. *The following inequality holds:*

$$ClCost(U, t(d)) - ClCost(S(d), St(S(d))) \leq RobCost(t)\ \forall t \in T(d) \tag{15}$$

Proof. By definition of the tour $t(d)$ associated with a search-tree node d, and since set $out(d)$ is never reduced when new branching operations are carried out, the following inequality holds:

$$ClCost(U, t) \geq ClCost(U, t(d))\ \forall t \in T(d) \tag{16}$$

Coupling the result of Lemma 1 with inequality (16), we obtain inequality (15):

$$RobCost(t) = ClCost(U, t) - ClCost(Ind(t), St(Ind(t))) \geq$$

$$\geq ClCost(U, t(d)) - ClCost(S(d), St(S(d))) \tag{17}$$

□

The lower bound defined by Theorem 2 allows us to cut dominated nodes in the search-tree. However, the lower bound can be further refined according to the result we will describe in the remainder of this section.

The idea is to make the estimate provided by inequality (15) tighter by improving the accuracy of its last term. We are therefore looking for an appropriate numeric term $\delta(t)$, based on some intrinsic properties of the problem and of the original estimate, that can be added to the original bound.

According to Theorem 1, each scenario of interest (i.e. those scenarios where at least a tour maximizes its deviation) has exactly $|V|$ edges with costs at the maximum possible value (the edges of the tour under investigation), while the remaining costs are at their lowest possible value.

We now define $\alpha(d) := |in(d) \setminus St(S(d))|$ and $\beta(d) := |St(S(d)) \cap out(d)|$, i.e. $\alpha(d)$ represents the number of edges contained in the set $in(d)$ but not in the tour $St(S(d))$, while $\beta(d)$ contains the number of edges that set $out(d)$ and tour $St(S(d))$ have in common. We can now give the following definition:

$$\delta(d) := \begin{cases} \sum_{\{i,j\} \in Rank(d)} (u_{ij} - lij) & \text{if } \alpha(d) - \beta(d) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

with $Rank(d) := \{(\alpha(d) - \beta(d))$ edges of $St(S(d)) \setminus \{in(d) \cup out(d)\}$ with the smallest values of $(u_{ij} - l_{ij})\}$.

We are ready to define the lower bound $lb(d)$ used within algorithm BB:

$$lb(d) := ClCost(U, t(d)) - ClCost(S(d), St(S(d))) + \delta(d) \tag{19}$$

Theorem 3. *$lb(d)$ is a valid lower bound for the robustness cost of the tours associated with the search-subtree $T(d)$: $lb(d) \leq RobCost(t)$ $\forall t \in T(d)$*

Proof. Set $Rank(d)$ contains the most pessimistic estimate (in terms of cost improving) for the edges that are counted at their maximum possible cost in inequality (15), but should have been at their minimum possible cost instead. Since $\delta(d)$ sums up the difference between the maximum and the minimum possible cost of the edges contained in $Rank(d)$, can be added to the estimate provided by inequality (15), obtaining a reinforced lower bound. □

It is interesting to observe that a result similar to that of Theorem 3, could have been derived and applied also to the branch and bound algorithm developed in Montemanni and Gambardella [13] for the robust spanning tree problem with interval data.

4.2 Branch and cut algorithm BC

The second algorithm we present is a branch and cut (BC) algorithm based on formulation $RTSP$. The algorithm has many common elements with algorithm BB, presented in Section 4.1.

The first step into the description of algorithm BC, is to express constraints (7) for formulation $RTSP$ in terms of (mixed integer) linear programming. It is convenient (from a computational point of view) to describe $\mathcal{TSP}$ through the following formulation, originally given in Dantzig et al. [5]:

$$(CTSP) \quad \sum_{\{j,i\}\in E} x_{ji} + \sum_{\{i,k\}\in E} x_{ik} = 2 \qquad \forall\, i \in V \qquad (20)$$

$$\sum_{\{j,i\}\in E_S} x_{ji} + \sum_{\{i,k\}\in E_S} x_{ik} < |S| - 1 \qquad \forall\, S \subset G, |S| > 2 \qquad (21)$$

$$x_{ij} \in \{0,1\} \qquad \forall\, \{i,j\} \in E \qquad (22)$$

where constraints (20) state that exactly two edges, among those incident to a given node, must be active in a feasible solution. Inequalities (21), where E_S is the set of edges with both endpoints in $S \subset G$, limit the number of active edges in each possible subgraph S, in order to avoid cycles, and consequently disconnected solutions. Constraints (22) state that x variables are binary variables.

The drawback of formulation $CTSP$ is that *subtour elimination constraints* (21) are in a huge number, notwithstanding only a few of them will be saturated at optimality. Common practice is then to solve the formulation in an iterative fashion, dynamically adding at each iteration those subtour elimination constraints that are violated in the last available solution (see, for example, Lawler et al. [11]). We will adopt a similar strategy.

Method BC, like algorithm BB, builds and visits a search-tree, and each search-tree node can be identified by some of the elements already adopted in Section 4.1 for the description of search-tree nodes of algorithm BB. Namely, search-tree node d is fully defined by a set of compulsory edges $in(d)$, a set of forbidden edges $out(d)$, and a lower bound $lb(d)$ for the robustness cost of all the tours associated with the search-subtree routed at d. What is different from algorithm BB is the strategy used to calculate the lower bound $lb(d)$.

In algorithm BC, also a linear program $RTSP(d)$ is somehow associated with the search-tree node d. It is a (strongly relaxed) version of $RTSP$ reinforced with some cuts originated by sets $in(d)$ and $out(d)$, that are added at each search-tree node d. More precisely, at each search-tree node d:

- Constraints (22) are substituted by their linear relaxation:

$$0 \le x_{ij} \le 1 \quad \forall\, \{i,j\} \in E \qquad (23)$$

- Branch and bound constraints derived by $in(d)$ are added:

$$x_{ij} = 1 \quad \forall\, \{i,j\} \in in(d) \qquad (24)$$

- Branch and bound constraints derived by $out(d)$ are added:

$$x_{ij} = 0 \quad \forall\, \{i,j\} \in out(d) \qquad (25)$$

- Only a subset of robustness constraints (6) is taken into account. Formally, the following constraints substitute the original set of inequalities (6):

$$r \le \sum_{\{i,j\}\in E} y_{ij} l_{ij} + \sum_{\{i,j\}\in E} y_{ij}(u_{ij} - l_{ij}) x_{ij} \quad \forall\, y \in \overline{\mathcal{TSP}} \qquad (26)$$

  with $\overline{\mathcal{TSP}} \subseteq \mathcal{TSP}$.

- Only a subset of subtour elimination constraints (21) is taken into account. Formally, the following constraints substitute the original set of inequalities 21:

$$\sum_{\{j,i\}\in E_S} x_{ji} + \sum_{\{i,k\}\in E_S} x_{ik} < |S| - 1 \quad \forall\ S \in \overline{\mathcal{SE}} \tag{27}$$

  with $\overline{\mathcal{SE}} \subseteq \{S \subset G \mid |S| \geq 2\}$.

What makes our approach somehow unusual is the way inequalities (26) and (27) are associated with $RTSP(d)$, given a search-tree node d. They do not depend on the depth of node d in the search-tree. Sets $\overline{\mathcal{TSP}}$ and $\overline{\mathcal{SE}}$ are empty at the beginning of the computation, and are incremented each time a new search-tree node is visited. Consequently the number of constraints available for the search-tree node d only depend on the position of d in the sequence of visit.

Notice that at the first iteration of such an approach, when no constraint of type (6) is present, we need to constrain variable r to be bounded, otherwise it would generate an unbounded solution of problem $RTSP$. For this purpose, we can add the following constraint to formulation $RTSP$:

$$r \leq \sum_{\{i,j\}\in E} u_{ij} x_{ij} \tag{28}$$

At each search-tree node d, the lower bound $lb(d)$ is given by the optimal solution of $RTSP(d)$. Moreover, in case the optimal solution of $RTSP(d)$ is integer, no violated subtour elimination constraint (21) exists, and no violated robustness constraint (7) exists, $lb(d)$ also provides an upper bound for the optimal solution, that could improve ub. In case one of these conditions is not matched, we execute a branching operation (see Section 4.2.1 for details) and two new search-tree nodes are generated and take the place of d. The algorithm stops when all the unvisited nodes have a lower bound greater than or equal to the current upper bound ub, that is consequently the cost of the optimal solution. Set Q contains the search-tree nodes to be visited.

In our implementation, parameter $ncuts$ regulates the maximum number of robustness inequalities (26) added at each search-tree node. Algorithm BC can be summarized as follows:

- **Step 0:**

  Set $\overline{\mathcal{TSP}} := \emptyset$; $\overline{\mathcal{SE}} := \emptyset$ and $ub := +\infty$.

  Generate the root of the search-tree and insert it into the set of search-tree nodes to be visited: $in(r) := \emptyset$; $out(r) := \emptyset$; $lb(r) = 0$ and $Q := \{r\}$.

- **Step 1:**

  If $Q = \emptyset$, the optimal solution is that associated with ub. **Stop**.

  Select the search-tree node d with the smallest lower bound: $d := \arg\min_{g\in Q}\{lb(g)\}$.

  Initialize the counter for the new cuts added at node d: $count := 0$.

- **Step 2:**

  Build $RTSP(d)$ as previously described.

  Solve $RTSP(d)$ and let (x^d, r^d) be its optimal solution.

  Set $lb(d) :=$ cost of solution (x^d, r^d).

  If $lb(d) \geq ub$, the search-subtree routed at node d is dominated. Set $Q := Q \setminus \{d\}$ and **goto step 1**.

  If solution x^d is fractional **goto step 4**.

- **Step 3:**

  If x^d is integer and a violated subtour elimination constraint (21) se exists, we add it to $\overline{\mathcal{SE}}$: $\overline{\mathcal{SE}} := \overline{\mathcal{SE}} \cup \{se\}$. **Goto step 2**.

- **Step 4:**

  If solution x^d is not integer and $count >= ncuts$, we execute the branching routine. **Goto step 5**.

  Check whether a violated robustness constraint (7) exists. According to Theorem 1, such a violated constraint can be identified by solving the following auxiliary problem $AUX(x^d)$. $AUX(x^d)$ is a classic traveling salesman problem where y is the vector of the variables and costs c are defined as follows, based on the value of x^d:

  $$c_{ij} := l_{ij} + (u_{ij} - l_{ij})x^d_{ij} \quad \forall\{i,j\} \in E \tag{29}$$

  Notice that x^d acts here as a vector of constants (not variables).

  If a violated robustness constraint (7) tsp exists, we add it to $\overline{\mathcal{TSP}}$: $\overline{\mathcal{TSP}} := \overline{\mathcal{TSP}} \cup \{tsp\}$.

  Set $count := count + 1$ and **Goto step 2**.

  Otherwise we have reached a leaf of the search-tree and x^d is a tour with robustness cost $lb(d)$. If $lb(d) < ub$ we update the upper bound and we delete dominated nodes from Q: $ub := lb(d); \forall g \in Q$ such that $lb(g) \geq ub$; $q := Q \setminus \{g\}$. **Goto step 1**.

- **Step 5:**

  Branching phase.

  Select edge e according to the branching strategy described in Section 4.2.1 and generate, from d, two new search-tree nodes d' and d''.

  Set $in(d') := in(d) \cup \{e\}; out(d') := out(d); lb(d') := lb(d)$; $in(d'') := in(d); out(d'') := out(d) \cup \{e\}$ and $lb(d'') := lb(d)$.

  Update search-tree nodes set Q: $Q := Q \cup \{d', d''\} \setminus \{d\}$.

  Goto step 1.

Preliminary computational experiments indicated that the algorithm clearly performs better, in terms of computation times, when parameter $ncuts$ is set to 1. We will therefore use this setting for all the experiments reported in Section 5.

Notice that any exact algorithm for the classic symmetric traveling salesman problem can be used to solve the auxiliary problem $AUX(x^d)$. A violated subtour elimination constraint can be finally identified by inspecting the current solution x^d.

4.2.1 Branching strategy

Branching edge e is of course an edge with a non integer value in the solution x^d of the problem $RTSP(d)$, associated with the search-tree node d, under investigation by the algorithm.

To select the branching edge we first check whether it exists an edge e such that x^d_e has a fractional value and e has a common vertex with an edge of set $in(d)$. In case such an edge e exists, we select it for the branching, in order to profit of the results of Proposition 1.

If such an edge e does not exist, we select a random edge with fractional value in solution x^d.

More complex branching strategies have been tested, but in all the cases preliminary results clearly suggested that the (eventual) gain in the number of visited search-tree nodes, was dominated by the computational effort spent to select branching edges, resulting in longer computation times.

4.3 Benders decomposition algorithm BD

As observed in Section 3, formulation $RTSP$ has a huge number of constraints (6). One strategy to handle it, is then to decompose the problem in a Benders Decomposition (BD) fashion.

Benders partitioning method was originally proposed in 1962 in Benders [3] (see also Geoffrion [6]). Applications of this approach to robust optimization problems can be found, for example, in Kouvelis and Yu [9], Montemanni [12] and Montemanni and Gambardella [14].

The key idea of Benders decomposition is that only a few constraints of type (6) - in our case - will be active at optimality, and therefore these constraints are generated in an iterative fashion only when violated.

We then start with no constraint of type (6) and we iteratively solve a relaxed version of $RTSP$ and seek for the most violated constraint of type (6), based on the last available solution of the relaxed $RTSP$. Also in this case constraint (28) has to be added to $RTSP$ in order to avoid unbounded solutions during the first iteration of the method.

Let now ψ represent the iteration number and let $\mathcal{TSP}^\psi$ represent the restricted set of tours of $\mathcal{TSP}$ available at iteration ψ. Algorithm BD can be summarized as follows:

- **Step 0:**

  Set $\psi := 1$ and $\mathcal{TSP}^1 := \emptyset$.

- **Step 1:**

  Solve the following mixed integer problem, $RTSP^\psi$, which is the relaxed version of $RTSP$ (the so-called master problem) obtained by considering the tours available at iteration ψ only.

  Formally the mixed integer program $RTSP^\psi$ is obtained by changing constraints (6) with the following ones:

$$r \leq \sum_{\{i,j\} \in E} y_{ij} l_{ij} + \sum_{\{i,j\} \in E} y_{ij}(u_{ij} - l_{ij}) x_{ij} \quad \forall\ y \in \mathcal{TSP}^\psi \qquad (30)$$

Let now (x^ψ, r^ψ) be an optimal solution of $RTSP^\psi$, and let z^ψ be the cost of this solution.

Solve the auxiliary problem $AUX(x^\psi)$ (see Section 4.2), i.e. a classic traveling salesman problem on the scenario induced by the tour defined by the x^ψ variables.

We refer to the optimal tour of the classic traveling salesman problem $AUX(x^\psi)$ as y (i.e. $y_{ij} = 1$ if edge $\{i, j\}$ is on the optimal tour, 0 otherwise). We also define:

$$w^\psi := \sum_{\{i,j\} \in E} (u_{ij} x_{ij}^\psi) - \sum_{\{i,j\} \in E} (l_{ij} + (u_{ij} - l_{ij}) x_{ij}^\psi) y_{ij}^\psi \tag{31}$$

According to Benders [3], it is possible to observe that w^ψ and z^ψ are respectively an upper bound and a lower bound for the optimal cost of the original problem $RTSP$.

If $\min_{\nu \in \{1,2,\ldots,\psi\}} w^\nu \leq z^\psi$ then the optimal solution of $RTSP$ has been found, **stop**.

Otherwise set $\mathcal{RTSP}^{\psi+1} := \mathcal{RTSP}^\psi \cup \{y^\psi\}$; $\psi := \psi + 1$ and **repeat step 1**.

What remains to be specified is the formulation used to make the original constraint (7) explicit. Evidence suggested that, also in this case, the best choice is to use formulation $CTSP$ (see Section 4.2), originally described in Dantzig et al. [5].

In the context of algorithm BD, a sequence of (relaxed) problems of type ($RTSP$) has to be solved. It is consequently convenient, from a computational point of view, to consider at each iteration ψ all the constraints (21) already generated at previous iterations $0, \ldots, \psi - 1$. This prevents us from generating the same constraints in different (similar) problems.

Any exact algorithm for the classic symmetric traveling salesman problem can finally be used to solve the auxiliary problem.

5 Computational experiments

In this section the algorithms presented in Section 4 are compared from an experimental point of view. Characteristics of the instances adopted for the tests are described in Section 5.1, while the comparison among the methods we propose can be found in Section 5.2.

All the algorithms have been coded in *ANSI C* and *ILOG CPLEX 9.0* (http://www.ilog.com/products/cplex) together with *ILOG Concert Technology 2.0* (http://www.ilog.com/products/optimization/tech/concert.cfm) has been used to handle and solve mixed integer programs. In our implementation of the methods we propose, we also used the callable libraries based on algorithms *LKH 1.3* (http://www.akira.ruc.dk/~keld/research/LKH) and *Concorde 03.12.19* (http://www.tsp.gatech.edu/concorde), that represent the state of the art of heuristic and exact algorithms for the classic symmetric traveling salesman problem, respectively.

In all our algorithms, each time a classic traveling salesman problem has to be solved to optimality, we use the following strategy: heuristic algorithm *LKH*

is run (with standard settings). It returns a tour (heuristic solution) together with the cost of this tour and an optimality flag, set to 1 in case a lower bound with the same cost of the heuristic solution has been produced, 0 otherwise. If optimality is not guaranteed, we run the exact algorithm *Concorde* (with standard settings). Preliminary experiments on the instances considered in this paper highlighted a strong improvement in computation times (compared to the use of algorithm *Concorde* only) when this strategy is implemented. A more accurate setting of *Concorde*'s parameter might lead to different results, but these refinements are beyond the scope of our work.

All the tests we present have been carried out on a Intel Pentium 4 1.5 GHz / 256 MB machine.

5.1 Description of the test problems

No benchmark problem is available from the literature for the robust traveling salesman problem with interval data. Consequently we had to generate a new benchmark set. We created two families of problems.

Problems with the same characteristics of those of all the families considered can be easily reproduced. Moreover, all the instances considered in this paper are available as a zip archive (http://www.idsia.ch/~roberto/RTSPinst.zip).

5.1.1 Random instances

The first family is a set of non-euclidian random instances, generated according to the following schema: a problem of type R-N-M has N nodes ($|V| = N$) and $\forall i, j \in V$ cost u_{ij} is picked up at random from the set $\{0, 1, \ldots, M\}$, while l_{ij} is selected again at random from the set $\{0, 1, \ldots, u_{ij}\}$.

5.1.2 TSPLIB instances

The second family of problems is generated from some classic traveling salesman problems available at the web-site *TSPLIB* (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95). All the problems considered in this paper for this second family are euclidian problems. Given a problem *Prob* from TSPLIB, each new instance named *Prob*-β (with $\beta \in [0, 1]$) will have the same vertices of the original instance *Prob*, while interval costs will be generated at random, in such a way that $\forall i, j \in V$, l_{ij} is selected at random as an integer number from the interval $\left[\lfloor (1-\beta)c^P_{ij} \rfloor, c^P_{ij}\right]$, while u_{ij} is an integer number picked up from $\left[c^P_{ij}, \lceil (1+\beta)c^P_{ij} \rceil\right]$, with c^P_{ij} being the original cost in problem *Prob* (e.g. the euclidian distance between vertices i and j). With this procedure, different levels of uncertainty can be implemented starting from the same original problem by modifying parameter b.

5.2 Computational results

The results obtained by the exact algorithms described in Section 4 are presented in Tables 1, 2 and 3. The same 10 instances of each problem are considered for each algorithm, and a maximum computation time of 3600 seconds has been set for each instance. In case an algorithm is able to solve to optimality all the 10 instances of a give problem within the time limit, average and standard

deviation for the number of iterations and the time required, are reported. In case some of the instances are not solved to optimality, information about the optimality gap are summarized. The optimality gap is given by $100 \cdot \frac{ub-lb}{ub}$, where lb and ub are the lower and upper bound provided by the algorithm at the end of the computation time.

Our current implementation of algorithm BB privileges speed with respect to memory usage, and consequently it is not able to handle the biggest problems considered. The corresponding entries of Table 1 are marked with the label "out of memory".

Table 1: Exact algorithm BB. Averages over 10 instances.

Problems	Iterations		Seconds		Gap (%)	
	Avg	StDev	Avg	StDev	Avg	StDev
R-10-100	569,70	759,73	6,86	9,17	-	
R-10-1000	251,00	170,35	2,95	1,99	-	
R-20-100	-				0,65	2,06
R-20-1000	30130,00	34415,87	667,41	766,95	-	
R-30-100	-				22,39	7,58
R-30-1000	-				22,43	9,10
R-40-100	out of memory					
R-40-1000	out of memory					
R-50-100	out of memory					
R-50-1000	out of memory					
R-60-100	out of memory					
R-60-1000	out of memory					
R-30-10	-				46,23	4,96
R-30-10000	out of memory					
gr17-0.25	5633,60	4902,59	103,38	88,94	-	
gr17-0.50	-				0,41	0,87
gr21-0.25	396,70	234,52	10,11	6,54	-	
gr21-0.50	49411,60	38840,09	1134,34	888,77	-	
gr24-0.25	10678,20	12158,31	287,72	331,30	-	
gr24-0.50	-				5,37	6,89
fri26-0.25	7399,10	4737,54	227,17	152,53	-	
fri26-0.50	-				8,78	9,56
swiss42-0.25	out of memory					
swiss42-0.50	out of memory					
dantzig42-0.25	out of memory					
dantzig42-0.50	out of memory					
gr48-0.25	out of memory					
gr48-0.50	out of memory					
hk48-0.25	out of memory					
hk48-0.50	out of memory					
brazil58-0.25	out of memory					
brazil58-.50	out of memory					

Tables 1, 2 and 3 clearly suggest that algorithm BD, based on Benders decomposition, is the best method of the pool for the optimization problem considered. A strict correlation between the number of iterations required by the each method and the respective computation time is also highlighted, as could have been expected.

Standard deviations for computation times, optimality gaps and number of iterations, are high for all the methods presented. This suggests that the difficulty of instances is strongly related to the conformation of each uncertainty interval, with respect to the others. This aspect would deserve a more detailed study, and could be the topic of a future work.

Methods BB and BC could, in principle, be combined together in order to enhance the quality of lower bounds. Preliminary results suggested however to abandon this idea, since the gain in the quality of the lower bounds available

Table 2: Exact algorithm BC. Averages over 10 instances.

Problems	Iterations		Seconds		Gap (%)	
	Avg	StDev	Avg	StDev	Avg	StDev
R-10-100	55,50	83,10	1,62	3,31	-	
R-10-1000	24,00	16,07	0,33	0,23	-	
R-20-100	233,60	166,32	44,30	68,30	-	
R-20-1000	148,90	167,35	30,04	67,26	-	
R-30-100	674,40	442,36	1081,66	1224,05	-	
R-30-1000			-		0,52	1,45
R-40-100			-		2,93	2,88
R-40-1000			-		1,55	2,06
R-50-100			-		3,69	3,05
R-50-1000			-		3,31	1,48
R-60-100			-		4,04	1,64
R-60-1000			-		3,81	1,36
R-30-10			-		0,25	0,79
R-30-10000			-		0,60	1,26
gr17-0.25	18,90	6,84	0,62	0,20	-	
gr17-0.50	74,60	44,50	4,28	4,96	-	
gr21-0.25	7,80	3,46	0,38	0,14	-	
gr21-0.50	83,00	51,02	6,31	6,14	-	
gr24-0.25	67,20	37,04	5,75	4,06	-	
gr24-0.50			-		0,30	0,95
fri26-0.25	47,20	35,07	5,03	4,09	-	
fri26-0.50			-		0,79	2,50
swiss42-0.25			-		11,05	17,10
swiss42-0.50			-		28,27	4,11
dantzig42-0.25			-		9,51	9,64
dantzig42-0.50			-		19,30	6,12
gr48-0.25			-		27,77	12,06
gr48-0.50			-		28,98	3,60
hk48-0.25			-		4,70	5,43
hk48-0.50			-		21,03	3,13
brazil58-0.25			-		100,00	6,08
brazil58-.50			-		46,86	11,32

was clearly dominated by the increasing in the computation time.

It is finally interesting to observe that many of the theoretical results developed to better describe the polytope associated with the classic traveling salesman problem (see, for example, Lawler et al. [11]), could be easily integrated in methods BC and BD, considerably speeding up the solving process of the (relaxed) mixed integer program $RTSP$. Since the bottleneck of both the methods is represented by this formulation, that is solved many times, the algorithms themselves would for sure perform much faster. This optimization is however beyond the scope of this paper.

6 Conclusion

The robust traveling salesman problem with interval data where uncertainty about edge costs is taken into account, has been introduced and studied in this paper.

A mixed integer programming formulation, based on a property of the problem we identified, has been presented. Some exact algorithms have been discussed and compared from a computational point of view.

The methodological approach presented can be easily extended to the interval data counterpart of many other classic NP-hard combinatorial optimization problems.

Table 3: Exact algorithm BD. Averages over 10 instances.

Problems	Iterations		Seconds		Gap (%)	
	Avg	StDev	Avg	StDev	Avg	StDev
R-10-100	9,00	3,71	0,38	0,28	-	
R-10-1000	5,90	3,00	0,19	0,16	-	
R-20-100	17,70	5,98	6,26	4,59	-	
R-20-1000	13,00	4,94	3,00	2,22	-	
R-30-100	30,30	13,15	37,37	32,86	-	
R-30-1000	26,10	7,92	33,11	16,94	-	
R-40-100	62,50	31,32	657,18	670,56	-	
R-40-1000	36,78	17,79	263,75	397,79	-	
R-50-100			-		1,78	3,60
R-50-1000	50,30	25,15	1406,04	1289,39	-	
R-60-100			-		3,06	3,05
R-60-1000			-		0,79	1,04
R-30-10	32,00	14,64	37,60	32,78	-	
R-30-10000	17,10	4,75	11,47	8,06	-	
gr17-0.25	6,00	1,49	0,38	0,22	-	
gr17-0.50	15,10	7,46	2,25	1,82	-	
gr21-0.25	3,20	0,63	0,13	0,04	-	
gr21-0.50	10,50	3,66	1,77	1,46	-	
gr24-0.25	9,50	2,92	1,61	0,38	-	
gr24-0.50	30,80	13,75	20,31	18,87	-	
fri26-0.25	5,70	2,11	0,78	0,37	-	
fri26-0.50	20,40	8,81	10,39	7,27	-	
swiss42-0.25	15,90	8,70	7,32	4,64	-	
swiss42-0.50			-		0,24	0,76
dantzig42-0.25	15,20	5,01	11,37	5,38	-	
dantzig42-0.50	66,20	15,05	437,73	249,96	-	
gr48-0.25	34,70	15,67	198,83	157,63	-	
gr48-0.50			-		7,65	5,56
hk48-0.25	22,30	8,50	33,85	22,38	-	
hk48-0.50			-		3,26	3,47
brazil58-0.25	52,70	13,42	198,64	93,69	-	
brazil58-.50			-		2,94	1,46

7 Acknowledgements

The authors would like to thank Keld Helsgaun and William Cook for their suggestions and help in the use of programs *LKH* and *Concorde* respectively, and Leonora Bianchi for the helpful discussions.

References

[1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. On the solution of traveling salesman problems. *Documenta Mathematica*, Extra volume ICM 1998(III):645–656, 1998.

[2] I. Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, pages 263–272, 2001.

[3] J.F. Benders. Partitioning procedures for solving mixed integer variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[4] R.L. Daniels and P. Kouvelis. Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41(2):363–376, 1995.

[5] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson. Solutions of a large scale travelling salesman problem. *Operations Research*, 2:393–410, 1954.

[6] A.M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[7] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[8] O.E. Karaşan, M.Ç. Pinar, and H. Yaman. The robust shortest path problem with interval data. Bilkent University, 2001.

[9] P. Kouvelis and G. Yu. *Robust Discrete Optimization and its applications*. Kluwer Academic Publishers, 1997.

[10] A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman formulations. *Operations Research Letters*, 9:127–132, 1990.

[11] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem*. Wiley, Chichester, 1985.

[12] R. Montemanni. A Benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, to appear.

[13] R. Montemanni and L.M. Gambardella. A branch and bound algorithm for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 161(3):771–779, 2005.

[14] R. Montemanni and L.M. Gambardella. The robust shortest path problem with interval data via Benders decomposition. *4OR*, to appear.

[15] R. Montemanni, L.M. Gambardella, and A.V. Donati. A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters*, 32(3):225–232, 2004.

[16] A.J. Orman and H.P. Williams. A survey of different integer programming formulations of the travelling salesman problem. Technical Report LSEOR 04.67, London School of Economics and Political Science, 2004.

[17] G. Reinelt. *The Traveling Salesman: Computational solutions for TSP applications*. Springer-Verlag, Berlin, 1994.

[18] H. Yaman, O.E. Karaşan, and M.Ç. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29:31–40, 2001.

[19] G. Yu and J. Yang. On the robust shortest path problem. *Computers and Operations Research*, 25(6):457–468, 1998.