Noname manuscript No.
(will be inserted by the editor)

On The Recoverable Robust Traveling Salesman Problem

André Chassein · Marc Goerigk

Received: date / Accepted: date

Abstract We consider an uncertain traveling salesman problem, where distances between nodes are not known exactly, but may stem from an uncertainty set of possible scenarios. This uncertainty set is given as intervals with an additional bound on the number of distances that may deviate from their expected, nominal values.

A recoverable robust model is proposed, that allows a tour to change a bounded number of edges once a scenario becomes known. As the model contains an exponential number of constraints and variables, an iterative algorithm is proposed, in which tours and scenarios are computed alternately.

While this approach is able to find a provably optimal solution to the robust model, it also needs to solve increasingly complex subproblems. Therefore, we also consider heuristic solution procedures based on local search moves using a heuristic estimate of the actual objective function. In computational experiments, these approaches are compared.

1 Introduction

The traveling salesman problem (TSP) is one of the most researched NP-complete optimization problems in the operations research community (see, e.g., [LLKS85]). Given a (directed, complete) graph with arc lengths, the problem is to determine the shortest circuit visiting all vertices exactly once.

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-13-1-3066. The U.S Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright notation thereon.

A. Chassein · M. Goerigk (✉)
Technische Universität Kaiserslautern, Paul-Ehrlich-Str. 14, 67653 Kaiserslautern, Germany
E-mail: chassein@mathematik.uni-kl.de, goerigk@mathematik.uni-kl.de

Depending on the application at hand it may happen that the input data (i.e., the arc lengths) are affected by uncertainty and are not known in advance. As an example, when traveling through a road network, there may be congestion on arcs leading to unplanned, increased travel times.

Robust optimization is an approach to handle such uncertain optimization problems. For an overview on the topic, we refer to the textbooks [BTGN09, KY97] and the surveys [ABV09, BBC11, GS15].

In [MBMG07], a robust TSP model is considered, based on the so-called regret or min-max-min approach (see [KY97, ABV09]). For every edge there is an interval given that describes the possible values of lengths this edge may have (the *scenario*). The regret approach asks for a tour that minimizes the worst-case objective over all possible scenarios, where the objective is determined as the difference between the length of the tour, and the best possible tour length in this scenario.

While the regret approach is a highly interesting method to handle uncertain optimization problems, it does not capture the possibility to modify a solution on-the-fly, when the scenario becomes known. In the example of a road network, a tour driver might decide to use a different route when radio traffic service informs him of congested sections. Such an approach amounts to a two-stage problem formulation, and – with slight differences between the concepts – is known as adjustable robustness (see [BTGGN03]), or recoverable robustness (see [LLMS09]). Other examples where the concept of recoverable robustness was applied include, e.g., the shortest path problem [Büs12], and the timetable information problem [GHMH$^+$13].

In this work we consider a recoverable robust TSP with exponentially many scenarios, where the decision maker is allowed to change a bounded number of arcs, once the scenario becomes known. To solve this model, we follow a decomposition approach. Similar approaches have also been considered in [ZZ13, FM12, BAvdH11, Mon06].

The remainder of this paper is structured as follows. In Section 2, we develop a recoverable robust model formulation for the TSP, and consider a solution approach based on iterative scenario generation in Section 3. As the resulting subproblems may still be computationally hard to solve, we introduce heuristic solution methods in Section 4. These solution methods are compared in an experimental study in Section 5. We consider problem extensions and conclude the paper in Section 6.

2 Problem Formulation

2.1 Nominal Problem

We begin with a repetition of the basic TSP. Let a complete graph of n vertices V and (possibly asymmetric) distances d_{ij} for all $i, j \in V$ be given. The TSP consists of finding a directed cycle containing all n vertices with minimal

length. A well-known formulation is the following:

$$\min \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{\substack{j \in V \\ j \neq i}} x_{ij} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{\substack{j \in V \\ j \neq i}} x_{ji} = 1 \qquad \forall i \in V \tag{3}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \qquad \forall \emptyset \subsetneq S \subsetneq V \tag{4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{5}$$

where x_{ij} is a binary variable that indicates if edge (i, j) is used in the tour or not. Constraints (4) ensure that there are no subtours. As there are exponentially many such constraints, one typically generates them during the solution procedure, using a separation algorithm. In the following, we denote by $\mathcal{T}$ the set of all feasible traveling salesman tours, i.e., all vectors x fulfilling Constraints (2–5).

2.2 Recoverable Robust Problem

We now introduce a recoverable robust problem variant. Let a TSP instance be given. Additionally, let $\mathcal{U}$ be an *uncertainty set* of distance matrices, also referred to as *scenarios*. Each scenario denotes one possible realization that we consider, but it is not known in advance which of these scenarios will actually occur. Since we consider a robust approach, we need to find a tour that performs well under all of these scenarios; and, applying a recoverable robust approach, we are allowed to modify our solution in each scenario. In this setting we allow that the modified solution differs from the original solution by at most L edges, i.e. the size of the set of edges that are part of exactly one of the two solutions has to be bounded by L. We denote this problem as TSP($\mathcal{U}$). The problem is formally stated as

$$\min_{x \in \mathcal{T}} \max_{d \in \mathcal{U}} \min_{x' \in \mathcal{T}(x)} d^T x', \tag{6}$$

where $\mathcal{T}(x) = \{x' \in \mathcal{T} \mid \Delta(x', x) \leq L\}$ is the set of all tours that can be generated from x by applying the recovery action, and $\Delta(x, x') = |\{(i, j) \in V \times V \ : \ x_{ij} \neq x'_{ij}\}|$.

Using such a restriction on the recovery action, we take the practical consideration into account that solutions cannot be changed arbitrarily in many applications, and that solution modifications may result in undesired costs. As examples, we mention the delivery of rail cargo wagons (where additional shunting operations should be avoided), the programming of drilling robots (when reprogramming operations are expensive), or the planning of circular

bus lines (which should not change too much if construction sites make routes impracticable). In these applications, the number of changes within the tour should be small. The number of these changes is counted, using the distance measure $\Delta(x, x')$.

In the following, we specify uncertainty sets $\mathcal{U}$ for the recoverable robust TSP.

2.3 Discrete Scenario Sets

We assume in this section that the uncertainty set $\mathcal{U} = \{d^1, \dots, d^N\}$ is given by a finite number of scenarios. Let $\mathcal{N} = \{1, \dots, N\}$. Using a set of variables x^k for every scenario $k \in \mathcal{N}$, we can simplify the $\min - \max - \min$ structure of TSP($\mathcal{U}$) to a $\min - \max$ problem. The resulting mixed-integer linear program (once the maximum in the objective is reformulated as a constraint) is the following:

$$\min \max_{k\in\mathcal{N}} \sum_{i\in V}\sum_{j\in V} d^k_{ij}x^k_{ij} \tag{7}$$

$$\text{s.t. } -y^k_{ij} \le x_{ij} - x^k_{ij} \le y^k_{ij} \qquad \forall i,j \in V, k \in \mathcal{N} \tag{8}$$

$$\sum_{i\in V}\sum_{j\in V} y^k_{ij} \le L \qquad \forall k \in \mathcal{N} \tag{9}$$

$$x \in \mathcal{T} \tag{10}$$

$$x^k \in \mathcal{T} \qquad \forall k \in \mathcal{N} \tag{11}$$

$$y^k_{ij} \ge 0 \qquad \forall i,j \in V, k \in \mathcal{N} \tag{12}$$

We use variables x to denote the nominal tour (i.e., the first-stage solution) and x^k to denote a recovery tour for each scenario k. Then, the Objective (7) denotes the worst-case performance over all scenarios, under the additional constraints that solutions may not differ too much. To this end, we use variables y^k_{ij} to denote if x_{ij} and x^k_{ij} differ in Constraint (8), and bound the number of such differences for every scenario in Constraint (9). Note that $L \ge 6$ is necessary to perform any recovery action at all.

Problem TSP($\mathcal{U}$) contains $N + 1$ TSP problems that need to be solved simultaneously, and which are connected by the distance constraints. While, e.g., generated cuts can be used for all variables x and x^k, an increasing number of scenarios still results in a highly increased computational complexity. Containing the classic TSP, the recoverable robust TSP is also NP-hard, even for a fixed number of scenarios.

2.4 Γ-Scenario Sets

We now consider the recoverable robust TSP with Γ-scenario sets, which were first introduced by Bertsimas and Sim in [BS04]. Specifically, they assume an

interval $[\underline{d}_{ij}, \overline{d}_{ij}]$ is given that describes the possible outcomes for each distance. As such an uncertainty set also contains unrealistically bad scenarios (e.g., all distances have simultaneously their worst-case length $\overline{d}$), and thus a robust solution would be too conservative, they restrict the number of edges which do not have their best-case length $\underline{d}_{ij}$ to be less than or equal to some parameter $\Gamma \in \mathbb{N}$. This gives the following uncertainty set:

$$\mathcal{U}(\Gamma) := \left\{ d \in \mathbb{R}^{|V|\times|V|} \mid d_{ij} = \underline{d}_{ij} + (\overline{d}_{ij} - \underline{d}_{ij})w_{ij}, \sum_{i,j\in V} w_{ij} \leq \Gamma, w_{ij} \in \{0,1\} \right\}$$

We will sometimes identify a scenario $d \in \mathcal{U}(\Gamma)$ with its defining boolean matrix $w \in \{0,1\}^{n\times n}$. Hence, the recoverable robust TSP with Γ-scenario sets can also be considered as a recoverable robust TSP with exponentially many scenarios.

3 Exact Solution Method

To solve the recoverable robust TSP with respect to some uncertainty set $\mathcal{U}$ exactly, we decompose it into subproblems which need to be solved iteratively (see, e.g., [ZZ13]). The first idea is to restrict the uncertainty set $\mathcal{U}$ to some smaller finite subset $\mathcal{U}' \subsetneq \mathcal{U}$, to avoid the exponential size of $\mathcal{U}$. This problem is denoted as TSP($\mathcal{U}'$):

$$\min_{x\in\mathcal{T}} \max_{d\in\mathcal{U}'} \min_{x'\in\mathcal{T}(x)} d^T x'$$

Note that TSP($\mathcal{U}'$) can be solved exactly by solving the mixed-integer programming formulation given by $(7)-(12)$. The second problem is the problem to evaluate a given solution x. This problem can be written as

$$\text{EVAL}(x) = \max_{d\in\mathcal{U}} \min_{x'\in\mathcal{T}(x)} d^T x'.$$

We write P^* to denote the optimal value of some problem P. Assume that there exists a tour $x^* \in \mathcal{T}$ such that $\text{EVAL}(x^*) = \text{TSP}(\mathcal{U}')^*$. As TSP($\mathcal{U}'$) is a relaxation of TSP($\mathcal{U}$), we can then conclude that x^* is also optimal for TSP($\mathcal{U}$).

Denote by $\hat{d}(x) \in \mathcal{U}$ a worst-case scenario that can happen for solution x, i.e., a maximizer of EVAL(x). We use the iterative procedure that is described in Algorithm 1 to solve TSP($\mathcal{U}$). The algorithm adds one additional scenario to the uncertainty set $\mathcal{U}'$ in every step. This approach can even be used for continuous uncertainty sets $\mathcal{U}$, if they can be represented as the convex hull of a finite set $\mathcal{P}$ (i.e., if $\mathcal{U}$ is a polytope). It is easy to see that all worst-case scenarios are contained in $\mathcal{P}$. Hence, the algorithm must terminate after finitely many steps, if it computes in every iteration a scenario from $\mathcal{P}$.

Algorithm 1 (Exact Algorithm to solve TSP($\mathcal{U}$))

Require: An instance of TSP($\mathcal{U}$).
1: Set $\mathcal{U}' := \{\underline{d}\}$
2: Solve TSP($\mathcal{U}'$). Denote by $\bar{x}$ the obtained solution.
3: Compute EVAL($\bar{x}$). Denote by $\hat{d}(\bar{x})$ the worst-case scenario for solution $\bar{x}$.
4: **if** TSP($\mathcal{U}'$)* = EVAL($\bar{x}$) **then**
5: **return** Optimal solution $\bar{x}$ with objective value EVAL($\bar{x}$).
6: **else**
7: Set $\mathcal{U}' := \mathcal{U}' \cup \{\hat{d}(\bar{x})\}$
8: Goto 2.
9: **end if**

It remains to discuss how to compute EVAL($\bar{x}$) and how to find $\hat{d}(\bar{x})$ for a given $\bar{x}$. We solve these problems again by an iterative procedure. We decompose the problems in two subproblems WC($\mathcal{T}'$) and REC(d'), where the first finds the worst-case scenario for a given set $\mathcal{T}' \subset \mathcal{T}(\bar{x})$:

$$\max_{d \in \mathcal{U}} \min_{x' \in \mathcal{T}'} d^T x', \tag{WC($\mathcal{T}'$)}$$

and the second computes the best recovery option for tour $\bar{x}$ if scenario d' is realized:

$$\min_{x' \in \mathcal{T}(\bar{x})} d'^T x' \tag{REC(d')}$$

The algorithmic approach is similar to the first one, and summarized as Algorithm 2. As $\mathcal{T}(\bar{x})$ is finite the algorithm terminates after a finite number of iterations.

Algorithm 2 (Algorithm to compute EVAL($\bar{x}$) and $\hat{d}(\bar{x})$)

Require: An instance of TSP($\mathcal{U}$) and a solution $\bar{x}$.
1: Set $\mathcal{T}' := \{\bar{x}\}$.
2: Solve WC($\mathcal{T}'$). Denote by d' the obtained solution.
3: Compute REC(d'). Denote by x'' the obtained solution.
4: **if** WC($\mathcal{T}'$)* = REC(d')* **then**
5: **return** The value of REC(d')* and the scenario d'.
6: **else**
7: Set $\mathcal{T}' := \mathcal{T}' \cup \{x''\}$
8: Goto 2.
9: **end if**

REC(d') can be solved as an integer program using formulation (7) – (12) with only one scenario and fixing all x_{ij} variables. For uncertainty sets of the form $\mathcal{U}(\Gamma)$, problem WC($\mathcal{T}'$) can be described by the following formulation, which is straightforward to convert to an integer linear programming problem (as the set $\mathcal{T}'$ is given at each iteration of Algorithm 2).

$$\max \min_{x' \in \mathcal{T}'} \sum_{i \in V} \sum_{j \in V} \left(\underline{d}_{ij} + \left(\overline{d}_{ij} - \underline{d}_{ij}\right) w_{ij}\right) x'_{ij} \tag{13}$$

$$\text{s.t.} \sum_{i\in V}\sum_{j\in V} w_{ij} \le \Gamma \tag{14}$$

$$w_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{15}$$

4 Heuristic Solution Methods

As the evaluation of a solution is in itself already NP-hard, we consider several heuristic algorithms with non-polynomial runtime. While we cannot avoid NP-hardness, the computationally more demanding steps in the exact algorithm is the computation of $\text{TSP}(\mathcal{U}')$ with increasingly large discrete sets $\mathcal{U}'$. Therefore, the number of solution evaluations a heuristic performs will have a large impact on the computation time.

4.1 Starting Solutions

We first consider several ways to construct feasible heuristic solutions. The easiest ways to do so are to solve the nominal problem (i.e., the TSP with respect to lengths $\underline{d}$), or the worst-case problem (i.e., the TSP with respect to lengths $\overline{d}$). We consider two more elaborate alternatives.

Γ-scenarios without recovery. We consider the strict robust solution with respect to the Γ-scenario set $\mathcal{U}$ (i.e., a solution in the spirit of [BS04]). The corresponding optimization problem is given as

$$\min \left\{ \max_{\substack{S\subseteq V\times V\\ |S|\le\Gamma}} \left(\sum_{(i,j)\in S} \overline{d}_{ij}x_{ij} + \sum_{(i,j)\in (V\times V)\setminus S} \underline{d}_{ij}x_{ij} \right) \;:\; x\in\mathcal{T} \right\}$$

As shown in [BS04], one can relax the inner maximization problem and dualize it to get the following mixed integer programming formulation:

$$\begin{aligned}
&\min \Gamma\alpha + \sum_{i\in V}\sum_{j\in V}\beta_{ij} + \sum_{i\in V}\sum_{j\in V}\underline{d}_{ij}x_{ij} \\
&\text{s.t. } x\in\mathcal{T} \\
&\quad \alpha + \beta_{ij} \ge \left(\overline{d}_{ij} - \underline{d}_{ij}\right)x_{ij} && \forall i,j\in V \\
&\quad \alpha \ge 0 \\
&\quad \beta_{ij} \ge 0 && \forall i,j\in V
\end{aligned}$$

A solution of the above problem can be interpreted as an optimal solution to the recoverable robust TSP with respect to any Γ and to $L = 0$, and may thus perform well as a heuristic for small $L > 0$.

Recovery to optimality. For the second heuristic, we also take the recovery action into account. We follow the idea of RecOpt [GS14] and interpret the computation of a robust solution as a location problem in the solution space. Taking a discrete uncertainty set, we compute an optimal TSP solution for each scenario. Then, a robust solution is a tour that minimizes the distance (given as recovery costs) to these solutions.

This means, we first sample randomly a discrete set of scenarios $\mathcal{U}' \subseteq \mathcal{U}$. Let $\mathcal{U}' = \{d^1, \dots, d^{N'}\}$. Then, we solve the corresponding TSP for each such scenario. Let x^k, $k \in \mathcal{N}' = \{1, \dots, N'\}$, be the solution computed this way. We solve

$$\begin{aligned}
\min \ & \sum_{k\in\mathcal{N}'} \sum_{i\in V} \sum_{j\in V} y_{ij}^k \\
\text{s.t. } & x \in \mathcal{T} \\
& -y_{ij}^k \le x_{ij} - x_{ij}^k \le y_{ij}^k && \forall k \in \mathcal{N}', i,j \in V \\
& y_{ij}^k \ge 0 && \forall k \in \mathcal{N}', i,j \in V
\end{aligned}$$

which is equivalent to $\min\{\sum_{i\in V}\sum_{j\in V}\left(\sum_{k\in\mathcal{N}'} \chi_{ij}^k\right) x_{ij} : x \in \mathcal{T}\}$, where $\chi_{ij}^k = -1$, if $x_{ij}^k = 1$, and $\chi_{ij}^k = 1$ otherwise.

4.2 Local Search

Given a feasible starting tour, we now consider methods to improve their objective value. We may reuse well-known neighborhoods from the literature (such as k-opt, see, e.g., [Hel00]), while search guiding mechanisms based on the original TSP objective function are not directly applicable.

As computing the robust objective function $\mathrm{EVAL}(x)$ is considerably more time consuming than the computation of the classic TSP objective function, we are limited in the size of search space we evaluate.

Therefore, we consider a heuristic evaluation procedure to speed-up the local search and to make a larger search space possible. To this end, we relax the computation of $\mathrm{REC}(d)$ by considering continuous decision variables, and ignoring subtour elimination constraints. Dualizing this problem and inserting it into the computation of $\mathrm{EVAL}(x)$ yields

$$\max \sum_{i\in V} \lambda_i + \sum_{j\in V} \mu_j + (n-L)\pi \tag{16}$$

$$\text{s.t. } \lambda_i + \mu_j + \nu_{ij} - \chi_{ij}\pi \le \underline{d}_{ij} + (\overline{d}_{ij} - \underline{d}_{ij})w_{ij} \qquad \forall i,j \in V \tag{17}$$

$$\sum_{i\in V}\sum_{j\in V} w_{ij} \le \Gamma \tag{18}$$

$$w_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{19}$$

$$\lambda_i, \mu_j \gtrless 0 \qquad \forall i,j \in V \tag{20}$$

$$\pi \ge 0 \qquad \forall i,j \in V \tag{21}$$

which we denote as $\text{EVAL}'(x)$. Here, w is the matrix that describes scenario d and $\chi_{ij} = -1$, if $x_{ij} = 1$, and 1 otherwise. We can use $\text{EVAL}'(x)$ as an estimate on the quality of a solution. Its accuracy is considered as part of the following experimental study.

5 Computational Experiments

We describe the setup and results of three experiments. In the first one, we use small instances that can be solved (almost) to optimality, and compare the results to the heuristic solutions from Section 4.1. In the second experiment, we compare the estimated objective function $\text{EVAL}'(x)$ from Section 4.2 with the exact objective $\text{EVAL}(x)$. The third experiment concerns larger instances, where we compare the best solutions found by the local search algorithm and the iterative algorithm.

Environment and Instances. All experiments were conducted on a computer with a 16-core Intel Xeon E5-2670 processor, running at 2.60 GHz with 20MB cache, and Ubuntu 12.04. Processes were pinned to one core. Mixed-integer programs were solved with CPLEX v. 12.6 using C++ programs compiled with gcc v. 4.5.4. and flag -O3. Subtour elimination constraints were included using lazy constraint callbacks.

To create an instance of size n, we generated two $n \times n$ matrices $\underline{d}$ and $\overline{d}$ with $\underline{d}_{ij} \in [0, 100]$ and $\overline{d}_{ij} \in [0, 100] + 100$ uniformly at random. Note that such uncorrelated uncertain instances are usually harder to solve than correlated ones for robust optimization problems (see, e.g. [Goe14]).

Experiment 1: Starting Solutions. We first compare the quality of the "one-shot" solutions from Section 4.1. To this end, we consider 10 instances of each size $n = 5, \ldots, 12$. Furthermore, we use four sets of parameters for (Γ, L): $(1, 6)$, $(1, 10)$, $(3, 6)$, and $(3, 10)$. The sample size to compute solutions of type RecOpt is 10 scenarios.

The results are summarized in Table 1. The columns "NOM" refer to the nominal solution, "WC" to the worst-case solution, "BS" to the Γ-scenario sets without recovery solution, and "RCO" to the RecOpt solution. Values are in percent, averaged over 10 instances per instance size n, and normalized with respect to the optimal objective value (e.g., the value 11.77 in the first column and row means that on average, the nominal solution has a robust objective value which is 11.77% larger than the optimal robust objective value). The last column "It" shows the average number of iterations for the exact solution algorithm, i.e., the number of generated worst-case scenarios.

We find that the worst-case solution shows considerably worse performance than all other approaches. This is because the danger of an edge obtaining a higher distance as in the nominal case is much overrated, as both the number of worst-case scenarios Γ and the recovery action L are ignored. Accordingly, solutions of type BS perform better, as they include the parameter Γ. Nominal

n	Γ	L	NOM	WC	BS	RCO	It
5	1	6	11.77	20.35	13.59	11.77	2.9
6			6.30	31.88	6.94	6.30	3.1
7			7.71	62.22	9.64	7.71	3.2
8			4.10	83.92	6.29	4.10	3.7
9			9.54	83.96	10.86	9.54	3.8
10			4.69	125.73	3.95	4.69	4.2
11			6.33	129.67	6.13	6.33	5.2
12			7.18	164.38	9.50	7.18	5.1
5	1	10	0.00	0.00	0.00	0.00	2.0
6			0.00	3.89	1.89	0.00	2.0
7			2.05	23.05	2.20	2.05	2.4
8			0.12	41.28	1.54	0.12	2.3
9			3.63	48.62	4.82	3.63	2.5
10			4.12	80.72	2.99	4.12	2.6
11			3.51	89.53	4.78	3.51	3.4
12			4.49	120.07	6.84	4.49	3.4
5	3	6	8.47	6.47	7.04	10.94	14.2
6			5.05	11.80	7.54	5.92	28.2
7			(3.22)	(28.55)	(5.54)	(3.11)	(40.8)
8			(3.36)	(38.71)	(6.17)	(3.07)	(34.0)
9			(2.47)	(41.35)	(9.33)	(2.47)	(25.1)
10			(1.79)	(61.50)	(4.60)	(1.79)	(20.7)
11			(0.68)	(68.17)	(9.14)	(0.68)	(17.9)
12			(0.28)	(78.69)	(5.69)	(0.08)	(15.4)
5	3	10	0.00	0.00	0.00	0.00	2.0
6			0.69	3.25	1.51	0.69	3.7
7			4.61	15.43	7.00	4.61	11.3
8			4.42	32.10	7.61	4.42	15.1
9			(6.45)	(40.44)	(11.90)	(6.45)	(25.0)
10			(3.41)	(63.07)	(6.97)	(3.41)	(23.6)
11			(2.32)	(71.86)	(12.87)	(2.32)	(18.9)
12			(2.42)	(91.92)	(5.86)	(2.54)	(16.0)

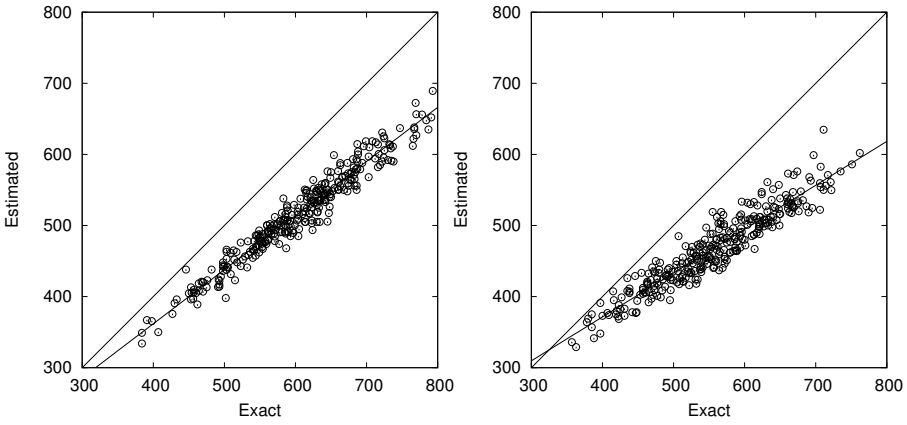
Table 1 Average objective values of 10 instances for experiment 1. Brackets $(\cdot)$ indicate that on some instances, the exact algorithm was stopped after $2h$ of computation time.

and RecOpt solutions are identical for most instances, and perform best for the considered parameter sets. Due to the recovery action, ignoring the possible worst-case of an edge is not expensive. In general, higher values for L improve the quality of the heuristic solutions (as more first-stage decisions can be changed in the second stage).

Comparing the parameter sets for Γ and L, we find an interesting behavior: While the relative objective value of algorithms NOM, BS and RCO stay in roughly the same order of magnitude for different values of n, they are especially large for $n = 5$ and $\Gamma = 1$, $L = 10$, but especially small for $n = 6$ and $\Gamma = 3$ (for $n = 5$ and $L = 10$, all solutions are optimal). The relative objective value of algorithm WC increases in both cases. Generally, the decreasing quality of a heuristic one-shot solution can be expected for an increasing instance size; in this case, the uncertainty was small enough that one single edge on the worst-case becomes decreasingly important when the instance size grows.

The computational problem complexity is mirrored in the number of iterations necessary to find an optimal solution, see the last column (decreasing numbers of iterations with increasing n are due to the timelimit of two hours). In particular, we find that increasing L tends to decrease the number of iterations, while an increased Γ results in more iterations.

Experiment 2: Objective Estimation. We now consider the usage of the estimated objective function $\text{EVAL}'(x)$ as described in Section 4.2 in more detail. To this end, we sampled 300 random tours for instances of size $n = 12$, and evaluated both the exact and the estimated objective function. The results are visualized in Figures 1(a) and 1(b), respectively, where we also show the linear regression of the data.



(a) $\Gamma = 1$, $L = 6$ (b) $\Gamma = 3$, $L = 10$

Fig. 1 Exact objective values against estimated objective values.

We find both values highly correlated for both parameter sets, with an observed correlation of 0.98 for $\Gamma = 1$ and $L = 6$, and 0.94 for $\Gamma = 3$ and $L = 10$. Furthermore, estimated objective values tend to be too optimistic, as they relax the subproblem of finding a recovery tour. The high correlation is an indicator that a local search that uses the heuristic evaluation EVAL'(x) instead of the exact evaluation $\text{EVAL}(x)$ may find better results, as EVAL'(x) is faster to solve, and a close estimator of $\text{EVAL}(x)$.

Experiment 3: Local Search. In this experiment we evaluate the quality of the exact algorithm when being used as a heuristic with a timelimit for larger instances. We used 20 minutes of maximum computation time for instances of size $n = 10, \ldots, 18$, with 10 instances of each size. As a comparison, we use the local search from Section 4.2 with the nominal solution as starting solution. To overcome local optima, we used a tabu search implementation, where objective

values are slightly randomly perturbed to avoid early cycling[1]. Four variants of this tabu search are used: We either use a 2-opt or a 3-opt neighborhood, and we either use the exact objective values $\text{EVAL}(x)$ or the estimated objective values $\text{EVAL}'(x)$. In the latter case, the estimated values are computed to assess the neighborhood, but the actual objective value is calculated after each performed move to determine if a new current best solution has been found.

The average objective values are presented in Table 2, where we normalized with respect to the best solution found over all five solution approaches. Column "EX" shows the objective value for the exact algorithm, columns T(X)F the tabu search using the (full) computation $\text{EVAL}(x)$, columns T(X)A the tabu search using the (approximated) computation $\text{EVAL}'(x)$, and the number in the middle stands for the 2-opt or 3-opt neighborhood, respectively.

The results show that while the exact algorithm performs best for instances with $\Gamma = 1$ (as can be expected from experiment 1), the increased problem complexity for $\Gamma = 3$ makes the tabu search the better choice in most cases. Furthermore, algorithms which use the exact objective value are outperformed by those using the estimated objective value, resulting in algorithm T3A to be the best choice for the more difficult instances.

Regarding the numbers of iterations in Table 2, the improved performance is explained by the considerably larger number of iterations that are possible within the 20 minutes time limit. Even from $n = 12$ on, not all neighbors from the 3-opt neighborhood can be evaluated within the time limit for $\Gamma = 3$, $L = 10$.

6 Extensions and Conclusion

We introduced a new variant of the traveling salesman problem, in which edge lengths are uncertain and a bounded number of them may become larger than in the nominal case, but we are allowed to use a recovery action once a scenario is realized, which allows to swap a bounded number of edges with alternatives. We developed an iterative algorithm to find an optimal solution to this problem.

As even the computation of the robust objective value is NP-hard, this is also done using a decomposition based on problem relaxations of increasing size. Several heuristic solution procedures are investigated. Due to the complex objective evaluation, we formulated a compact mixed-integer program to estimate the actual objective, which can be used within a local search algorithm.

Using experimental data, the high correlation between estimated and exact objective value can be confirmed, and the resulting heuristic algorithm seems to find solutions with better objective value than the iterative solution approach within the same time for the more complex instances.

[1] Naturally, many other meta-heuristic approaches are possible here. A simulated annealing algorithm has also been tested; however, results are not discussed, as tabu search showed a better performance.

			EX		T2F		T3F		T2A		T3A	
n	Γ	L	Obj	It	Obj	It	Obj	It	Obj	It	Obj	It
10	1	6	0.27	4.2	1.74	152.1	0.55	34.3	0.38	877.5	0.00	191.9
11			0.00	5.2	2.34	91.1	0.77	16.9	0.09	517.6	0.00	94.3
12			0.00	5.1	4.34	55.2	2.01	9.1	0.80	301.3	0.00	46.4
13			0.00	4.9	3.15	38.4	1.35	5.9	2.01	223.3	0.00	30.0
14			0.25	5.7	2.39	26.9	2.53	4.0	2.94	188.7	2.33	22.5
15			0.00	6.2	4.02	21.2	3.18	2.9	3.12	143.8	0.65	15.2
16			0.00	6.9	4.27	15.6	2.33	2.0	3.46	109.7	2.64	10.7
17			0.00	5.8	2.69	11.4	1.90	1.7	2.11	89.4	1.24	8.0
18			0.00	5.9	2.53	8.9	2.14	1.0	2.36	73.8	1.36	6.2
10	1	10	0.00	2.6	0.00	103.3	0.00	24.2	0.00	762.8	0.00	170.0
11			0.20	3.4	0.69	60.2	0.00	12.1	0.00	490.3	0.22	90.8
12			0.64	3.4	2.65	38.1	0.59	6.7	1.04	277.7	0.78	44.3
13			0.29	3.2	2.00	25.3	1.82	4.1	0.73	197.5	0.65	26.5
14			0.30	3.4	0.00	18.3	0.30	2.7	0.00	166.0	0.75	19.8
15			0.00	4.5	3.38	13.3	2.00	2.0	2.39	124.7	1.81	13.2
16			0.00	4.3	3.00	9.5	1.96	1.0	3.00	97.2	1.72	9.4
17			0.07	4.4	2.12	7.2	1.18	1.0	2.12	79.5	1.65	6.9
18			0.00	3.6	2.42	5.8	1.88	1.0	1.98	64.2	2.16	5.5
10	3	6	1.66	13.1	1.27	41.9	0.63	8.1	0.00	584.4	0.00	153.2
11			1.28	11.9	1.50	22.4	0.73	3.5	0.29	364.5	0.20	81.8
12			2.36	10.3	2.55	12.4	2.14	1.8	0.95	219.0	0.00	43.5
13			1.52	9.2	1.34	9.0	1.28	1.3	0.74	127.8	0.28	19.9
14			0.93	8.4	1.44	4.9	1.89	1.0	0.70	94.0	0.62	13.0
15			0.21	7.9	1.10	3.6	1.10	1.0	0.90	72.9	1.00	8.5
16			1.48	7.3	1.75	2.7	1.62	1.0	1.59	50.8	0.46	5.6
17			0.00	7.3	0.36	1.8	0.36	1.0	0.36	40.5	0.28	4.3
18			0.09	6.9	0.09	1.5	0.09	1.0	0.09	34.3	0.00	3.4
10	3	10	1.20	15.9	1.62	20.0	1.18	4.0	0.48	390.8	0.48	113.0
11			2.01	12.5	1.59	10.6	1.98	1.9	0.14	251.8	0.07	63.5
12			1.73	10.9	2.34	5.1	3.42	1.0	0.48	148.6	0.04	32.6
13			0.98	9.1	2.43	3.7	1.17	1.0	1.12	94.7	0.07	17.9
14			0.67	8.1	1.10	2.1	0.96	1.0	0.88	56.6	0.35	8.4
15			0.07	7.8	1.24	1.4	1.24	1.0	1.10	39.2	0.68	5.4
16			0.04	7.0	1.50	1.0	1.50	1.0	1.50	26.7	1.08	3.6
17			0.00	6.4	0.39	1.0	0.39	1.0	0.39	19.7	0.39	2.8
18			0.39	6.1	0.77	1.0	0.77	1.0	0.71	14.8	0.04	2.0

Table 2 Avarage objective values of 10 instances (normalized w.r.t. the best solution found) and the average numbers of iterations for experiment 3.

The recovery model described in Section 2 includes the possibility to change the current tour once a scenario becomes known, but under the condition that the recovery action results in a new traveling salesman tour. One can also consider the case that it is allowed to skip a node along a tour instead, i.e., when the scenario becomes known, we are allowed to modify our tour by leaving out up to L cities. Then, the optimal recovery action can be found in polynomial time by solving a shortest path problem. The evaluation problem can be solved in polyonomial time by using strong duality on the inner shortest path problem for polyhedral uncertainty sets; however, for $\mathcal{U}(\Gamma)$, the evaluation is not solvable in polynomial time, as can be seen by a reduction from the max-scenario-problem in [Büs09].

References

[ABV09] H. Aissi, C. Bazgan, and D. Vanderpooten. Minmax and minmax regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427 – 438, 2009.

[BAvdH11] P.C. Bouman, J.M. Akker, and J.A. van den Hoogeveen. Recoverable robustness by column generation. In *European Symposium on Algorithms*, volume 6942 of *Lecture Notes in Computer Science*, pages 215–226. Springer, 2011.

[BBC11] D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[BS04] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[BTGGN03] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Math. Programming A*, 99:351–376, 2003.

[BTGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, Princeton and Oxford, 2009.

[Büs09] C. Büsing. The exact subgraph recoverable robust shortest path problem. In R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 231–248. Springer Berlin Heidelberg, 2009.

[Büs12] C. Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

[FM12] M. Fischetti and M. Monaci. Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation*, 4(3):239–273, 2012.

[GHMH+13] M. Goerigk, S. Heße, M. Müller-Hannemann, M. Schmidt, and A. Schöbel. Recoverable Robust Timetable Information. In D. Frigioni and S. Stiller, editors, *Proc. of ATMOS 13*, volume 33 of *OASIcs*, pages 1–14, 2013.

[Goe14] M. Goerigk. A note on upper bounds to the robust knapsack problem with discrete scenarios. *Annals of Operations Research*, pages 1–9, 2014.

[GS14] M. Goerigk and A. Schöbel. Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. *Computers & Operations Research*, 52, Part A(0):1 – 15, 2014.

[GS15] M. Goerigk and A. Schöbel. Algorithm engineering in robust optimization. *LNCS State-of-the-Art Surveys Springer*, 2015. To appear.

[Hel00] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106 – 130, 2000.

[KY97] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, 1997.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The traveling salesman problem: a guided tour of combinatorial optimization*, volume 3. Wiley New York, 1985.

[LLMS09] C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Note on Computer Science*, pages 1–27. Springer, 2009.

[MBMG07] R. Montemanni, J. Barta, M. Mastrolilli, and L. M. Gambardella. The robust traveling salesman problem with interval data. *Transportation Science*, 41(3):366–381, 2007.

[Mon06] R. Montemanni. A benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 174(3):1479 – 1490, 2006.

[ZZ13] B. Zeng and L. Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457 – 461, 2013.