A Robust Approach to the Capacitated Vehicle Routing Problem with Uncertain Travel Times

Lars Eufinger† Jannis Kurtz‡ Christoph Buchheim† Uwe Clausen‡

November 7, 2017

Abstract

We investigate a robust approach for solving the capacitated vehicle routing problem (CVRP) with uncertain travel times. It is based on the concept of K-adaptability, which allows to calculate a set of k feasible solutions in a preprocessing phase. Once a scenario occurs, the corresponding best solution may be picked out of the set of candidates. The aim is to determine the k candidates such that the respective best one of them is worst-case optimal, which leads to a min-max-min problem.

In this paper, we propose an oracle-based algorithm for solving the resulting min-max-min CVRP, calling an exact algorithm for the deterministic problem in each iteration. Moreover, we adjust this approach such that also heuristics for the CVRP can be used. In this way, we derive a heuristic algorithm for the min-max-min problem, which turns out to yield good solutions in a short running time. All algorithms are tested on standard benchmark instances of the CVRP.

Capacitated Vehicle Routing Problem, Robust Optimization, K-Adaptability

1 Introduction.

The Vehicle Routing Problem (VRP) is one of the most widely studied optimization problems in the field of Operations Research. Dantzig and Ramser [16] introduced the Truck Dispatching Problem in 1959, modeling how a fleet of homogeneous trucks could serve the demand for oil of a number of gas stations from a central hub and with a minimum traveled distance. Five years later in 1964, Clarke and Wright [13] generalized this problem to a linear optimization

---

*This work has been supported by the German Research Foundation (DFG) within the Research Training Group 1855 and under grant no. BU 2313/2 – CL 318/14.
†Department for Mathematics, TU Dortmund University, Vogelpothsweg 87, 44227 Dortmund, Germany
‡Institute of Transport Logistics, TU Dortmund University, Leonhard-Euler-Straße 2, 44227 Dortmund, Germany
problem, which became known as the VRP. The VRP is commonly encountered in the domain of logistics and transport: the question is how to serve a set of customers, which are located geographically distributed around a central depot, using a fleet of trucks. The Capacitated Vehicle Routing Problem (CVRP) is an extension of the classical VRP, where the trucks have a given capacity. Several surveys on solution methods for the VRP and the CVRP have been published by Toth and Vigo [37, 38, 39], Baldacci et al. [3, 4], Laporte [28], Golden et al. [19] and Cordeau et al. [15]. Although the VRP in its deterministic version is extensively studied in the literature it is still not possible to solve the VRP for large size instances arising in practical applications. Therefore many heuristic algorithms were developed to compute feasible solutions for the VRP without a guarantee for the quality of the solution [14, 26, 29, 39].

In practical applications of the VRP, many parameters may be unknown in advance or disturbances can occur. For example, unknown traffic situations or unknown future demands of customers have to be considered during the optimization process, in order to find a solution which works well under all possible (or at least probable) realizations of the uncertain parameters. Usually, it is not enough to consider an average traffic situation, since an optimal solution for this scenario can perform very badly in other traffic situations. Furthermore, a solution which satisfies certain demands need not be feasible for all realizations of future demands. To address such problems, besides other approaches, robust optimization has been proposed as a paradigm. Here an uncertainty set \( U \) is given that contains all possible scenarios, i.e. all outcomes of the uncertain parameters that should be taken into account. The aim is then to find a solution which is feasible for all scenarios in \( U \) and which is worst-case optimal, i.e. the worst objective value over all scenarios in \( U \) is optimized. The robust optimization approach has been studied intensively for different classes of uncertainty sets \( U \) and for several combinatorial optimization problems [2, 6, 7, 10, 25, 35].

One well-known drawback of robust solutions is that, since all possible scenarios in \( U \) are assumed to be equiprobable and only the worst case is relevant, they often perform badly in most of the scenarios. Another drawback is that the calculated robust solution afterwards is static and has to be used in any scenario. At the other extreme, computing a different optimal solution from scratch every time a new scenario materializes is too time-consuming for the VRP, as discussed above. Moreover, a human user, in this case the driver, might prefer solutions that do not change every day. In this paper, we investigate an intermediate approach to tackle these drawbacks: we propose to solve vehicle routing problems by an extended model called min-max-min robustness, which aims to calculate \( k \) different solutions for the underlying problem in a preprocessing step such that the best of them can be chosen each time a new scenario occurs. In the case of the CVRP, a company which has to serve the same customers every day could calculate an optimal set of \( k \) solutions once by the proposed model and, each morning when the traffic scenario is known, choose the best of the \( k \) solutions.

This approach is based on the concept of K-adaptability for robust two stage problems [8, 23] and was introduced in [12]. The resulting problem of finding the
best set of $k$ solutions can be modeled as a min-max-min optimization problem. Clearly, this approach is less conservative than the standard robust approach: since we cover the uncertainty by $k$ solutions instead of one, in each scenario we obtain solutions which are not worse (and usually better) than the classical robust optimal solution. In [12] the authors derived an oracle based polynomial time algorithm to solve the min-max-min problem for large enough $k$ and showed that it works very well on random knapsack instances.

1.1 Related Literature

There are only a few works in the literature which consider robust versions of the VRP. A variant with time windows and uncertain travel times has been studied by Agra et. al. [1]. Here among others the authors apply the framework of adjustable robustness to the problem. In [20, 36, 30] the CVRP with uncertain demands has been studied. Wohlgemuth et. al [40] studied the pickup and delivery problem (PDP), a variant of the VRP, with variable demand and transport conditions. The dynamic PDP is modeled as a multi-stage mixed integer problem and solved via tabu search.

The lack of literature dealing with the robust VRP can probably be explained by the fact that even robust versions of classical tractable combinatorial optimization problems such as the shortest path problem can become $NP$-hard, even in the case $|U| = 2$ [25]. Furthermore, these problems are hard to solve in practice for most of the general uncertainty classes studied in the literature. Since the VRP is already hard in its deterministic version, there is not much hope to achieve tractable robust versions of the problem in general.

1.2 Contribution and Overview

In this paper, we investigate the potential of the new min-max-min-robust approach for the CVRP. As mentioned above, in [12] an algorithm has been proposed to solve the min-max-min problem for any combinatorial problem, given by an algorithm for the deterministic version. In this paper we apply the latter algorithm to the CVRP and run it on several benchmark instances. Furthermore, we adjusted the algorithm to make use of heuristics for the CVRP and derive a heuristic algorithm for the min-max-min problem by this idea.

It turns out that the adjusted algorithm which combines exact and heuristic algorithms for the CVRP does not yield an improvement of the running time. On the other hand the heuristic version of the algorithm is very fast and calculates solutions which are not too far from optimal in most of the instances. Furthermore the number of solutions calculated by the heuristic is very low in average.

In Section 2, we will define the CVRP and give an overview over the exact and heuristic methods which we used in our computations. In Section 3, we recall the min-max-min robust approach and introduce the aforementioned algorithms. We present the results of our computations for the min-max-min robust CVRP in Section 4, followed by a conclusion.
2 The Capacitated Vehicle Routing Problem.

In the literature many different variants of the vehicle routing problem were presented [27, 39]. In this paper we will study the capacitated vehicle routing problem and apply the min-max-min robust approach to it. In the following assume $G = (V, E)$ to be a directed complete graph with nodes $V = \{0\} \cup V_C$ and arcs $E = \{(i, j) \mid i, j \in V, i \neq j\}$. Node 0 represents the depot and the nodes in $V_C$ represent the customers. Each customer $i \in V_C$ has a positive demand $d_i \in \mathbb{R}_+$ while we set $d_0 = 0$. Furthermore we have a set of vehicles $K = \{1, \ldots, m\}$ where each vehicle has the same capacity $C \in \mathbb{R}_+$ and a cost function $c : E \rightarrow \mathbb{R}$ on the edges of $G$ where the cost can be interpreted as traveling times. A tour $T \subset E$ in $G$ is a directed cycle in $G$ which traverses the depot. This can be interpreted as a tour of a vehicle which starts from the depot and supplies a subset of customers before it returns to the depot. The cost of a tour $T$ is defined by

$$c(T) := \sum_{e \in T} c(e)$$

and the demand of a tour is the sum over all demands of the customers which are served by the tour, i.e.

$$d(T) := \sum_{(v, w) \in T} d_v.$$

The capacitated vehicle routing problem is now to find a set of $m$ tours which minimize the total cost such that the sum of all demands on each tour does not exceed the capacity of the vehicle. Formally we define the problem as follows.

**Problem 2.1 (CVRP)** Let $G = (V, E)$ be a complete directed graph and $c$ and $d$ defined like above. Find a set of tours $T_1, \ldots, T_m \subset E$ with $d(T_i) \leq C$ for $i = 1, \ldots, m$ and with minimal cost

$$c(T_1, \ldots, T_m) := c(T_1) + \ldots + c(T_m)$$

such that each customer $i \in V_C$ is traversed by exactly one of the tours.

Note that in Problem (2.1) every vehicle has to be used. In the case of a variable number of vehicles often costs on the use of vehicles are defined. Another assumption we make is that the graph has to be directed. This is due to the fact that in real-world applications the travel time from $A$ to $B$ can be different from the travel time from $B$ to $A$ in some of the possible traffic scenarios.

In the literature several methods and algorithms were presented to solve the capacitated vehicle routing problem [15, 19, 28, 39]. Besides dynamic programming methods one way to solve the CVRP is to formulate it as an integer program and to solve it by Branch & Bound methods. One integer programming formulation presented in the literature and which we will use in our computations in Section 4 is the *Miller-Tucker-Zemlin (MTZ)* formulation [33]. Here
we identify each edge $e = (i, j) \in E$ with a variable $x_{ij} \in \{0, 1\}$ which has value one if the edge is contained in any of the $m$ tours and zero otherwise. Additionally we add variables $u_i \geq 0$ for each customer $i \in V_C$ which represent the total demand a vehicle supplies up to the point immediately after it leaves customer $i$. The MTZ problem is then of the form

$$\min \quad c^\top x$$

$$\text{s.t.} \quad \sum_{i \in V, i \neq j} x_{ij} = \sum_{i \in V, i \neq j} x_{ji} = 1 \quad \forall j \in V_C$$

$$\sum_{i \in V_C} x_{i0} = \sum_{i \in V_C} x_{0i} = m$$

$$u_j - u_i + C(1 - x_{ij}) \geq d_j \quad \forall i, j \in V_C, \ i \neq j$$

$$d_i \leq u_i \leq C \quad \forall i \in V_C$$

$$x_{ij} \in \{0, 1\} \quad i, j \in V$$

$$u_i \in \mathbb{R}_+ \quad i \in V_C$$

Here the constraints (2) and (3) ensure that each customer is visited exactly once on a tour and that exactly $m$ vehicles leave the depot and return to it. The constraints (4) ensure that if the edge $(i, j)$ is passed by a vehicle then $u_j = u_i + q_j$ holds and otherwise constraints (5) ensure that $u_i$ does not exceed the capacity and has at least the value $q_i$. It can be easily verified that each feasible solution of the latter problem induces a feasible solution for Problem 2.1 and that each feasible solution for Problem 2.1 is feasible for the latter problem.

Since the vehicle routing problem is an NP-hard problem [31] and very hard to solve exactly in high dimensions, often heuristic algorithms are used to calculate feasible solutions. One of the most used heuristics for the VRP is the Clarke-Wright algorithm [13], also known as the savings algorithm. Given an instance of Problem 2.1, the Clarke-Wright algorithm starts with $|V_C|$ tours where on each tour one single customer is visited. These routes are then merged, this process is guided by the savings criterion. Let $c_{ij}$ be the costs of edge $(i, j)$. Then the initial solution has total costs of $\sum_{i \in V_C} 2c_{0i}$. For each pair of customers $(i, j)$ the (asymmetric) savings value $s_{ij}$ is defined by

$$s_{ij} = (c_{0i} + c_{i0} + c_{0j} + c_{j0}) - (c_{0i} + c_{ij} + c_{j0}),$$

which is the decrease in the objective function obtained by merging the two initial routes servicing nodes $i$ and $j$. Note that in the asymmetric case we get two different savings values $s_{ij}$ and $s_{ji}$ for each pair of customers $(i, j)$. The Clarke-Wright algorithm uses these savings values to determine the order in which the initial routes are merged to form larger routes. For the CVRP routes are only merged if the vehicle capacity of the resulting route is not exceeded. The latter merging process is performed iteratively until no more routes can be merged without leading to infeasible solutions or the merging would result in higher costs. For the computations in Section 4 a variant of the Clarke-Wright algorithm [41] is used, which incorporates a shape parameter $\lambda$ into the savings
One-Point move: Relocate an Existing node into a new position.

Two-Point move: Swap the position of two nodes.

Inter-route Two-Opt move: Remove one edge from two different routes and replace them with two new edges.

Intra-route Two-Opt move: Remove two edges from a single route and replace them with two new edges.

Three-Point move: Swap the position of a pair of adjacent nodes with the position of a third node.

Three-Point move: Swap the position of a pair of adjacent nodes with the position of a third node.

\[ \tilde{s}_{ij} = c_{0i} - \lambda c_{ij} + c_{0j} \] Note that not necessarily a feasible solution can be found by the latter procedure due to the restriction for the number of vehicles.

To improve feasible solutions of the vehicle routing problem generated by the Clarke-Wright algorithm, local search operators can be applied. For the VRP, the neighborhoods are typically defined in terms of a heuristic operation, where a node in a given solution is moved to a new position, or several edges are removed and replaced with new edges. In other words, given a solution and a heuristic operation, the neighborhood consists of all solutions that can be created by applying the heuristic operation to the current solution. The six different local search operators are shown in Figure 1. For each operator, a diagram illustrating the change to the solution is shown and described briefly. Local search has proven to be very effective for the VRP and this technique is used in many VRP metaheuristics.
3 Min-max-min Robustness.

In the classical robust optimization approach a so called uncertainty set \( U \subseteq \mathbb{R}^n \) is given which contains all possible scenarios of the uncertain parameters. In this paper we assume that only the costs of the CVRP are uncertain such that we can formulate the CVRP as a linear problem of the form

\[
\min_{x \in X} c^\top x
\]

where \( X \subseteq \{0,1\}^n \) is the set of incidence vectors of all feasible solutions and the set \( U \) contains all relevant cost vectors \( c \). The robust counterpart is then given by the min-max problem

\[
\min_{x \in X} \max_{c \in U} c^\top x. \tag{M^2}
\]

The latter problem has been studied intensively in the robust optimization literature for different classes of uncertainty sets \( U \). The two main classes which are studied are finite and convex sets. If \( |U| \) is finite we say that \( U \) is a discrete uncertainty set. In the convex case, mostly polyhedral or ellipsoidal sets are studied. One well-studied subclass of the polyhedral uncertainty sets are the budgeted uncertainty sets introduced in [10]. Given an interval \([l_i, u_i]\) in which each parameter can vary, it is very unlikely in practice that all parameters differ from their nominal value at the same time. To this end the authors introduce a fixed parameter \( \Gamma \) which gives the maximal number of parameters which can differ from their nominal values. This set can be modeled by a polyhedron (see Section 4). It was shown in [9] that the min-max problem \((M^2)\) with budgeted uncertainty can be solved by solving \( n + 1 \) instances of the underlying deterministic problem. Therefore it remains tractable whenever the underlying problem is tractable. In contrast to this, it was shown that Problem \((M^2)\) is \(NP\)-hard for several tractable combinatorial problems under general discrete, polyhedral and ellipsoidal uncertainty [2, 35]. Moreover, Problem \((M^2)\) is rarely used for practical applications since it can produce solutions which can be far from optimal in many scenarios; see [10] and Example 3.1. To tackle the latter problems, several extensions of the robust approach have been introduced; see e.g. [5, 18, 23, 32, 34]. Another new approach is the so called min-max-min robustness introduced by [12]. Here the authors propose to calculate \( k \) feasible solutions for the underlying combinatorial problem by solving the problem

\[
\min_{x^{(1)}, \ldots, x^{(k)} \in X} \max_{c \in U} \min_{i=1, \ldots, k} c^\top x^{(i)}. \tag{M^3}
\]

The idea of the approach is to calculate \( k \) different solutions \( x^{(1)}, \ldots, x^{(k)} \) instead of one, such that the worst-case over all scenarios of the respectively best solution in each scenario is optimized. Regarding a practical application, after calculating the optimal set of \( k \) solutions once, each time a scenario occurs the best of the \( k \) solutions for this scenario can be chosen by comparing the \( k \) objective values. Therefore problem \((M^3)\) only has to be solved once and the
calculated solutions hedge against the uncertainty in a robust way. Clearly this approach is less conservative than the standard min-max approach in the sense that for each upcoming scenario there exists always a solution $x^{(i)}$ in the optimal solution of Problem (M$^3$) which is at least as good as the classical min-max solution. This is emphasized by the following example.

**Example 3.1** Consider an CVRP instance with 2 vehicles on the graph in Figure 2. The capacity of the vehicles is defined such that all pairs of customers can be served, but not all customers together. The costs on the edges are the same in both directions and we assume budgeted uncertainty with $\Gamma = 1$. The related interval $[l_i, u_i]$ for the possible costs on each edge is given in Figure 2.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{example_graph.png}
\caption{Figure 2:}
\end{figure}

Then one of the optimal min-max solutions with optimal value 300 is the solution where customers 2 and 3 are served by one vehicle and customer 1 by the other vehicle. The worst case scenario is the one where edge $\{1, 2\}$ has costs 20 and the edge $\{2, 3\}$ has costs 100. The other optimal solution is the solution where customer 1 and 2 are served by one vehicle and customer 3 by the other. Here the worst case scenario is the one where edge $\{1, 2\}$ has costs 100 and the edge $\{2, 3\}$ has costs 20. If we consider the min-max-min problem with $k = 2$, the optimal solution consists of the both latter solutions but has an optimal value of 260 since the worst case scenario has costs 60 on both edges $\{1, 2\}$ and $\{2, 3\}$. Note that the absolute deviation from the min-max optimal value can be arbitrarily high if we increase the upper bound for the costs on the edges $\{1, 2\}$ and $\{2, 3\}$.

A practical motivation for problem (M$^3$) is a supplier which has to deliver goods to the same customers every day. Depending on the traffic situation every day the company wants to find the best solution to serve all customers with the available fleet of vehicles, i.e. it has to solve a vehicle routing problem with uncertain travel times. This problem is known to be hard to solve and in high dimension it cannot be solved in appropriate time every morning when the actual traffic scenario is known. In contrast Problem (M$^3$) only has to be solved once in a perhaps expensive preprocessing. Afterwards the best of
the k solutions can be chosen every morning depending on the current traffic situation. This can be done easily by comparing the objective values of the calculated solutions for the current traffic situation. Therefore even if a very short time-window is given by a company to calculate a solution for the scenario which came up, the latter approach can be used instead of heuristic algorithms.

In [12] it was shown that the min-max-min problem (M$^3$) can be solved in oracle-polynomial time if the uncertainty set $U$ is convex and if $k \geq n+1$. More precisely the authors present a polynomial time algorithm which calls an oracle for the deterministic problem (M) at most polynomial many times. Moreover, the authors show that the optimal value of Problem (M$^3$) is the same for all $k \geq n$. Note that an optimal solution for the problem with $k = n$ must not consist of $n$ pairwise different solutions and can therefore be optimal even for the problem with a lower $k$ (see Section 4).

For the case $k \geq n$ in [12] Algorithm 1 is provided to solve the min-max-min problem. Note that the problem which has to be solved in Step 4 depends on the uncertainty set $U$ while in Step 5 the deterministic problem has to be solved for the given cost vector $c^*$. For polyhedral or ellipsoidal uncertainty sets the problem in Step 4 is a continuous linear or quadratic problem respectively. This can be solved by the latest versions of optimization software like CPLEX [24]. Therefore Algorithm 1 can be implemented for any combinatorial problem of the form (M) and any algorithm for the combinatorial problem can be used. Note that we exclude all redundant solutions in the return statement. In [12] the authors applied the latter algorithm to the knapsack problem and showed that on random instances the algorithm calculates significantly less than $n$ solutions in general.

In the following section, we will apply Algorithm 1 to the vehicle routing problem defined in Section 2. The vehicle routing problem is already hard to solve in its deterministic version, which has to be done in Step 5, and therefore several heuristic algorithms have been developed in the literature (see Section 2).

---

**Algorithm 1** Algorithm to solve Problem (M$^3$) for $k \geq n$

**Input:** convex $U \subset \mathbb{R}^n$, $X \subseteq \{0, 1\}^n$

**Output:** optimal solution of Problem (M$^3$)

1. $i := 0$
2. choose any $x^*_0 \in X$
3. repeat
   4. calculate optimal solution $(z^*, c^*)$ of
      \[ \max \{ z \mid c^T x_j^* \geq z \quad \forall j = 0, \ldots, i, \quad z \in \mathbb{R}, \quad c \in U \} \]
   5. calculate optimal solution $x^*_{i+1}$ of
      \[ \min_{x \in X} (c^*)^T x \]
   6. $i := i + 1$
4. until $(c^*)^T x^*_i \geq z^*$
7. return \{ $x^*_j \mid (c^*)^T x^*_j = z^*$ \}
To make use of these algorithms we adjusted Algorithm 1 as follows. Instead of solving the deterministic problem exactly in Step 5 we calculate a feasible solution by using a heuristic algorithm as long as the stopping criterion of the loop is not fulfilled. If the heuristic algorithm does not return an improving solution anymore we switch back to the exact algorithm. Let \( H : \mathbb{R}^n \to X \cup \{ \emptyset \} \) be a function which returns for any cost vector \( c \in \mathbb{R}^n \) a feasible solution \( H(c) \) or, in the case that no feasible solution could be found, the empty set. Then the adjusted algorithm is given by Algorithm 2. Note that the result is still a provably optimal solution to the min-max-min problem \((M^3)\).

\[ \begin{align*} \text{Algorithm 2} \quad \text{Algorithm to solve Problem (M^3)} & \quad \text{for} \quad k \geq n \\ \text{Input}: & \quad \text{convex } U \subset \mathbb{R}^n, \quad X \subseteq \{0,1\}^n \\ \text{Output}: & \quad \text{optimal solution of Problem (M^3)} \\ 1: \quad i := 0 \\ 2: \quad \text{choose any } x_0^* \in X \\ 3: \quad \text{repeat} \\ 4: \quad \text{calculate optimal solution } (z^*, c^*) \text{ of } \max \{ z \mid c^T x_j^* \geq z \forall j = 0, \ldots, i, \quad z \in \mathbb{R}, \quad c \in U \} \\ 5: \quad \text{calculate } H(c^*) \\ 6: \quad \text{if } (c^*)^T H(c^*) < z^* \text{ then} \\ 7: \quad x_{i+1}^* := H(c^*) \\ 8: \quad \text{else} \\ 9: \quad \text{calculate optimal solution } x_{i+1}^* \text{ of } \min_{x \in X} (c^*)^T x \\ 10: \quad \text{end if} \\ 11: \quad i := i + 1 \\ 12: \quad \text{until } (c^*)^T x_i^* \geq z^* \\ 13: \quad \text{return } \{x_j^* \mid (c^*)^T x_j^* = z^*\} \end{align*} \]

Several of the vehicle routing instances given in [17] were not solved to optimality yet even in the deterministic case where no uncertainty is considered. Clearly for these instances we are also not able to solve Problem \((M^3)\) exactly by the algorithms above. Since for many of the latter instances heuristic solutions can be calculated very fast, we propose a heuristic variant of the algorithm which only uses a heuristic algorithm for the deterministic problem instead of the exact algorithm in Algorithm 1. Algorithm 3 calculates a feasible solution for Problem \((M^3)\) but without any guarantee for the quality of the objective value. Nevertheless in our computations it turned out that the heuristic variant produces good solutions and runs very fast, as shown in the next section.
Algorithm 3  Heuristic for Problem (M³) for \( k \geq n \)

**Input:** convex \( U \subseteq \mathbb{R}^n \), \( X \subseteq \{0,1\}^n \)

**Output:** feasible solution of Problem (M³)

1: \( i := 0 \)
2: choose any \( x^*_0 \in X \)
3: repeat
4: calculate optimal solution \( (z^*, c^*) \) of
   \[
   \max \{ z \mid c^T x_j^* \geq z \quad \forall j = 0, \ldots, i, \quad z \in \mathbb{R}, \ c \in U \} 
   \]
5: calculate \( x^*_{i+1} = H(c^*) \)
6: \( i := i + 1 \)
7: until \( (c^*)^T x^*_{i} \geq z^* \)
8: return \( \{ x_j^* \mid (c^*)^T x_j^* = z^* \} \)

4  Computations.

In the following we present our results on Problem (M³) for the CVRP. As mentioned before this problem is very hard to solve in practice even in its deterministic version. Since Algorithm 1 has to solve the deterministic problem several times, clearly it is at most as efficient as the algorithm for the underlying deterministic version. In particular, the instances which have not been solved to optimality yet even in the deterministic version cannot be solved by our algorithm as well.

For the computations below we implemented the original Algorithm 1 introduced by [12] and the two variants of it shown in Section 3. For the exact CVRP oracle we implemented the MTZ formulation in CPLEX 12.4 [24]. The dual problem in Step 4 was also implemented in CPLEX 12.4. As a heuristic algorithm for the deterministic problem we used the open source library VRPH [22]. Here 3 initial solutions are calculated via the modified Clarke-Wright algorithm described in Section 2 using different values of the shape parameter \( \lambda \). The best solution is then used as a starting solution for the simulated annealing heuristic.

For the simulated annealing the operations One-Point, Two-Point and Two-Opt move are used; see Fig. 1 again. After the simulated annealing stage, a tour cleanup is performed. During the cleanup, each tour is improved individually via local search if possible. The local search operators used in the cleanup are One-Point, Two-Point, Two-Opt, Three-Opt and Three-Point move. Note that during the simulated annealing stage the local search operators can be inter-route and intra-route operators, but during the cleanup stage the used operators can be intra-route only. All computations were calculated on a cluster of 64-bit Intel(R) Xeon(R) E5-2670 processors running at 2.60 GHz with 20MB cache.

For our computations we chose several instances from [17]. For each instance we created 10 ellipsoidal and 10 budgeted uncertainty sets. To this end as the mean cost vector \( \bar{c} \) (i.e. the center of the ellipsoid or the lower interval limit \( l \) for the budgeted uncertainty set respectively) we chose the Euclidean distances between the coordinates of the nodes given by the instances. If no coordinates
are defined then we chose the edge-weights which are given by the instance. In
the following \( n \) is the number of edges in the underlying graph of the CVRP
instance, i.e. \( n = |V|(|V| - 1) \).

The ellipsoidal uncertainty sets are then given by
\[
U^E := \left\{ c \in \mathbb{R}^n \mid (c - \bar{c})^T \Sigma^{-1} (c - \bar{c}) \leq \Omega^2 \right\} ,
\]
where \( \Sigma \in \mathbb{Q}^{n \times n} \) is a symmetric positive semidefinite matrix and \( \Omega \in \mathbb{N} \). To
create the matrix \( \Sigma \) we created random orthonormal bases which give the direc-
tion of the extreme rays of the ellipsoid. Similar to [11] the length of the rays
were chosen as \( \sqrt{\delta_j c_j} \), where \( \delta_j \) is a random number in \([0, 1]\). Note that the
resulting ellipsoids are not axis-parallel in general and therefore more general
than the ellipsoids used by [11]. For any instance, we scaled the ellipsoid by
varying the parameter \( \Omega \in \{1, 3, 5\} \).

We created budgeted uncertainty sets by
\[
U^\Gamma := \left\{ c \in \mathbb{R}^n \mid c_i = l_i + \delta_i (u_i - l_i), \sum_{i=1}^n \delta_i \leq \Gamma \right\} ,
\]
where \( \Gamma \) is a given parameter and \( l \) is the mean cost vector described above.
Here we chose \( (u_i - l_i) \) randomly between 0 and \( l_i \) for all \( i = 1, \ldots, n \). Each
instance has been solved for all values of \( \Gamma \) from the set
\( \{0.15(|V_C| + m), 0.5(|V_C| + m), 0.75(|V_C| + m)\} \),
rounded down if fractional. The latter choice is motivated by the fact that each
feasible solution of Problem (M3) uses exactly \(|V_C| + m\) edges in the graph.
For \( \Gamma \geq |V_C| + m \) our computations showed that often only one solution is
needed to solve Problem (M3) exactly; in this case, the solution agrees with the
optimal solution of Problem (M2).

| Instance | \(|V_C|\) | \(|E\) | \(m\) | \(|\bar{c}\) | \(\delta\) | \(\Omega\) | \(\bar{\delta}\) | \(\bar{\Omega}\) | \(\bar{\delta}\) | \(\bar{\Omega}\) | \(\bar{\delta}\) | \(\bar{\Omega}\) |
|----------|----------|--------|------|--------|------|------|--------|--------|------|--------|--------|------|--------|
| E-n13-k2 | 6        | 42     | 2    | 1      | 13.4 | 13.0 | 13.0   | 0.3    | 0.0   | 0.3    | 0.3    | 11.8 | 8.5   |
|          |          |        |      |        |      |      |        |        |      |        |        |      |        |
|          |          |        |      |        |      |      |        |        |      |        |        |      |        |
|          |          |        |      |        |      |      |        |        |      |        |        |      |        |

Table 1: Results of Algorithm 1 and 2 for CVRP with ellipsoidal uncertainty.

In the Tables 1, 2, 3, and 4 we list the computational results for a selection
of instances. For each combination of \(|V_C|\) and \( \Omega \) or \(|V_C|\) and \( \Gamma \), respectively,
we show the average over all 10 instances of the following numbers (from left
to right): the difference (in percent) of the objective value of Problem (M3)
to the value of the certain problem with the ellipsoid center \( \bar{c} \) or the lower interval
limit \( l \), respectively, as cost function; the number of solutions in the computed set \( X^* \); the number of major iterations; the run-times used by the two oracles (\( t_{dual} \) for the dual problem in Step 4 and \( t_{comb} \) for solving the deterministic vehicle routing problem (M) in Step 5) and the total run-time \( t_{tot} \). Furthermore for Algorithm 2 where we use both the exact and the heuristic oracle, we show the number of calls of the exact oracle in column iter\(_e\) and the number of calls of the heuristic oracle in column iter\(_h\). Table 3 and 4 which show the results for Algorithm 3, which only uses the heuristic algorithm, includes the column diff\(_h\). In this column the difference (in percent) of the objective value of the calculated heuristic solution and of the optimal value of Problem (M) is shown if the latter could be calculated. All times are given in CPU seconds and all numbers are rounded to one decimal.

| Inst   | \( |V| \) | \( |C| \) | \( |E| \) | \( m \) | \( c \) | \( \Omega \) | \( X^* \) | \( \#s \) | \( t_{dual} \) | \( t_{comb} \) | \( t_{tot} \) | \( t_{dual} \) | \( t_{comb} \) | \( t_{tot} \) |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| E-n13-k4 | 42   | 156   | 42   | 2    | 4.8   | 11.2   | 18.4   | 96.7   | 0.0   | 76.7   | 11.3   | 11.0   | 9.7   | 55.4   | 0.0   | 55.4   |
| P-n16-k8 | 15   | 240   | 15   | 3    | 9.0   | 6.2    | 8.4    | 9.3    | 0.0   | 8.4    | 5.5    | 5.4    | 6.2    | 4.9    | 0.0   | 3.9    |
| P-n16-k8 | 15   | 240   | 15   | 3    | 9.0   | 6.2    | 8.4    | 9.3    | 0.0   | 8.4    | 5.5    | 5.4    | 6.2    | 4.9    | 0.0   | 3.9    |
| P-n20-k2 | 19   | 380   | 38   | 3    | 12.5  | 21.1   | 37.0   | 2259.8 | 0.0   | 2259.8 | 21.1   | 24.7   | 37.2   | 1860.3 | 0.0   | 1860.1 |
| E-n13-k4 | 42   | 156   | 42   | 2    | 4.8   | 11.2   | 18.4   | 96.7   | 0.0   | 76.7   | 11.3   | 11.0   | 9.7   | 55.4   | 0.0   | 55.4   |
| P-n16-k8 | 15   | 240   | 15   | 3    | 9.0   | 6.2    | 8.4    | 9.3    | 0.0   | 8.4    | 5.5    | 5.4    | 6.2    | 4.9    | 0.0   | 3.9    |
| P-n16-k8 | 15   | 240   | 15   | 3    | 9.0   | 6.2    | 8.4    | 9.3    | 0.0   | 8.4    | 5.5    | 5.4    | 6.2    | 4.9    | 0.0   | 3.9    |
| P-n20-k2 | 19   | 380   | 38   | 3    | 12.5  | 21.1   | 37.0   | 2259.8 | 0.0   | 2259.8 | 21.1   | 24.7   | 37.2   | 1860.3 | 0.0   | 1860.1 |

Table 2: Results of Algorithm 1 and 2 for CVRP with budgeted uncertainty.

In Table 1 we show the results of Algorithm 1 and Algorithm 2 for ellipsoidal uncertainty sets. The number of calculated solutions and the number of iterations clearly grows with increasing \( \Omega \). The same holds for the difference of the objective value. Note that in instance gr-n17-k3 the difference is nearly 95% for \( \Omega = 5 \), which is not surprising, since a larger uncertainty set clearly leads to more conservative solutions even in the min-max-min approach. Furthermore the total run-time as well as the number of solutions increase with the dimension. Instance P-n16-k8 could be solved very fast compared to the other instances since the deterministic versions were solved faster by the MTZ-formulation. The deterministic oracle takes most of the run-time, while the dual oracle ran in a few seconds. Note that the combination of the exact oracle and the heuristic oracle does only improve the run-time in instance E-n13-k4. This is possibly due to the fact that the heuristic solutions do not improve the dual problem significantly in each iteration and the resulting higher number of iterations increases the run-time. Nevertheless it turned out that for instances with more than 16 customers we could not solve all configurations for the 10 ellipsoidal instances within days.

The results for budgeted uncertainty in Table 2 are very similar but the total run-time is much lower. Here the combination of the exact and heuristic oracles is faster in instance P-n16-k8 and P-n20-k2 Overall the total run-time is much lower than for ellipsoidal uncertainty sets which is due to the lower number of iterations, also leading to a lower number of calculated solutions. In contrast to
Table 3: Results of Algorithm 3 for CVRP with ellipsoidal uncertainty.

| Inst. | | m | | |
|-------|--------|---|---|---|---|---|
|  | [V] | [E] | m | | |
| E-n7-k2 | 6 | 42 | 2 | 1 | 4.4 | 4.6 | 5.9 | 0.2 | 0.0 | 0.4 |
|  | 3 | 11.7 | 5.1 | 6.1 | 0.2 | 0.0 | 0.2 |
|  | 5 | 19.6 | 4.8 | 5.9 | 0.2 | 0.0 | 0.1 |
| E-n13-k4 | 12 | 156 | 4 | 1 | 1.6 | 8.2 | 11.0 | 1.5 | 0.4 | 0.9 |
|  | 3 | 4.2 | 9.8 | 12.3 | 1.6 | 0.4 | 1.0 |
|  | 5 | 8.3 | 10.3 | 12.7 | 1.6 | 0.5 | 1.0 |
| P-n16-k8 | 15 | 248 | 8 | 1 | 3.7 | 4.2 | 5.3 | 1.6 | 0.5 | 0.8 |
|  | 3 | 5.4 | 5.5 | 6.5 | 1.8 | 0.6 | 0.9 |
|  | 5 | 10.7 | 9.0 | 11.0 | 1.8 | 0.7 | 1.0 |
| gr-n17-k3 | 16 | 272 | 3 | 1 | 7.6 | 5.0 | 7.2 | 2.7 | 0.9 | 1.3 |
|  | 3 | 6.3 | 14.4 | 21.7 | 7.8 | 3.5 | 3.9 |
|  | 5 | 7.6 | 15.5 | 21.4 | 7.7 | 3.5 | 3.7 |
| gr-n21-k3 | 20 | 420 | 3 | 1 | - | 5.8 | 5.7 | 6.8 | 4.4 | 2.8 |
|  | 3 | - | 14.4 | 17.7 | 14.4 | 8.5 | 4.8 |
|  | 5 | - | 18.6 | 24.8 | 21.2 | 13.5 | 6.5 |

Table 4: Results of Algorithm 3 for CVRP with budgeted uncertainty.

| Inst. | | m | | |
|-------|--------|---|---|---|---|---|
|  | [V] | [E] | m | | |
| E-n13-k4 | 12 | 156 | 4 | 2 | 4.8 | 3.5 | 5.1 | 0.5 | 0.0 | 0.58 |
|  | 8 | 7.6 | 7.1 | 10.6 | 1.0 | 0.0 | 0.9 |
|  | 12 | 8.9 | 7.0 | 11.2 | 1.1 | 0.0 | 0.9 |
| P-n16-k8 | 15 | 248 | 8 | 3 | 4.9 | 2.6 | 4.0 | 0.5 | 0.0 | 0.5 |
|  | 11 | 9.4 | 3.8 | 6.1 | 0.8 | 0.0 | 0.7 |
|  | 17 | 8.7 | 4.0 | 6.0 | 0.8 | 0.0 | 0.7 |
| gr-n17-k3 | 16 | 272 | 3 | 2 | 9.6 | 2.5 | 4.5 | 1.0 | 0.0 | 0.8 |
|  | 9 | 11.4 | 5.8 | 8.3 | 1.6 | 0.0 | 1.5 |
|  | 14 | 8.9 | 8.3 | 12.5 | 2.4 | 0.0 | 2.3 |
| P-n20-k2 | 19 | 380 | 2 | 3 | 19.1 | 6.1 | 9.4 | 2.1 | 0.0 | 1.9 |
|  | 10 | 42.6 | 6.9 | 9.3 | 2.0 | 0.0 | 1.8 |
|  | 15 | 27.8 | 6.6 | 9.1 | 1.9 | 0.0 | 1.8 |
| gr-n21-k3 | 20 | 420 | 3 | 3 | - | 3.8 | 3.5 | 3.8 | 0.0 | 1.8 |
|  | 11 | - | 7.7 | 10.4 | 3.1 | 0.0 | 2.8 |
|  | 17 | - | 9.9 | 13.9 | 4.0 | 0.0 | 3.8 |
| A-n39-k5 | 48 | 1482 | 5 | 6 | - | 12.0 | 15.0 | 18.9 | 0.0 | 14.9 |
|  | 21 | - | 18.5 | 23.9 | 19.1 | 0.0 | 18.2 |
|  | 32 | - | 24.2 | 32.0 | 25.0 | 0.0 | 24.1 |
| A-n45-k6 | 44 | 1980 | 6 | 7 | - | 1.6 | 2.1 | 3.2 | 0.0 | 2.1 |
|  | 25 | - | 1.1 | 1.6 | 2.7 | 0.0 | 1.6 |
|  | 37 | - | 1.2 | 1.5 | 2.6 | 0.0 | 1.5 |

However, the results for Algorithm 3 are very promising. Both for ellipsoidal and budgeted uncertainty sets the total run-time and the number of calculated solutions is very low compared to the exact versions above. On average the total run-time never exceeded 25 seconds. Furthermore the difference to the exact value is often not higher than 12% in average. Nevertheless for instance P-n20-k2 it is nearly 43% for \( \Gamma = 10 \). We could even solve instance gr-n21-k3 in at most 22 seconds in average for ellipsoidal uncertainty sets, while we were not able to solve all the configurations for the exact version to optimality within days. For budgeted uncertainty we could even solve instances with around 40 customers e.g. A-n39-k5 and A-n45-k6 in at most 25 seconds on average. Here for the
latter instance Algorithm 3 calculated not more than 2 solutions on average, possibly because the heuristic algorithm does not produce good solutions for this instance. Possibly in this case the min-max-min solutions are not good as well.

In the following we present all solutions of an exact optimal solution of Problem (M3) for a selected instance. Since we assumed a directed graph which can have different costs on edges \((i,j)\) and \((j,i)\) for any customers \(i,j \in V_C\), it can happen that in one optimal solution one set of tours can occur more than once but at least one tour is oriented in a different direction (see solutions 8 and 10 in Fig. 3).

5 Conclusion.

In this paper we use a robust approach introduced in [12] to solve the capacitated vehicle routing problem (CVRP) with uncertain travel times. We calculate up to \(k\) feasible solutions which are optimal for the min-max-min problem and therefore hedge against the uncertain travel times of the vehicles in a robust way. We implemented the algorithm proposed in [12] and derived a heuristic algorithm for the problem by replacing the exact oracle, used in each iteration, by a heuristic oracle. Our results on several benchmark instances for different kinds of uncertainty sets look very promising. By using heuristic algorithms for the deterministic CVRP, we could not speed up the exact algorithm for the min-max-min problem, but the heuristic variant of the algorithm runs very fast and could also solve larger instances.

Several directions for future research can be identified. First, it would be promising to use different exact and heuristic algorithms for the deterministic CVRP in each iteration of Algorithm 1. Second, since often pairs of different solutions differ only in one tour (see Figure 3), one could try to adjust the idea of the min-max-min model and only calculate a set of tours instead of complete solutions. These tours could then be combined to a feasible solution by the user each time a scenario occurs. This would require however to define specific rules for the feasible combination of tours.

It also seems promising to adapt the min-max-min robust approach to other logistical problems, such as pickup and delivery problems or even hub location problems. Finally, vehicle routing problems with time windows or uncertain demands could be investigated. However, these versions of the VRP are more challenging than the considered CVRP with uncertain travel times: as the uncertainty could now affect the feasibility of the solutions, these problems would require an adaptation of the min-max-min robust approach and, in particular, of Algorithm 1.
Figure 3: The optimal solution of Problem (M^3) for instance P-n16-k8 with a budgeted uncertainty set.
References


