Oracle-Based Algorithms for Binary Two-Stage Robust Optimization*

Nicolas Kämmerling † Jannis Kurtz‡

Abstract

In this work we study binary two-stage robust optimization problems with objective uncertainty. The concept of two-stage robustness is tailored for problems under uncertainty which have two different kinds of decision variables, first-stage decisions which have to be made here-and-now and second-stage decisions which can be determined each time after an uncertain scenario occured.

We adapt an oracle-based algorithm, originally introduced to solve binary min-max-min robust optimization problems, to efficiently calculate lower bounds for the binary two-stage robust problem by using an oracle for the underlying deterministic problem. We show that the latter lower bound can be implemented in a branch & bound procedure, where the branching is performed only over the first-stage decision variables. Additionally we show that the same procedure can be extended for non-linear binary problems which can be linearized. As an alternative solution method we apply a famous column-and-constraint generation algorithm to the binary two-stage robust problem with objective uncertainty.

We test both algorithms on benchmark instances of the uncapacitated single-allocation hub-location problem, which is classically modeled by a quadratic binary formulation and show that the branch & bound procedure outperforms the column-and-constraint generation algorithm.

Two-Stage Robust Optimization, Non-linear Binary Optimization, Single-Allocation Hub-Location Problem, Branch & Bound Algorithm

1 Introduction.

The concept of robust optimization was created to tackle optimization problems with uncertain parameters. The basic idea behind this concept is to use uncertainty sets instead of probability distributions to model uncertainty. More precisely it is assumed that all realizations of the uncertain parameters, called *scenarios*, are contained in a known uncertainty set. Instead of optimizing the expected objective value or a given risk-measure as common in the field of stochastic optimization, in the robust optimization framework we calculate solutions which are optimal in the worst case and which are feasible for all scenarios in the uncertainty set.

The concept was first introduced in [71]. Later it was studied for combinatorial optimization problems with discrete uncertainty sets in [56], for conic and ellipsoidal uncertainty in [12, 13], for semi-definite and least-square problems in [42, 41] and for budgeted uncertainty in [26, 25]. An overview about the robust optimization literature can be found in [31, 14, 2, 9].

The so called robust counterpart is known to be *NP*-hard for most of the classical combinatorial problems, although most of them can be solved in polynomial time in its deterministic version; see [56]. Furthermore it is a well-known drawback of this approach that the optimal solutions are often too conservative for practical issues [26]. To obtain better and less-conservative solutions several new ideas have been developed to improve the concept of robustness; see e.g. [56, 45, 68, 58, 1].

Inspired by the concept of two-stage stochastic programming a further extension of the classical robust approach which attained increasing attention in the last decade is the concept of *two-stage robustness*, or

*This work has been supported by the German Research Foundation (DFG) under grant no. BU 2313/2 – CL 318/14.

†Institute of Transport Logistics, TU Dortmund University, Leonhard-Euler-Straße 2, 44227 Dortmund, Germany

‡Chair for Mathematics of Information Processing, RWTH Aachen University, Pontdriesch 12-14, 52062 Aachen, Germany

sometimes called *adjustable robustness*, first introduced in [11]. The idea behind this approach is tailored for problems which have two different kinds of decision variables, first-stage decisions which have to be made *here-and-now* and second-stage decisions which can be determined after the uncertain parameters are known, sometimes called *wait-and-see* decisions. As in the classical robust framework it is assumed that all uncertain scenarios are contained in a known uncertainty set and the worst-case objective value is optimized. The main difference to the classical approach is that the second-stage decisions do not have to be made in advance but can be chosen as the best reaction to a scenario after it occured. This approach can be modeled by min-max-min problems in general. Famous applications occur in the field of network design problems where in the first stage a capacity on an edge must be bought such that, after the real costs on each edge are known, a minimum cost flow is sent from a source to a sink which can only use the bought capacities [21]. An overview about recent results for two-stage robustness can be found in [76]. Several concepts closely related to the two-stage robust concept were introduced in [58, 1, 29].

In this work we study binary two-stage robust optimization problems. We consider underlying deterministic problems of the form

$$\min_{(x,y)\in Z} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix} \tag{CP}$$

where $Z \subseteq \{0,1\}^{n_1+n_2}$ contains all incidence vectors of the feasible solutions, $A \in \mathbb{R}^{(n_1+n_2)\times m}$ and $c \in \mathbb{R}^m$. The variables x are called *first-stage solutions* and the variables y are called *second-stage solutions*. We assume that the vector c and hence the cost vector Ac is uncertain and all possible realizations c are contained in a convex uncertainty set $U \subset \mathbb{R}^m$. The binary two-stage robust problem is then defined by

$$\min_{x\in X} \max_{c\in U} \min_{y\in Y(x)} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix} \tag{2RP}$$

where $X \subset \{0,1\}^{n_1}$ is the projection of Z onto the x-variables and $Y(x) = \{y \in \{0,1\}^{n_2} \mid (x,y) \in Z\}$. Note that we do not consider uncertainty affecting the constraints of the problem which is a reasonable assumption for most of the classical combinatorial optimization problems. Problem (2RP) can be interpretated as follows: in the first stage, before knowing the precice uncertain vector c, the decisions $x \in X$ have to be made. Afterwards when the cost-vectors are known we can choose the best feasible second-stage decisions $y \in Y(x)$ for the given costs. As usual in robust optimization we measure the worst-case over all possible scenarios in U. Note that by our definition of the set $Y(x)$ and since the uncertainty only affects the objective function, for each first-stage solution $x \in X$ there always exists a feasible second-stage solution $y \in Y(x)$.

Problem (2RP) has been already studied in the literature and several exact algorithms as well as approximation algorithms have been proposed; see Section 1.1. While several of the existing methods are able to handle uncertainty in the constraints it is often assumed that a polyhedral description of the sets X and $Y(x)$ is given. Besides the latter limitation most of the methods are based on dualizations or reformulations which destroy the structure of the original problem (CP). Often the uncertainty set is even restricted to be a polyhedron. In this work we derive the first oracle-based exact algorithm which solves Problem (2RP) for any deterministic problem by iteratively using an oracle for the deterministic Problem (CP). Therefore the structure of the underlying problem is preserved and any preliminary algorithms which were derived for the underlying problem can be used in our method. Furthermore our algorithm works for most of the common convex uncertainty sets. Additionally we apply the column-and-constraint generation algorithm (CCG) presented in [77] to Problem (2RP) and compare it to our new method.

In Section 1.1 we will give an overview about the literature related to two-stage robust optimization problems. In Section 2 we derive an oracle-based branch & bound procedure to solve Problem (2RP). Furthermore we apply the results in [77] to Problem (2RP). Afterwards in Section 2.3 we show that the latter procedures even work for a wide class of non-linear binary problems. Finally in Section 3 we apply both methods to the uncapacitated singe-allocation hub-location problem and test it on classical benchmark instances from the literature.

Our main contributions:

- We adapt the oracle-based algorithm derived in [28] and show that it can be used to calculate a lower bound for Problem (2RP) for the common convex uncertainty sets by iteratively calling an oracle

which returns an optimal solution of Problem (CP) for a given scenario $c \in U$. Therefore any solution algorithm of the deterministic problem can be used to calculate this lower bound.

- We show that the latter lower bound can be implemented in a branch & bound procedure to solve Problem (2RP) exactly if the oracle of the deterministic problem can handle fixations on its variables. The branching has to be done only over the first-stage solutions.
- We show that if the dimension of the first-stage solutions is fixed then the robust two-stage problem can be solved in polynomial time given an oracle for the second-stage problem and under further assumptions.
- We apply the CCG algorithm presented in [77] to Problem (2RP) and show that calculating the upper bound can also be done by the same oracle-based algorithm as above.
- We show that by linearizing the non-linear terms in the objective function the latter branch & bound procedure even works for a wide class of non-linear binary optimization problems and remains oracle-based. The same holds for the CCG algorithm. To the best of our knowledge these are the first general methods for linear and non-linear binary optimization problems which solve the robust two-stage problem exactly for any (not necessarily linear) deterministic problem given by an oracle.
- We apply the branch & bound procedure and the CCG algorithm to the uncapacitated single-allocation hub-location problem which is classically modeled by a binary quadratic formulation. We test the algorithms on benchmark instances for hub-location problems and show that the number of iterations and the number of subproblem of both algorithms is very low on realistic instances. Furthermore we show that our branch & bound algorithm yields much better computation times.

1.1 Related Literature

Two-stage robust optimization or sometimes called adjustable robust optimization was first introduced in [11]. The authors show that the problem is NP-hard even if X and Y are given by linear uncertain constraints and all variables are real; see also [61]. In [11] the authors propose to approximate the problem by assuming that the optimal values of the wait and see variables y are affine functions of the uncertain parameters. These so called *affine decision rules* were studied in the robust context in several articles for the case of real recourse; see e.g. [5, 10, 33, 36, 50, 57, 69, 75]. Furthermore in several works special cases are derived for which a decision rules structure is known which is optimal; see [22, 51, 20]. Further non-linear decision rules are studied in [76].

Lower bounds for robust two-stage problems can be derived by considering a finite subset of scenarios in U. Then for each selected scenario c a duplication of the second-stage solution y^c is added to the problem, see [48, 35, 6]. The authors in [16] first dualize the inner minimization and maximization problem and then apply the latter finite scenario approach to the dual problem to obtain stronger lower bounds. Note that while the finite scenario approach can also be applied to the case when the second-stage solutions are integer, for the dualization approach the second-stage variables have to be relaxed to real variables. Unfortunately both lower bounds can not be used in a branch & bound scheme since for a complete fixation of the first-stage variables the bounds are not necessarily exact.

Exact methods for real recourse are based on the idea of Benders' decomposition, see [74, 23, 54, 46] or column-and-constraint generation [77, 24]. Note that for the Benders' decomposition approaches the second-stage solutions have to be real since dualizations of the second-stage problem are used. In contrast to this the CCG algorithm even works for integer recourse, see [78]. We will apply the latter method to our problem in Section 2.2.

For the case of integer recourse, i.e. the second-stage variables y are modeled as integer variables, decision rules have been applied to Problem (2RP) in [19, 18] to approximate the problem. Another approximation approach is called k-adaptability and was introduced in [15]. The idea is to calculate k second-stage solutions in the first-stage and allow to choose the best out of these solutions in the second-stage. Clearly since the

set of possible second-stage solutions is restricted compared to the original problem, this idea leads to an approximation of the problem. Solution methods and the quality of this approximation were studied in [22, 49, 72]. In [49] it is shown that the k-adaptability problem is exact if k is chosen larger than the dimension of the problem. The authors in [29, 30, 44] apply the k-adaptability concept to combinatorial problems without a second-stage to calculate a set of solutions which is worst-case optimal if for each scenario the best of these solutions can be chosen. They furthermore show that solving this problem can be done in polynomial time if an oracle for the deterministic problem exists and if the number of calculated solutions is larger or equal to the dimension of the problem. To solve the problem in the latter case they present an oracle-based algorithm which we will use in Section 2. The k-adaptability concept was also applied to the case that the uncertain parameters follow a discrete probability distibution [32].

Besides the algorithm in [78, 77] the only general exact method derived in the literature to solve two-stage robust problems with integer recourse is based on uncertainty set splitting; see [65, 17].

On the topic of non-linear two-stage robust problems there the literature is very sparse. Decision rules have been applied to problems with non-linear robust constraints in [73, 62]. The two-stage problem is studied for second order conic optimization problems in [27]. In [8, 59] the authors derive robust counterparts of uncertain nonlinear constraints. Note that all the latter results were developed for real second-stage solutions.

2 Binary Two-Stage Robustness

In this section we analyze the binary two-stage robust problem (2RP) with convex uncertainty sets U and derive general lower bounds which can be calculated by an oracle-based algorithm and which can be implemented in a branch & bound procedure. The branching will be done over the first-stage solutions.

The classical approach to derive lower bounds in a branch & bound procedure is relaxing the integrality and solving the relaxed problem. Applying this approach to the second-stage decisions of problem (2RP) is not useful, since for a given $x \in X$ and $c \in U$ an optimal solution of the relaxed second-stage problem must not be contained in $\operatorname{conv}(Y(x))$. It may be even the case that a linear description of $\operatorname{conv}(Y(x))$ is not known. Therefore, even if all first-stage variables are fixed, the lower bound obtained by relaxing the second-stage solution variables would not necessarily be exact and an optimal solution can not be guaranteed using a branch & bound scheme. In the following lemma we derive a lower bound for Problem (2RP) which is exact if all first-stage solutions are fixed.

Lemma 2.1. *If $U \subset \mathbb{R}^m$ is convex, then*

$$\min_{(x,y)\in\operatorname{conv}(Z)} \max_{c\in U} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix} \tag{LB}$$

is a lower bound for Problem (2RP).

Proof. For each $x \in X$ and $c \in U$ in the second-stage problem of Problem (2RP),

$$\min_{y\in Y(x)} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix},$$

we minimize an affine linear function over $Y(x)$. Hence we can replace $Y(x)$ by its convex hull and therefore Problem (2RP) is equivalent to

$$\min_{x\in X} \max_{c\in U} \min_{y\in\operatorname{conv}(Y(x))} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix}.$$

Using the classical min-max theorem (see e.g. [66]) we can swap the maximum and the inner minimum and obtain the problem

$$\min_{x\in X, y\in\operatorname{conv}(Y(x))} \max_{c\in U} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix}.$$

Since for each (x, y) where $x \in X$ and $y \in \operatorname{conv}(Y(x))$ we have $(x, y) \in \operatorname{conv}(Z)$ this proves the result. □

Note that Problem (LB) is a convex problem, since the maximum over linear functions is always convex. In [29] the authors analyze Problem (LB) for the case that A is the identity matrix. They prove that it can be solved in oracle-polynomial time, i.e. by a polynomial time algorithm if solving the deterministic problem (CP) is done by an oracle in constant time. Furthermore if we fix a solution $x \in X$, then the bound (LB) is exact, which we prove in the following.

Proposition 2.2. *If all first-stage variables are fixed then* (LB) *is equal to the exact objective value of the fixed first-stage solution.*

Proof. Let $\bar{x}$ be the fixed first-stage solution, then it holds

$$\{(x, y) \in \operatorname{conv}(Z) \mid x = \bar{x}\} = \{\bar{x}\} \times \operatorname{conv}(Y(\bar{x})).$$

Following the proof of Lemma 2.1 problem

$$\min_{(x,y)\in\{\bar{x}\}\times\operatorname{conv}(Y(\bar{x}))} \max_{c\in U} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix}$$

is equivalent to

$$\min_{x\in X, x=\bar{x}} \max_{c\in U} \min_{y\in Y(x)} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix} = \max_{c\in U} \min_{y\in Y(\bar{x})} (Ac)^\top \begin{pmatrix} \bar{x} \\ y \end{pmatrix}$$

which proves the result. □

The result of Proposition 2.2 indicates that the lower bound (LB) can be integrated in a branch & bound procedure.

In the following lemma we show that if the dimension of the first-stage solutions is fixed then the robust two-stage problem can be solved in polynomial time given an oracle for the optimization problem over $Y(x)$ for each $x \in X$. Assume that U is a non-empty convex set for which we have a weak maximization oracle, i.e. for a given $(x, y) \in \mathbb{Q}^{n_1+n_2}$ and rational $\varepsilon > 0$ we can compute in polynomial time a vector $\tilde{c} \in U \cap \mathbb{Q}^m$ with

$$(A\tilde{c})^\top x \geq (Ac)^\top x - \varepsilon \quad \text{for all } c \in U .$$

Moreover, we assume that U is bounded by a constant M, i.e.

$$\|c\| \leq M \quad \text{for all } c \in U .$$

Note that the latter assumptions hold for the classical uncertainty classes as polytopal uncertainty or ellipsoidal uncertainty.

Lemma 2.3. *Let* $\operatorname{conv}(Y(x))$ *be full-dimensional for each* $x \in X$, $\varepsilon \in (0, 1) \cap \mathbb{Q}$, *and* U *convex with the properties above. If* n_1 *is fixed and we have an oracle which returns an optimal solution of Problem*

$$\min_{y\in Y(x)} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix}$$

for each $x \in X$ *in constant time, then Problem* (2RP) *can be solved up to an accuracy of at most* ε *in time polynomial in* $(n_2, \log M + \log \|A\|_{op}, \log \varepsilon)$ *where* $\|\cdot\|_{op}$ *is the operator norm.*

Proof. As already shown in the proof of Lemma 2.1, we can reformulate Problem (2RP) by

$$\min_{x\in X, y\in\operatorname{conv}(Y(x))} \max_{c\in U} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix}.$$

Since n_1 is fixed, we have $|X| \leq 2^{n_1}$ and therefore to obtain an optimal solution we can calculate the objective value for each first-stage solution $x \in X$ and return the solution with the smallest one. The objective value for a given solution x can be calculated by solving problem

$$\min_{y\in\operatorname{conv}(Y(x))} \max_{c\in U} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix}.$$

As it was shown in [29] the latter problem can be solved in polynomial time if A is the identity matrix. Following the proof in [29] and using $\|Ac\| \leq \|A\|_{\text{op}}\|c\| \leq \|A\|_{\text{op}}M$ proves the result. □

For practical implementations we have to calculate the lower bound (LB). The authors in [29] present a practical algorithm, based on the idea of column-generation for the case that A is the identity matrix. Applied to the more general Problem (LB) the algorithm can be derived as follows: The algorithm starts with a subset of solutions $Z' \subset Z$, leading to problem

$$\min_{z \in \text{conv}(Z')} \max_{c \in U} (Ac)^\top z, \tag{1}$$

and then iteratively adds new solutions to Z' until optimality can be ensured. The solution which is added in each iteration is the one which has the largest impact on the optimal value. To find this solution the authors reformulate Problem (1) by changing the order of the minimum and the maximum, using the classical min-max theorem, and then applying a level set transformation. The reformulation is given by

$$\begin{aligned} \max \quad & \mu \\ s.t. \quad & (Ac)^\top z \geq \mu \quad \forall z \in Z' \\ & \mu \in \mathbb{R},\ c \in U \end{aligned} \tag{2}$$

For an optimal solution (μ^*, c^*) of the latter problem, we search for the solution $z \in Z$ which most violates the constraint $(Ac^*)^\top z \geq \mu^*$, i.e. the solution with the largest improvement on the optimal value of Problem (1). The latter task can be done by minimizing the objective function $(Ac^*)^\top z$ over all $z \in Z$, i.e. solving the deterministic problem (CP) under scenario c^* by any exact algorithm. If we can find a $z^* \in Z$ such that $(Ac^*)^\top z^* < \mu^*$, then we add z^* to Z' and repeat the procedure. If no such solution can be found, then $(Ac^*)^\top z \geq \mu^*$ holds for all $z \in Z$ and therefore μ^* is the optimal value of (LB). The procedure described above is presented in Algorithm 1.

Algorithm 1 Algorithm to calculate the lower bound (LB)

Input: Convex $U \subset \mathbb{R}^m$, $Z \subseteq \{0,1\}^{n_1+n_2}$
Output: Optimal value of Problem (LB) and a set of feasible solutions $Z' \subseteq Z$
1: Choose any $z_0 \in Z$ and set $Z' := \{z_0\}$
2: **repeat**
3: Calculate optimal solution (μ^*, c^*) of
$$\max \{\mu \mid (Ac)^\top z \geq \mu \quad \forall z \in Z',\ \mu \in \mathbb{R},\ c \in U\}$$
4: Calculate optimal solution z^* of $\min_{z \in Z} (Ac^*)^\top z$
5: Add z^* to Z'
6: **until** $(Ac^*)^\top z^* \geq \mu^*$
7: **return** μ^*, Z'

Note that the Problem in Step 3 depends on the uncertainty set U. For polyhedral or ellipsoidal uncertainty sets this is a continuous linear or quadratic problem, respectively. Both problems can be solved by the latest versions of optimization software like CPLEX [52]. Therefore the algorithm can be implemented for each deterministic problem by using any exact algorithm to solve the deterministic problem in Step 4. In [44] the authors applied the latter algorithm to the min-max-min robust capacitated vehicle routing problem and showed that on classical benchmark instances the number of iterations of Algorithm 1 is significantly less than the dimension of Z in general.

Note that besides the optimal value of Problem (2RP) the algorithm returns a set of feasible solutions $Z' \subseteq Z$ and not an optimal solution in conv (Z). By the correctness of the algorithm the optimal solution in conv (Z) must be contained in conv (Z') and could be calculated by finding the optimal convex combination of the solutions in Z' which can also be done in polynomial time using the given oracle for the deterministic

problem; see [29]. Nevertheless for our branch & bound procedure the set Z' is sufficient as we will see in Section 2.1. A practical advantage of the set Z' is that it contains a set of second-stage policies which can be used for practical applications. Instead of solving the second-stage problem each time after a scenario occured, which may be a computationally hard problem, we can choose the best of the pre-calculated second-stage policies in Z' for the actual scenario. The latter task can be done by just comparing the objective values of all solutions in Z' for the given scenario. Note that the returned set of solutions must not contain the optimal solution for each scenario. Nevertheless we will show in Section 3.1.1 that the calculated solutions perform very well in average over random scenarios in U.

2.1 Oracle-Based Branch & Bound Algorithm

Using the results of the latter section we can easily derive a classical branch & bound procedure to solve Problem (2RP). The idea is to branch over the first-stage solutions $x \in X$ and to calculate the lower bound (LB) in each subproblem of the branch & bound tree to possibly prune the actual branch of subproblems. All necessary details needed to implement a branch & bound procedure are presented in the following.

Handling Fixations In each subproblem of the branch & bound tree we have a given set of fixations for the x-variables, i.e. a set of indices $I_0 \subset [n_1]$ such that $x_i = 0$ for each $i \in I_0$ and a given set of indices $I_1 \subset [n_1] \setminus I_0$ such that $x_i = 1$ for each $i \in I_1$. All indices in $[n_1] \setminus (I_0 \cup I_1)$ are free. Therefore in each subproblem for the given fixations we have to solve the problem

$$\min_{\substack{(x,y)\in \mathrm{conv}(Z) \\ x_i=0 \ \forall \ i\in I_0 \\ x_i=1 \ \forall \ i\in I_1}} \max_{c\in U} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix} \tag{3}$$

or to decide if the latter problem is infeasible. It is easy to see that the latter problem, if it is feasible, can be solved by Algorithm 1 by including the given fixations into the set Z. Note that here the oracle for the deterministic problem must be able to handle variable-fixations. Nevertheless for most of the classical problems fixations can easily be implemented in most algorithms.

Calculating Feasible Solutions In each subproblem of the branch & bound tree we want to find a feasible solution to update the upper bound on our optimal value. We do this as follows: In each branch & bound node Algorithm 1 calculates a set Z' of feasible solutions $z \in Z$. If all of the generated solutions in Z' have the same first-stage solution x, then the optimal solution of (3) has binary first-stage variables and we obtain a feasible solution $x \in X$ which has the objective value μ^* returned by the algorithm. If the first-stage variables are not the same for all $z \in Z'$ then we can either choose an arbitrary first-stage solution given by any $z \in Z'$ or we can calculate the objective value of all first stage solutions included in Z' and choose the one with the best objective value. To this end we have to solve

$$\max_{c\in U} \min_{y\in Y(\tilde{x})} (Ac)^\top \begin{pmatrix} \tilde{x} \\ y \end{pmatrix},$$

for any first-stage solution $\tilde{x}$ given in Z'. Note that by changing the order of the minimum and the maximum, the latter problem can be transformed to the convex problem

$$\min_{y\in \mathrm{conv}(Y(\tilde{x}))} \max_{c\in U} (Ac)^\top \begin{pmatrix} \tilde{x} \\ y \end{pmatrix}$$

which again can be solved by Algorithm 1 replacing the deterministic problem in Step 4 by

$$\min_{y\in Y(\tilde{x})} (Ac^*)^\top \begin{pmatrix} \tilde{x} \\ y \end{pmatrix}.$$

If $X = \{0,1\}^{n_1}$, as it is the case for the hub-location problem (see Section 3), then calculating all objective values as above can be avoided and finding a good feasible solution can be done by a rounding procedure which we will present in Section 3.

Branching Strategy An easy branching strategy can be established as follows: for the calculated set of solutions Z' returned by Algorithm 1 we define the vector $\bar{x} \in [0,1]^{n_1}$ by

$$\bar{x}_i = \frac{1}{|Z'|} \sum_{(x,y) \in Z'} x_i$$

for all $i \in [n_1]$, i.e. the value $\bar{x}_i$ is the fraction of solutions in Z' for which $x_i = 1$ holds. We can then use any of the classical branching rules, e.g. we can decide to branch on the index i for which the value $\bar{x}_i$ is closest to 0 or 1.

2.2 Oracle-Based Column-and-Constraint Algorithm

In [77] a column-and-constraint generation method (CCG) was introduced to solve two-stage robust problems with real recourse variables. In [78] the authors show how the algorithm can be applied to two-stage robust problems with mixed-integer recourse variables. In both cases the algorithm is studied for problems with uncertain constraints. In this section we will apply the algorithm to Problem (2RP), i.e. to the special case of objective uncertainty, and show that we can again use Algorithm 1 to solve one crucial step in the CCG. In Section 2.3 we show that the same procedure can easily be applied to a wide class of non-linear deterministic problems. In the following we derive the CCG algorithm for Problem (2RP). For more details see [77, 78].

Using a level set transformation Problem (2RP) can be reformulated by

$$\begin{aligned} \min \quad & \mu \\ s.t. \quad & \mu \geq \max_{c \in U} \min_{y \in Y(x)} (Ac)^\top \begin{pmatrix} x \\ y \end{pmatrix} \\ & x \in X,\ \mu \in \mathbb{R}. \end{aligned}$$

If we choose any finite subset of scenarios $\left\{c^1, \ldots, c^l\right\} \in U$ we obtain the lower bound

$$\begin{aligned} \min \quad & \mu \\ s.t. \quad & \mu \geq \min_{y \in Y(x)} \left(Ac^i\right)^\top \begin{pmatrix} x \\ y \end{pmatrix} \quad i = 1, \ldots, l \\ & x \in X,\ \mu \in \mathbb{R}, \end{aligned}$$

which is equivalent to problem

$$\begin{aligned} \min \quad & \mu \\ s.t. \quad & \mu \geq \left(Ac^i\right)^\top \begin{pmatrix} x \\ y^i \end{pmatrix} \quad i = 1, \ldots, l \\ & x \in X,\ \mu \in \mathbb{R},\ y^i \in Y(x) \quad i = 1, \ldots, l. \end{aligned} \tag{4}$$

The algorithm in [77] now iteratively calculates an optimal solution (x^*, μ^*) of the latter problem (4), which is a lower bound for Problem (2RP), and afterwards calculates a worst-case scenario $c^{l+1} \in U$ by

$$c^{l+1} = \operatorname*{argmax}_{c \in U} \min_{y \in Y(x^*)} (Ac)^\top \begin{pmatrix} x^* \\ y \end{pmatrix}. \tag{5}$$

The optimal value of Problem (5) is the objective value of solution $x^* \in X$ and therefore an upper bound for Problem (2RP). Afterwards new variables y^{l+1} and the constraint

$$\mu \geq \left(Ac^{l+1}\right)^\top \begin{pmatrix} x \\ y^{l+1} \end{pmatrix}$$

are added to Problem (4) and we iterate the latter procedure until

$$\mu^* \geq \max_{c \in U} \min_{y \in Y(x^*)} (Ac)^\top \begin{pmatrix} x^* \\ y \end{pmatrix}.$$

Clearly a solution (x^*, μ^*) fulfilling the latter condition is optimal for Problem (2RP). Following the proof of Proposition 2.2 the worst-case scenario in (5) can be calculated by Algorithm 1. This can be done since we do not consider uncertainty in the constraints, while in the more general framework in [77] this is not possible.

The main difference of the latter procedure to our branch & bound algorithm is that in a branch & bound subproblem only a subset of first-stage variables is fixed while the rest is relaxed. Then we use Algorithm 1 to calculate a lower bound regarding the given fixations. In the CCG procedure in each iteration a first-stage solution is calculated by Problem (4) and therefore all variables are fixed when Algorithm 1 is applied to calculate the worst-case scenario. Nevertheless the number of constraints and the number of variables of Problem (4) increases iteratively, since each second-stage variable has to be duplicated in each iteration, while in the branch & bound procedure we always iterate over the same number of first-stage variables. In Section 3.1.1 we will compare both algorithms on benchmark instances of the uncapacitated single-allocation hub location problem.

2.3 Non-linear Binary Two-Stage Robustness

In this section we apply the idea of two-stage robustness to non-linear combinatorial optimization problems. More precicely we consider deterministic problems of the form

$$\min_{(x,y) \in Z} f_c\left((x,y)\right) \tag{NLCP}$$

where $Z \subseteq \{0,1\}^{n_1+n_2}$ and $f_c : Z \to \mathbb{R}$ is a given objective function which depends on the parameters $c \in \mathbb{R}^m$. Again we denote by X the projection of Z onto the first-stage variables and define $Y(x) = \{y \in \{0,1\}^{n_2} \mid (x,y) \in Z\}$. As in Section 2 we assume the parameters c to be uncertain and that a convex uncertainty set $U \subset \mathbb{R}^m$ is given which contains all possible realizations of the vector c.

We assume that there exists a linear problem

$$\min_{(u,v) \in W} (Ac)^\top \begin{pmatrix} u \\ v \end{pmatrix} \tag{LCP}$$

where $W \subseteq \{0,1\}^{n_1'+n_2'}$, $A \in \mathbb{R}^{(n_1'+n_2') \times m}$ and a bijective mapping $L : Z \to W$ such that for each $c \in U$ and each $(x,y) \in Z$ it holds

$$f_c((x,y)) = (Ac)^\top L(x,y). \tag{6}$$

As above we define $W_1 \subseteq \{0,1\}^{n_1'}$ as the projection of W onto the first-stage solutions and

$$V(u) := \left\{ v \in \{0,1\}^{n_2'} \mid (u,v) \in W \right\}$$

the set of second-stage solutions. The problem (LCP)is also called *linearization* of the non-linear Problem (NLCP).

Example 2.4. *Consider the problem*

$$\min_{x \in X} f_c(x) \tag{7}$$

where $X \subseteq \{0,1\}^n$, $c \in U \subseteq \mathbb{R}^{\binom{n}{p}}$ *for* $p \in \mathbb{N}$ *and*

$$f_c(x) = \sum_{i_1,\ldots,i_p \in [n]} c_{i_1 \ldots i_p} x_{i_1} \cdots x_{i_p}$$

is a polynomial. Using the classical linearization in [47], Problem (7) *is equivalent to the linear combinatorial problem*

$$\begin{aligned}
\min \quad & \sum_{i_1,\dots,i_p} c_{i_1\dots i_p} u_{i_1\dots i_p} \\
s.t. \quad & u_{i_1\dots i_p} \leq x_{i_j} \quad j = 1,\dots,p \;\forall\; i_1,\dots,i_p \in [n] \\
& \sum_{j=1}^{p} x_{i_j} - (p-1) \leq u_{i_1\dots i_p} \;\forall\; i_1,\dots,i_p \in [n] \\
& u \in \{0,1\}^{\binom{n}{p}}
\end{aligned}$$

Note that the bijective mapping $L : X \to u \in \{0,1\}^{\binom{n}{p}}$ *is given by*

$$L(x)_{i_1\dots i_p} = x_{i_1} \cdots x_{i_p}.$$

Using the notation above, the non-linear robust two-stage problem is defined as

$$\min_{x\in X} \max_{c\in U} \min_{y\in Y(x)} f_c\left((x,y)\right). \tag{NL2RP}$$

A practical application of the latter problem is the uncapacitated single-allocation hub-location problem with uncertain demands. Here a set of hub-locations has to be determined in the first stage before the demands of the customers are known. Afterwards the requested amount of flow has to be send from each origin, passing at most two hubs, to the destinations. This problem can be modeled as a quadratic problem with binary variables; for more details see Section 3.

Next we prove that we can use the linear transformation above to calculate a lower bound for the non-linear robust two-stage problem (NL2RP).

Lemma 2.5. *If* $U \subset \mathbb{R}^m$ *is convex, then*

$$\min_{w\in \mathrm{conv}(L(Z))} \max_{c\in U} (Ac)^\top w \tag{LBNL}$$

is a lower bound for Problem (NL2RP)*.*

Proof. Using property (6) we can rewrite Problem (NL2RP) by

$$\min_{x\in X} \max_{c\in U} \min_{y\in Y(x)} (Ac)^\top L(x,y).$$

which is equivalent to

$$\min_{u\in W_1} \max_{c\in U} \min_{v\in V(u)} (Ac)^\top \begin{pmatrix} u \\ v \end{pmatrix}.$$

Following the proof of Lemma 2.1 we can reformulate the latter problem to

$$\min_{u\in W_1, v\in \mathrm{conv}(V(u))} \max_{c\in U} (Ac)^\top \begin{pmatrix} u \\ v \end{pmatrix}.$$

Since for each (u,v) where $u \in W_1$ and $v \in \mathrm{conv}\left(V(u)\right)$ we also have $(u,v) \in \mathrm{conv}\left(L(Z)\right)$, this proves the result. □

To calculate the lower bound derived in Lemma 2.5, we adapt Algorithm 1 for the non-linear Problem (NLCP).

Theorem 2.6. *Algorithm 2 calculates the optimal value of Problem* (LBNL)*.*

Algorithm 2 Algorithm to calculate the lower bound (LBNL)

Input: Convex $U \subset \mathbb{R}^m$, $Z \subseteq \{0,1\}^{n_1+n_2}$
Output: Optimal value of Problem (LBNL) and set of feasible solutions $Z' \subseteq Z$.
1: Choose any $z_0 \in Z$ and set $Z' := \{z_0\}$
2: **repeat**
3: Calculate optimal solution (μ^*, c^*) of
$$\max \{\mu \mid (Ac)^\top L(z) \geq \mu \quad \forall z \in Z',\ \mu \in \mathbb{R},\ c \in U\}$$
4: Calculate optimal solution z^* of $\min_{z \in Z} f_{c^*}(z)$
5: Add z^* to Z'
6: **until** $(Ac^*)^\top L(z^*) \geq \mu^*$
7: **return** μ^*, Z'

Proof. Because of Property 6 it holds

$$\min_{z \in Z} (Ac)^\top L(z) = \min_{z \in Z} f_c(z)$$

in Step 4 for all $c \in U$. Since we use the linear formulation in Step 3 by the correctness of Algorithm 1 it follows that Algorithm 2 calculates the optimal value of the linearized Problem (LBNL). □

Note that for the latter results we just require the existance of a linearization with property (6) and without uncertainty appearing in the constraints. Nonetheless the procedure we use to solve Step 4 can be chosen arbitrarily and can be independent of the latter linearization. It is even possible to solve Step 4 by a linearization where the uncertain coefficients appear in the constraints (see Section 3) while such a linearization could not be used to derive the results of Lemma 2.5 since swapping the maximum and the minimum expression is not possible in this case.

Equivalent to Section 2.1 we can derive a branch & bound procedure for the non-linear robust two-stage problem (NL2RP). Note that again the oracle for the deterministic problem in Step 4 has to be able to handle fixations on the x-variables. Furthermore the CCG algorithm presented in Section 2.2 can easily be adapted to the non-linear case by using the linearization (LCP).

3 The Uncapacitated Single-Allocation Hub Location Problem with Uncertain Demands

In this section the oracle-based branch & bound algorithm is exemplarily applied to the single-allocation hub location problem which can be naturally defined as a two-stage problem. Furthermore due to its quadratic objective function it perfectly fits into the non-linear framework studied in Section 2.3.

Hub-location problems address the strategic planning of a transportation network with many sources and sinks. In many applications sending all commodities over direct connections would be too expensive in operation. Instead, some locations are considered to serve as transshipment points and are then called hubs. Thus, strongly consolidated transportation links are established. The bundling of shipments usually outweighs the additional costs of hubs and detours. Important applications of this problem arise in air freight [53], postal and parcel transport services [43], telecommunication networks [55] and public transport networks [63]. The recent surveys of [3] and [34] provide a comprehensive overview of the various variations and solution approaches of the hub location problem.

Operational network data is usually unknown in the strategic planning phase and can only be approximated beforehand. One main source of uncertainty in single-allocation hub location problems are demand fluctuations. Thus, it is important to include this uncertainty when deciding hub locations and allocations of the nodes to the hubs. Installing a hub is a long-term decision which last for many years or even for several decades. Nonetheless, the allocation to the hub nodes are mid-to-short-term decisions as they can

be changed over time. Rostami et al. [67] propose the variable allocation variant for single-allocation hub location problems under stochastic demand uncertainty. In this concept, hub locations are regarded as first-stage decisions. The allocation decisions are considered to be more flexible and can be adjusted when the uncertainty in the demand is revealed. Thus, the allocations are decided in the second-stage. The two-stage robust optimization problem for singe-allocation hub location problems with variable allocation is investigated in this chapter.

Before analyzing the robust formulation, the notation of the deterministic single allocation hub location problem (SAHLP) is introduced. We consider a directed graph $G = (N, A)$, where $N = \{1, 2, \ldots, n\}$ corresponds to the set of nodes that denote the origins, destinations, and possible hub locations, and A is a set of arcs that indicate possible direct links between the different nodes. Let $w_{ij} \geq 0$ be the amount of flow to be transported from node i to node j and d_{ij} the distance between two nodes i and j. We denote by $O_i = \sum_{j \in N} w_{ij}$ and $D_i = \sum_{j \in N} w_{ji}$ the total outgoing flow from node i and the total incoming flow to node i, respectively. For each $k \in N$, the value f_k represents the fixed set-up cost for locating a hub at node k. The cost per unit of flow for each path $i - k - m - j$ from an origin node i to a destination node j passing through hubs k and m respectively, is $\chi d_{ik} + \alpha d_{km} + \delta d_{mj}$, where χ, α, and δ are the nonnegative collection, transfer, and distribution costs respectively and d_{ik}, d_{km}, and d_{mj} are the distances between the given pairs of nodes. Typically $\alpha \leq \min\{\chi, \delta\}$ since otherwise using a hub would not be beneficial. Note that if hub nodes are fully interconnected, every path between an origin and a destination node will contain at least one and at most two hubs. The SAHLP consists of selecting a subset of nodes as hubs and assigning the remaining nodes to these hubs such that each spoke node, is assigned to exactly one hub with the objective of minimizing the overall costs of the network.

To formulate the SAHLP, we follow the first formulation of this problem introduced by O'Kelly [64]. Two types of decision variables are introduced. First, the

$$x_k = \begin{cases} 1 & \text{if node } k \text{ is a hub node} \\ 0 & \text{otherwise.} \end{cases}$$

variables indicate whether a node is used as hub in the transportation network. Second, the

$$y_{ik} = \begin{cases} 1 & \text{if node } i \text{ is allocated to a hub located at node } k \\ 0 & \text{otherwise.} \end{cases}$$

variables show how the nodes are allocated to the hub nodes. SAHLP can then be formulated as the following binary quadratic program:

$$\min \quad \sum_{k \in N} f_k x_k + \sum_{i \in N} \sum_{k \in N} d_{ik} \left(\chi O_i + \delta D_i\right) y_{ik} + \sum_{i,k,j,m \in N} \alpha \, w_{ij} d_{km} y_{ik} y_{jm} \tag{8}$$

$$\text{s.t.} \quad \sum_{k \in N} y_{ik} = 1 \qquad i \in N \tag{9}$$

$$y_{ik} \leq x_k \qquad i, k \in N \tag{10}$$

$$y_{ik} \in \{0, 1\} \qquad i, k \in N \tag{11}$$

$$x_k \in \{0, 1\} \qquad i, k \in N. \tag{12}$$

The objective is to minimize the total costs of the network which includes the costs of setting up the hubs, the costs of collection and distribution of items between the spoke nodes and the hubs, and the costs of transfer between the hubs. Constraints (9) indicate that each node i is allocated to precisely one hub (i.e. single allocation) while Constraints (10) enforce that node i is allocated to a node k only if k is selected as a hub node. The binary conditions are enforced by Constraints (11) and (12).

In order to solve SAHLP, many solution methods have been proposed in the literature. The classical approach to obtain an exact solution is to linearize the quadratic objective function. In [70] and [43] two

mixed-integer linear programming (MILP) formulations for the problem have been proposed which are based on a path and a flow representation, respectively. The path-based formulation in [70] has $O(|N|^4)$ variables and $O(|N|^3)$ constraints and its linear programming (LP) relaxation was shown to provide tight lower bounds. However, due to the large number of variables and constraints, the path-based formulation can only be solved for instances of relatively small sizes. Alternatively, the flow-based formulation of [43] uses only $O(|N|^3)$ variables and $O(|N|^2)$ constraints to linearize the problem. To formulate the flow-based SAHLP model (SAHLP-flow), new variables z_{ikm} are defined as the total amount of flow originating at node i and routed via hubs located at nodes k then m, respectively. SAHLP-flow is formulated as

$$\begin{aligned}
\min \quad & \sum_{k\in N} f_k x_k + \sum_{i\in N}\sum_{k\in N} d_{ik}\left(\chi\, O_i + \delta\, D_i\right) y_{ik} + \sum_{i\in N}\sum_{k\in N}\sum_{m\in N} \alpha\, d_{km} z_{ikm} \\
\text{s.t.} \quad & (9), (10), (11), (12) \\
& \sum_{m\in N} z_{ikm} - \sum_{m\in N} z_{imk} = O_i y_{ik} - \sum_{j\in N} w_{ij} y_{jk} \quad \forall i,k && (13) \\
& \sum_{m\in N} z_{ikm} \le O_i y_{ik} \quad \forall i,k && (14) \\
& z_{ikm} \ge 0 \quad \forall i,k,m. && (15)
\end{aligned}$$

Similar to SAHLP, the objective function minimizes the hub setup costs, the costs of collection and distribution, and the inter-hub transfer costs. Besides Constraints (9), (10), (11), (12) which are also used in SAHLP, Constraints (13) are flow balance constraints while Constraints (14) ensure that a flow is possible from spoke i to hub k only if node i is allocated to hub k [40]. Finally, Constraints (15) indicate the non-negativity restriction on the variables z.

The presented flow-based formulation is typically regarded to be the most effective linearized formulation in order to obtain exact solutions for the single-allocation hub location problem. In our computations we use this simple solution method to solve Step 4 in Algorithm 2. Basically, any other solution method for the deterministic single-allocation hub location problem could also be used for the oracle. More involved solution methods for single-allocation hub location problems take use of Lagrangian relaxation [38], Benders' decomposition [37], branch-and-price [39], and cutting plane methods based on Euclidean projection [60].

3.1 Two-Stage Robust SAHLP

The SAHLP splits up naturally in first- and second-stage problems as the decision variables in the SAHLP are subject to different planning horizons as discussed above. Therefore, the robust two-stage SAHLP can be modeled as follows:

$$\min_{x\in\{0,1\}^N} \max_{w\in U} \min_{y\in Y(x)} \sum_{k\in N} f_k x_k + \sum_{i\in N}\sum_{k\in N} d_{ik}\left(\chi\, O_i + \delta\, D_i\right) y_{ik} + \sum_{i,k,j,m\in N} \alpha\, w_{ij} d_{km} y_{ik} y_{jm}, \quad \text{(SAHLP-2RP)}$$

where

$$Y(x) = \{y \in \{0,1\}^{N\times N} : \sum_{k\in N} y_{ik} = 1, y_{ik} \le x_k \quad \forall i,k \in N\}.$$

We assume that $U \subset \mathbb{R}_+^{n^2}$ is a convex uncertainty set. Note that this classical formulation is a quadratic robust two-stage problem. To solve Problem (SAHLP-2RP) we use the branch & bound procedure described in Section 2.3. To this end lower bounds can be calculated by Algorithm 2 implementing the flow linearization SAHLP-flow in CPLEX ([52]) to solve the oracle in Step 4. The variable fixations in each node of the branch & bound tree can be added as constraints to the SAHLP-flow formulation. Note that in the SAHLP-flow formulation the uncertain parameters w_{ij} appear in the constraints. Therefore this formulation could not be used to derive the results of Lemma 2.5. Nonetheless we can use this formulation as an oracle for Step 4 in Algorithm 2.

Besides the lower bound Algorithm 2 returns a set of feasible solutions Z' where the solutions do not necessarily use the same hub locations. We define $\tilde{x}$ by

$$\tilde{x}_i = \frac{\sum_{(x,y)\in Z'} x_i}{|Z'|},$$

i.e. $\tilde{x}_i$ is the fraction of returned solutions which use a hub in location i. To obtain an upper bound in each subproblem we define a feasible first-stage solution $\bar{x} \in \{0,1\}^N$ for Problem (SAHLP-2RP) by

$$\bar{x}_i = \begin{cases} 1 & \text{if } \tilde{x}_i \geq 0.5 \\ 0 & \text{otherwise,} \end{cases}$$

i.e. if a hub is used by at least 50% of the returned solutions, then we use it in $\bar{x}$. We calculate the upper bound, i.e. the objective value of $\bar{x}$ by again running Algorithm 2 with total fixations $\bar{x}$. Furthermore we use the classical fractional branching strategy on $\tilde{x}$, i.e. in the subsequent subproblems we branch on the variable $\tilde{x}_i$ which has the closest distance to 0 or 1.

As our computations point out (see Section 3.1.1), for a fixed choice of hub variables $x \in \{0,1\}^N$ the number of second-stage solutions Z' generated by Algorithm 2 is very low in average and increases with increasing α. The reason is that under certain conditions for a given solution re-allocating a node i to a different hub than the actual one does not improve the objective value in any scenario. Therefore Algorithm 2 does not find a better second-stage solution in the next iteration if all hubs are fixed. More precicely, we can prove the following lemma.

Lemma 3.1. *Assume the distances are symmetric and fulfill the triangle inequality, i.e. $d_{ij} = d_{ji}$ and $d_{ij} \leq d_{ik} + d_{kj}$ holds for all $i,j,k \in N$. If in a given solution (x,y) of the SAHLP a node $l \in N$ is allocated to hub $h \in N$, then for all scenarios $w \in \mathbb{R}^{n^2}$ and independent of all other allocation variables y, re-allocating node l to a different hub $h' \in N$ does not improve the objective value if*

$$d_{lh'} - d_{lh} \geq \frac{\alpha}{\min\{\chi,\delta\}} d_{hh'}. \tag{16}$$

Proof. For the given solution (x,y) denote the change of the objective value by re-allocating node l from hub h to hub h' by $\Delta(l,h,h')$. Then using the objective function of the SAHLP formulation we have

$$\Delta(l,h,h') = (\chi O_l + \delta D_l)(d_{lh'} - d_{lh}) + \sum_{j,m\in N} \alpha (d_{h'm} - d_{hm}) y_{jm} (w_{lj} + w_{jl}).$$

Applying the triangle inequality we have

$$d_{h'm} - d_{hm} \geq -d_{hh'} \quad \forall m \in N$$

and since for each $j \in N$ by Constraint (9) it holds $\sum_{m\in N} y_{jm} = 1$, we can bound the change of objective value by

$$\begin{aligned} \Delta(l,h,h') &\geq (\chi O_l + \delta D_l)(d_{lh'} - d_{lh}) - d_{hh'} \sum_{j\in N} \alpha (w_{lj} + w_{jl}) \\ &= (\chi O_l + \delta D_l)(d_{lh'} - d_{lh}) - d_{hh'} \sum_{j\in N} \alpha (O_l + D_l). \end{aligned}$$

Substituting Condition (16) and using the inequality $(\chi O_l + \delta D_l) \geq \min\{\chi,\delta\}(O_l + D_l)$ we obtain

$$\Delta(l,h,h') \geq 0.$$

Note that the latter result holds for all scenarios $w \in \mathbb{R}^{n^2}$ and for all second-stage solutions y which proves the result. □

Condition (16) is always violated if $\alpha > \min\{\chi,\delta\}$, while it may be true if α is small compared to $\min\{\chi,\delta\}$. This indicates that for decreasing α re-allocating a node to a new hub in the next iteration of Algorithm 2 is less likely and the algorithm terminates after a few iterations. This effect is confirmed by our computational results in Section 3.1.1.

3.1.1 Computations.

In this section we apply the branch & bound method derived in Section 2.3 and the CCG method presented in Section 2.2 to the SAHLP. Both algorithms were implemented in C++. For the branch & bound procedure we calculate the lower and upper bounds by Algorithm 2 as discussed in the previous sections. The branching is performed as discussed in Section 3.1. For the CCG algorithm we implemented Problem (4) in CPLEX 12.8 while Problem (5) is solved by Algorithm 2. In Algorithm 2 the dual problem in Step 3 is solved by CPLEX 12.8 [52]. As deterministic oracle in Step 4 we use the flow linearization SAHLP-flow presented in Section 3 which was also implemented in CPLEX 12.8. After termination of Algorithm 2 we delete all solutions z from the calculated set Z' which have a non-zero slack in the dual problem in Step 3, i.e. for which $(Ac^*)^\top L(z) > \mu^*$ in the last iteration of Algorithm 2. By dualizing the dual problem in Step 3 it can be shown that the optimal value does not change by throwing out all calculated solutions with non-zero slack.

Generation of Random Instances We generated random instances as follows: As basis for our instances we use a selection of instances of the AP and the CAB datasets which were intensively studied in the hub location literature. The AP instances are based on the mail flows of Australia Post and were introduced in [43]. The CAB instances contain airline passenger interactions between 25 major cities in the United States of America and were first studied in [64]. Both datasets can be found in [7]. Since there is only one CAB instance available, we introduce three additional instances (cab1 to cab3) by variing the demand values as follows: For each node pair $i, j \in N$, the demand values are drawn randomly from the interval $[0.01\bar{w}_{ij}, 10\bar{w}_{ij}]$, where $\bar{w}_{ij}$ is the demand value of the original cab instance. The demands are normalized, i.e.

$$\bar{w}_{ij} = \frac{O_i}{\sum_{i,j\in N} w_{ij}}.$$

The number of locations n together with its pairwise distances d_{ij} are given by the instance data. The set-up costs for hub locations are also given by the instance data in case of the AP instances. Accordingly to [4], the set-up cost at node k are set to $15\log(O_k)$ for the CAB instances. The collection, transfer and distribution costs are set to $\chi = 3$, $\alpha = 0.75$ and $\delta = 2$ for the AP instances while for the CAB instances $\chi = 1$, $\delta = 1$ and α is varied in $\{0.2, 1\}$. For each instance and each $\Gamma \in \left\{0.02n^2, 0.1n^2\right\}$, rounded down if fractional, we generate 10 random budgeted uncertainty sets which are defined by

$$U_\Gamma = \left\{ w \in \mathbb{R}^{n^2} \mid w_{ij} = \bar{w}_{ij} + \delta_{ij}\hat{w}_{ij},\ \sum_{i,j\in N} \delta_{ij} \le \Gamma,\ \delta_{ij} \in [0,1] \right\}.$$

Here $\bar{w}$ are the flows given by the AP or CAB instances, respectively, while $\hat{w}_{ij}$ is chosen randomly in $[0, \bar{w}_{ij}]$ for each $i, j \in N$, i.e. the change in demand can be at most 100% of the given mean $\bar{w}_{ij}$.

Analysis of Results The results for the branch & bound procedure are presented in Table 1 and 2. Each row shows the average over all 10 random instances of the following values from left to right: the instance name; the number of locations n for the AP instances; the value Γ of the budgeted uncertainty set U_Γ; the value of α for the CAB instances; the total solution time t in seconds; the number of subproblems solved in the branch & bound tree; the percental difference of the lower bound calculated in the root problem to the optimal value of Problem (2RP); the average number of iterations i_{lb} of Algorithm 2 to calculate the lower bounds; the average number of iterations i_{ub} of Algorithm 2 to calculate the upper bounds; the number of solutions $|Z'|$ Algorithm 2 returned for the optimal first-stage solution x; the average percental difference Δ (over 10 random scenarios in U_Γ) between the best solution in Z' and the deterministic optimal solution in each scenario w. To be more precicely, to obtain the value Δ we generate 10 random scenarios in U_Γ by the following procedure: we first create n^2 equally distributed random numbers s_i in $[0, \Gamma]$ and define $s_0 := 0$. Assume the numbers are given in increasing order. We then define $\delta_i := s_i - s_{i-1}$. If $\delta \le \mathbf{1}$ is not true we

Inst.	n	Γ	t	# Subp.	Root Gap	i_{lb}	i_{ub}	# Sol.	Δ
10LL	10	2	0.9	2.2	2.5	2.3	1.6	1.0	0.0
10LL	10	10	1.8	3.6	4.5	2.8	1.4	1.0	0.0
20LL	20	8	3.4	1.0	0.0	2.0	1.0	1.0	0.0
20LL	20	40	10.4	2.4	9.4	2.4	1.0	1.0	0.0
25LL	25	12	10.1	1.0	0.0	2.0	2.0	1.0	0.0
25LL	25	62	11.7	1.0	0.0	2.2	2.2	1.2	0.0
40LL	40	32	150.5	1.2	26.6	2.1	1.0	1.0	0.0
40LL	40	160	223.0	1.6	5.6	2.6	1.1	1.0	0.0
50LL	50	50	530.3	1.4	10.7	2.3	1.9	1.0	0.0
50LL	50	250	1308.7	3.2	33.9	2.4	1.6	1.2	0.2
60LL	60	72	888.9	1.0	0.0	2.0	2.0	1.0	-
60LL	60	360	1001.3	1.0	0.0	2.0	2.0	1.0	-
70LL	70	98	1977.4	1.0	0.0	2.1	2.1	1.1	-
70LL	70	490	8632.2	3.2	11.6	3.0	2.1	1.0	-
75LL	75	112	5956.1	1.8	7.6	2.4	1.9	1.0	-
75LL	75	562	3349.2	1.0	0.0	2.0	2.0	1.0	-
90LL	90	162	7460.1	1.0	0.0	2.0	2.0	1.0	-
90LL	90	810	12681.1	1.6	0.0	2.1	2.0	1.0	-

Table 1: Results of the branch & bound procedure for AP instances.

Inst.	Γ	α	t	# Subp.	Root Gap	i_{lb}	i_{ub}	# Sol.	Δ
cab	12	0.2	17.0	1.4	1.4	2.2	1.1	1.0	0.0
cab	12	1.0	217.9	2.6	9.4	2.7	1.8	1.0	0.0
cab	62	0.2	21.1	1.8	3.3	2.1	1.2	1.0	0.0
cab	62	1.0	258.3	2.8	4.8	2.3	1.8	1.0	0.0
cab1	12	0.2	19.3	1.6	1.7	2.1	1.9	1.0	0.0
cab1	12	1.0	405.1	7.8	5.2	2.8	1.9	1.0	0.0
cab1	62	0.2	11.3	1.0	0.0	2.0	2.0	1.0	0.0
cab1	62	1.0	353.9	5.2	6.2	2.8	2.1	1.1	0.0
cab2	12	0.2	19.5	1.4	1.6	2.2	1.1	1.0	0.0
cab2	12	1.0	145.6	2.0	6.1	2.3	1.8	1.0	0.0
cab2	62	0.2	17.4	1.4	1.7	2.2	1.1	1.0	0.0
cab2	62	1.0	114.2	1.4	0.4	2.4	2.1	1.2	0.0
cab3	12	0.2	57.5	3.4	29.8	2.9	1.4	1.0	1.9
cab3	12	1.0	171.8	2.6	7.4	2.4	2.1	1.1	0.0
cab3	62	0.2	27.5	1.8	5.1	2.4	1.1	1.0	0.0
cab3	62	1.0	173.3	2.4	2.8	2.6	2.3	1.3	0.0

Table 2: Results of the branch & bound procedure for CAB instances. All instances have $n = 25$ locations.

start the procedure again. The random scenario is then given by w with

$$w_i = \bar{w}_i + \delta_i \hat{w}_i.$$

Here for the ease of notation we do not use the double index w_{ij}. After generating 10 random scenarios $w^1, \dots w^{10}$, in each scenario we compare the costs of the best solution in Z' to the costs of the optimal solution in the scenario, i.e.

$$\Delta_i := \frac{\min_{z \in Z'} f_{w^i}(z) - \min_{z \in Z} f_{w^i}(z)}{\min_{z \in Z'} f_{w^i}(z)}$$

and set Δ to the average of all Δ_i.

The results for the AP instances are shown in Table 1. The number of calculated subproblems in the branch & bound tree are in most cases close to 1 and seems to remain constant with increasing dimension. Nevertheless the run-time increases with the dimension and with Γ which is mainly due to the increasing run-time of Algorithm 2. Here with higher dimension the calculation time of the deterministic problem increases, while with increasing Γ the number of iterations of Algorithm 2 increases which was already observed in

[29, 44]. Another positive observation is that the root gap is very small in general, mostly 0 and never larger than 34%. The number of iterations of Algorithm 2 is larger for the calculations of the lower bound than for the upper bound, which is because not all hub variables are fixed in the former case. Nevertheless the number of iterations is very low and never larger than 2.2 for the lower bound and 1.2 for the upper bound. This leads to a very small number of policies calculated by Algorithm 2. Nevertheless the values of Δ indicate that the returned second-stage solutions are optimal in most of the scenarios, as Δ is 0 for most of the instances. Note that for larger dimensions due to the time consuming computations we did not determine the Δ values.

The computations for the CAB instances are presented in Table 2. The results look similar to the results related to the AP instances. The root gap is again very small for most of the instances and never larger than 30%. The number of subproblems in the branch & bound tree is very low, but in general higher than for the AP instances. Nevertheless it is never larger than 8% in average. In contrast to the AP instances the total run-time does not increase much with increasing Γ. Instead the run-time increases significantly with increasing α. The reason for this is the larger number of iterations performed by Algorithm 2 to calculate the lower and the upper bounds. This can be explained by the result of Lemma 3.1 since for $\alpha = 1$ the condition (16) is only true if all 3 hubs lie on the same line segment and therefore re-allocating for a given scenario may often improve the objective value. The latter effect leads to a larger number of calculated solutions for the instances with $\alpha = 1$. Comparing the calculated solutions to the optimal values on random scenarios, the percental difference Δ is again 0 for most of the instances. Nevertheless for two instances the difference can increase up to 5.8%.

Inst.	n	Γ	t	t_{lb}	t_{ub}	# Sol.	Δ
10LL	10	2	1.8	0.4	0.1	3.7	0.0
10LL	10	10	5.9	1.1	0.1	4.3	0.0
20LL	20	8	9.1	2.3	0.4	3.0	0.0
20LL	20	40	73.6	14.6	0.4	3.8	0.0
25LL	25	12	31.1	8.5	1.1	3.0	0.0
25LL	25	62	37.3	10.5	1.1	3.0	0.0
40LL	40	32	4682.0	1093.9	5.3	3.9	0.0
40LL	40	160	2660.3	720.9	5.5	3.3	0.0
50LL	50	50	8606.9	2230.8	15.2	3.6	-
50LL	50	250	57557.4	12632.5	14.1	4.1	-

Table 3: Results of the CCG algorithm for AP instances.

All results for the CCG algorithm are presented in Table 3 and 4. Each row shows the average over all 10 random instances of the following values from left to right: the instance name; the number of locations n for the AP instances; the value Γ of the budgeted uncertainty set U_Γ; the value of α for the CAB instances; the total solution time t in seconds; the average time t_{lb} in seconds to solve the lower bound Problem (4); the average time t_{ub} in seconds to solve the upper bound Problem (5); the number of solutions l calculated by Problem (4) which is equal to the number of iterations of the CCG algorithm; the average percental difference Δ (over 10 random scenarios $\tilde{w} \in U_\Gamma$) between the best of the solutions calculated in the last iteration by Problem (4) and the deterministic optimal solution in each scenario w; find a more detailed explanation above.

The results of the CCG algorithm are less convincing. We could solve AP instances up to 50 locations in reasonable time, while for the branch & bound procedure we managed to solve instances with 90 locations. Furthermore the runtime is at least three times as large as for the branch & bound method for most of the instances and even larger for growing dimension. The same effect holds for the CAB instances. Here the runtime is much higher for the instances with $\alpha = 1$. The large runtime of the CCG is mainly caused by the lower bound problem (4). The calculations of the upper bound, solved by Algorithm 2, are less time consuming, at most 6 seconds in average. The number of calculated solutions, i.e. the number of iterations, is slightly larger than for the branch & bound procedure but still very small, never larger than 5. A positiv effect is that the performance Δ of the calculated solutions on random scenarios is very close to 0 for all

Inst.	Γ	α	t	t_{lb}	t_{ub}	# Sol.	Δ
cab	12	0.2	56.6	15.7	0.9	3.1	0.0
cab	12	1	4526.9	850.7	1.1	4.0	0.0
cab	62	0.2	78.8	19.8	0.9	3.4	0.0
cab	62	1	9663.3	1798.0	1.2	4.1	0.0
cab1	12	0.2	53.1	12.8	1.0	3.2	0.0
cab1	12	1	3131.2	660.4	1.0	4.6	0.0
cab1	62	0.2	32.2	8.8	1.0	3.0	0.0
cab1	62	1	10760.1	1914.9	1.0	5.0	0.0
cab2	12	0.2	67.3	18.2	0.8	3.2	0.0
cab2	12	1	1616.5	402.0	1.0	3.6	0.0
cab2	62	0.2	63.0	16.2	0.8	3.3	0.0
cab2	62	1	1000.3	294.4	1.1	3.2	0.0
cab3	12	0.2	285.4	57.3	0.9	4.0	0.0
cab3	12	1	1382.1	366.1	1.0	3.5	0.0
cab3	62	0.2	121.4	30.9	0.8	3.5	0.0
cab3	62	1	1899.3	492.3	1.1	3.7	0.0

Table 4: Results of the CCG algorithm for CAB instances. All instances have $n = 25$ locations.

instances.

In Figure 1 we compare the runtimes in seconds of both algorithms. The results show that the runtime of the CCG method increases rapidly for more than 25 locations and is always much larger than the runtime of the branch & bound method. For the larger value of Γ the run-time of the CCG method explodes if n is larger than 40.

Analysis of Results for Hard Instances For the realistic instances calculated above the number of subproblems in the branch & bound tree, the number of iterations of the CCG as well as the number of iterations of Algorithm 2 is very low. The same effect occurs for most of the randomly generated instances we tested. To test the boundaries of our algorithm we generated further instances which are generated as the instances above with the only difference that the values $\hat{w}_{ij}$ are randomly drawn in $[0, 10w_{ij}]$, i.e. the

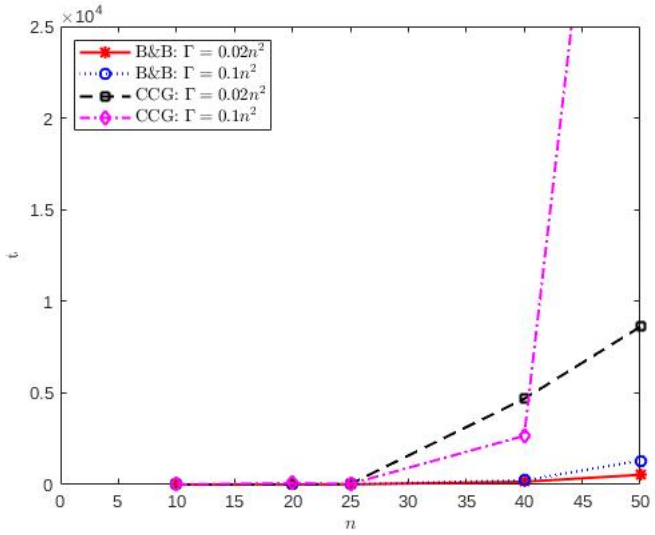

Figure 1: Development of the runtime in seconds of both algorithms for different n.

uncertainty sets are much larger. Furthermore for the AP instances we varied $\alpha \in \{0.75, 1.5\}$. The results for the branch & bound procedure are presented in Table 5. For the CCG algorithm we could not even solve instances with 25 locations in reasonable time.

Inst.	n	α	t	# Subp.	Root Gap	i_{lb}	i_{ub}	# Sol.	Δ
10LL	10	0.75	14.3	21.4	5.5	5.7	1.9	1.1	1.0
10LL	10	1.50	8.8	11.4	2.9	5.7	2.5	2.2	0.0
20LL	20	0.75	152.1	13.4	6.8	6.6	2.0	1.3	0.9
20LL	20	1.5	192.0	10	5.6	6.1	3.4	2.6	4.9
25LL	25	0.75	403.3	17.8	3.9	5.5	1.9	1.1	1.0
25LL	25	1.5	726.5	15.4	6.0	7.5	3.3	2.6	6.7
40LL	40	0.75	4095.7	13.2	4.7	6.4	2.4	1.2	10.0
40LL	40	1.50	5671.4	6.8	1.5	6.6	3.8	2.3	7.5
50LL	50	0.75	2542.0	5.4	2.4	3.3	2.2	1.1	-
50LL	50	1.50	19425.2	6.2	0.5	7.3	4.5	3.4	-
cab	25	0.2	884.8	32.8	3.2	7.4	1.8	1.4	1.0
cab	25	1	13718.3	22.0	1.1	11.1	7.4	4.8	15.7
cab1	25	0.2	865.6	34.2	5.0	7.1	2.3	1.8	0.3
cab1	25	1	15129.7	22.6	0.6	12.6	7.8	4.8	27.3
cab2	25	0.2	570.9	21.0	3.6	6.5	1.7	1.3	0.3
cab2	25	1	16103.7	18.8	1.0	11.4	7.3	6.0	15.7
cab3	25	0.2	312.5	12.8	3.0	5.9	1.5	1.0	1.2
cab3	25	1	10954.0	15.0	0.8	10.0	6.1	4.9	16.4

Table 5: Results of the branch & bound procedure for instances with large deviations and $\Gamma = 0.1n^2$.

Inst.	n	α	t	t_{lb}	t_{ub}	# Sol.	Δ
10LL	10	0.75	455.9	20.9	0.1	14.3	0.0
10LL	10	1.5	124.0	9.9	0.1	9.8	0.0
20LL	20	0.75	59150.5	3471.2	0.6	15.5	0.0
20LL	20	1.5	8445.2	880.2	0.8	8.2	-0.1

Table 6: Results of the column-and-constraint algorithm for instances with large deviations and $\Gamma = 0.1n^2$.

The results in Table 5 show that the number of subproblems in the branch & bound tree and the number of iterations of Algorithm 2 are larger than for the realistic instances above but still never get larger than 33 and 12 respectively. Both values are larger for the CAB instances. The number of subproblems decreases with increasing dimension and with increasing α. The same holds for the root gap which is lower than for the realistic instances for most of the instances. Similar to the results above the number of iterations for the calculations of the lower and the upper bounds seems to be more or less constant over the dimension. The same holds for the number of calculated second-stage solutions. The performance of these solutions over random scenarios is worse than for the realistic instances above, but still is never larger than 10% for the AP instances. For the CAB instances it is larger for $\alpha = 1$ but at most 28%. For the CCG algorithm the results are not very convincing. Even for instances with 20 locations finding an optimal solution took more than 16 hours in average for $\alpha = 0.75$. Interestingly here the instances with smaller α were harder to solve.

In Figure 2 we present the development of several problem parameters over α for the $20LL$ instance. All values are the average over 10 random uncertainty sets with random deviations $\hat{w}_{ij} \in [0, 10\bar{w}_{ij}]$. Cost parameters are defined as above by $\chi = 3$ and $\delta = 2$. Figure 2 shows that the number of subproblems in the branch & bound tree rapidly decreases with increasing α. Furthermore the number of iterations performed by Algorithm 2 to calculate the upper bounds and the number of returned policies in Z' increases until $\alpha = \min\{\chi, \delta\} = 2$ and afterwards slowly decreases. This practically verifies the theoretical result of Lemma 3.1 since for $\alpha > \min\{\chi, \delta\}$ the condition of Lemma 3.1 is always violated and re-allocating is more likely which leads to a larger number of iterations. The number of iterations performed by Algorithm 2 to calculate the lower bounds is nearly constant and slightly decreases. The root gap of the branch & bound procedure

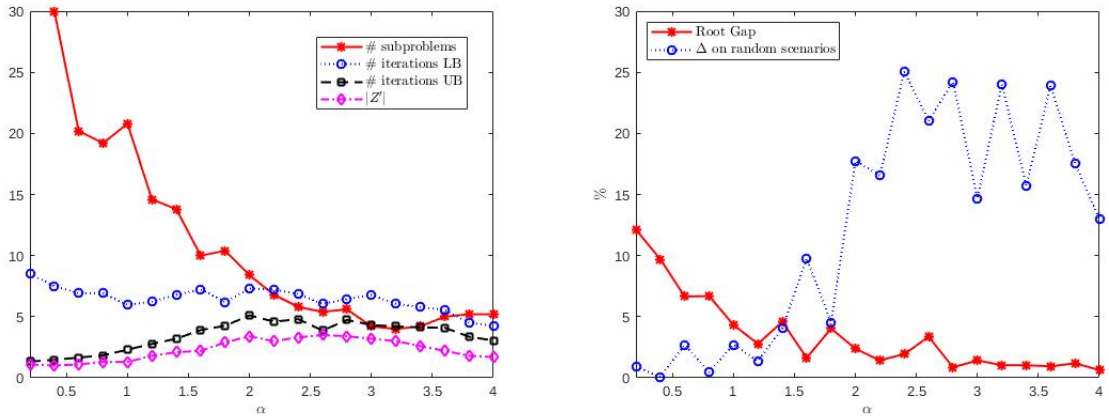

Figure 2: Development of the parameters of the branch & bound procedure over α for the $20LL$ instance with random deviations $\hat{w}_{ij} \in [0, 10\bar{w}_{ij}]$, $\chi = 3$ and $\delta = 2$.

decreases with increasing α and tends to 0. In contrast to this the performance of the returned policies in Z', indicated by Δ, seems to get worse with increasing α, and seems to be constant for $\alpha \geq \min\{\chi, \delta\}$, subtracting out the large fluctuations. The latter Δ values fluctuate around 20%.

In summary the results show that the number of subproblems of the branch & bound procedure and the number of iterations of Algorithm 2 is very low for the realistic instances of the SAHLP. Hence we could solve instances with up to 90 in less than 4 hours. Furthermore the number of calculated policies $|Z'|$ is very low for the hub location problem but they perform very well on random scenarios. For the larger uncertainty sets, the number of subproblems of the branch & bound procedure and the number of iterations of Algorithm 2 is larger but is still very low compared to the dimension of the problem. Furthermore the latter values seem to be nearly constant with increasing dimension. The runtime and the number of iterations of Algorithm 2 increases with increasing α while the number of subproblems in the branch & bound tree decreases.

An example of an optimal solution of a random instance with 20 locations and $\hat{w}_{ij}$ randomly drawn in $[0, 10\bar{w}_{ij}]$ is shown in Figure 3. The figure shows the optimal solution of the nominal scenario $\bar{w}$ and the three returned solutions in Z'. The number of hubs is larger in the two-stage robust solution than in the deterministic solution since for flexible re-allocation after a scenario occured it can be beneficial to build further hubs in advance. Furthermore the figure indicates that a hub which is used by many locations in the deterministic solution must not be used by the second-stage reactions of the two-stage solution.

4 Conclusion.

In this paper we derive a branch & bound procedure to solve linear and non-linear robust binary two-stage problems. We show that the oracle-based column generation algorithm presented in [29] can be adapted to calculate lower bounds for both, the linear and the non-linear case and that the whole procedure can be implemented for any algorithm solving the underlying deterministic problem. Furthermore we apply the famous column-and-constraint generation algorithm studied in [77] to our problem and show that again the oracle based algorithm in [29] can be used to solve one step of the procedure. We test both algorithms on classical benchmark instances of the single-allocation hub location problem which has a quadratic structure and show that the number of subproblems in the branch & bound tree, the number of iterations of the CCG algorithm as well as the number of iterations of the column generation algorithm is very low. Nevertheless our branch & bound procedure is much faster than the CCG algorithm and can solve larger instances in reasonable time. Furthermore our computational results indicate that for both algorithms the precalculated second-stage solutions perform very well on random scenarios.

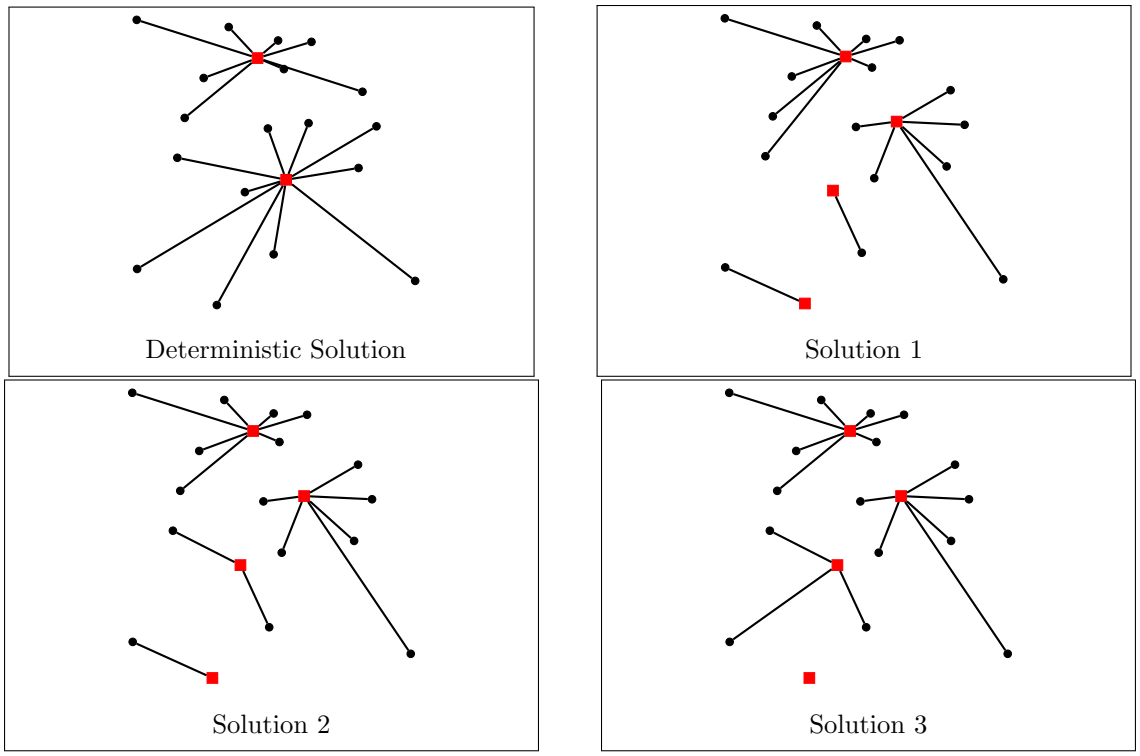

Figure 3: The optimal solution of the nominal scenario $\bar{w}$ (top left) and the optimal two-stage robust solution presented by all 3 solutions in Z' returned by Algorithm 2 in the optimal branch & bound subproblem for a $20LL$ instance with $\alpha = 1.5$ and $\Gamma = 40$.

References

[1] D. Adjiashvili, S. Stiller, and R. Zenklusen. Bulk-robust combinatorial optimization. *Mathematical Programming*, 149(1-2):361–390, 2015.

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

[3] S. A. Alumur and B. Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21, 2008.

[4] S. A. Alumur, S. Nickel, and F. Saldanha-da Gama. Hub location under uncertainty. *Transportation Research Part B: Methodological*, 46(4):529–543, 2012.

[5] A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[6] J. Ayoub and M. Poss. Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Computational Management Science*, 13(2):219–239, 2016.

[7] J. E. Beasley. OR library, 2012.

[8] A. Ben-Tal, D. Den Hertog, and J.-P. Vial. Deriving robust counterparts of nonlinear uncertain inequalities. *Mathematical programming*, 149(1-2):265–299, 2015.

[9] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[10] A. Ben-Tal, B. Golany, A. Nemirovski, and J.-P. Vial. Retailer-supplier flexible commitments contracts: A robust optimization approach. *Manufacturing & Service Operations Management*, 7(3):248–271, 2005.

[11] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[12] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[13] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

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

[15] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *Automatic Control, IEEE Transactions on*, 55(12):2751–2766, 2010.

[16] D. Bertsimas and F. J. de Ruiter. Duality in two-stage adaptive linear optimization: Faster computation and stronger bounds. *INFORMS Journal on Computing*, 28(3):500–511, 2016.

[17] D. Bertsimas and I. Dunning. Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research*, 64(4):980–998, 2016.

[18] D. Bertsimas and A. Georghiou. Binary decision rules for multistage adaptive mixed-integer optimization. *Mathematical Programming*, pages 1–39, 2014.

[19] D. Bertsimas and A. Georghiou. Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*, 63(3):610–627, 2015.

[20] D. Bertsimas and V. Goyal. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical programming*, 134(2):491–531, 2012.

[21] D. Bertsimas and V. Goyal. On the approximability of adjustable robust convex optimization under uncertainty. *Mathematical Methods of Operations Research*, 77(3):323–343, 2013.

[22] D. Bertsimas, D. A. Iancu, and P. A. Parrilo. Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394, 2010.

[23] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE transactions on power systems*, 28(1):52–63, 2013.

[24] D. Bertsimas and S. Shtern. A scalable algorithm for two-stage adaptive linear optimization. *arXiv preprint arXiv:1807.02812*, 2018.

[25] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1-3):49–71, 2003.

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

[27] O. Boni and A. Ben-Tal. Adjustable robust counterpart of conic quadratic problems. *Mathematical Methods of Operations Research*, 68(2):211, 2008.

[28] C. Buchheim and J. Kurtz. Min-max-min robustness: a new approach to combinatorial optimization under uncertainty based on multiple solutions. *Electronic Notes in Discrete Mathematics*, 52:45–52, 2016.

[29] C. Buchheim and J. Kurtz. Min–max–min robust combinatorial optimization. *Mathematical Programming*, 163(1):1–23, 2017.

[30] C. Buchheim and J. Kurtz. Complexity of min–max–min robustness for combinatorial optimization under discrete uncertainty. *Discrete Optimization*, 28:1–15, 2018.

[31] C. Buchheim and J. Kurtz. Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211–238, 2018.

[32] C. Buchheim and J. Pruente. K-adaptability in stochastic combinatorial optimization under objective uncertainty. *European Journal of Operational Research*, 2019.

[33] G. C. Calafiore. Multi-period portfolio optimization with linear control policies. *Automatica*, 44(10):2463–2473, 2008.

[34] J. F. Campbell and M. E. O'Kelly. Twenty-five years of hub location research. *Transportation Science*, 46(2):153–169, 2012.

[35] M. Campi and G. Calafiore. Decision making in an uncertain environment: the scenario-based optimization approach. *Multiple Participant Decision Making*, pages 99–111, 2004.

[36] X. Chen and Y. Zhang. Uncertain linear programs: Extended affinely adjustable robust counterparts. *Operations Research*, 57(6):1469–1482, 2009.

[37] I. Contreras, J.-F. Cordeau, and G. Laporte. Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, 59(6):1477–1490, 2011.

[38] I. Contreras, J. A. Díaz, and E. Fernández. Lagrangean relaxation for the capacitated hub location problem with single assignment. *OR spectrum*, 31(3):483–505, 2009.

[39] I. Contreras, J. A. Díaz, and E. Fernández. Branch and price for large-scale capacitated hub location problems with single assignment. *INFORMS Journal on Computing*, 23(1):41–55, 2011.

[40] I. Correia, S. Nickel, and F. Saldanha-da Gama. Single-assignment hub location problems with multiple capacity levels. *Transportation Research Part B: Methodological*, 44(8):1047–1066, 2010.

[41] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

[42] L. El Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33–52, 1998.

[43] A. T. Ernst and M. Krishnamoorthy. Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location science*, 4(3):139–154, 1996.

[44] L. Eufinger, J. Kurtz, C. Buchheim, and U. Clausen. A robust approach to the capacitated vehicle routing problem with uncertain costs. Technical report.

[45] M. Fischetti and M. Monaci. Light robustness. In *Robust and online large-scale optimization*, pages 61–84. Springer, 2009.

[46] V. Gabrel, M. Lacroix, C. Murat, and N. Remli. Robust location transportation problems under uncertain demands. *Discrete Applied Mathematics*, 164:100–111, 2014.

[47] F. Glover and E. Woolsey. Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Operations research*, 22(1):180–182, 1974.

[48] M. J. Hadjiyiannis, P. J. Goulart, and D. Kuhn. A scenario approach for estimating the suboptimality of linear decision rules in two-stage robust optimization. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7386–7391. IEEE, 2011.

[49] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[50] D. A. Iancu. *Adaptive robust optimization with applications in inventory and revenue management*. PhD thesis, Massachusetts Institute of Technology, 2010.

[51] D. A. Iancu, M. Sharma, and M. Sviridenko. Supermodularity and affine policies in dynamic robust optimization. *Operations Research*, 61(4):941–956, 2013.

[52] IBM Corporation. IBM ILOG CPLEX Optimization Studio V12.8.0, 2017.

[53] P. Jaillet, G. Song, and G. Yu. Airline network design and hub location problems. *Location science*, 4(3):195–212, 1996.

[54] R. Jiang, M. Zhang, G. Li, and Y. Guan. Benders' decomposition for the two-stage security constrained robust unit commitment problem. In *IIE Annual Conference. Proceedings*, page 1. Institute of Industrial and Systems Engineers (IISE), 2012.

[55] J. G. Klincewicz. Hub location in backbone/tributary network design: a review. *Location Science*, 6(1):307–335, 1998.

[56] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Springer, 1996.

[57] D. Kuhn, W. Wiesemann, and A. Georghiou. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130(1):177–209, 2011.

[58] C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and online large-scale optimization*, pages 1–27. Springer, 2009.

[59] A. Marandi, A. Ben-Tal, D. den Hertog, and B. Melenberg. Extending the scope of robust quadratic optimization. Technical report, 2017.

[60] J. F. Meier and U. Clausen. Solving single allocation hub location problems on euclidean data. *Transportation Science*, 52(5):1141–1155, 2017.

[61] M. Minoux. On 2-stage robust lp with rhs uncertainty: complexity results and applications. *Journal of Global Optimization*, 49(3):521–537, 2011.

[62] Z. K. Nagy and R. D. Braatz. Robust nonlinear model predictive control of batch processes. *AIChE Journal*, 49(7):1776–1786, 2003.

[63] S. Nickel, A. Schöbel, and T. Sonneborn. Hub location problems in urban traffic networks. In *Mathematical methods on optimization in transportation systems*, pages 95–107. Springer, 2001.

[64] M. E. O'Kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404, 1987.

[65] K. Postek and D. den Hertog. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3):553–574, 2016.

[66] R. T. Rockafellar. *Convex analysis*. Princeton university press, 2015.

[67] B. Rostami, N. Kämmerling, C. Buchheim, J. Naoum-Sawaya, and U. Clausen. Stochastic single-allocation hub location. 2018.

[68] A. Schöbel. Generalized light robustness and the trade-off between robustness and nominal quality. *Mathematical Methods of Operations Research*, pages 1–31, 2014.

[69] A. Shapiro. A dynamic programming approach to adjustable robust optimization. *Operations Research Letters*, 39(2):83–87, 2011.

[70] D. Skorin-Kapov, J. Skorin-Kapov, and M. O'Kelly. Tight linear programming relaxations of uncapacitated p-hub median problems. *European Journal of Operational Research*, 94(3):582–593, 1996.

[71] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

[72] A. Subramanyam, C. E. Gounaris, and W. Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *arXiv preprint arXiv:1706.07097*, 2017.

[73] A. Takeda, S. Taguchi, and R. H. Tütüncü. Adjustable robust optimization models for a nonlinear two-period system. *Journal of Optimization Theory and Applications*, 136(2):275–295, 2008.

[74] A. Thiele, T. Terry, and M. Epelman. Robust linear optimization with recourse. *Rapport technique*, pages 4–37, 2009.

[75] P. Vayanos, D. Kuhn, and B. Rustem. A constraint sampling approach for multi-stage robust optimization. *Automatica*, 48(3):459–471, 2012.

[76] İ. Yanıkoğlu, B. Gorissen, and D. den Hertog. Adjustable robust optimization–a survey and tutorial. *Available online at ResearchGate*, 2017.

[77] 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.

[78] L. Zhao and B. Zeng. An exact algorithm for two-stage robust optimization with mixed integer recourse problems. *submitted, available on Optimization-Online. org*, 2012.