A Bicriteria Approach to Robust Optimization*

André Chassein[†] and Marc Goerigk

Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany

Abstract

The classic approach in robust optimization is to optimize the solution with respect to the worst case scenario. This pessimistic approach yields solutions that perform best if the worst scenario happens, but also usually perform bad on average. A solution that optimizes the average performance on the other hand lacks in worst-case performance guarantee.

In practice it is important to find a good compromise between these two solutions. We propose to deal with this problem by considering it from a bicriteria perspective. The Pareto curve of the bicriteria problem visualizes exactly how costly it is to ensure robustness and helps to choose the solution with the best balance between expected and guaranteed performance.

Building upon a theoretical observation on the structure of Pareto solutions for problems with polyhedral feasible sets, we present a column generation approach that requires no direct solution of the computationally expensive worst-case problem. In computational experiments we demonstrate the effectivity of both the proposed algorithm, and the bicriteria perspective in general.

Keywords Robust optimization; Column Generation; Minimum Cost Flow Problem; Bicriteria Optimization; Linear Programming.

1 Introduction

Robust and stochastic optimization are paradigms for optimization under uncertainty, that have been receiving increasing attention over the last two decades. Optimization under uncertainty means that the exact parameters that describe the optimization problem are not known exactly and can only be estimated. In contrast to stochastic optimization, where one assumes to have enough knowledge to estimate the probability distribution of the input data, robust optimization deals with problems without any or only very few information about the underlying distributions. In robust optimization one defines an uncertainty set that describes all possible realizations of the input data. This can be done, for

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

†Corresponding author. Email: chassein@mathematik.uni-kl.de

example, by defining a finite set of different scenarios for the parameter values, but also continuous uncertainty sets are possible. The aim is to find a solution that is feasible for all realization of input data and yields also the best performance if the worst possible realization has occurred.

This approach follows a pessimistic point of view and, hence, it is not surprising that optimizing only the worst case performance yields in most cases a solution that performs poor in the average case, what makes the solution unpractical for many applications. Even though it is not really clear what the average realization of the data is, as there is no information about the distribution of data available, a lot of effort has been put into the development of robustness concepts that reduce the conservatism of the solution and give a better performance in the average case.

Several approaches to overcome this conservatism have been proposed. Following the ideas of Ben-Tal and Nemirovski [4] it is a matter of choosing the right uncertainty set to get a solution that performs good in the average case and in the worst case. Bertsimas and Sim [8] introduce a parameter Γ that allows to control the conservatism of a solution. Fischetti [10] proposes to identify a nominal scenario and to demand from the robust solution a performance guarantee for this scenario. For general surveys on robust optimization, we refer to [2, 3, 7, 11]. In this paper, we take a simpler and more direct approach to relax the conservatism of a solution: We propose to include the average-case performance as an objective function, thus resulting in a bicriteria problem.

A frequently used assumption is that the objective function of the optimization problem is certain, as every objective function can be represented as a constraint by using the epigraph transformation. While this is a valid method, it has the drawback that feasibility and performance guarantee are mixed into one criterion, what is questionable for most practical problems. Therefore, we focus in this paper explicitly on problems that are affected by uncertainty only in the objective function. This can be done by restricting the set of feasible solutions to solutions that are feasible for all possible parameter values. We define the bicriteria optimization problem with the two objective functions average and worst case performance, and the set of optimal solutions of this problem as the *average case–worst case curve* (*AC–WC curve*). We argue that the AC–WC curve is a valuable tool to assess the trade-off between average and worst-case performance, and should play a vital role within a robust decision making process. Note that we do not start with an uncertain multi-criteria problem, as considered in [9, 12]. Instead, we begin with an uncertain single-objective problem and extend it to a robust bicriteria problem.

To compute the AC–WC curve, we make algorithmic use of the observation that a robust solution can be interpreted as a special point on the Pareto curve with respect to a multi-criteria problem where every possible scenario outcome leads to its own objective (see [2, 13]). To the best of our knowledge, this is the first time that this observation is used.

The paper is organized as follows. In Section 2 we introduce basic definitions and notations that are used throughout the paper. In Section 3 we show a theoretical result that allows us to develop a column generation approach to compute the AC–WC curve. We evaluate different experiments in Section 4. The first experiments compares the new developed column generation approach to compute the AC–WC curve with a straightforward approach. The second experiment uses an approximation algorithm from the literature to approximate the AC–WC curve. The gains that can be obtained by using the AC–WC curve instead of only considering the average and worst case solutions are shown in the third experiment, and in the last experiment we use the AC–WC curve to directly compare the performance of two frequently used robustness concepts from the literature. For all experiments we use the minimum cost flow problem as a benchmark. We conclude the paper and point to further research questions in Section 5.

2 Notation and Definitions

Many optimization problems (P) can be solved efficiently with specialized combinatorial algorithms. Another way to solve them is to use their linear programming formulation (LP) and solve this formulation with a linear programming solver. We consider linear programs of the form

$$\begin{aligned} \min\ & c^T x \\ s.t.\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{LP}$$

where $x \in \mathbb{R}^N$ are the decision variables. The uncertainty is introduced by an uncertainty set $\mathcal{U}$. A frequently used uncertainty set is the interval uncertainty, which is given as a hyperrectangle $\mathcal{U} = \times_{i=1}^{N} [\underline{c}_i, \overline{c}_i]$. It is a standard assumption that the average scenario is given by the midpoint of the interval $\hat{c} = 0.5(\underline{c} + \overline{c})$. A second important kind of uncertainty are discrete uncertainty sets of the form $\mathcal{U} = \{c_1, \dots, c_n\}$ that specify n different cost vectors for the objective function for each scenario. Denote by $\hat{c} = \frac{1}{n} \sum_{i=1}^{n} c_i$ the average cost vector, assuming a uniform probability distribution. If any other probability distribution p is available, where p_i is the probability that scenario i is realized, and one is interested in optimizing the expected value, the cost vector $\hat{c}(p) = \sum_{i=1}^{n} p_i c_i$ can be used instead. As there is no structural difference between these two cases, we will deal in the following only with vectors $\hat{c}$ that implicitly assume that every scenario is equally likely. We assume in this chapter to have discrete uncertainty to be able to exploit the discrete structure of $\mathcal{U}$. Using this notation, the average case optimization problem (AC) has the form

$$\begin{aligned} \min\ & \hat{c}^T x \\ s.t.\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{AC}$$

Note that solving problem (AC) has the same computational complexity as solving the original problem (LP). In general, this does not hold for the worst-case (robust) optimization problem (WC) that looks as follows

$$\begin{aligned} \min\ & z \\ s.t.\ & c_k^T x \leq z & & k = 1, \dots, n \\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{WC}$$

Both problems (AC) and (WC) are tractable, as they are formulated as linear programs. However, starting from combinatorial optimization problems, problem (WC) may not yield an integral solution anymore; in particular, the robust counterpart for most combinatorial optimization problems with discrete scenarios is NP-hard [2]. Both optimization goals – the average case as well as the worst case – are important criteria to evaluate. But in most cases these two functions are contradicting. A good performance in the average case often has to be paid with worse performance in a single scenario and vice versa: good performance in the worst case objective leads to a bad performance in the average case. A common approach to deal with contradicting objective functions is to translate the problem into a multiobjective optimization problem. Applied to our situation this yields a bicriteria optimization problem $(BILP)$ with the two objective functions average and worst case performance.

$$\begin{aligned} vec - \min\ & (z, \hat{c}^T x) \\ s.t.\ & c_k^T x \leq z \qquad k = 1, \ldots, n \\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{BILP}$$

Note that in most cases there is not a single solution that optimizes problem $(BILP)$. On the contrary, it is rather common that there are many solutions that can all be seen as optimal solutions for problem $(BILP)$. A common approach to solve such a multiobjective optimization problem is to compute the set of all solutions that are Pareto efficient. A solution x is called Pareto efficient if there exists no solution y that performance in every objective function as least as good as x and is strictly better in at least one objective function. The Pareto front of an multiobjective optimization problem is obtained by mapping all Pareto efficient solutions of the problem into the multiobjective space with the corresponding objective functions. We define the *AC–WC curve* as the Pareto front of problem $(BILP)$.

3 Computing the AC–WC Curve

The set of Pareto efficient solutions of a convex bicriteria optimization problem can be computed effectively, e.g., by the SANDWICH algorithm [14]. This algorithm basically solves a sequence of one dimensional optimization problems that have as objective function a weighted sum of the two objective functions of the bicriteria problem. Hence, for the problem types under consideration, only linear programs need to be solved. We call this approach the *straightforward approach*. The drawback of the straightforward approach is that problems of the type (WC) needs to be solved and the original problem structure of problem (P) is lost. In the next chapter we show how we can overcome this drawback. We explain the theoretical principles that allow us to compute the AC–WC curve without solving problems that have the same structure as problem (WC). Instead, we only need to solve problems that maintain the structure of the original problem (P), and one additional linear master program with easily solvable structure. This has the practical advantage that specialized algorithms for problems of type (P) can be reused.

3.1 Dantzig-Wolfe Decomposition in the Objective Space

In [2] it is shown that the set of efficient solutions to the problem

$$\begin{aligned} vec - \min\ & (c_1^T x, \dots, c_n^T x) \\ s.t.\ & Ax = b \\ & x \geq 0 \end{aligned}$$

contains an optimal solution to (WC). Following a similar idea, we introduce a second multiobjective optimization problem besides problem $(BILP)$. It uses the same constraints as problem (LP) but the objective function is lifted into the multidimensional objective space. So, we get a $(n+1)-$dimensional optimization problem $(MOLP)$

$$\begin{aligned} vec - \min\ & (c_1^T x, \dots, c_n^T x, \hat{c}^T x) \\ s.t.\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{MOLP}$$

Denote by $\mathcal{P}_{MO}$ and $\mathcal{P}_{BI}$ the set of all Pareto efficient solutions of problems $(MOLP)$ and $(BILP)$, respectively. We arrive at the following first observation.

Lemma 3.1. $\mathcal{P}_{BI} \subseteq \mathcal{P}_{MO}$

Proof. Let x be an arbitrary element of $\mathcal{P}_{BI}$. Hence, for all solutions x' of problem $(BILP)$ with $\hat{c}^T x' < \hat{c}^T x$ follows that the worst case performance of x' is worse than that of x, i.e. $\exists k : c_k^T x' > c_k^T x$. We assume that $x \notin \mathcal{P}_{MO}$.

It is trivial that x is feasible for $(MOLP)$, hence, there must exist a solution x'' of $(MOLP)$ with $c_i^T x'' \leq c_i^T x$ for $i = 1, \dots, n$ and $\hat{c}^T x'' \leq \hat{c}^T x$ where at least one inequality is strict. Summing up over all scenarios we get that $\hat{c}^T x'' = \frac{1}{n} \sum_{i=1}^n c_i^T x'' < \frac{1}{n} \sum_{i=1}^n c_i^T x = \hat{c}^T x$. As x'' is feasible for $(BILP)$ it follows from the efficiency property of x that $\exists k : c_k^T x'' > c_k^T x$, which yields a contradiction. □

We use in the following the assumption that the set of feasible solutions $\mathcal{X} = \{x \in \mathbb{R}^N \mid Ax = b, x \geq 0\}$ is bounded. Therefore, there exists a finite set $\mathcal{X}' = \{x'_1, \dots, x'_s\}$ of corners of $\mathcal{X}$ such that $\mathcal{X} = conv\{\mathcal{X}'\}$, where *conv* denotes the convex hull of a set of points.

Lemma 3.2. $\mathcal{P}_{MO} \subseteq conv\{\mathcal{P}_{MO} \cap \mathcal{X}'\}$

Proof. Let x be an arbitrary element of $\mathcal{P}_{MO}$. As $x \in \mathcal{X}$ it follows that $\exists I \subseteq \{1, \dots, s\}, \lambda \in \mathbb{R}^{|I|}_{>0} : x = \sum_{j \in I} \lambda_j x'_j$ and $\sum_{j \in I} \lambda_j = 1$. Assume that $\exists k \in I : x'_k \notin \mathcal{P}_{MO}$. From this follows that it must exist an other feasible solution $\hat{x}_k$ that dominates x'_k, i.e. $c_i^T \hat{x}_k \leq c_i^T x'_k$ for $i = 1, \dots, n$ and $\hat{c}^T \hat{x}_k \leq \hat{c}^T x'_k$ where at least one inequality is strict. As $\lambda_k > 0$ it follows that $\hat{x} = \sum_{j \in I \setminus \{k\}} \lambda_j x'_j + \lambda_k \hat{x}_k$ dominates x, which is a contradiction. Therefore, the assumption was wrong and it holds that $\forall i \in I : x'_i \in \mathcal{P}_{MO}$. Hence, it follows that $\{x'_j \mid j \in I\} \subseteq \mathcal{P}_{MO} \cap \mathcal{X}'$. Therefore, $x \in conv\{\{x'_j \mid j \in I\}\} \subseteq conv\{\mathcal{P}_{MO} \cap \mathcal{X}'\}$. □

We define the set of Pareto efficient corners of the polyhedron by $\mathcal{P}'_{MO} = \mathcal{P}_{MO} \cap \mathcal{X}'$ and the linear map $C(x) = (c_1^T x\ \dots\ c_n^T x\ \hat{c}^T x)^T$. The next lemma is a direct consequence of Lemma 3.1 and Lemma 3.2.

Lemma 3.3. $C(\mathcal{P}_{BI}) \subseteq conv\{C(\mathcal{P}'_{MO})\}$

Proof. $C(\mathcal{P}_{BI}) \subset C(\mathcal{P}_{MO}) \subset C(conv(\mathcal{P}'_{MO})) = conv(C(\mathcal{P}'_{MO}))$.
The first and second step follows from Lemma 3.1 and Lemma 3.2 and the last step uses the linearity of map C. □

Next consider the parametric problem $(\lambda - ACWC)$ $(\lambda \in [0,1])$ that tries to optimizes the average and worst case performance simultaneously.

$$\begin{aligned} &\min \lambda \hat{c}^T x + (1-\lambda) z \\ &s.t.\ c_k^T x \leq z \qquad\qquad k = 1,\dots,n \qquad\qquad (\lambda - ACWC) \\ &\quad Ax = b \\ &\quad x \geq 0 \end{aligned}$$

Observation 3.4. *The $AC - WC$ curve can be computed by solving a sequence of $(\lambda - ACWC)$ problems, where $\lambda \in [0,1]$.*

To see that Observation 3.4 holds we refer to [14]. The authors of this paper deal with the approximate computation of the Pareto front of bicriteria optimization problems. They define the algorithms SANDWICH and $\epsilon-$SANDWICH that can be used to compute the complete Pareto front of a bicriteria problem respectively only a set that approximates this Pareto front. As the exact description of SANDWICH and $\epsilon-$SANDWICH are rather long we refer to [14] for more information. The only important observation for our theoretical result is that the application of these algorithms in our context leads to a sequence of $(\lambda - ACWC)$ problems.

The main idea is to translate problem $(\lambda - ACWC)$ into the objective space. We use variables a and z, where a represents the average case and z the worst case performance.

$$\begin{aligned} &\min \lambda a + (1-\lambda) z \\ &s.t.\ y_{n+1} = a \\ &\quad y_i \leq z \qquad\qquad i = 1,\dots,n \\ &\quad y \in C(\mathcal{P}_{BI}) \end{aligned}$$

Note that we do not change the optimal solution of the problem if we replace $C(\mathcal{P}_{BI})$ by the set $conv\{C(\mathcal{P}'_{MO})\}$, as $C(\mathcal{P}_{BI}) \subset conv\{C(\mathcal{P}'_{MO})\}$ due to Lemma 3.3 and optimal solutions must be contained in $C(\mathcal{P}_{BI})$.

$$\begin{aligned} &\min \lambda a + (1-\lambda) z \\ &s.t.\ y_{n+1} = a \\ &\quad y_i \leq z \qquad\qquad i = 1,\dots,n \\ &\quad y \in conv\{C(\mathcal{P}'_{MO})\} \end{aligned}$$

Assume that we would know the set $C(\mathcal{P}'_{MO}) = \{y^1, \dots, y^r\}$ completely then we can reformulate the problem in the following way.

$$
\begin{aligned}
&\min \lambda a + (1-\lambda) z \\
&s.t.\ \hat{y} = \sum_{i=1}^{r} \alpha_i y^i \\
&\qquad \sum_{i=1}^{r} \alpha_i = 1 \\
&\qquad \hat{y}_{n+1} = a \\
&\qquad \hat{y}_i \leq z && i = 1, \dots, n \\
&\qquad \alpha \geq 0
\end{aligned}
$$

Introducing slack variables and removing the superfluous variable $\hat{y}$ we get the parameterized master program $(\lambda - M)$.

$$
\begin{aligned}
&\min \lambda a + (1-\lambda) z \\
&s.t. \begin{pmatrix}
-1 & 0 & 1 & & y_1^1 & & y_1^r \\
\vdots & \vdots & & \ddots & \vdots & \cdots & \vdots \\
-1 & 0 & & 1 & y_n^1 & & y_n^r \\
0 & -1 & 0 & & y_{n+1}^1 & \cdots & y_{n+1}^r \\
0 & 0 & 0 & & 1 & \cdots & 1
\end{pmatrix}
\begin{pmatrix} z \\ a \\ s_1 \\ \vdots \\ s_n \\ \alpha_1 \\ \vdots \\ \alpha_r \end{pmatrix}
= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad (\lambda - M) \\
&\qquad (z, a, \alpha, s) \geq 0
\end{aligned}
$$

Note that the linear program $(\lambda - M)$ might contain exponential many variables but we can still solve it to optimality by using column generation. In column generation one does not solve $(\lambda - M)$ directly instead one solves a restricted problem that contains only a subset of all columns and adds columns if they have negative reduced costs. If no columns with negative reduce cost exist anymore the problem is solved to optimality. For a successful application of the column generation idea it is necessary that one can find columns with negative reduced costs very fast. It is easy to see that the problem of finding the column with the smallest reduced cost value is an original problem of type (P). This result combined with Theorem 3.4 gives the main theorem of the section.

Theorem 3.5. *The $AC-WC$ curve can be computed by solving linear programs of the form $(\lambda - M)$ and instances of the original problem (P).*

3.2 Algorithmic Details

In this section we discuss the benefits and the drawbacks of solving $(\lambda - M)$ using column generation instead of $(\lambda - ACWC)$ directly as linear programs. The most obvious benefit is the fact that the original problem structure (P) remains preserved. Hence, the column generation approach is especially useful if there exists specialized algorithms for problem (P) that allow to solve (P) faster than it could be solved by linear programming solvers (e.g., one can use the network simplex algorithm to solve uncertain network flow problems). But note that it is necessary to solve several problems of type (P) to solve a single $(\lambda - M)$ problem – for every column we generate in the master program, we need to solve one subproblem. To solve $(\lambda - ACWC)$ directly we have only to solve a single linear program.

The drawback that the column generation approach needs to generate probably a lot of columns to solve problem $(\lambda - M)$ can be compensated by the fact that columns that are generated to solve $(\lambda - M)$ can be reused in the master program to solve another problem $(\tilde{\lambda} - M)$. This makes the column generation approach valuable for computing the AC–WC curve, as in the SANDWICH algorithm a probably long sequence of these problems $(\lambda - M)$ needs to be solved for different values of λ.

In the case where computation time is limited and, therefore, one cannot compute the complete AC–WC curve, another benefit from the column generation approach can be obtained. If the SANDWICH algorithm is stopped prematurely, an incomplete set of Pareto efficient solutions is provided by the algorithm. Using the column generation approach one can still try to generate some additional Pareto efficient solutions by solving the remaining $(\lambda - M)$ problems heuristically using only the columns that were generated so far. Note that the heuristic solution of $(\lambda - M)$ needs almost no additional computation time.

The performance of the column generation method depends heavily on the number of scenarios. Whereas one additional scenario increases the dimension of the multidimensional projection that is used in the column generation framework by one, the additional scenario can be represented by only one additional inequality in the linear programming formulation of $(\lambda - ACWC)$. Hence, one can expect the column generation approach to perform relatively well for a small number of scenarios and relatively badly for a large number of scenarios.

4 Experiments

4.1 Setup and Instances

As a benchmark for the following experiments we use the minimum cost flow problem [1] on randomly generated graphs. An instance of the minimum cost flow problem is described by the quadruple (G, c, u, b), where $G = (V, A)$ is a directed graph, c_a is the cost of sending one unit of flow over arc $a \in A$, u_a is the capacity of flow that can be sent over arc $a \in A$, and b_v is the supply/demand of node $v \in V$. The goal is to find a flow with minimal costs that fulfills the capacity constraints on all arcs as well as the supply and the demand constraints of every node. This well-known problem can be stated as the following linear program

$$\begin{aligned} \min & \sum_{a \in A} c_a x_a & \\ s.t. & \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = b_v & \forall\, v \in V \\ & 0 \leq x_a \leq u_a & \forall\, a \in A \end{aligned}$$

The instances for the minimum cost flow problems are generated randomly. We consider two types of instances: One with discrete uncertainty, and one with interval-based uncertainty. The instances R with discrete cost scenarios are described by six parameters R-N-p-n-c_{max}-u_{max}-b_{max}. The number of nodes is given by N. We create an arc between two arbitrary nodes with probability p. The number of different scenarios is given by n. The costs $c_j(a)$ of an arc a in scenario j are generated by the formula

$$c_j(a) = \begin{cases} Y \cdot c_{max} \cdot \left(\frac{j}{n}\right)^2 & \text{, if } D \leq \frac{1}{2} \\ Y \cdot c_{max} \cdot \left(\frac{n-j}{n}\right)^2 & \text{, else} \end{cases} \quad \forall a \in A,\ j = 1, \ldots, n$$

where Y and D are independent and uniformly distributed in $[0, 1]$. We use this formula to generate a cost structure with varying spreads, which increases the size of the Pareto front. The capacity u_a of an arc a is set to u_{max} for all arcs. The supply/demand b_v of a node v is chosen uniformly from the interval $[-b_{max}, b_{max}]$, except for the supply/demand $b_{v_N} = -\sum_{v \neq v_N} b_v$ of the last node v_N. Note that the choice of b_{v_N} is necessary to get a feasible minimum cost flow instance.

The instances I with interval uncertainty for the arc costs are also described by six parameters I-N-p-var-c_{max}-u_{max}-b_{max}. The parameters N, p, u_{max}, and b_{max} are used in the same way as for R instances to generate the arcs and their capacities as well as the nodes and their corresponding demands/supplies. The average costs $\hat{c}(a)$ of an arc a are chosen uniformly at random from the interval $[0, c_{max}]$. Next, the worst case cost $\overline{c}(a)$ of an arc a are chosen uniformly at random from the interval $[\hat{c}(a), \hat{c}(a)(1 + var)]$.

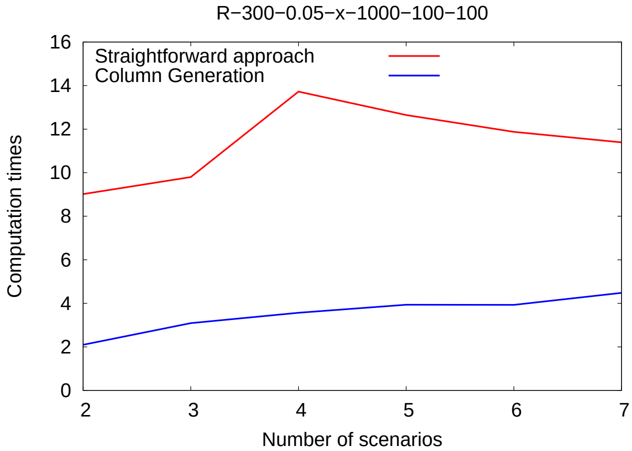
All experiments were conducted on an Intel Core i5-3470 processor, running at 3.20 GHz, 8 GB RAM under Windows 7. The linear and quadratic programs are solved with CPLEX version 12.6. The minimum cost flow problems are solved with the network simplex algorithm of the LEMON graph library version 1.3.1. Algorithms were implemented with Microsoft Visual C++ 2010 Express.

4.2 Computation of the AC–WC curve

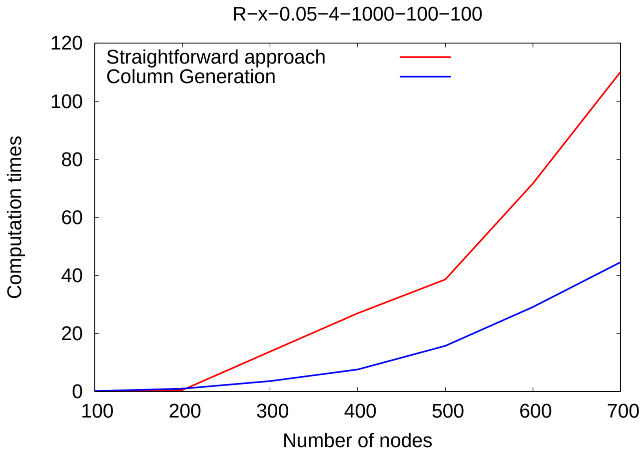
4.2.1 Exact Computation

In this section we want to compare the straightforward implementation of the SANDWICH algorithm to compute the AC–WC curve with the version we have introduced in this paper that is based on column generation. We compare the times needed for both algorithms to compute the complete AC–WC curve.

In the first experiment we fix the graph type and vary the number of scenarios and in the second experiment we fix the number of scenarios and vary the number of nodes in the graph. The values are averaged over 10 runs per instance.



(a) All graphs are of the type R-300-0.05-x-1000-100-100. The x-axis shows the different number of scenarios and the y-axis the time (in seconds) needed to compute the complete AC–WC curve.



(b) All graphs are of the type R-x-0.05-4-1000-100-100. The x-axis shows the different number of nodes and the y-axis the time (in seconds) needed to compute the complete AC–WC curve.

Figure 1: Comparison of computation times.

Discussion. The results of the first experiment, shown in Figure 1(a), confirm that the column generation approach outperforms the straightforward approach if the number of scenarios is small. Additional experiments suggest that if the number of scenarios exceeds a certain amount, the straightforward approach may become the better alternative. It is interesting to note that the computation time decreases for the straightforward approach if the number of scenarios increases. This is explained by the observation that an increasing amount of scenarios can lead to a decreasing number of Pareto efficient solutions.

The second experiment (Figure 1(b)) confirms that the column generation approach outperforms the straightforward approach if the instance size increases, assumed that the number of scenarios is small.

4.2.2 Approximate Computation

As the number of Pareto efficient points is not polynomially bounded in the input size, the exact computation of the AC–WC curve might be an unreasonable approach. This problem can be resolved in practice by computing not all Pareto efficient points but only a subset of them.

Denote by P the set of all points of the AC–WC curve. We call a set $S \subseteq P$ an $\epsilon-$AC–WC curve if $\forall\ p \in P\ \exists s \in S : s_{ac} \leq (1+\epsilon)p_{ac}$ and $s_{wc} \leq (1+\epsilon)p_{wc}$.

For any practical application one should be able to choose an appropriate ϵ such that the $\epsilon-$AC–WC curve is sufficient. Note that there exists efficient algorithms to compute the $\epsilon-$AC–WC curve. We use the $\epsilon-$SANDWICH algorithm [14] that is very similar to the SANDWICH algorithm. The only difference is the choice of the weights used for the weighted sum computations, roughly speaking. We used the column generation method to solve the weighted sum problems that have to be solved during the execution of the $\epsilon-$SANDWICH algorithm.

We did three different experiments. In the first experiment we bound the running time of the $\epsilon-$SANDWICH algorithm by a multiple of the time needed to compute the average and worst case solution. In the second experiment we bound the number of iterations used in the $\epsilon-$SANDWICH algorithm. In the last experiment we stopped the generation of new columns after 5 steps of the algorithm and solved all remaining weighted sum problems only approximately by solving the restricted master program. Note that the set consisting only of the average and worst case solution is also an $\epsilon-$AC–WC curve for an ϵ that is sufficiently large. The values are averaged over 10 runs per instance.

Approximation guarantee ϵ	0 %	50 %	100 %	500 %	1000 %
R-300-0.05-4-1000-100-100	1.16e-01	3.50e-02	8.78e-03	1.16e-04	1.83e-05
R-400-0.05-4-1000-100-100	1.28e-01	3.83e-02	7.68e-03	2.32e-04	3.74e-05
R-500-0.05-4-1000-100-100	1.31e-01	4.41e-02	1.18e-02	3.30e-04	6.82e-05
R-1000-0.05-10-1000-100-100	5.38e-02	2.22e-02	5.57e-03	1.12e-04	1.46e-05
R-1000-0.05-20-1000-100-100	1.18e-02	3.32e-03	1.54e-03	2.27e-05	5.32e-06
R-1000-0.05-40-1000-100-100	9.87e-03	7.82e-03	4.07e-03	2.96e-04	8.66e-05

Table 1: The numbers in the columns give the approximation guarantee ϵ of the $\epsilon-$AC–WC curve that can be obtained in x% of the time needed to compute the average and worst case solution.

Approximation guarantee ϵ	0	1	5	10	100
R-300-0.05-4-1000-100-100	1.16e-01	5.12e-02	2.87e-03	8.07e-04	6.16e-06
R-400-0.05-4-1000-100-100	1.28e-01	5.48e-02	3.27e-03	8.50e-04	7.50e-06
R-500-0.05-4-1000-100-100	1.31e-01	4.98e-02	3.24e-03	7.02e-04	8.43e-06
R-1000-0.05-10-1000-100-100	5.38e-02	2.61e-02	1.40e-03	3.93e-04	3.51e-06
R-1000-0.05-20-1000-100-100	1.18e-02	6.47e-03	3.20e-04	8.20e-05	1.02e-07
R-1000-0.05-40-1000-100-100	9.87e-03	4.30e-03	3.07e-04	8.97e-05	7.48e-07

Table 2: The numbers in the columns give the approximation guarantee ϵ of the $\epsilon-$AC–WC curve that can be obtained within x steps of the $\epsilon-$SANDWICH algorithm.

Approximation guarantee ϵ	After 5 steps	Improved guarantee
R-300-0.05-4-1000-100-100	2.87e-03	1.88e-03
R-400-0.05-4-1000-100-100	3.72e-03	2.49e-03
R-500-0.05-4-1000-100-100	3.24e-03	2.31e-03
R-1000-0.05-10-1000-100-100	1.40e-03	7.26e-04
R-1000-0.05-20-1000-100-100	3.20e-04	2.25e-04
R-1000-0.05-40-1000-100-100	3.07e-04	2.39e-04

Table 3: The numbers in the first column give the approximation guarantee ϵ of the $\epsilon-$AC–WC curve that can be obtained within 5 steps of the $\epsilon-$SANDWICH algorithm. The second column gives the improved guarantee after additional solutions are computed using the already generated columns of the master program.

Discussion. The second column in Table 1 and Table 2 describe how well the AC–WC curve is approximated by the set that consists only of the average and worst case solution. We observe that an increasing number of scenarios improves the approximation guarantee of the average and worst case solution. This emphasizes the observation that the size of the AC–WC curve decreases with an increasing number of scenarios. We observe in general that already relatively few steps of the $\epsilon-$ SANDWICH algorithm suffice to get a $\epsilon-$AC–WC curve with good approximation guarantee. Note especially the values for ϵ in Table 1. They indicate that the time needed to compute a sufficiently good $\epsilon-$AC–WC curve is in the same order as the time needed to compute the average and worst case solution. In a practical setting this means that whenever one is able to compute the average and the worst case solution of a problem, one should also be able to compute an $\epsilon-$AC–WC curve that provides on the one hand clear insight in the different trade-offs between average and worst case objective and on the other hand a rich set of alternative solutions where the decision maker can choose from.

The last experiment (Table 3) shows a benefit of using the column generation method for generating a $\epsilon-$AC–WC curve. The already generated columns can be used to compute additional solutions that can improve the approximation guarantee considerably. Note that the computational effort to compute these additional solutions is almost for free as no further instances of the original problem (P) needs to be solved.

4.3 Properties of the AC–WC curve

In the first part of this section we show how the AC–WC curve can be used to find compromise solutions that are reasonable for the average and the worst case objective. In the second part we compare two common robustness concepts and measure how they perform in comparison with the AC–WC curve.

4.3.1 Gains by using the AC–WC curve

In this section we compare the trade-offs between the average case and the worst case solution. We use the AC–WC curve to answer the questions: Allowing the worst case value to increase, by how much can we decrease the average case value? And vice versa: Allowing the average case value to increase, by how much decreases the worst case value? The numbers in the tables are averaged over 100 runs per instance.

Higher WC performance	+0.1 %	+1.0 %	+5.0 %	+10 %
R-200-0.05-2-1000-100-100	99.80	98.06	91.34	84.94
R-200-0.05-3-1000-100-100	99.88	98.86	95.26	92.48
R-200-0.05-4-1000-100-100	99.90	99.08	96.43	94.42
R-200-0.05-5-1000-100-100	99.92	99.32	97.58	96.34
R-200-0.05-6-1000-100-100	99.92	99.35	97.99	97.17
R-300-0.05-2-1000-100-100	99.77	97.72	89.81	82.37
R-300-0.05-3-1000-100-100	99.83	98.39	93.25	89.24
R-300-0.05-4-1000-100-100	99.88	98.89	95.62	93.21
R-300-0.05-5-1000-100-100	99.90	99.06	96.46	94.46
R-300-0.05-6-1000-100-100	99.92	99.26	97.32	95.87

Table 4: The numbers in the columns give the average case performance of a solution that has a $x\%$ higher worst case performance than the worst case solution. The average case performance of the worst case solution is normalized to 100.

Higher AC performance	+0.1 %	+1.0 %	+5.0 %	+10 %
R-200-0.05-2-1000-100-100	98.63	95.46	89.99	86.06
R-200-0.05-3-1000-100-100	96.93	91.70	85.15	81.75
R-200-0.05-4-1000-100-100	96.62	89.65	79.52	75.46
R-200-0.05-5-1000-100-100	95.96	88.58	80.92	79.95
R-200-0.05-6-1000-100-100	95.79	89.01	83.60	83.32
R-300-0.05-2-1000-100-100	98.72	96.12	91.44	88.06
R-300-0.05-3-1000-100-100	97.08	92.86	87.40	84.03
R-300-0.05-4-1000-100-100	96.42	89.20	77.74	71.96
R-300-0.05-5-1000-100-100	95.81	88.83	79.64	76.00
R-300-0.05-6-1000-100-100	96.04	88.54	79.63	77.86

Table 5: The numbers in the columns give the worst case performance of a solution that has a $x\%$ higher average case performance than the average case solution. The worst case performance of the average case solution is normalized to 100.

Discussion. The most obvious observation by comparing Tables 4 and 5 is the fact that almost all numbers of Table 4 are bigger than the corresponding numbers of Table 5. This yields to the following statement. Increasing the worst case performance of the average case solution is cheaper than increasing the average case performance of the worst case solution.

Small increases in the average case performance of the average case solution guarantee in most cases already considerably decreases in the worst case performance. But also an increase in the worst case performance of $x\%$ can lead to average case reductions that are above $x\%$. Note that an increasing number of scenarios reduces this positive effect.

4.3.2 Comparison with Other Robustness Concepts

In contrast to the other experiments where we assumed to have a discrete uncertainty set, we consider in this section interval uncertainty. This kind of uncertainty is typical if the data is obtained, e.g., by measurements. Also more general versions of uncertainty sets are studied in the literature, e.g. polyhedral or ellipsoidal uncertainty sets [5]. The other concepts are important if the values of the parameters are correlated and, therefore, a hyperrectangle is not the correct form for the uncertainty set.

The problem with interval uncertainty or continuous uncertainty sets in general is often that protecting against the complete set might give an over-conservative solution that performs bad in the average case. Therefore, researchers have tried to find robustness concepts that improve the average case performance of the solution and still give a rather good protection against most realizations of the uncertainty set.

Note that most robustness concepts normally only deal with uncertainty in the constraints and not in the objective function, as it is claimed that the objective function can be represented by a constraint.

The goal of this section is to compare two different robustness approaches that define a parameter that helps the modeler to control the conservatism of the solution with the AC–WC curve. We analyze if the transformation of an uncertain objective to an uncertain constraint has undesired side effects. As benchmark problem we use the minimum cost flow problem on randomly generated graphs.

$\Gamma-$Robustness. The concept of $\Gamma-$robustness was introduced by Bertsimas and Sim [8]. They motivate their concept by the observation that the situation where all parameters take their worst case value simultaneously should be rather unlikely. Hence, they propose to use a parameter Γ that defines an upper bound for the number of parameters that can deviate from their nominal value.

The application of the Γ-robustness concept to the minimum cost flow problem leads to the following linear program

$$\begin{aligned}
\min \sum_{a \in A} & \hat{c}_a x_a + p_a + \Gamma z \\
s.t. \sum_{a \in \delta^+(v)} & x_a - \sum_{a \in \delta^-(v)} x_a = b(v) && \forall\ v \in V \\
& (\overline{c}_a - \hat{c}_a) x_a \leq p_a + z && \forall\ a \in A \\
& 0 \leq x_a \leq u_a && \forall\ a \in A \\
& 0 \leq p_a && \forall\ a \in A \\
& 0 \leq z
\end{aligned}$$

Controlling parameter Γ allows the modeler to control the conservatism of the solution. If Γ is chosen to be 0, z can be chosen arbitrary large in an optimal solution and, therefore, $p_a = 0\ \forall a \in A$. In this case the objective function is equal to the average case objective function $\hat{c}^T x$. If, on the other hand, Γ is chosen to be as large as the number of uncertain parameters, $z = 0$ in an optimal solution and $p_a = (\overline{c}_a - \hat{c}_a) x_a\ \forall a \in A$. Hence, the objective function is equal to the worst case objective function $\overline{c}^T x$.

$\Omega-$**Robustness.** The second concept was proposed by Ben-Tal and Nemirovski [6]. We call it $\Omega-$robustness as the problems are parameterized by Ω. The main idea of the concept is to include chance constraints to deal with possible constraint violations. For a feasible solution of the Ω–robust problem might exist some parameter realization such that the solution is infeasible for the original problem. However, the probability that such a parameter realization happens should be small under the assumption that the parameters are distributed symmetrically and independently. Before one can solve the robust counterpart one has to specifies a value κ that defines a bound for the probability of constraint violation. Ben-Tal and Nemirovski showed how the probability constraint can be relaxed by a second order cone constraint. The value κ is then replaced by $\Omega = \sqrt{-2ln(\kappa)}$. The application of the Ω-robustness concept to the minimum cost flow problem leads to the following quadratically constraint program

$$
\begin{aligned}
\min \sum_{a\in A} & \overline{c}_a x_a - (\overline{c}_a - \hat{c}_a) z_a + \Omega q & \\
s.t. \sum_{a\in\delta^+(v)} & x_a - \sum_{a\in\delta^-(v)} x_a = b(v) & \forall\, v\in V \\
& \sum_{a\in A} (\overline{c}_a - \hat{c}_a)^2 z_a^2 \le q^2 & \\
& 0 \le z_a \le x_a \le u_a & \forall\, a \in A \\
& 0 \le q &
\end{aligned}
$$

Ignoring the relation between Ω and κ one can interpret Ω as a parameter that can be used to control the conservatism of the solution. If Ω is set to 0, $z_a = x_a\ \forall a \in A$ in an optimal solution and, hence, the objective function is equal to the average case objective function $\hat{c}^T x$. On the other hand for large enough values for Ω, $z_a = 0\ \forall a \in A$ in an optimal solution and, therefore, the objective function is equal to the worst case objective function $\overline{c}^T x$.

Hence, we see that the parameter Ω plays a similar role for $\Omega-$robustness as Γ for $\Gamma-$robustness.

Experiment description For every instance we generate 5 graphs and compute for each the AC–WC curve. We define the $\Gamma\backslash\Omega-$curves as all points that are generated by the $\Gamma\backslash\Omega-$robustness concept if $\Gamma\backslash\Omega$ is integral. To compute the $\Gamma\backslash\Omega-$curves we set $\Gamma\backslash\Omega = 0$ and compute the solution of the $\Gamma\backslash\Omega-$robustness concept. Next, we increase both parameters by 1 and resolve the problem. We repeat this process until an optimal solution of the worst case problem is found. For every solution found so far we compute the average and worst case performance and compare it to the AC–WC curve. As the AC–WC curve is a Pareto front of minimization problems it is clear that all points of the $\Gamma\backslash\Omega-$curves must lie on or above the AC–WC curve.

To measure the distance between the AC–WC curve and the $\Gamma\backslash\Omega-$curves we use two different measures. First we choose the solution of the AC–WC curve that minimizes the sum of average and worst case performance. We call this solution in the following the $ac - wc$ solution. Next, we select the points from the $\Gamma\backslash\Omega-$curves that have the same average case performance as the $ac - wc$ solution and compare the worst case performance of these points with the worst case performance of the $ac - wc$ solution. We repeat this step and compare the average case performances if we require equal worst case performance. The values obtained by this computation indicate how close the $\Gamma \backslash \Omega-$curves is to the center of the AC–WC curve.

As second measure we use a more pessimistic measurement to represent the maximum distance between the $\Gamma \backslash \Omega-$curves and the AC–WC curve. Every point of the $\Gamma \backslash \Omega-$curves could become a point of the AC–WC curve if it would be scaled by a factor of $\frac{1}{(1+\epsilon)}$ for large enough ϵ. For every point of the $\Gamma \backslash \Omega-$curves we compute the smallest ϵ such that the scaled version of the point belongs to the AC–WC curve. To measure the furthest distance between the curves we take the maximum over all such computed ϵ's.

	Γ		Ω	
Increased WC\AC	Same AC	Same WC	Same AC	Same WC
I-100-0.05-1-1000-100-100	102.2	102.2	101.2	102.1
I-200-0.05-1-1000-100-100	103.5	103.0	101.9	102.8
I-300-0.05-1-1000-100-100	102.7	102.7	101.5	102.6
I-400-0.05-1-1000-100-100	103.0	102.7	101.8	102.7
I-500-0.05-1-1000-100-100	103.2	103.1	101.6	103.1

Table 6: Comparison of the $ac-wc$ solution with the $\Gamma \backslash \Omega-$curves. The numbers in the columns with label *Same AC* give the worst case value of the solution of the $\Gamma \backslash \Omega-$curve that has the same average case performance as the $ac - wc$ solution. The number in the columns with label *Same WC* are defined vice versa. The corresponding performance of the $ac - wc$ solution is normalized to 100.

Maximum ϵ used to scale	Γ	Ω
I-100-0.05-1-1000-100-100	0.029	0.012
I-200-0.05-1-1000-100-100	0.037	0.018
I-300-0.05-1-1000-100-100	0.032	0.014
I-400-0.05-1-1000-100-100	0.034	0.016
I-500-0.05-1-1000-100-100	0.037	0.017

Table 7: Scaling a point of the $\Gamma \backslash \Omega-$ curves by $\frac{1}{(1+\epsilon)}$ may generate a point of the AC–WC curve. The number in the columns give the smallest ϵ such that every scaled point of the corresponding $\Gamma \backslash \Omega-$ curves lies below or on the AC–WC curve.

Discussion. Assume that instead of computing the AC–WC curve one would rely on $\Gamma \backslash \Omega-$robustness. If one is asked to provide a solution that has a certain average case or worst case performance one would provide in most cases a solution that is strictly dominated by a point of the AC–WC curve. The resulting loss is given in Table 6 where the $ac-wc$ solution defines the demanded average or worst case performance. We see that if a fixed worst case value is required both robustness concepts perform almost equally. But if a certain average case value is demanded the $\Gamma-$robustness concept provides a solution that is worse in the worst case as the solution provided by the $\Omega-$robustness concept.

The second measure, shown in Table 7, indicates how far the $\Gamma\backslash\Omega-$curves can deviate from the AC–WC curve. It is obvious that the $\Omega-$curve sticks rather close to the AC–WC curve compared to the $\Gamma-$curve.

We conjecture that the relatively bad performance of the $\Gamma-$robustness concept in this experiment is due to the fact that the underlying minimum cost flow problem tends to generate solutions that are rather dense and the $\Gamma-$robustness on the contrary is more effective for problems with sparse solutions.

We plot the $\Gamma\backslash\Omega-$curves and the AC–WC curve for the instances I-400-0.05-1.0-1000-100-100 and I-500-0.05-1.0-1000-100-100 in Figures 2 and 3 to visualize the last statements. Note that both robustness concepts generate solutions that are strictly dominated by the worst case solution.

Furthermore, the computation time of the $\Omega-$curve is considerably longer than for the $\Gamma-$curve which is in turn also considerably longer than for the AC–WC curve. The presented figures are both highly surprising and representative for other instances. They underline the value for the AC–WC perspective in the decision process for a practitioner.

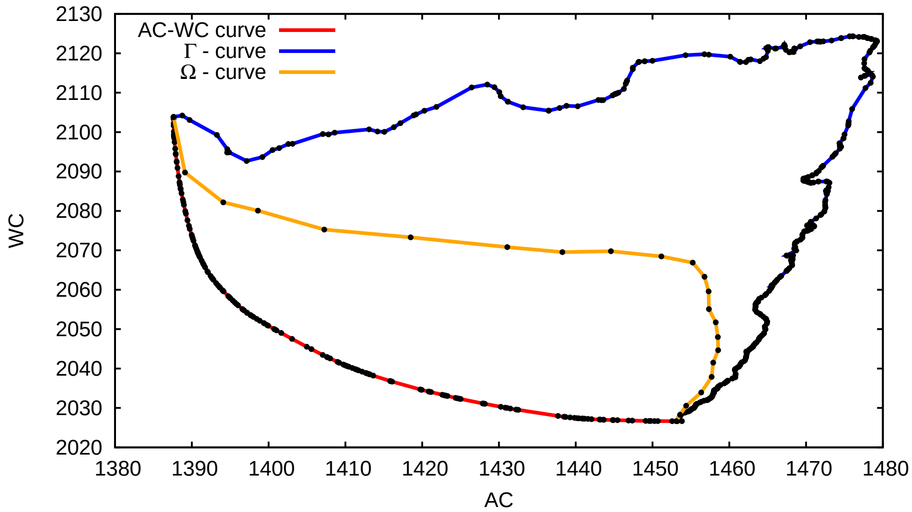


Figure 2: The AC–WC curve and the $\Gamma\backslash\Omega-$ curves for an graph of type I-400-0.05-1.0-1000-100-100. The axis are scaled by 1000.

5 Conclusions and Future Research

In this paper we considered optimization problems with convex feasibility sets and uncertainty in the objective function. To deal with the uncertainty we propose to consider the problem as a bicriteria optimization problem with the two objectives average and worst case performance. We call the Pareto front of the resulting optimization problem the AC–WC curve. We develop a column generation approach that can be combined with existing methods from the literature to compute the AC–WC curve efficiently. The experiments are used to emphasize our statements.

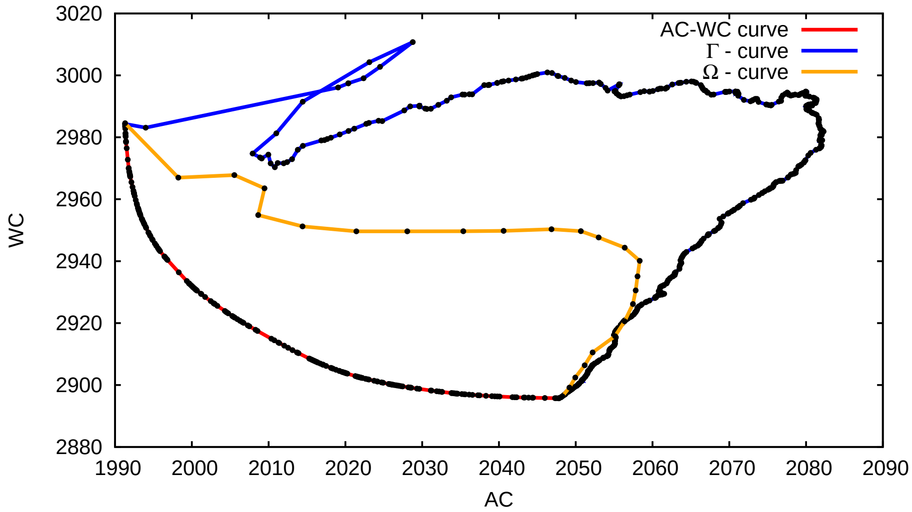


Figure 3: The AC–WC curve and the $\Gamma \backslash \Omega-$ curves for an graph of type I-500-0.05-1.0-1000-100-100. The axis are scaled by 1000.

Firstly, the new developed algorithm is effective, especially for a small number of scenarios. Secondly, the time needed to compute an approximate version of the AC–WC curve, that should suffice completely in practice, is of the same magnitude as the time needed to compute the average and the worst case solution. Thirdly, the AC–WC curve provides valuable information for a decision maker and allows him to choose the solution that gives the best trade-off between average and worst case performance in his situation.

In the last experiment we recall two famous robustness concepts from literature and use the AC–WC curve to compare them. We observe that the approach suggested by Ben-Tal and Nemirovski is closer to the AC–WC curve than the approach suggested by Bertsimas and Sim. But both approaches reveal a considerable gap to the AC–WC curve.

For future research it is interesting to investigate the AC–WC curve of more complex problems that have discrete feasibility sets. Another interesting question is whether the observed gap between common robustness concepts and the AC–WC curve can be re-observed for problems that have a certain objective function but uncertain constraints.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

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

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

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

[6] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

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

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

[9] M. Ehrgott, J. Ide, and A. Schöbel. Minmax robustness for multi-objective optimization problems. *European Journal of Operational Research*, 239(1):17 – 31, 2014.

[10] M. Fischetti and M. Monaci. Light robustness. In *Robust and online large-scale optimization*, volume 5868 of *Lecture Note on Computer Science*, pages 61–84. Springer, 2009.

[11] M. Goerigk and A. Schöbel. Algorithm engineering in robust optimization. Technical report, Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Universität Göttingen, 2013. Submitted.

[12] J. Ide and E. Köbis. Concepts of efficiency for uncertain multi-objective optimization problems based on set order relations. *Mathematical Methods of Operations Research*, 80(1):99–127, 2014.

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

[14] G. Ruhe and B. Fruhwirth. ϵ-Optimality for bicriteria programs and its application to minimum cost flows. *Computing*, 44(1):21–34, 1990.