Minmax regret solutions for minimax optimization problems with uncertainty

Igor Averbakh *

Mathematics Department, MS-9063, Western Washington University, Bellingham, Washington 98225, USA

Received 1 June 1999; received in revised form 1 January 2000

Abstract

We propose a general approach for finding minmax regret solutions for a class of combinatorial optimization problems with an objective function of minimax type and uncertain objective function coefficients. The approach is based on reducing a problem with uncertainty to a number of problems without uncertainty. The method is illustrated on bottleneck combinatorial optimization problems, minimax multifacility location problems and maximum weighted tardiness scheduling problems with uncertainty. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: Optimization under uncertainty; Polynomial algorithms; Minmax regret optimization

1. Introduction

Combinatorial optimization problems with uncertainty in input data have attracted significant research efforts because of their importance for practice. Two ways of modeling uncertainty are usually used: the stochastic approach and the worst-case approach.

In the stochastic approach, uncertainty is modelled by means of assuming some probability distribution over the space of all possible scenarios (where a scenario is a specific realization of all parameters of the problem), and the objective is to find a solution with good probabilistic performance. Models of this type are handled using stochastic programming techniques [15].

In the worst-case approach, the set of possible scenarios is described deterministically, and the objective is to find a solution that performs reasonably well for all scenarios, i.e. that has the best “worst-case” performance and hedges against the most hostile scenario. The performance of a feasible solution under a specific scenario may be either the objective function value, or the absolute (or relative) deviation of the objective function value from the best possible one under this scenario (“regret”, or “opportunity loss”). Solutions that optimize the worst-case performance are called in [18] robust solutions; we will use this terminology, although there are also other robustness concepts in the literature.

Uncertainty in parameters of a model can have different origins in different application areas. It may be due to either incomplete information, or fluctuations inherent for the system, or unpredictable changes in
the future; when quantitative parameters have a subjective nature (e.g. priorities of customers in location models or priorities of jobs in scheduling models), it may be convenient for a decision maker to specify intervals instead of specific values for parameters and adjust this input interactively, to get more intuition about the model. We refer the reader to [18] for a discussion of sources of uncertainty in optimization models.

Minmax regret optimization models have received increasing attention over the last decade, and by now it is a well-established area of research. A comprehensive treatment of the state of art in minmax regret discrete optimization by 1997 and extensive references can be found in the book [18]. We also refer the reader to the book [18] for a comprehensive discussion of the motivation for the minmax regret approach and its advantages; in the paper, we discuss only technical issues.

To put minmax regret optimization in a broader context, we note that concepts similar to minmax regret have been used in areas such as competitive analysis of online algorithms, machine learning, and information theory. In the area of competitive analysis of online algorithms, problems are considered in which information arrives one piece at a time, and decisions must be made based only on input received so far without knowing what will occur in the future. In these problems one typically looks at the “competitive ratio” of an algorithm, which is the worst-case ratio of the cost of the solution obtained by the algorithm to the cost of the optimal solution (in hindsight) given the sequence of events that occurred. For a recent textbook on competitive analysis of online algorithms, see [5]. In the area of machine learning, one often attempts to bound the difference between the loss of a learning algorithm and the loss of the best solution in hindsight from some set of candidates determined in advance (see, e.g., [7]). In information theory, one uses minmax regret and regret ratios to study notions such as universal prediction, universal portfolios, etc. (see, e.g., [11]).

We focus on finding minmax regret solutions for a class of combinatorial optimization problems with a minimax type of objective function and interval structure of uncertainty (i.e. each parameter can take on any value from the corresponding interval of uncertainty). The class of problems that we consider is quite general, it includes many combinatorial optimization problems with minimax objective. The main idea of our approach is to reduce a problem with uncertainty to a number of problems without uncertainty; then, an efficient algorithm for the problem without uncertainty can be used to obtain an efficient algorithm for the problem with uncertainty. The approach is illustrated on such examples as bottleneck combinatorial optimization problems, minimax multi-facility location problems, maximum weighted tardiness scheduling problems with uncertainty. Low-order polynomial complexity bounds are given for the minmax regret versions of the bottleneck spanning tree problem, the maximum capacity path problem, bottleneck Steiner subnetwork problems with k-connectivity constraints ($k \leq 3$).

The paper is organized as follows. In Section 2, we study the problems in the most general form. In Section 3, we specialize the approach to bottleneck combinatorial optimization problems. In Sections 4 and 5, we discuss how to apply the approach for minmax multifacility location problems and maximum weighted tardiness scheduling problems, respectively.

2. Notation and general framework

Let $A$ be a set (solution space); elements of $A$ will be called feasible solutions. Suppose that $m$ functions $f_i(X), i \in [1 : m]$ are defined on $A$; we assume that $f_i(X) \geq 0$ for any $X \in A$ and $i \in [1 : m]$. A vector $S = \{w_i^S, i \in [1 : m]\}$ of $m$ positive real numbers will be called a scenario and will represent an assignment of weights $w_i^S$ to functions $f_i(X)$. When the choice of a scenario is clear from the context or is irrelevant, we drop the superscript $S$ and denote the corresponding weight just $w_i$. For a scenario $S$ and a feasible solution $X \in A$, let us define $F(S, X) = \max_{i \in [1 : m]} \{w_i^S f_i(X)\}$. Consider the following generic minmax optimization problem:

**Problem OPT(S).** Minimize $\{F(S, X) | X \in A\}$.

Let $F^*(S)$ denote the optimal objective function value for Problem OPT($S$) (we assume that $A$ is either a finite set or a compact metric space, and in the latter case functions $f_i(X)$ are assumed to be continuous, so the optimum exists).
Most combinatorial optimization problems with minimax objective functions can be represented as special cases of Problem OPT(S). In this paper, we consider in detail the following four examples: bottleneck combinatorial optimization problems, the distance-constrained multicenter location problem with mutual communication, the weighted p-center location problem, the maximum weighted tardiness scheduling problem.

Everywhere in the paper, for notational convenience, for any ratio \( a/b \) of nonnegative numbers we assume that \( a/b = +\infty \) when \( b = 0 \) and \( a \neq 0 \), and \( a/b = 0 \) when \( a = 0 \) and \( b = 0 \).

For an \( X \in A \), the value \( F(S,X) - F^*(S) \) is called the absolute regret for the feasible solution \( X \) under scenario \( S \), and the value \( [F(S,X) - F^*(S)]/[F^*(S)] \) is called the relative regret for \( X \) under scenario \( S \).

Suppose that a set \( SS \) of possible scenarios is fixed. For any \( X_1, X_2 \in A \), let us define values \( \text{REGR}1(X_1, X_2) \) and \( \text{REGR}2(X_1, X_2) \) as

\[
\text{REGR}1(X_1, X_2) = \max_{S \in SS} (F(S,X_1) - F(S,X_2)), \tag{1}
\]

\[
\text{REGR}2(X_1, X_2) = \max_{S \in SS} \frac{F(S,X_1) - F(S,X_2)}{F(S,X_2)}. \tag{2}
\]

For an \( X \in A \), let us define the worst-case absolute regret \( Z_1(X) \) and the worst-case relative regret \( Z_2(X) \) as follows:

\[
Z_1(X) = \max_{S \in SS} \{F(S,X) - F^*(S)\}, \tag{3}
\]

\[
Z_2(X) = \max_{S \in SS} \left\{ \frac{F(S,X) - F^*(S)}{F^*(S)} \right\}. \tag{4}
\]

Alternative ways to represent \( Z_1(X) \) and \( Z_2(X) \) are

\[
Z_1(X) = \max_{S \in SS, X' \in A} \{F(S,X) - F(S,X')\}, \tag{5}
\]

\[
Z_2(X) = \max_{S \in SS, X' \in A} \left\{ \frac{F(S,X) - F(S,X')}{F(S,X')} \right\}, \tag{6}
\]

\[
Z_1(X) = \max_{X' \in A} \text{REGR}1(X, X'), \tag{7}
\]

\[
Z_2(X) = \max_{X' \in A} \text{REGR}2(X, X'). \tag{8}
\]

An optimal solution to the right-hand side of (7) (of (8)) is called an absolute (relative) worst-case alternative for \( X \). Also, we define the worst-case objective function value for \( X \) as

\[
Z_0(X) = \max_{S \in SS} F(S,X). \tag{9}
\]

The following problems are considered in the paper:

**Problem ROB-0.** Find \( X \in A \) that minimizes \( Z_0(X) \).

**Problem ROB-1.** Find \( X \in A \) that minimizes \( Z_1(X) \).

**Problem ROB-2.** Find \( X \in A \) that minimizes \( Z_2(X) \).

Following [18], we call Problems ROB-0, ROB-1, and ROB-2 robust versions of Problem OPT(S).

We say that two optimization problems are equivalent if their sets of optimal solutions and their optimal objective function values are equal.

We assume that the set of scenarios \( SS \) has the following structure: for each weight \( w_i \), a lower bound \( w_i^- \) and an upper bound \( w_i^+ \) are given, \( w_i^- \geq w_i^+ > 0 \), and the weight \( w_i \) can take on any value in the corresponding interval of uncertainty \( [w_i^-, w_i^+] \). Therefore, \( SS \) is the Cartesian product of the corresponding intervals of uncertainty \( [w_i^-, w_i^+] \), \( i \in [1 : m] \).

**Observation 1.** To solve Problem ROB-0, it is sufficient to solve Problem OPT\( (S^+) \), where \( S^+ \) is the scenario assigning all weights \( w_i \) equal to the corresponding upper bounds \( w_i^+ \).

According to Observation 1, Problem ROB-0 can be solved straightforwardly with the same order of complexity as Problem OPT(S). In the remainder of the paper, we focus on Problems ROB-1 and ROB-2 ("minmax regret" problems).

Notice that Problem OPT(S) is a special case of Problems ROB-1 and ROB-2 (corresponding to set \( SS \) consisting of a single scenario). To get a better intuition about the minmax regret problems, the following interpretation is useful. For an \( \varepsilon > 0 \) and a scenario \( S \), an \( X \in A \) is called an \( \varepsilon \)-optimal solution to Problem OPT(S) if \( F(S,X) - F^*(S) \leq \varepsilon \). Let \( X_{\varepsilon}(S) \) denote the set of all \( \varepsilon \)-optimal solutions to Problem OPT(S). It is reasonable to look for a solution that is \( \varepsilon \)-optimal (for a given \( \varepsilon > 0 \) for all possible scenarios, that is, to look for an \( X \in \bigcap_{S \in SS} X_{\varepsilon}(S) \). For some values of \( \varepsilon \) such a solution exists, but for some (smaller) values of \( \varepsilon \) such a solution may not exist, because solutions good for one scenario may be bad for some other
scenarios. Then, the solution $X^*$ obtained by solving Problem ROB-1 will be $\varepsilon$-optimal for all scenarios $S \in S^*$ for any $\varepsilon \geq Z_1(X^*)$; also, for any $\varepsilon < Z_1(X^*)$ we have $\bigcap_{S \in S^*} X_i(S) = \emptyset$, and for any $\varepsilon \geq Z_1(X^*)$ we have $\bigcap_{S \in S^*} X_i(S) \neq \emptyset$. So, value $Z_1(X^*)$ has the interpretation of the minimum possible $\varepsilon$ such that there exists a solution $\varepsilon$-optimal for Problem $\text{OPT}(S)$ for all scenarios $S \in S^*$; this value can be used as a measure of uncertainty. Problem ROB-2 has a similar interpretation if we change the definition of $\varepsilon$-optimality and call an $X \in A$ an $\varepsilon$-optimal solution to Problem $\text{OPT}(S)$ if $[F(S, X) - F^*(S)]/[F^*(S)] \leq \varepsilon$.

Let $S_i$ denote the scenario where $w_j = w_i^+$ and $w_j = w_i^-$ for all $j \neq i$, i.e. scenario $S_i$ assigns weight $w_i$ equal to the corresponding upper bound and all other weights equal to the corresponding lower bounds. For simplicity of presentation of results for Problem ROB-2, we assume $F^*(S_i) > 0$ for all $i \in [1 : m]$ (this assumption can be ignored in the context of Problem ROB-1).

For any $X \in A$, let

$$h_1^i(X) = \max \{w_i^+ f_i(X) - F^*(S_i), 0\},$$

$$h_2^i(X) = \frac{w_i^+ f_i(X) - F^*(S_i)}{F^*(S_i)}.$$  \hspace{1cm} (10)

Consider the following problems:

**Problem B1.** Minimize $\{\max_{i \in [1 : m]} h_1^i(X) \mid X \in A\}$.

**Problem B2.** Minimize $\{\max_{i \in [1 : m]} h_2^i(X) \mid X \in A\}$.

**Theorem 1.**

(a) For any $X \in A$, $Z_i(X) = \max_{i \in [1 : m]} h_1^i(X)$;

(b) For any $X \in A$, $Z_2(X) = \max_{i \in [1 : m]} h_2^i(X)$.

**Proof.** See the appendix. \hfill $\Box$

**Corollary 1.** Problems B1 and B2 are equivalent to Problems ROB-1 and ROB-2, respectively.

**Corollary 2.** For any $X \in A$, there is an absolute (relative) worst-case scenario for $X$ that belongs to $S^* = \{S_i, i \in [1 : m]\}$. Therefore, the sets of optimal solutions for Problems ROB-1 and ROB-2 will not change if we replace $S^*$ with $S^{SS^*}$.

**Proof.** See the proof of Theorem 1 in the appendix. \hfill $\Box$

Let $\xi_i = w_i^+ / F^*(S_i)$, $i \in [1 : m]$. Let $\hat{S}$ be the scenario that assigns weights $\xi_i$ to functions $f_i(X)$, i.e. $\hat{S} = \{\xi_i, i \in [1 : m]\}$.

**Corollary 3.** The set of optimal solutions to Problem $\text{OPT}(\hat{S})$ is equal to the set of optimal solutions to Problem ROB-2, i.e. to solve Problem ROB-2 it is sufficient to solve Problem $\text{OPT}(\hat{S})$.

**Proof.** It is sufficient to notice that $h_2^i(X) = \xi_i f_i(X) - 1$, and to use Theorem 1(b). \hfill $\Box$

According to Corollary 3, it is possible to solve Problem ROB-2 with complexity $O(m \cdot \text{complexity of Problem } \text{OPT}(\hat{S}))$, by solving $m$ problems $\text{OPT}(S_i)$, $i \in [1 : m]$ to obtain values $F^*(S_i)$, $i \in [1 : m]$ and Problem $\text{OPT}(\hat{S})$. As we will see later, in many cases it is possible to avoid solving all problems $\text{OPT}(S_i)$, $i \in [1 : m]$, using the structure of a specific combinatorial problem at hand.

Corollary 3 and Observation 1 show that it is possible to obtain efficient algorithms for Problems ROB-0 and ROB-2 immediately from an efficient algorithm for Problem $\text{OPT}(S)$. This is not necessarily the case for Problem ROB-1, since, in general, Problem B1 may not have the same structure as Problem $\text{OPT}(S)$. However, we will see that quite often Problem B1 can be reformulated in terms of Problem $\text{OPT}(S)$, or an algorithm for Problem $\text{OPT}(S)$ can be modified to handle Problem B1 using the structure of the specific combinatorial optimization problem under consideration.

### 3. Robust bottleneck combinatorial optimization problems

Let $E = \{e_1, \ldots, e_m\}$ be a finite set (called “ground set”), and $P \subseteq 2^E$ be a set of feasible subsets of $E$. Let us define functions $f_i(X)$, $i \in [1 : m]$ on $A$ as follows: for an $X \in A$,

$$f_i(X) = \begin{cases} 0 & \text{if } e_i \not\subseteq X, \\ 1 & \text{if } e_i \subseteq X. \end{cases}$$  \hspace{1cm} (12)

For any scenario $S = (w_1^S, \ldots, w_m^S)$, Problem $\text{OPT}(S)$ in this setting will be referred to as
Problem BOTTLE(S). Problems ROB-0, ROB-1 and ROB-2 in this setting will be referred to as Problems ROBBOT-0, ROBBOT-1 and ROBBOT-2, respectively. Problem BOTTLE(S) is the generic bottleneck combinatorial optimization problem and has the following interpretation: elements $e_i$ of the ground set $E$ have weights $w_i^S$, and it is required to find a feasible subset that minimizes the weight of its heaviest element. For example, $E$ can be the set of edges of a network; then if $A$ is the set of paths from a specified source node to a specified sink node, Problem BOTTLE(S) is the bottleneck path problem [25]; if $A$ is the set of spanning trees, Problem BOTTLE(S) is the bottleneck spanning tree problem [6].

Consider Problem ROBBOT-1. According to Corollary 1 from Theorem 1, this problem can be replaced with Problem B1. Notice that in our case, according to (10) and (12)

$$h_i^1(X) = \max \{w_i^+ f_i(X) - F^*(S_i), 0\}$$

$$= \max \{(w_i^+ - F^*(S_i)), 0\} : f_i(X),$$

$i \in [1:m]$. Therefore, Problem B1 is nothing but Problem BOTTLE($S'$) where scenario $S'$ is defined by assigning weights $w_i = \max \{(w_i^+ - F^*(S_i)), 0\}$ to elements $e_i$, $i \in [1:m]$. Thus, Problems ROBBOT-1 and ROBBOT-2 can be solved by the following

Algorithm 1.

Begin

Step 1. Obtain values $F^*(S_i)$, $i \in [1:m]$.

Step 2. Solve Problem BOTTLE($S'$) and Problem BOTTLE($\hat{S}$), where scenario $S'$ is defined above, and scenario $\hat{S}$ was defined in the previous section.

End

An optimal solution for Problem BOTTLE($S'$) (Problem BOTTLE($\hat{S}$)) is an optimal solution for Problem ROBBOT-1 (Problem ROBBOT-2). If Step 1 of Algorithm 1 is implemented by solving $m$ problems BOTTLE($S_i$), $i \in [1:m]$, then the complexity of Algorithm 1 is $O(m \cdot$ complexity of solving Problem BOTTLE(S)). However, it is not necessary to solve all problems BOTTLE($S_i$), $i \in [1:m]$. Step 1 of Algorithm 1 can be implemented using the following procedure.

Procedure 1. Let $S^0$ denote the scenario where each $e_i \in E$ has the weight equal to the corresponding lower bound $w_i^-$. Begin

1. Solve Problem BOTTLE($S^0$) and obtain an optimal solution $X^0$ and the value $F^*(S^0)$.

2. For each $e_i \in E$ such that $e_i \notin X^0$, set $F^*(S_i) = F^*(S^0)$.

3. For each $e_i \in X^0$ such that $w_i^+ + F^*(S_i)$, set $F^*(S_i) = F^*(S^0)$.

4. For each $e_i \in X^0$ such that $w_i^+ > F^*(S_i)$, solve Problem BOTTLE(S) to obtain value $F^*(S_i)$.

End

In Procedure 1, Problem BOTTLE(S) is solved at most $|X^0| + 1$ times; therefore, the complexity of Procedure 1 is $O(m + |X^0| \cdot$ complexity of Problem BOTTLE(S)), which is also the (improved) complexity of Algorithm 1.

Let us state the complexity of Algorithm 1 for several specific bottleneck combinatorial optimization problems.

(a) Robust bottleneck spanning tree problems. Set $E$ is the set of edges of an undirected network $G = (V, E)$, $|V| = n$, $|E| = m$, $A = \{X \in 2^E| edges in X comprise a spanning tree of G\}$. Since Problem BOTTLE(S) can be solved in $O(m)$ time in this case [6] and $|X^0| \leq n - 1$, we obtain $O(nm)$ order of complexity for Algorithm 1.

(b) Robust bottleneck path problem. Set $E$ is as in the previous example; $A = \{X \in 2^E| edges in X comprise a path from a given source node $s$ to a given sink node $t$\}. Since Problem BOTTLE(S) can be solved in $O(m)$ time in this case [25] and $|X^0| \leq n - 1$, we obtain $O(nm)$ order of complexity for Algorithm 1. Bottleneck path problems (also known as maximum capacity path problems) have applications in reliability theory [20], bicriteria optimization [14] and service routing [4].

(c) Robust bottleneck Steiner subnetwork problems with k-connectivity constraints. Set $E$ is as in the previous examples. A connected network is called $k$-edge-connected, $k \geq 1$, if it is connected and deleting any $k - 1$ or less edges (without their endpoints) does not disconnect it. A connected network is called $k$-node-connected, $k \geq 1$, if it has at least $k + 1$ nodes and deleting any $k - 1$ or less nodes does not disconnect it. Let $A = \{X \in 2^E| edges in X comprise
a subnetwork which connects a given set \( T \subset V \) of
terminal nodes and which is \( k \)-node-connected (or \( k \)-edge-connected)). Bottleneck Steiner subnetwork
problems have applications in telecommunication sys-
tems and VLSI; \( k \)-connectivity requirements arise in
connection with reliability issues, where it is neces-
sary to take into account the possibility of failure of
several edges or nodes. For \( k = 2 \), the problem with-
out uncertainty (i.e. Problem BOTTLE(\( S \))) has been
studied in [23,8,26]; for a general \( k \geq 1 \) the problem
has been studied in [1]. If \( k \leq 3 \), Problem BOTTLE(\( S \))
can be solved in \( O(m + n \log n) \) time (\( O(m) \) time if
\( k = 1 \)) obtaining a solution of cardinality not greater
than \( k(n - 1) \) [26,1]; therefore, \( |X^0| \leq k(n - 1) \), and
we obtain \( O(nm + n^2 \log n) \) order of complexity for
Algorithm 1 (\( O(nm) \) if \( k = 1 \)).

4. Minmax regret minimax multifacility location
problems

Minmax regret approach was first applied to a loca-
tion problem by Kouvelis et al. [17] (see also [18]);
they obtained polynomial algorithms for the minmax
regret 1-median problem on a tree with uncertainty
in weights of nodes. For the same problem on a tree,
Chen and Lin [9] and subsequently Averbakh and
Berman [2] also developed algorithms with improved or-
ders of complexity; Averbakh and Berman [2] also
presented a polynomial algorithm for the problem
on a general network. Vairaktarakis and Kouvelis
[28] considered the 1-median problem on a tree with
minimax-regret objective where both uncertainty and
dynamic evolution of node demands and transporta-
tion costs are present. Labbe et al. [19] study sensitiv-
ity analysis issues for the 1-median problem on a tree.
In this section, we apply the general methodology
of Section 2 to minmax-regret multicenter location
models.

Let \( G = (V,E) \) be a transportation network with
\( V = (v_1,\ldots,v_n) \) the set of nodes and \( E \) the set of edges.
\( G \) will also denote the set of all points of the network.
For any \( a,b \in G \), \( d(a,b) \) denotes the shortest distance
between \( a \) and \( b \). Nodes of the network represent cus-
tomers; the customers need service from facilities that
should be located on the network. It is required to
locate \( p \) facilities, \( 1 \leq p < n \). Background informa-
tion on facilities location theory can be found in [22].

4.1. Minmax regret multicenter problems with
mutual communication

In this model, the facilities are assumed to be distin-
guishable, i.e. they provide different types of service.
A customer may need service from several (or all) fa-
cilities, and the facilities themselves may need service
from each other. Given are \( np + \frac{1}{2} p(p - 1) \) numbers
\( q_{ij}, r_{tk}, i \in [1:n], j \in [1:p], 1 \leq t < k \leq p \); \( q_{ij} \) is a
nonnegative upper bound on the distance between the
customer located at \( v_i \) and facility \( j \); \( r_{tk} \) is a nonneg-
avtive upper bound on the distance between facilities
\( t \) and \( k \). A vector \( X = (x_1,\ldots,x_p) \in G^p \) is called a
location vector; \( k \)th component \( x_k \) of \( X \) represents the
location for \( k \)th facility. A location vector \( X \) is called a
feasible location vector if it satisfies the constraints
\( d(v_i,x_j) \leq q_{ij}, i \in [1:n], j \in [1:p], \) \( (13) \)
\( d(x_t,x_k) \leq r_{tk}, 1 \leq t < k \leq p. \) \( (14) \)

Let \( A \) be the set of all feasible location vectors. A
scenario \( S \) represents an assignment of interaction
weights for all “customer-facility” and “facility-
facility” pairs. Specifically, a scenario \( S \) is a set of
the following \( m = np + \frac{1}{2} p(p - 1) \) weights:
\( w_{ij}^S \) – interaction weight between customer \( v_i \) and fa-
cility \( j \);
\( u_{tk}^S \) – interaction weight between facilities \( t \) and \( k \),
\( i \in [1:n], j \in [1:p], 1 \leq t < k \leq p \). Define \( m \) func-
tions \( f_{(v_i,j)}(X), f_{(t,k)}(X) \) as follows: \( f_{(v_i,j)}(X) =
d(v_i,x_j) \), \( f_{(t,k)}(X) = d(x_t,x_k) \). Problem OPT(\( S \))
in this setting is the problem of finding a feasible
location vector that minimizes the maximum of
weighted distances between customers and facili-
ties and between pairs of facilities; it is known as
the Distance-Constrained Multicenter Problem
with Mutual Communication [12,10] and will be re-
ferred to as Problem MMC(\( S \)). Problems ROB-1 and
ROB-2 in this setting will be referred to as Problems
ROBMCC-1 and ROBMCC-2, respectively.

We assume that upper bounds \( w_{ij}^+, w_{hk}^- \) and lower
bounds \( w_{ij}^-, w_{hk}^+ \) are given for interaction weights
\( w_{ij}^S, u_{tk}^S \); i.e. \( S^S \) is the Cartesian product of \( m \) intervals
of uncertainty \( [w_{ij}^-, w_{ij}^+], [u_{tk}^-, u_{tk}^+] \). Let \( S_{(v_i,j)} \) denote
the scenario where weight \( w_{ij} \) is equal to the corre-
ponding upper bound \( w_{ij}^+ \) and all other interaction
weights are equal to the corresponding lower bounds.
Let $S(t,k)$ denote the scenario where weight $u_{t,k}$ is equal to the upper bound $u^*_t$ and all other interaction weights are equal to the corresponding lower bounds.

We will use the following Distance-Constrained Feasibility Problem that has been extensively studied in the literature [13,27]:

**Problem DCF.** Find $X = (x_1, \ldots, x_p) \in G^p$ that satisfies the constraints

$$d(v_i, x_j) \leq c^i_j, \quad i \in [1:n], \quad j \in [1:p],$$

$$d(x_t, x_k) \leq c^t_k, \quad 1 \leq t < k \leq p,$$

where $c^i_j, c^t_k$ are nonnegative constants.

Consider the feasibility version of Problem ROBMMC-1.

**Problem C1.** For a given $\gamma \geq 0$, find $X \in A$ such that $Z_0(X) \leq \gamma$.

According to Theorem 1 and its corollaries, Problem C1 can be reformulated as follows:

**Problem D1.** For a given $\gamma \geq 0$, find $X \in A$ satisfying constraints

$$w^{\gamma}_{ij}d(v_i, x_j) - F^*(S(v_i, j)) \leq \gamma,$$

$$i \in [1:n], \quad j \in [1:p],$$

$$u^{\gamma}_{t,k}d(x_t, x_k) - F^*(S(t,k)) \leq \gamma, \quad 1 \leq t < k \leq p.$$  (15)

Let $\gamma^*$ be the smallest value of $\gamma$ such that Problem D1 is feasible; $\gamma^*$ is the optimal objective function value for Problem ROBMMC-1. Given value $\gamma^*$, an optimal solution for Problem ROBMMC-1 can be found by solving Problem D1 with $\gamma = \gamma^*$. Problem D1 is feasible for any $\gamma \geq \gamma^*$ and infeasible for any $\gamma < \gamma^*$. Problem D1 is equivalent to Problem DCF with

$$c^i_j = \min \left\{ q_{ij}, \frac{1}{w^i_{ij}}(\gamma + F^*(S(v_i, j))) \right\},$$

$$c^t_k = \min \left\{ r_{t,k}, \frac{1}{u^t_{t,k}}(\gamma + F^*(S(t,k))) \right\}.$$  (16)

The feasibility version of Problem MMC(S) is also an instance of Problem DCF [12].

Problem DCF is NP-hard in the general case [16], but has a number of important polynomially solvable special cases; for example, if network $G$ is a tree [13], or if the interaction between the facilities has a special structure [10,27]. Problem MMC(S) is usually solved by converting an algorithm for its feasibility version (Problem DCF) into an algorithm for the optimization version [12]. The same approach can be used for Problem ROBMMC-1. If all values $F^*(S(v_i, j))$, $F^*(S(t,k))$ are known, a polynomial combinatorial algorithm that solves Problem DCF (and, therefore, Problem D1) can be converted to a polynomial algorithm for problem ROBMMC-1 using binary search over $\gamma$ with rational representation of data or using Meggido’s parametric approach [21,12]; the methodology of such conversions (that use either binary search or the parametric approach) has been developed in [12], where implementation details are given in the context of problems on trees (in [12], a strongly polynomial algorithm for Problem DCF on a tree from [13] is used to obtain polynomial and strongly polynomial algorithms for Problem MMC(S) on trees). If the same conversion technique is used for Problems MMC(S) and ROBMMC-1, we obtain an algorithm for solving Problem ROBMMC-1 with computational effort roughly equivalent to solving $np + \frac{1}{2}p(p - 1) + 1$ Problems MMC(S) (including $np + \frac{1}{2}p(p - 1)$ problems to obtain values $F^*(S(v_i, j))$, $F^*(S(t,k)))$. Problem ROBMMC-2 can be solved with the same order of complexity, according to Corollary 3 from Theorem 1. So, whenever Problem MMC(S) can be solved in polynomial time, we obtain polynomial algorithms for Problems ROBMMC-1 and ROBMMC-2 (for example, when $G$ is a tree).

4.2. Minmax regret weighted $p$-center problems

In this model, it is assumed that the facilities are identical, and each customer is served by the closest facility. Let $A$ be the set of all $p$-element subsets of $G$; $X \in A$ represents chosen locations for the facilities and is called a location variant. Let $m = n$. Define functions $f_i(X)$, $i \in [1:m]$ as $f_i(X) = d(v_i, X)$, where $d(v_i, X) = \min\{d(v_i, x) \mid x \in X\}$ is the distance between node $v_i$ and the closest point of $X$. For any scenario $S = (w^1_1, \ldots, w^m_m)$, Problem OPT(S) in this setting will be referred to as Problem PC(S). Problem PC(S) is the classical weighted $p$-center problem; weights $w^i_j$, $i \in [1:m]$ can be interpreted as priorities of nodes (customers) $v_i$, $i \in [1:m]$. Problems ROB-1 and ROB-2 in this setting will be referred to as Problems ROBPC-1 and ROBPC-2, respectively.
Problem ROBPC-1 has been studied in [3]. The approach of [3] can be viewed as an application of the general methodology suggested in the current paper: reducing a problem with uncertainty to a similar problem without uncertainty using Theorem 1 and its Corollary 1. Namely, in [3] Problem ROBPC-1 is reduced to Problem PC(S) on an auxiliary network \( G' \), which is obtained from the original network \( G \) by appending a “dummy” edge to each node. The auxiliary network \( G' \) preserves the main structural properties of the original network \( G \) (e.g. if \( G \) is a tree, so is \( G' \); see details in [3]).

Problem ROBPC-2 is not studied in [3]; according to Corollary 3 from Theorem 1, it can be solved with the order of complexity \( O(m^3) \), which is polynomial whenever Problem PC(S) is polynomially solvable (e.g. when \( G \) is a general network and \( p = 1 \)).

5. Minmax regret maximum weighted tardiness scheduling problems

Suppose that \( m \) jobs have to be processed on a machine; the machine cannot process more than one job at any time. Let \( d_i, p_i \) denote the due date and the processing time of job \( i \), respectively. Processing of the jobs starts at time 0. Let \( A \) be the set of all permutations of \( \{1, 2, \ldots, m\} \); an \( X \in A \) represents a sequence of processing the jobs. If there are precedence constraints for the jobs, then \( A \) is the set of permutations satisfying the precedence constraints; all the subsequent results hold also for the case of precedence constraints. We assume that the jobs are processed according to a nondelay schedule, that is, between time 0 and the completion of all jobs the machine is never idle. Therefore, a sequence \( X \in A \) uniquely defines the completion times \( C_i = C_i(X) \) of all the jobs \( i \in [1:m] \). The tardiness of job \( i \) is defined as \( T_i(X) = \max\{C_i(X) - d_i, 0\} \). If we define \( f_i(X) = T_i(X), i \in [1:m] \) then for any scenario \( S = (w^1, \ldots, w^m) \), Problem OPT(S) in this setting is the maximum weighted tardiness scheduling problem and will be referred to as Problem WT(S). Problems ROB-1 and ROB2 in this setting will be referred to as Problems ROBWT-1 and ROBWT-2, respectively. Since \( w^i f_i(X) \) is a nondecreasing function of \( C_i \), Problem WT(S) can be solved in \( O(m^2) \) time using the backward dynamic programming algorithm from [24, p. 33] (even with precedence constraints).

Consider Problem ROBWT-1. According to Corollary 1 from Theorem 1, this problem can be replaced with Problem B1. Functions \( h_i^0(X) \) are nonincreasing functions of \( C_i \); therefore, given values \( F^*(S), i \in [1:m] \), problem B1 can also be solved in \( O(m^2) \) time using the backward dynamic programming algorithm from [24, p. 33]. Since for any \( i \in [1:m] \) value \( F^*(S) \) can be obtained in \( O(m^2) \) time using the same algorithm, we obtain that Problem B1 (and therefore Problem ROBWT-1) can be solved in \( O(m^3) \) time. With Corollary 3 from Theorem 1, we obtain that Problem ROBWT-2 can also be solved in \( O(m^3) \) time. The results hold without any changes if instead of \( T_i(X) \) we consider any other nondecreasing functions of \( C_i \).

For an analysis of a number of other minmax regret scheduling problems, see [18].

Appendix

Proof of Theorem 1. First, let us prove part (a). Consider an arbitrary \( X \in A \). Let \( S' \) be an absolute worst-case scenario for \( X \), and let \( X' \) be an optimal solution for Problem OPT(S'). Let \( j \in \text{Arg max}_{i \in [1:m]} w^j f_j(X) \) (i.e. \( w^j f_j(X) \geq w^j f_j(X) \) for any \( i \in [1:m] \)). Observe that \( f_j(X) \geq f_j(X') \) (since \( F(S', X) \geq F(S', X') = F^*(S'), F(S', X) = w^j f_j(X)', F(S', X') = w^j f_j(X') \) and \( w^j > 0 \). Therefore, value \( F(S', X) - F(S', X') \) cannot decrease if in scenario \( S' \) we increase \( w^j \) and decrease \( w^j \) for any \( i \neq j \). It cannot increase either, since \( S' \) is an absolute worst-case scenario for \( X \). Therefore, scenario \( S' \) is also an absolute worst-case scenario for \( X, X' \) is an optimal solution for Problem OPT(S), and \( Z_i(X) = F(S, X) - F^*(S) = w^j f_j(X) - F^*(S) = h_i^0(X) \). Since \( h_i^0(X) \leq Z_i(X) \) for any \( i \in [1:m] \), part (a) of the theorem is proven.

Let us now prove part (b). Consider an arbitrary \( X \in A \). Let \( S'' \) be the relative worst-case scenario for \( X \), and let \( X'' \) be an optimal solution for Problem OPT(S''). Let \( j \in \text{Arg max}_{i \in [1:m]} w^j f_j(X) \) (i.e. \( w^j f_j(X) \geq w^j f_j(X) \) for any \( i \in [1:m] \)). Observe that \( f_j(X) \geq f_j(X'') \) (since \( F(S'', X) \geq F(S'', X'') = F^*(S''), F(S'', X) = w^j f_j(X) \) and \( F(S'', X') \geq w^j f_j(X) \) for any \( i \in [1:m] \)).
$w_j^{*''} f_j(X'')$ and $w_j^{*''} > 0$). Therefore, value $[F(S'', X) - F(S'', X'')]/[F(S'', X'')]$ cannot decrease if in scenario $S''$ we increase $w_j^{*''}$ and decrease $w_i^{*''}$ for any $i \neq j$. It cannot increase either, since $S''$ is a relative worst-case scenario for $X$. Therefore, scenario $S_j$ is also a relative worst-case scenario for $X$, $X''$ is an optimal solution for Problem OPT($S_j$), and

$$Z_2(X) = \frac{F(S_j, X) - F^*(S_j)}{F^*(S_j)} = h_2^j(X).$$

Since $h_2^j(X) \leq Z_2(X)$ for any $i \in [1 : n]$, part (b) of the theorem is proven. \qed

References

[17] P. Kouvelis, G. Vairaktarakis, G. Yu, Robust 1-median location on a tree in the presence of demand and transportation cost uncertainty, Working paper 93/94-3-4, Department of Management Science and Information Systems, Graduate School of Business, The University of Texas at Austin.