Exact and Heuristic Algorithms for the Interval Min-Max Regret Generalized Assignment Problem[☆]

Wei Wu[a,*], Manuel Iori[b], Silvano Martello[c], Mutsunori Yagiura[d]

[a]*Seikei University, Tokyo, Japan*
[b]*University of Modena and Reggio Emilia, Reggio Emilia, Italy*
[c]*University of Bologna, Bologna, Italy*
[d]*Nagoya University, Nagoya, Japan*

Abstract

We consider the *generalized assignment problem* (GAP) with min-max regret criterion under interval costs. This problem models many real-world applications in which jobs must be assigned to agents but the costs of assignment may vary after the decision has been taken. We computationally examine two heuristic methods: a fixed-scenario approach and a dual substitution algorithm. We also examine exact algorithmic approaches (Benders-like decomposition and branch-and-cut) and further introduce a more sophisticated algorithm that incorporates various methodologies, including Lagrangian relaxation and variable fixing. The resulting Lagrangian-based branch-and-cut algorithm performs satisfactorily on benchmark instances.

Keywords: Combinatorial optimization, Min-max regret generalized assignment problem, Branch-and-cut, Lagrangian relaxation, Variable fixing

1. Introduction

Several optimization problems arising in real world applications do not have accurate estimates of the problem parameters when the optimization decision is taken. Stochastic programming (SP) and robust optimization (RO) are two common approaches for the solution of optimization problems under uncertainty. A decision maker should select a model and a solution approach based on careful examination of the input parameters. In an SP approach, the decision maker has the probability distribution for the random parameters observed in the problem. SP aims to minimize the expected cost, and in most cases it is able to obtain high-quality solutions based on the fact that probability distributions of the parameters are known or can be estimated. However, collecting such distributions can be very time-consuming in many read-world situations, and sometimes they are even unable to obtain. On the other hand, RO aims to minimize the worst-case cost or to minimize the maximum regret without typically requiring any probability distribution of the problem parameters. However, the biggest disadvantage of an RO approach is that a min-max result may be extremely pessimistic. The min-max criterion, known as a typical approach for RO, looks for a solution with the best worst-case value across all scenarios. To deal with the trade-off between robustness and performance, the min-max regret criterion is defined to minimize the worst-case regret, where the *regret* is the difference between the actual cost and the optimal cost that would have been obtained if a different solution had been chosen.

Robust optimization has been one of the hottest topic of the last decade. Buhmann et al. [1] proposed a generic approach based on a measure of similarity of typical scenarios. Snyder and Daskin [2] proposed the stochastic p-robust model in which the object is to minimize the expected cost while bounding the relative regret in each scenario.

[☆]A preliminary version of this paper was presented at the IEEE International Conference on Industrial Engineering and Engineering Management 2014 (IEEM 2014).

[*]Corresponding author. Tel: +81-422-37-3440; fax: +81-422-37-3871.
Email address: `wuwei@st.seikei.ac.jp` (Wei Wu)

Poss [3] provided exact and approximation algorithms to min-max robust combinatorial optimization problems for an uncertainty polytope that is defined by knapsack constraints, where the uncertainty set is an extension of the one considered by Bertsimas and Sim [4]. Lappas and Gounaris [5] considered RO under endogenous uncertainty which is affected by the decision maker's strategy. Recently, RO has also been applied to multi-objective optimization problems [6, 7]. Moreover, RO has many important applications in industry, such as energy planning [8, 9] and aircraft routing [10].

The min-max regret version of a number of important combinatorial optimization problems has been recently studied, such as the traveling salesman problem [11], the assignment problem [12], the minimum spanning tree problem [13, 14], scheduling problems [15, 16], the shortest path problem [17], the facility location problem [18], the resource allocation problem [19], the set covering problem [20], and the 0-1 knapsack problem [21]. The min-max regret criterion has also been applied to revenue management [22], and the assembly line worker assignment and balancing problem [23]. In the literature, the solution of this kind of problems has been tackled with a number of exact and heuristic techniques. Concerning exact algorithms, Benders-like decomposition methods iteratively solve a relaxed master problem in which only a subset of scenarios is considered. Cuts corresponding to violated scenarios are computed by the solution of a slave problem and possibly added to the master problem, until a solution satisfying all scenarios is found (see, e.g., Montemanni [24]). Branch-and-cut methods extend the Benders-like decomposition by including the solution of the slave problem at all nodes of the enumeration tree, and have been applied to several min-max regret problems (see Pereira and Averbakh [20]). Other authors used instead the structure of the min-max regret problem at hand to devise tailored combinatorial branch-and-bound algorithms, such as the one by Montemanni and Gambardella [25]. Concerning heuristic methods, several attempts have been made, including constructive heuristics based on the solution of a fixed scenario (Kasperski and Zielinski [16]) or on the inclusion of a dual relaxation component (Furini et al. [21]), as well as more elaborated metaheuristics such as genetic algorithms and filter-and-fan methods (Pereira and Averbakh [20]). For a deeper analysis of the existing literature, we refer the interested reader to Kouvelis and Yu [26], Kasperski [27], Aissi, Bazgan and Vanderpooten [28], and Candia-Véjar, Álvarez-Miranda, and Maculan [29].

In this paper we consider the *generalized assignment problem* (GAP) with min-max regret criterion under interval costs. The classical GAP is an NP-hard combinatorial optimization problem [30] having many applications in different real-world areas, such as production planning, scheduling, timetabling, telecommunication, investment and transportation (see [31], [32], [33], and [34]). The *interval min-max regret generalized assignment problem* (MMR-GAP) is a generalization of the GAP to the case in which the cost coefficients are uncertain. Consider, for example, the case in which a seller aims at serving customers by assigning their demands to different shippers. This problem is easily formalized by the classical GAP, but in a more general situation, the final assignment costs are often affected by variability. The seller could aim at minimizing a posteriori regret of the selected choice with respect to any possible assignment-cost scenario. Then, the solution of a MMR-GAP would allow the seller to obtain a robust assignment with low regret. As another real-life example, the assignment cost in many cases represents risk aversion (or penalty) in an investment problem (or in a scheduling problem) which is known to be an important application of the generalized assignment problem (see [32]). The degree of risk aversion or penalty is affected by many factors and is hard to accurately quantify at the optimization stage. Then, a min-max regret result would be robust to avoid high risk or penalty. We assume that every cost coefficient can take any value in a corresponding given interval, regardless of the values taken by the other cost coefficients. The problem requires to find a robust solution that minimizes the maximum regret. We computationally evaluate a heuristic algorithm for the MMR-GAP that solves the underlying GAP to optimality under a fixed scenario. We consider three scenarios: lowest cost, highest cost, and median cost. The median cost scenario leads to a solution of the MMR-GAP whose objective value is within twice the optimal value. We also computationally evaluate a dual substitution heuristic based on a *mixed integer programming* (MIP) model obtained by replacing some constraints with the dual of their continuous relaxation.

We also examine three exact algorithmic approaches that iteratively solve the problem by only including a subset of scenarios. The first approach is based on Benders-like decomposition: it solves a MIP with incomplete scenarios and iteratively supplements the scenarios corresponding to violated constraints. The second approach is a basic branch-and-cut algorithm, in which the violated constraints are added at the nodes of a MIP enumeration tree. We then introduce a new Lagrangean-based branch-and-cut algorithm and enhance it through: (i) effective Lagrangian relaxations, to provide tighter lower bounds than those produced by the linear programming relaxation; (ii) an efficient variable fixing technique; (iii) a two-direction dynamic programming approach to effectively solve the Lagrangian subproblems. We evaluate the algorithms through computational experiments on different benchmarks.

The classical GAP is a very general and difficult optimization problem and exploring its min-max regret version is important both from a theoretical and a practical point of view. In this work, we consider both heuristic and exact approaches for the MMR-GAP. The algorithms have been obtained both by a non-trivial use of results from the literature and by original developments of new ideas. We provide extensive computational experiments, and the results show that a Lagrangian-based branch-and-cut approach can exactly solve 102 out of 240 instances with up to 10 agents and 80 jobs within reasonable running times. Some of the results presented in the next sections were sketched, without proofs, in the proceeding paper [35], together with a small set of preliminary computational results. We hope our results can stimulate further research in the area of min-max regret optimization problems.

2. Problem Description

2.1. Generalized Assignment Problem

The *generalized assignment problem* (GAP) is defined as follows. Given a set of n jobs $J = \{1, \ldots, n\}$ and a set of m agents $I = \{1, \ldots, m\}$, we look for a minimum cost assignment, subject to assigning each job to exactly one agent and satisfying a resource constraint for each agent. Assigning job j to agent i incurs a cost c_{ij} and consumes an amount a_{ij} of a resource, whereas the total amount of the resource available at agent i (agent capacity) is b_i.

A natural formulation of the GAP is defined over a two-dimensional binary variable x_{ij} indicating that job j is assigned to agent i if and only if $x_{ij} = 1$:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i \qquad \forall i \in I \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in J \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in I,\ \forall j \in J. \tag{4}$$

For convenience, we define X_0 to be the set of all feasible solutions of the GAP, i.e.,

$$X_0 = \{x \mid x \text{ satisfies constraints (2)–(4)}\}. \tag{5}$$

2.2. Interval Min-Max Regret Generalized Assignment Problem

In this paper we assume that the cost c_{ij} can take any value within a given range $[c_{ij}^-, c_{ij}^+]$. An array of costs c_{ij}^s satisfying $c_{ij}^s \in [c_{ij}^-, c_{ij}^+]$ $\forall i \in I,\ j \in J$, is called a *scenario* and is denoted by s. We define $z^s(x)$ to be the objective value of solution $x \in X_0$ under scenario s:

$$z^s(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^s x_{ij}. \tag{6}$$

We denote by z_*^s the optimal solution value under scenario s, i.e., $z_*^s = \min_{y \in X_0} z^s(y)$. The *regret* $r^s(x)$ associated with solution x under scenario s is then the difference between these two values:

$$r^s(x) = z^s(x) - z_*^s. \tag{7}$$

Let S denote the set of all possible scenarios, i.e., $S = \{s \mid c_{ij}^s \in [c_{ij}^-, c_{ij}^+]\ \forall i \in I, j \in J\}$. The *maximum regret* of a solution x is then the maximum $r^s(x)$ value over all scenarios:

$$r_{\max}(x) = \max_{s \in S} r^s(x). \tag{8}$$

The MMR-GAP is to find a feasible solution x such that the maximum regret is minimized:

$$\min_{x\in X_0} r_{\max}(x) = \min_{x\in X_0}\max_{s\in S} r^s(x) = \min_{x\in X_0}\max_{\substack{y\in X_0\\ s\in S}}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^s x_{ij} - \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^s y_{ij}\right\}. \tag{9}$$

This formulation can be rewritten using the following classical general result that was proposed by Aissi et al. [28] (whose roots are in Yaman et al. [14]), which also applies to many other interval min-max problems.

Lemma 2.1. *The regret of a solution $x \in X_0$ is maximized under the following scenario $\sigma(x)$:*

$$c_{ij}^{\sigma(x)} = \begin{cases} c_{ij}^+ & \text{if } x_{ij} = 1 \\ c_{ij}^- & \text{otherwise} \end{cases} \qquad \forall i \in I,\ \forall j \in J. \tag{10}$$

In other words, the value $r_{\max}(x)$ is achieved by the scenario that gives the worst costs to the job-agent pairs selected by the assignment x, and the best costs to the non-selected job-agent pairs. Since x_{ij} is a binary variable, $c_{ij}^{\sigma(x)}$ can also be written as $c_{ij}^{\sigma(x)} = c_{ij}^- + (c_{ij}^+ - c_{ij}^-)x_{ij}$.

From Lemma 2.1, the maximum regret is achieved if we apply the worst scenario $\sigma(x)$ to s, and, from (9), the MMR-GAP can be rewritten as

$$\min_{x\in X_0} r_{\max}(x) = \min_{x\in X_0}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^+ x_{ij} - \min_{y\in X_0}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{\sigma(x)} y_{ij}\right\}. \tag{11}$$

We assume in the following that all input data are integers.

3. Heuristic Algorithms

In this section we present heuristic algorithms that, in view of the considerations of the previous section, are not guaranteed to run in polynomial time.

3.1. Fixed-Scenario Algorithm

We introduce a heuristic approach based on the observation that a feasible solution to an MMR-GAP instance can be obtained by fixing a scenario, solving the resulting GAP instance to optimality, and evaluating the maximum regret of the obtained solution using (11). This approach was used for other interval min-max regret problems (see, e.g., [16] and [20]).

We consider three scenarios: the lowest cost $c_{ij}^s = c_{ij}^-$, the highest cost $c_{ij}^s = c_{ij}^+$, and the median cost $c_{ij}^s = (c_{ij}^- + c_{ij}^+)/2$.

For the median-cost scenario, the following result is a special case of a more general result proved in [16].

Lemma 3.1. *Let $\widetilde{s}$ be the median-cost scenario, i.e., $c_{ij}^{\widetilde{s}} = (c_{ij}^- + c_{ij}^+)/2\ \forall i \in I,\ \forall j \in J$, and let $\widetilde{x}$ be an optimal solution to the GAP under $\widetilde{s}$. Then, $r_{\max}(\widetilde{x}) \le 2r_{\max}(x)$ holds for all $x \in X_0$.*

Proof. For any two solutions x, $y \in X_0$ and scenario $s \in S$, we have

$$z^s(x) - z^s(y) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^s (x_{ij} - y_{ij}) \tag{12}$$

from definition (6). Using (12) and definitions (7) and (8), the following inequality holds:

$$\begin{aligned} r_{\max}(y) &= \max_{s\in S}\{z^s(y) - z_*^s\} \\ &\ge \max_{s\in S}\{z^s(y) - z^s(x)\} \\ &= \max_{s\in S}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^s (y_{ij} - x_{ij}) \\ &= \sum_{(i,j):\, y_{ij} > x_{ij}} c_{ij}^+ (y_{ij} - x_{ij}) + \sum_{(i,j):\, y_{ij} < x_{ij}} c_{ij}^- (y_{ij} - x_{ij}). \end{aligned} \tag{13}$$

From (12), we can also obtain

$$z^{\sigma(x)}(x) = z^{\sigma(x)}(y) + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{\sigma(x)}(x_{ij} - y_{ij}) = z^{\sigma(x)}(y) + \sum_{(i,j):\, x_{ij}>y_{ij}} c_{ij}^{+}(x_{ij} - y_{ij}) + \sum_{(i,j):\, x_{ij}<y_{ij}} c_{ij}^{-}(x_{ij} - y_{ij}), \tag{14}$$

where $\sigma(x)$ is the worst-case scenario defined by Lemma 2.1. Then, by subtracting $z_*^{\sigma(x)}$ from both sides of (14), we get

$$r_{\max}(x) = z^{\sigma(x)}(x) - z_*^{\sigma(x)} = z^{\sigma(x)}(y) - z_*^{\sigma(x)} + \sum_{(i,j):\, x_{ij}>y_{ij}} c_{ij}^{+}(x_{ij} - y_{ij}) + \sum_{(i,j):\, x_{ij}<y_{ij}} c_{ij}^{-}(x_{ij} - y_{ij}). \tag{15}$$

By combining this with $r_{\max}(y) \geq z^{\sigma(x)}(y) - z_*^{\sigma(x)}$, we get

$$r_{\max}(x) \leq r_{\max}(y) + \sum_{(i,j):\, x_{ij}>y_{ij}} c_{ij}^{+}(x_{ij} - y_{ij}) + \sum_{(i,j):\, x_{ij}<y_{ij}} c_{ij}^{-}(x_{ij} - y_{ij}). \tag{16}$$

Now we consider the median-cost scenario $\widetilde{s}$. Since $\widetilde{x}$ is an optimal solution under $\widetilde{s}$, we have $z^{\widetilde{s}}(\widetilde{x}) \leq z^{\widetilde{s}}(y)$, and according to (12),

$$\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{1}{2}(c_{ij}^{+} + c_{ij}^{-})(\widetilde{x}_{ij} - y_{ij}) = z^{\widetilde{s}}(\widetilde{x}) - z^{\widetilde{s}}(y) \leq 0 \tag{17}$$

holds, which is equivalent to

$$\sum_{(i,j):\, y_{ij}>\widetilde{x}_{ij}} c_{ij}^{+}(y_{ij} - \widetilde{x}_{ij}) + \sum_{(i,j):\, y_{ij}<\widetilde{x}_{ij}} c_{ij}^{-}(y_{ij} - \widetilde{x}_{ij}) \geq \sum_{(i,j):\, \widetilde{x}_{ij}>y_{ij}} c_{ij}^{+}(\widetilde{x}_{ij} - y_{ij}) + \sum_{(i,j):\, \widetilde{x}_{ij}<y_{ij}} c_{ij}^{-}(\widetilde{x}_{ij} - y_{ij}). \tag{18}$$

Then, by applying this to (13), we obtain

$$r_{\max}(y) \geq \sum_{(i,j):\, \widetilde{x}_{ij}>y_{ij}} c_{ij}^{+}(\widetilde{x}_{ij} - y_{ij}) + \sum_{(i,j):\, \widetilde{x}_{ij}<y_{ij}} c_{ij}^{-}(\widetilde{x}_{ij} - y_{ij}). \tag{19}$$

By combining (16) and (19), we finally obtain

$$r_{\max}(\widetilde{x}) \leq 2r_{\max}(y) \qquad \forall y \in X_0. \tag{20}$$

□

We found a tight example for the approximation ratio of Lemma 3.1.

Lemma 3.2. *There is an instance of the MMR-GAP for which an optimal solution to GAP under the median-cost scenario $\widetilde{s}$ has a regret twice as large as the optimal regret.*

Proof. Let $m = 3$, $n = 1$, $a_{11} = a_{21} = a_{31}$, $b_1 = 1$ with interval costs $c_{11} \in [1, 1]$, $c_{21} \in [0, 2]$, $c_{31} \in [0, 2]$. Assigning the unique job to any agent gives an optimal solution to GAP under the median-cost scenario, because all median costs have value 1. The optimal MMR-GAP solution assigns the job to agent 1, attaining the maximum regret of value 1, while assigning the job to agent 2 (or 3) gives the maximum regret of value 2. □

3.2. Dual Substitution Heuristic

The dual substitution heuristic introduced in this section is based on a MIP formulation in which some of the constraints are replaced by their dual counterpart in the linear relaxation of the problem. Kasperski provided a general technique in [27], and similar ideas of using MIP models have been used a number of times to produce exact algorithms for other min-max regret problems having zero duality gap, such as the min-max regret shortest problem [17], the min-max regret spanning tree problem [14]. To our knowledge, it is relatively new to use such ideas to design heuristic algorithms for problems with possibly non-zero duality gaps [21], and not much has been done to apply such ideas to various problems and computationally evaluate the resulting heuristics. In our research, we use it as a heuristic for the MMR-GAP.

The minimization problem over y in (11), $\min_{y\in X_0} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^{\sigma(x)} y_{ij}$, for every fixed x is an instance of the GAP. We consider the linear program obtained by replacing the integrality constraint $y_{ij} \in \{0,1\}$ with the weaker requirement $y_{ij} \geq 0$ for all $i \in I$ and $j \in J$, i.e.,

$$\begin{aligned}
\min\ & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^{\sigma(x)} y_{ij} && && (21)\\
\text{s.t.}\ & \sum_{j=1}^{n} a_{ij} y_{ij} \leq b_i && \forall i \in I && (22)\\
& \sum_{i=1}^{m} y_{ij} = 1 && \forall j \in J && (23)\\
& y_{ij} \geq 0 && \forall i \in I,\ \forall j \in J. && (24)
\end{aligned}$$

We introduce two types of dual variables: λ_i ($i \in I$) for constraints (22) and μ_j ($j \in J$) for constraints (23). The dual of (21)–(24) is then

$$\begin{aligned}
\max\ & -\sum_{i=1}^{m} b_i \lambda_i + \sum_{j=1}^{n} \mu_j && && (25)\\
\text{s.t.}\ & -a_{ij}\lambda_i + \mu_j \leq c_{ij}^{\sigma(x)} && \forall i \in I,\ \forall j \in J && (26)\\
& \lambda_i \geq 0 && \forall i \in I. && (27)
\end{aligned}$$

By embedding (25)–(27) into (11), we obtain the following dual substitution model (D-MMR-GAP):

$$\begin{aligned}
\min & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^{+} x_{ij} + \sum_{i=1}^{m} b_i \lambda_i - \sum_{j=1}^{n} \mu_j && && (28)\\
\text{s.t.}\ & -a_{ij}\lambda_i + \mu_j \leq c_{ij}^{-} + (c_{ij}^{+} - c_{ij}^{-}) x_{ij} && \forall i \in I, \forall j \in J && (29)\\
& \lambda_i \geq 0 && \forall i \in I && (30)\\
& x \in X_0. && && (31)
\end{aligned}$$

Intuitively, the D-MMR-GAP is not easier than the GAP, since it contains all the GAP constraints. In fact, the D-MMR-GAP has the following complexity:

Property 3.1. *Problem D-MMR-GAP is NP-hard in the strong sense.*

Proof. Given an instance of the classical GAP, we transform it to an instance of the D-MMR-GAP by considering intervals with no gap $c_{ij}^{+} = c_{ij}^{-} = c_{ij}$. Then, because the right-hand side of (29) becomes a constant value c_{ij}, the resulting D-MMR-GAP decomposes into two problems:

$$\begin{aligned}
\text{(i)}\ \min\ & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} && && (32)\\
\text{s.t.}\ & x \in X_0; && && (33)\\
\text{(ii)}\ \min\ & \sum_{i=1}^{m} b_i \lambda_i - \sum_{j=1}^{n} \mu_j && && (34)\\
\text{s.t.}\ & -a_{ij}\lambda_i + \mu_j \leq c_{ij} && \forall i \in I,\ \forall j \in J && (35)\\
& \lambda_i \geq 0 && \forall i \in I. && (36)
\end{aligned}$$

Note that these two problems are totally separated and the former problem is exactly the same as the given GAP instance. This implies that the GAP is polynomial-time reducible to the D-MMR-GAP, and in this reduction it is not necessary to introduce large numbers (i.e., the maximum among the absolute values of the coefficients c_{ij}^{+}, c_{ij}^{-}, a_{ij}, and b_i of the D-MMR-GAP instance is the same as that of the GAP instance). Since the GAP is known to be NP-hard in the strong sense, the same hods for the D-MMR-GAP. $\square$

Property 3.2. *The optimal solution value of the D-MMR-GAP is an upper bound for the MMR-GAP.*

Proof. Let $\hat{X}$ denote the set of feasible solutions for the continuous relaxation of the GAP:

$$\hat{X} = \{y \mid y \text{ satisfies constraints (22)–(24)}\}.$$

From (5), we have $X_0 \subseteq \hat{X}$, and hence

$$\min_{x\in X_0}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{+}x_{ij} - \min_{y\in X_0}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{\sigma(x)}y_{ij}\right\} \le \min_{x\in X_0}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{+}x_{ij} - \min_{y\in \hat{X}}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{\sigma(x)}y_{ij}\right\}. \tag{37}$$

Note that, according to (11), the left side of the above inequality is the optimal value of the MMR-GAP.

Let $D(x)$ denote the set of feasible solutions of the dual problem in (25)–(27):

$$D(x) = \{(u, v) \mid (u, v) \text{ satisfies constraints (26)–(27)}\}.$$

Due to the strong duality theorem, the optimal values of the two problems (21)–(24) and (25)–(27) are the same, and hence

$$\min_{x\in X_0}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{+}x_{ij} - \min_{y\in \hat{X}}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{\sigma(x)}y_{ij}\right\} = \min_{x\in X_0}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{+}x_{ij} - \max_{(u,v)\in D(x)}\left\{-\sum_{i=1}^{m} b_i\lambda_i + \sum_{j=1}^{n}\mu_j\right\}\right\}. \tag{38}$$

It is not hard to see that the D-MMR-GAP is equivalent to the right-hand side of equality (38). The thesis follows from (37) and (38). $\square$

In addition, it easily follows that a tighter upper bound can be obtained as follows.

Property 3.3. *The upper bound obtained by evaluating the maximum regret of any optimal solution of the D-MMR-GAP is at least as good as the optimal value of the D-MMR-GAP.*

Proof. For any feasible solution $x \in X_0$, the smallest objective value of the D-MMR-GAP (i.e., x is fixed and u and v are optimized) is

$$z^{\sigma(x)}(x) - \min_{y\in\hat{X}} z^{\sigma(x)}(y). \tag{39}$$

According to (7) and Lemma 2.1, the min-max regret of this solution x is

$$z^{\sigma(x)}(x) - \min_{y\in X_0} z^{\sigma(x)}(y). \tag{40}$$

Since $X_0 \subseteq \hat{X}$, we have

$$z^{\sigma(x)}(x) - \min_{y\in\hat{X}} z^{\sigma(x)}(y) \ge z^{\sigma(x)}(x) - \min_{y\in X_0} z^{\sigma(x)}(y). \tag{41}$$

$\square$

We observed in our experiments that the dual substitution heuristic tends to obtain better solutions compared to the fixed-scenario heuristic. However, unlike the fixed-scenario algorithm with the median-cost scenario, this algorithm cannot have a guarantee on its solution quality.

Property 3.4. *For any positive K, there exists an instance of the MMR-GAP for which the dual substitution heuristic obtains a solution whose maximum regret is at least K times the optimal value.*

Proof. Let $m = 2$, $n = 3$, $a_{11} = a_{21} = 1$, $a_{12} = a_{22} = 4$, $a_{13} = a_{23} = 2$, $b_1 = 5$, $b_2 = 3$ with interval costs $c_{11} \in [1, K_1]$, $c_{21} \in [1, 2]$, $c_{12} \in [K_2, K_2]$, $c_{22} = c_{13} = c_{23} \in [1, 1]$, where K_1 and K_2 satisfy $K_2 \gg K_1 \gg 1$ and $K_1 \geq K+1$. Obviously, jobs 2 and 3 have to be assigned to agents 1 and 2, respectively, while job 1 can be assigned to either agent. There are only two feasible solutions: solution x assigning job 1 to agent 1 with maximum regret $K_1 - 1$, and the optimal solution y assigning job 1 to agent 2 with maximum regret 1. On the other hand, since we applied LP relaxation in the D-MMR-GAP, solution x with maximum regret $(3/4)(K_2 - 1)$ is better than y with maximum regret $(3/4)(K_2 - 1) + 1$ under the formulation (28)–(31), and hence the dual substitution heuristic outputs solution x. Therefore, the ratio of the obtained objective value to the optimal value is $K_1 - 1 \geq K$, which can be arbitrarily large. □

4. Exact Algorithms

The two exact algorithms examined in this section are both rooted from a MIP model of the MMR-GAP. By using Lemma 2.1, and introducing a new continuous variable φ, along with a constraint that forces φ to satisfy $\varphi \leq z_*^s$ $\forall s \in S$, the MMR-GAP can be expressed by the following MIP model (MIP-MMR-GAP):

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^{+} x_{ij} - \varphi \tag{42}$$

$$\text{s.t. } \varphi \leq \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij}^{-} + (c_{ij}^{+} - c_{ij}^{-}) x_{ij}) y_{ij} \qquad \forall y \in X_0 \tag{43}$$

$$x \in X_0. \tag{44}$$

4.1. Benders-Like Decomposition

Benders' decomposition was originally proposed in [36]. Benders-like decompositions are standard techniques that have been frequently used for the exact solution of min-max regret problems [11, 20, 21, 24, 37, 38, 39].

Model (42)–(44) is hard to handle due to the exponential number of constraints (43). Let us define a *master problem* $P(X)$ as the relaxation of the MIP-MMR-GAP in which set X_0 in constraints (43) is replaced by a subset X of X_0:

$$\varphi \leq \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij}^{-} + (c_{ij}^{+} - c_{ij}^{-}) x_{ij}) y_{ij} \qquad \forall y \in X. \tag{45}$$

We name the constraints (45) Benders' cuts. For an optimal solution (x^*, φ^*) to the current master problem $P(X)$, we define a *slave problem* $Q(x^*)$ as:

$$\min_{y \in X_0} \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij}^{-} + (c_{ij}^{+} - c_{ij}^{-}) x_{ij}^{*}) y_{ij}. \tag{46}$$

Let $q(y)$ be the objective value of a solution y and let y^* be an optimal solution to $Q(x^*)$. If $q(y^*) < \varphi^*$ holds, then the specific constraint (43) induced by y^* is violated by the current optimal solution (x^*, φ^*) of $P(X)$, and it is called a Benders' cut. Whenever such a cut is found, the proposed algorithm adds the solution y^* to X, and it solves the updated $P(X)$. The process is iterated until the algorithm finds a solution (x^*, φ^*) for which no violated constraint exists.

Since $P(X)$ is a relaxation of the MIP-MMR-GAP, the optimal solution value at each iteration is a valid lower bound on the optimal solution value of the original MMR-GAP, and hence the final solution, which does not violate any constraint (43), is an optimal solution to the MMR-GAP.

The choice of the Benders' cuts added to X at each iteration can have a strong influence on the overall performance. We start with set $X = \{\widetilde{x}\}$, where $\widetilde{x}$ is the optimal solution obtained by the fixed-scenario heuristic under the median-cost scenario. When set X contains exactly one Benders' cut, the optimal solution of $P(X)$ has the following properties.

Property 4.1. *If $X = \{y\}$ for any $y \in X_0$, then the optimal value of $P(X)$ cannot be positive.*

Proof. In $P(X)$, φ only appears in constraint (43) and in the objective function, where its value has to be maximized. Hence we have

$$\varphi = \sum_{i=1}^{m} \sum_{j=1}^{n} (c^-_{ij} + (c^+_{ij} - c^-_{ij})x_{ij})y_{ij}.$$

Accordingly, $P(X)$ can be written as

$$\min_{x \in X_0} \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} c^+_{ij} x_{ij} - \sum_{i=1}^{m} \sum_{j=1}^{n} (c^-_{ij} + (c^+_{ij} - c^-_{ij})x_{ij})y_{ij} \right\} = \min_{x \in X_0} \left\{ \sum_{(i,j):\, y_{ij}=0} c^+_{ij} x_{ij} + \sum_{(i,j):\, y_{ij}=1} c^-_{ij}(x_{ij} - 1) \right\}. \tag{47}$$

Then, the optimal value cannot be positive, because the objective value becomes zero when $x = y$. □

On the other hand, if set X consists of an optimal solution y to the GAP instance with a fixed scenario, we have the following property.

Property 4.2. *If $X = \{y\}$ for an optimal solution y to the GAP instance obtained by fixing the scenario to any $s \in S$, then the optimal value of $P(X)$ cannot be negative.*

Proof. We prove the thesis by contradiction. Assume that the optimal value is negative, and let x^* be an optimal solution of $P(X)$. Then, from (47), we get

$$\sum_{\substack{(i,j):\, y_{ij}=0, \\ x^*_{ij}=1}} c^+_{ij} < \sum_{\substack{(i,j):\, y_{ij}=1, \\ x^*_{ij}=0}} c^-_{ij}. \tag{48}$$

Since $c^-_{ij} \le c^s_{ij} \le c^+_{ij}$ for any i, j and s, we have

$$\sum_{\substack{(i,j):\, y_{ij}=0, \\ x^*_{ij}=1}} c^s_{ij} < \sum_{\substack{(i,j):\, y_{ij}=1, \\ x^*_{ij}=0}} c^s_{ij} \qquad \forall s \in S. \tag{49}$$

This inequality indicates that the objective value $z^s(x^*)$ of x^* is strictly better than that of y for the GAP instance with scenario s. This contradicts the assumption that y is an optimal solution to this GAP instance, which completes the proof. □

In our Benders-like decomposition approach, we start with a set X consisting of the solution $\widetilde{x}$ obtained by the fixed-scenario heuristic with the median-cost scenario. From Properties 4.1 and 4.2, the optimal value of $P(X)$ is zero when $X = \{\widetilde{x}\}$.

4.2. A Branch-and-Cut Algorithm

Branch-and-cut is another basic approach widely applied as an exact algorithm for interval min-max regret problems [11, 20, 21]. Our second exact algorithm uses Benders' cuts in the context of a basic branch-and-cut framework. We define $\bar{P}(X)$ as the linear relaxation of $P(X)$, and we solve it (with respect to the free variables) at each node of the search tree. Its optimal value is a lower bound for the corresponding partial problem. Since the boundaries of the cost intervals c^- and c^+ for all the instances are integral, we strengthen this bound by rounding it up. If it is not smaller than the incumbent solution value, then we prune the current node. Otherwise, we look for a violated constraint (43) by solving the slave problem $Q(x^*)$ for an optimal solution x^* of the current $\bar{P}(X)$. If such a violation is found, we add an optimal solution y^* for $Q(x^*)$ to the current set X, and then we solve the updated $\bar{P}(X)$. The process continues until no violated constraint (43) exists. When this occurs, if the current optimal solution x^* to $\bar{P}(X)$ is integral, we update the incumbent solution and terminate the current node. If instead it is fractional, a branching follows.

The general framework of branch-and-cut can be implemented in various ways. We maintain the set X for constraints (45) as follows. The search starts with the set X that only contains an optimal solution $\widetilde{x}$ to the GAP instance

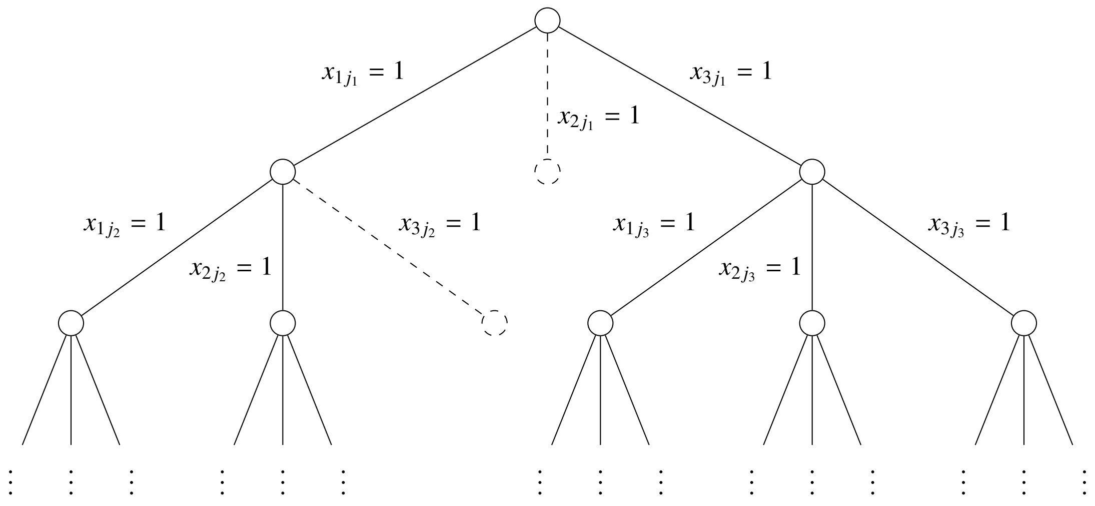

Figure 1: A branching tree for the MMR-GAP with $m = 3$

with the median-cost scenario $\widetilde{s}$. The cuts added to X at each active node are used throughout the entire computation of the branch-and-cut algorithm, i.e., they contribute to all other active nodes.

In the branching step, we branch on the most fractional variable, i.e., the one closest to 0.5, and we adopt a *depth-first search* strategy that chooses, as the next node to search, an active node at the maximum depth in the search tree. This tends to allow the upper bound to be improved quickly in the early phases of the algorithm.

The Benders-like decomposition approach of the previous section cannot provide a feasible solution until it terminates, while the branch-and-cut algorithm usually obtains feasible solutions before reaching optimality. Hence, branch-and-cut algorithms can also serve as heuristics by prematurely halting them.

5. Lagrangian-Based Branch-and-Cut Algorithm

To improve the performance of the basic branch-and-cut algorithm of the previous section, we introduce a Lagrangian-based branch-and-cut algorithm. A similar approach has been used in [21], but we make use of many new techniques tailored for the MMR-GAP. In particular, we propose a stronger lower bound, an efficient variable fixing method, and an effective solution of the Lagrangian subproblems through dynamic programming.

5.1. *Branching Strategy Based on Semi-Assignment Constraints*

In this section, we consider a branching strategy that exploits the structure of semi-assignment constraints (3). In the MMR-GAP, fixing a variable x_{ij} to one requires job j to be assigned to agent i and forbids x_{kj} to take the value one for all agents $k \neq i$. Based on this, we adopted the following branching rule: a partial problem branches to at most m partial problems by fixing the assignment of an unfixed job j to each possible agent $i \in I$, except for those agents i whose remaining capacity is less than a_{ij} or those for which x_{ij} has already been fixed by the variable fixing rules described in Section 5.3. A branching tree for the case with $m = 3$ is illustrated in Figure 1, in which the branches in dashed lines represent the nodes that are not generated due to earlier variable fixing or violation of capacity constraint.

We choose a job j for branching by utilizing the information from the lower bound computation. Let μ_j^* denote the optimal value of the dual variable associated with the semi-assignment constraints (3), which is obtained by solving $\bar{P}(X)$, the linear relaxation of the master problem $P(X)$, with respect to the free variables. Indeed, solving $\bar{P}(X)$ also provides an optimal solution to its dual $\bar{D}(X)$:

$$\max \quad \sum_{j=1}^{n} \mu_j - \sum_{i=1}^{m} b_i \lambda_i - \sum_{y^s \in X} \left(\nu^s \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^- y_{ij}^s \right) \tag{50}$$

$$\text{s.t.} \quad \sum_{y^s \in X} \left(c_{ij}^+ - c_{ij}^- \right) y_{ij}^s \nu^s - a_{ij} \lambda_i + \mu_j \le c_{ij}^+ \qquad \forall i \in I,\ \forall j \in J \tag{51}$$

$$\sum_{y^s \in X} \nu^s = 1 \tag{52}$$

$$\lambda_i \ge 0 \qquad \forall i \in I \tag{53}$$

$$\nu^s \ge 0 \qquad \forall y^s \in X, \tag{54}$$

where λ, μ, and ν are the dual variables associated with the capacity constraints (2), the semi-assignment constraints (3), and the Benders' constraints (43), respectively. For branching, we select the job $j \in J$ with the highest $|\mu_j^*|$ among those whose assignment has not been fixed yet. For this job j, a branch is considered for each agent that has sufficient remaining capacity to receive it, thus creating up to m child nodes. The highest $|\mu_j^*|$ value intuitively indicates that the corresponding semi-assignment constraint is critical, and hence it is expected that the LP lower bound of the current node can be strengthened after such fixing.

It is not hard to see that the resulting branching scheme reduces the number of nodes in the entire enumeration tree from $O(2^{mn})$ of the basic 0-1 branching to $O(m^n)$. (Note however that, in our case, the number of nodes that can actually be generated will not be that different, because the rule to choose the most fractional variable prevents the 0-1 branching from generating those nodes in which the semi-assignment constraints are violated.)

5.2. Lagrangian-Based Lower Bound

In this section we propose an improved lower bound computation based on Lagrangian relaxation. Other studies on min-max regret problems also applied Lagrangian methods (see [40] and [21]). By embedding constraints (3) and (45) in the objective function (42) through Lagrangian multipliers μ_j and ν^s, respectively, we get the following Lagrangian relaxation $L(X, \mu, \nu)$:

$$\min \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^+ x_{ij} - \varphi + \sum_{j=1}^{n} \mu_j \left(1 - \sum_{i=1}^{m} x_{ij} \right) + \sum_{y^s \in X} \nu^s \left(\varphi - \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij}^- + (c_{ij}^+ - c_{ij}^-) x_{ij}) y_{ij}^s \right) \right\} \tag{55}$$

s.t. (2) and (4).

We use the values of μ and ν in an optimal solution to $\bar{D}(X)$ as the Lagrangian multipliers for (55). Then we have $\sum_{y^s \in X} \nu^s = 1$ according to (52), and hence the objective function (55) of $L(X, \mu, \nu)$ can be rewritten as follows:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} \hat{c}_{ij} x_{ij} + \sum_{j=1}^{n} \mu_j - \sum_{y^s \in X} \left(\nu^s \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^- y_{ij}^s \right) \tag{56}$$

where $\hat{c}_{ij} = c_{ij}^+ - \mu_j - \sum_{y^s \in X} \left(c_{ij}^+ - c_{ij}^- \right) y_{ij}^s \nu^s$.

Note that $L(X, \mu, \nu)$ is independent from φ, and its optimal solution can be obtained by solving m 0-1 knapsack problems in the x variables. To solve such knapsack problems, we use a dynamic programming algorithm, introduced in Section 5.4.

It is not hard to show that $L(X, \mu, \nu)$ provides a lower bound at least as good as the LP lower bound of Section 4.2 when we use the values of μ and ν in an optimal solution to the dual $\bar{D}(X)$ of problem $\bar{P}(X)$.

At each active node, we first obtain a lower bound by solving $\bar{P}(X)$. If it is lower than the incumbent solution value, then a round of Benders' cut additions is performed. When no violated constraint (43) exists, we solve $L(X, \mu, \nu)$ by setting μ and ν to the values they have in an optimal solution to $\bar{D}(X)$. If the resulting lower bound is not smaller than the incumbent solution value, the node is fathomed.

5.3. Variable Fixing

An advantage of Lagrangian relaxation is that the obtained information can also be used for variable fixing in an efficient way. Let U be the incumbent solution value and $l^*(X,\mu,\nu)$ the optimal solution value of $L(X,\mu,\nu)$. We denote by Δ, the optimality gap at the current node: $\Delta = U - l^*(X,\mu,\nu)$.

For obtaining $l^*(X,\mu,\nu)$, recall that the Lagrangian relaxation decomposes into m independent 0-1 knapsack problems, one for each agent $i \in I$:

$$L_i(X,\mu,\nu) \quad \min \sum_{j=1}^{n} \hat{c}_{ij} x_{ij} \tag{57}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_{ij} \le b_i \tag{58}$$

$$x_{ij} \in \{0,1\} \qquad \forall j \in J. \tag{59}$$

For a given set of multipliers μ, ν and Benders' cut set X, let $l_i^*(X,\mu,\nu)$ denote the optimal solution value of $L_i(X,\mu,\nu)$. Using (57)–(59), the Lagrangian function (56) can be rewritten as

$$l^*(X,\mu,\nu) = \sum_{j=1}^{n} \mu_j - \sum_{y^s \in X} \left(\nu^s \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^- y_{ij}^s \right) + \sum_{i=1}^{m} l_i^*(X,\mu,\nu). \tag{60}$$

Now, let $\check{x}$ be an optimal solution of $L(X,\mu,\nu)$ and $l_i^*(X,\mu,\nu)_{x_{ij}=1-\check{x}_{ij}}$ be the optimal value of the knapsack problem for agent i in which we force x_{ij} to take value $1-\check{x}_{ij}$. The increase Ξ_{ij} on lower bound $l^*(X,\mu,\nu)$ can be written as follows when we force x_{ij} to take value $1-\check{x}_{ij}$:

$$\Xi_{ij} = \begin{cases} l_i^*(X,\mu,\nu)_{x_{ij}=0} - l_i^*(X,\mu,\nu) & \text{if } \check{x}_{ij} = 1 \\ l_i^*(X,\mu,\nu)_{x_{ij}=1} - l_i^*(X,\mu,\nu) & \text{if } \check{x}_{ij} = 0 \end{cases} \qquad \forall i \in I,\ \forall j \in J. \tag{61}$$

With the definitions above, we can consider many variable fixing rules:

- Rule 1: If $\check{x}_{ij} = 0$ and $\Xi_{ij} \ge \Delta$, then x_{ij} can be fixed to 0.
- Rule 2: If $\check{x}_{ij} = 1$ and $\Xi_{ij} \ge \Delta$, then x_{ij} can be fixed to 1 and x_{kj} to 0 for all $k \ne i$.
- Rule 3: This is a rule strengthened from Rule 1. Suppose that $\check{x}_{ij} = 0$. If x_{ij} took the value 1 in a feasible solution, then all other x_{kj} would have to take the value 0; therefore if $\Xi_{ij} + \sum_{k:\,\check{x}_{kj}=1} \Xi_{kj} \ge \Delta$, then x_{ij} can be fixed to 0.
- Rule 4: Suppose that $\check{x}_{ij} = 1$ and $\check{x}_{kj} = 0$ for all $k \ne i$. If x_{ij} took the value 0 in a feasible solution, then some x_{kj} with $k \ne i$ would have to take value 1; therefore if $\Xi_{ij} + \min\left\{\Xi_{kj} \mid k \ne i\right\} \ge \Delta$, then x_{ij} can be fixed to 1 and x_{kj} can be fixed to 0 for all $k \ne i$.
- Rule 5: If x_{kj} is fixed to 0 for all $k \ne i$ then x_{ij} must be fixed to 1. Moreover, if $\check{x}_{ij} = 0$ and $\Xi_{ij} \ge \Delta$, then the current node can be pruned.
- Rule 6: Suppose that $\sum_{i=1}^{m} \check{x}_{ij} = 0$. At least one unfixed variable x_{ij} for some $i \in I$ must take the value 1; therefore if $\min\left\{\Xi_{ij} \mid i \in I\right\} \ge \Delta$, then the current node can be pruned.
- Rule 7: If x_{ij} is fixed to 0 for all $i \in I$ then the current node can be pruned.
- Rule 8: Suppose that $\check{x}_{ij} = 1$ for more than one $i \in I$. As only one of them can take the value 1, if $\sum_{k:\,\check{x}_{kj}=1} \Xi_{kj} - \max\left\{\Xi_{kj} \mid \check{x}_{kj} = 1, k \in I\right\} \ge \Delta$, then the current node can be pruned.

Rules 1–7 have been proposed by Posta et al. [41] for the classical GAP. Preliminary computational experiments confirmed that they are also effective for the MMR-GAP, and hence they were incorporated in our variable fixing step.

5.4. Dynamic Programming Approach

As all input data are integers, the knapsack problems (57)–(59), which are needed to compute $l^*(X, \mu, \nu)$ as in (60), can be solved through the following two-direction dynamic programming approach, which was first proposed for the classical GAP by Posta et al. [41].

For each agent i, we introduce a quantity $f_i(j, k)$ for $j = 0, 1, \ldots, n$ and $k = 0, 1, \ldots, b_i$, where j is the number of jobs and k is an amount of resource. The value $f_i(j, k)$ gives the minimum cost when only jobs from 1 to j are available, and the resource limit is k instead of b_i in (2). We can compute $f_i(j, k)$ through the classical dynamic programming recursion

$$f_i(j,k) = \begin{cases} 0 & \text{if } \ j = 0; \\ f_i(j-1,k) & \text{if } \ j \geq 1 \text{ and } k < a_{ij}; \\ \min\left\{f_i(j-1,k), f_i(j-1,k-a_{ij}) + \hat{c}_{ij}\right\} & \text{otherwise.} \end{cases}$$

The computation can be implemented using an $(n + 1) \times (b_i + 1)$ array whose (j, k)-element contains the value of $f_i(j, k)$, and computing their values by increasing j. The value of $f_i(n, b_i)$ gives the optimal value $l_i^*(X, \mu, \nu)$. We call this dynamic programing a *head-to-tail* approach.

In order to efficiently determine on variable fixing, we additionally use a *tail-to-head* approach for each knapsack problem. For each agent i, we define $g_i(j, k)$ as the minimum cost when we are only allowed to use jobs from j to n, and we have a resource restriction of $b_i - k$. Values $g_i(j, k)$ are computed in a symmetric way, by decreasing j:

$$g_i(j,k) = \begin{cases} 0 & \text{if } \ j = n+1; \\ g_i(j+1,k) & \text{if} j \leq n \text{ and } k > b_i - a_{ij}; \\ \min\left\{g_i(j+1,k), g_i(j+1,k+a_{ij}) + \hat{c}_{ij}\right\} & \text{otherwise.} \end{cases}$$

Next, we show how this DP approach also provides $l_i^*(X, \mu, \nu)_{x_{ij}=1-\check{x}_{ij}}$ from the two DP tables, i.e., the table of $f_i(j, k)$ and that of $g_i(j, k)$. We compute $l_i^*(X, \mu, \nu)_{x_{ij}=1-\check{x}_{ij}}$ by considering the combination of two partial knapsack problems for each k: a knapsack problem on jobs from 1 to $j - 1$ with capacity k and another one on jobs from $j + 1$ to n with capacity $b_i - a_{ij}(1 - \check{x}_{ij}) - k$. Formally, when $n \geq 2$, $l_i^*(X, \mu, \nu)_{x_{ij}=1-\check{x}_{ij}}$ can be obtained by

$$l_i^*(X,\mu,\nu)_{x_{ij}=0} = \min_{0 \leq k \leq b_i} \begin{cases} g_i(j+1,k) & \text{if } \ j = 1; \\ f_i(j-1,k) & \text{if } \ j = n; \\ f_i(j-1,k) + g_i(j+1,k) & \text{otherwise} \end{cases} \tag{62}$$

or

$$l_i^*(X,\mu,\nu)_{x_{ij}=1} = \min_{0 \leq k \leq b_i - a_{ij}} \begin{cases} \hat{c}_{ij} + g_i(j+1,k+a_{ij}) & \text{if } \ j = 1; \\ \hat{c}_{ij} + f_i(j-1,k) & \text{if } \ j = n; \\ \hat{c}_{ij} + f_i(j-1,k) + g_i(j+1,k+a_{ij}) & \text{otherwise.} \end{cases} \tag{63}$$

The computation of (62) and (63) can be done in $O(b_i)$ time for each pair (i, j), and hence, this two-direction dynamic programming approach has time complexity $O(nb_i)$ for each i, i.e., $O(n \sum_{i=1}^m b_i)$ in total to compute Ξ_{ij} for all i and j. Figure 2 illustrates the case in which we fix x_{ij} to 1 and the value of a_{ij} is 1. The upper and lower parts represent the tables for $f_i(j, k)$ and $g_i(j, k)$, and the lines in the jth column connect the two elements in the table corresponding to the two terms in the last line of (63). Then, the value of $l_i^*(X, \mu, \nu)_{x_{ij}=1}$ can be obtained by searching for the minimum among the sums of the values at the two ends.

5.5. Benders' Cuts Management

In our Lagrangian-based branch-and-cut approach, it takes only polynomial or pseudo-polynomial time to solve the LP relaxation $\bar{P}(X)$ or the Lagrangian relaxation problems, while the slave problem $Q(\cdot)$ is NP-hard in the strong sense. We handle such difficulty by reducing the computation time needed to solve each $Q(\cdot)$, and by limiting the number of times we solve it. We consider the following methods.

	0	1	$\cdots$	$\cdots$	$b_i - 1$	b_i
1	$f(1,0)$	$f(1,1)$	$\cdots$	$\cdots$	$f(1,b_i-1)$	$f(1,b_i)$
2	$f(2,0)$	$f(2,1)$	$\cdots$	$\cdots$	$f(2,b_i-1)$	$f(2,b_i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$j-1$	$f(j-1,0)$	$f(j-1,1)$	$\cdots$	$\cdots$	$f(j-1,b_i-1)$	$f(j-1,b_i)$
j	↖↘	↖↘		↘	↖↘	
$j+1$	$g(j+1,0)$	$g(j+1,1)$	$\cdots$	$\cdots$	$g(j+1,b_i-1)$	$g(j+1,b_i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	$g(n-1,0)$	$g(n-1,1)$	$\cdots$	$\cdots$	$g(n-1,b_i-1)$	$g(n-1,b_i)$
n	$g(n,0)$	$g(n,1)$	$\cdots$	$\cdots$	$g(n,b_i-1)$	$g(n,b_i)$

Figure 2: Two-direction dynamic programing for fixing $x_{ij} = 1$, where $a_{ij} = 1$

Full-cut
This approach generates, at every node, as many Benders' cuts as possible, and adds them to X.
Cut-and-branch
At the root node we repeat the process of adding Benders' cuts to X until no such cut can be found. For the other nodes, we solve the LP relaxation $\bar{P}(X)$ first and continue our Benders' cut generation only if in the optimal solution (x^*, φ^*) to $\bar{P}(X)$, x^* is integral. Whenever a new Benders' cut is found, we add it to X and we solve the updated $\bar{P}(X)$ again. This process is repeated until the obtained x^* becomes fractional or we find a solution (x^*, φ^*) for which no constraint is violated. Note that we can terminate an active node if we find an optimal solution (x^*, φ^*) to $\bar{P}(X)$ such that x^* is integral and no violated constraint exists, because such a solution is also optimal to problem $P(X_0)$ (with respect to the free variables at the node). The rule for non-root nodes is adopted for this reason. In this approach, the number of solved $Q(\cdot)$ is drastically reduced with respect to the *full-cut* approach.
Adaptive generation
We now introduce two techniques to improve the performance of the cut-and-branch method. The cut-and-branch strategy concentrates on lowering the number of times cuts are generated, which is shown to be more effective than the full-cut approach by the computational experiments of Section 6. However, this comes at the expense of sacrificing the quality of lower bounds due to the lack of effective Benders' cuts. That is, there is a trade-off between the quality of lower bounds and the time to generate Benders' cuts. A simple idea to reduce the time for generating Benders' cuts without sacrificing the quality of the lower bound (i.e., without reducing the number of Benders' cuts) is to shorten the time for solving $Q(\cdot)$ by using heuristic approaches instead of exact ones. Note that even if we use heuristics to solve $Q(\cdot)$, the lower bounds are valid and hence the resulting branch-and-cut algorithm remains exact. It should also be noted, however, that when integral solutions of $\bar{P}(X)$ are obtained, they are not necessarily feasible with respect to the MIP-MMR-GAP (42)–(44) unless $Q(\cdot)$ is solved to optimality. Thus, we solve $Q(\cdot)$ exactly only if integral solutions of $\bar{P}(X)$ are obtained, and use heuristics otherwise. We tested two heuristics, originally developed for the classical GAP: an ejection chain approach [42] and a path relinking approach with ejection chains [43], with a limit T (a parameter) on the number of calls to EC probe, which is the basic local search component of these heuristic algorithms. If no violated Benders' constraint is found within T iterations, a branching follows. On the basis of preliminary computational experiments, we adopted the ejection chain approach.

The second improvement is the method we use for adding Benders' cuts at non-root nodes. For this approach, we impose an upper limit on the total number of Benders' cuts generated at non-root nodes, and we skip the cut generation once this number was been reached (except for the case where x^* is integral). The upper limit was set to W times the number of Benders' cuts generated at the root (W a parameter). Moreover, for each non-root node, we stop the Benders' cut generation process when the ratio $\alpha = \frac{\varphi^* - q(y^*)}{q(y^*)}$ becomes smaller than or equal to a parameter Υ (except for the case where x^* is integral), where (x^*, φ^*) is an optimal solution to $\bar{P}(X)$ and y^* is a (not necessarily optimal) solution to the slave problem $Q(x^*)$ defined by (46). Preliminary computational experiments showed that high increments in the lower bound mostly occur at early iterations, when $\alpha > 2\%$, while most cuts with $\alpha \leq 1\%$ only produce a tiny increase of the node lower bound. This suggests that it is possible to save computational effort by halting the generation of Benders' cuts when α becomes small. It turned out that the adaptive generation of Benders' cuts performs well with parameter setting $T = 100$, $W = 8$, and $\Upsilon = 1\%$.

6. Computational Experiments

6.1. Instance Generation

To the best of our knowledge, this is the first research on the MMR-GAP. In order to test our approaches, we generated MMR-GAP instances from the following well-known (see [44] and [45]) GAP benchmark instances known as Types A, B, C, D, and E (u.d. stands for "uniformly distributed"):

- **A:** $\forall i, j$, a_{ij} is a random integer u.d. in $[5, 25]$, c_{ij} is a random integer u.d. in $[10, 50]$; $b_i = 0.6(n/m)15 + 0.4\gamma$, where $\gamma = \max_{i \in I} \sum_{j \in J, \theta_j = i} a_{ij}$ and $\theta_j = \min\{i \mid c_{ij} \leq c_{kj}, \forall k \in I\}$.
- **B:** a_{ij} and c_{ij} as for Type A; b_i is set to 70% of the value in Type A.
- **C:** a_{ij} and c_{ij} as for Type A; $b_i = 0.8 \sum_{j=1}^{n} a_{ij}/m$.
- **D:** $\forall i, j$, a_{ij} is a random integer u.d. in $[1, 100]$ and $c_{ij} = 111 - a_{ij} + e_1$ (e_1 a random integer u.d. in $[-10, 10]$); $b_i = 0.8 \sum_{j=1}^{n} a_{ij}/m$.
- **E:** $\forall i, j$, $a_{ij} = 1 - 10 \ln e_2$ (e_2 a random number u.d. in $(0, 1]$), $c_{ij} = 1000/a_{ij} - 10e_3$ (e_3 a random number u.d. in $[0, 1]$); $b_i = 0.8 \sum_{j=1}^{n} a_{ij}/m$.

For each type, we generated GAP instances with $m \in \{5, 10\}$ and $n \in \{40, 80\}$. We then obtained MMR-GAP instances by setting the extremes of the cost intervals, c_{ij}^- and c_{ij}^+, to random integers u.d. in $[(1-\delta)c_{ij}, c_{ij}]$ and $[c_{ij}, (1+\delta)c_{ij}]$, respectively, with $\delta \in \{0.10, 0.25, 0.50\}$. We randomly generated 5 instances for each δ, thus obtaining 15 MMR-GAP instances by using each original GAP instance as a seed. As a result, we generated 300 MMR-GAP instances in total, which are available at `http://www.co.mi.i.nagoya-u.ac.jp/~goi/mmr-gap/`.

6.2. Implementation Details

The heuristic algorithms proposed for the GAP in [42] and [43] were coded in C, while the algorithms proposed in this paper were all coded in C++. We used IBM ILOG CPLEX, version 12.4, for solving linear programming and mixed integer linear programming problems. The CPLEX solver was also used to exactly solve the classical GAP, the Benders' master problems $P(X)$ and $\bar{P}(X)$, and problem D-MMR-GAP for the dual substitution heuristic. All experiments were carried out on a PC with two cores i7-5820K at 3.30 GHz and 32 GB RAM memory, where the computation was always conducted on a single core.

6.3. Heuristic Algorithms

We compared two heuristic algorithms, the fixed-scenario algorithm of Section 3.1 and the dual substitution heuristic of Section 3.2.

Table 1 shows the upper bounds computed by these algorithms for all instances. The GAP instances are denoted as $Xyyzz$, where X = type, $yy = m$ and $zz = n$. For each GAP instance, we give the results for the three values of δ. For every 5 random instances under the same δ, the entries give the average CPU time in seconds spent by the algorithm ("time"), the average optimality gap ("gap"), and the total number of failures ("#f"), i.e., the total number of instances not solved exactly within the time limit. Both the average CPU times and the average optimality gaps were taken for those instances whose optimal solutions were found within the time limit. The optimality gap of solution x is the quantity $(r_{\max}(x) - \mathrm{OPT})/\mathrm{OPT}$, where OPT is the optimal value obtained by the exact algorithms.

We used CPLEX to exactly solve the classical GAP for evaluating the maximum regret of the obtained solutions through Lemma 2.1. A time limit of 3600 seconds was assigned to each algorithm. Notation "t.l." indicates that the algorithm was not able to exactly solve even one of the five instances within the time and memory limit. Columns "DS" refer to the dual substitution algorithm. An empty entry indicates that no optimal value was available for calculating the optimality gap.

Among the fixed-scenario approaches, the median-cost scenario $(c^+ + c^-)/2$ obtains the best solutions for most of the instances. The dual substitution algorithm provides better solutions with objective values very close to the optimal values. However, for Types D and E, the computation of the dual substitution algorithm becomes very expensive. One possible explanation for the hardness of achieving optimality is that the number of mixed integer constraints (26) increases with the number of jobs n. On the other hand, we also observed that the increase in the value of δ did not change the size of the problem, while the mixed integer constraints (26) with a wide range of interval caused more consuming time for a branch-and-cut based solver.

The average gaps from the known optimal values for Types A, B, C, and E are reported in Table 2. The dual substitution heuristic obtained better upper bounds than the fixed-scenario heuristic for the instances of all types.

A reasonable conclusion can be drawn from these results. A decision maker that aims to find a near-optimal solution to the MMR-GAP could first adopt the fixed-scenario approach under the median-cost scenario. The advantage of the fixed-scenario approach is that it is fast and easy to implement. If he/she has sufficient computation time, at the second attempt, he/she could invoke the dual substitution heuristic, which may provide a better solution.

6.4. *Exact Algorithms*

We tested the exact algorithms of Section 4 on all instances. Recall that, for the Benders-like decomposition, when it is halted with a time limit, the objective value of the obtained solution is a valid lower bound, although it is usually an infeasible solution (see Section 4.1). On the other hand, the solution value obtained by the branch-and-cut framework provides a valid upper bound on the optimal regret when it terminates with a time limit. Each pair of entries in Table 3 shows the average CPU time in seconds spent by the algorithm (computed for those instances whose optimal solutions were found within the time limit), the average optimality gap from the optimal value or from a lower bound (taken for those instances whose optimal solutions or valid lower bounds were found within the time limit), and the number of instances where an optimal solution was found (in column "#opt") within the time limit. We calculated the optimality gap of solution x in the same way as described in Section 6.3, where for those instances whose optimal values are not known, the valid lower bounds obtained by Benders-like decomposition are used instead. An instance was not included into the average calculation if no feasible solution was obtained by the corresponding algorithm. We use B&C and C&B to denote branch-and-cut and cut-and-branch, respectively. "Benders" is the Benders-like decomposition approach of Section 4.1, while "basic B&C" is the basic branch-and-cut algorithm discussed in Section 4.2. Columns "Lagrangian-based B&C" refer to the approach proposed in Section 5: "full-cut," "C&B" and "adaptive" represent the three cut management approaches discussed in Section 5.5. A time limit of 3600 seconds was assigned to each algorithm. For the Lagrangian-based branch-and-cut algorithms, the time limit for solving the D-MMR-GAP to obtain an initial upper bound was set to 300 seconds. Notation "t.l." has the same meaning as in Table 1.

Table 3 shows that the Lagrangian-based branch-and-cut algorithms dominated the basic branch-and-cut algorithm for instances of Types B, C, and E, with few exceptions for instances of Type A.

With respect to Benders-like decomposition, the Lagrangian-based branch-and-cut algorithm with adaptive cut generation has on average shorter CPU times for Types A and E, while for Types B and C there is no clear winner. For Type D (the most difficult one), we omit the results since none of the algorithms was able to obtain optimal solutions, and there is a large gap between the lower bound obtained by the Benders-like decomposition approach and the upper bound obtained by the branch-and-cut algorithms.

Table 4 provides the overall results for the introduced exact algorithms. For each algorithm and instance type, the entries provide the number of instances for which the algorithm determined the optimal solution with the least computation time among the three methods. The value in parentheses gives the number of instances solved to optimality within one hour. The Lagrangian-based branch-and-cut algorithm with adaptive cut generation had the best results for the instances of Types A and E. In total, it solved to optimality 102 out of 240 instances of Types A, B, C, and E, in 64 cases with the lowest CPU time.

These results indicate that the branch-and-cut approach using Benders cut has a better performance than the simple cutting plane method (Benders-like decomposition). One of the important future research directions could be to consider improvement on the classical Benders-like decomposition. For the MMR-GAP, the Lagrangian-based branch-and-cut algorithm that takes advantage of the features of the GAP dominates the classical Benders-like decomposition and branch-and-cut approach. It indicates that when considering other combinatorial optimization problems under the min-max regret criterion, a problem-specialized algorithm based on two general frameworks may improve the overall CPU time.

Table 1: Results of heuristic algorithms

		Fixed Scenario											
		c^-			$(c^+ + c^-)/2$			c^+			DS		
instance	δ	time	#f	gap	time	#f	gap	time	#f	gap	time	#f	gap
A0540	0.10	0.03	0	4.3%	0.08	0	1.5%	0.01	0	7.6%	0.06	0	0.0%
	0.25	0.02	0	15.2%	0.22	0	1.4%	0.01	0	3.6%	0.50	0	0.7%
	0.50	0.02	0		0.05	0		0.01	0		0.60	0	
A0580	0.10	0.03	0	16.3%	0.02	0	1.7%	0.03	0	8.0%	0.19	0	1.0%
	0.25	0.01	0		0.01	0		0.01	0		0.67	0	
	0.50	0.02	0		0.01	0		0.03	0		6.45	0	
A1040	0.10	0.02	0	17.5%	0.02	0	4.9%	0.02	0	19.4%	0.21	0	0.0%
	0.25	0.03	0	9.6%	0.03	0	1.7%	0.04	0	9.9%	0.42	0	0.5%
	0.50	0.05	0		0.03	0		0.01	0		1.17	0	
A1080	0.10	0.04	0	15.6%	0.03	0	1.1%	0.02	0	6.7%	0.68	0	0.0%
	0.25	0.02	0		0.03	0		0.01	0		1.43	0	
	0.50	0.05	0		0.02	0		0.03	0		15.83	0	
B0540	0.10	0.31	0	4.6%	0.15	0	9.7%	0.22	0	12.7%	0.42	0	8.0%
	0.25	0.18	0	11.6%	0.34	0	3.1%	0.16	0	9.9%	1.05	0	0.6%
	0.50	0.37	0	16.9%	0.20	0	1.2%	0.16	0	3.8%	1.89	0	0.0%
B0580	0.10	0.55	0	15.2%	0.41	0	2.1%	0.41	0	1.1%	2.70	0	0.0%
	0.25	0.57	0		0.90	0		0.28	0		19.35	0	
	0.50	0.57	0		0.35	0		0.73	0		80.03	0	
B1040	0.10	0.34	0	11.6%	0.16	0	0.9%	0.12	0	2.5%	0.52	0	6.7%
	0.25	0.22	0	5.6%	0.24	0	3.0%	0.17	0	11.3%	4.09	0	3.5%
	0.50	0.23	0	24.4%	0.19	0	4.5%	0.14	0	8.7%	14.07	0	0.0%
B1080	0.10	0.90	0	18.0%	0.98	0	5.8%	0.58	0	13.2%	6.31	0	2.2%
	0.25	0.63	0		0.56	0		0.45	0		168.19	0	
	0.50	1.16	0		0.78	0		0.87	0		2052.81	1	
C0540	0.10	0.40	0	4.0%	0.26	0	0.0%	0.27	0	6.7%	0.55	0	0.0%
	0.25	0.48	0	11.0%	0.22	0	1.7%	0.38	0	6.4%	1.26	0	0.0%
	0.50	0.38	0	12.8%	0.45	0	3.2%	0.24	0	4.2%	5.82	0	0.0%
C0580	0.10	0.42	0	11.2%	0.69	0	0.0%	0.58	0	7.1%	3.11	0	0.0%
	0.25	0.74	0		0.70	0		0.38	0		13.24	0	
	0.50	0.51	0		0.44	0		0.53	0		677.74	0	
C1040	0.10	0.44	0	8.3%	0.47	0	7.1%	0.28	0	7.4%	1.08	0	3.8%
	0.25	0.40	0	16.6%	0.31	0	5.7%	0.24	0	7.0%	1.68	0	2.1%
	0.50	0.85	0	21.4%	0.28	0	0.0%	0.40	0	0.0%	14.00	0	0.0%
C1080	0.10	2.09	0	20.0%	0.88	0	3.5%	0.63	0	7.5%	12.38	0	3.9%
	0.25	0.93	0		1.48	0		1.00	0		75.95	0	
	0.50	1.66	0		1.24	0		0.70	0		1033.34	1	
D0540	0.10	10.83	0		11.89	0		21.71	0		121.95	0	
	0.25	7.77	0		4.89	0		7.53	0		931.42	0	
	0.50	2.46	0		6.74	0		9.30	0		574.27	0	
D0580	0.10	169.32	0		166.96	0		89.29	0		t.l.	5	
	0.25	98.41	0		178.03	0		214.99	0		t.l.	5	
	0.50	26.71	0		136.86	0		60.81	0		t.l.	5	
D1040	0.10	26.89	0		37.98	0		37.59	0		518.40	0	
	0.25	6.94	0		58.14	0		19.59	0		1301.18	0	
	0.50	3.14	0		16.02	0		4.09	0		387.35	0	
D1080	0.10	1652.03	4		739.86	3		t.l.	5		t.l.	5	
	0.25	1422.67	0		t.l.	5		1927.13	4		t.l.	5	
	0.50	260.87	0		849.46	3		1668.17	3		t.l.	5	
E0540	0.10	0.64	0	9.5%	0.51	0	2.5%	0.48	0	5.5%	3.05	0	0.4%
	0.25	0.26	0	10.8%	0.68	0	2.5%	0.61	0	4.6%	16.31	0	0.0%
	0.50	0.63	0		0.68	0		0.44	0		47.27	0	
E0580	0.10	1.06	0	9.9%	1.73	0	2.6%	0.99	0	4.3%	94.13	0	0.0%
	0.25	1.84	0		1.02	0		1.18	0		246.46	0	
	0.50	0.95	0		1.59	0		1.09	0		502.27	1	
E1040	0.10	4.03	0	10.6%	2.98	0	5.6%	3.31	0	11.6%	64.45	0	9.1%
	0.25	2.46	0		2.25	0		2.62	0		436.48	0	
	0.50	2.65	0		2.44	0		1.65	0		1185.87	0	
E1080	0.10	8.72	0		7.11	0		5.43	0		1477.02	4	
	0.25	8.76	0		5.14	0		7.70	0			5	
	0.50	3.69	0		9.87	0		8.57	0			5	

Table 2: Average optimality gap of heuristic algorithms

	Fixed-Scenario			
Type	c^-	$(c^+ + c^-)/2$	c^+	DS
A	13.0%	2.1%	9.3%	0.4%
B	12.5%	3.7%	7.7%	2.9%
C	12.5%	2.9%	6.3%	1.4%
E	10.2%	3.2%	6.4%	2.1%

The results also show that a decision maker could consider attacking through an exact algorithm a real-world problem that can be modeled as an MMR-GAP, provided the size of the instance is not very large. However, large-size instances, or instances with large uncertainty, should be preferably handled through heuristic algorithms.

7. Conclusions

The GAP has a number of real world managerial implications in production planning, scheduling, timetabling, transportation, etc. We studied the min-max regret GAP, a robust version of the GAP. Problems of this class are much more difficult than NP-hard problems; indeed it is considered very unlikely that they even belong to NP. We presented and compared heuristic and exact methods.

We computationally examined a fixed-scenario heuristic, for which three scenarios were considered. We also presented a dual substitution heuristic that uses a mixed integer programming formulation obtained by replacing a subproblem with its dual. We observed that this heuristic method in most cases provides better upper bounds than the fixed-scenario heuristic.

We also examined three exact methods: a Benders-like decomposition approach, a basic branch-and-cut algorithm, and a Lagrangian-based branch-and-cut algorithm that incorporates several ideas. We used a Lagrangian lower bound that is stronger than the LP lower bound and is obtained by solving m 0-1 knapsack problems through dynamic programming. This computation was further exploited for variable fixing. Compared with Benders-like decomposition, the Lagrangian-based branch-and-cut had better performance for instances of Types A and E, and it exactly solved 102 out of 240 instances of Types A, B, C, and E with m up to 10 and n up to 80.

Snyder and Daskin [2] proposed, for two classical facility location problems, a hybrid stochastic and robust optimization approach (*p-robust approach*). Future research could consider the development of a similar approach for the MMR-GAP.

Acknowledgements

This work was partially supported by JSPS KAKENHI [Grant Nos. 15K12460, 15H02969], by the Hori Sciences and Arts Foundation, and by CAPES/Brazil under grant PVE no. A007/2013, and by MIUR (Italy) under grant PRIN 2015. We are grateful to the referees of this paper for many useful suggestions.

References

[1] J. M. Buhmann, A. Y. Gronskiy, M. Mihalák, T. Pröger, R. Šrámek, P. Widmayer, Robust optimization in the presence of uncertainty: A generic approach, Journal of Computer and System Sciences 94 (2018) 135–166.

[2] L. V. Snyder, M. S. Daskin, Stochastic p-robust location problems, IIE Transactions 38 (11) (2006) 971–985.

[3] M. Poss, Robust combinatorial optimization with knapsack uncertainty, Discrete Optimization 27 (2018) 88–102.

[4] D. Bertsimas, M. Sim, The price of robustness, Computers & Operations Research 52 (2004) 35–53.

[5] N. H. Lappas, C. E. Gounaris, Robust optimization for decision-making under endogenous uncertainty, Computers & Chemical Engineering 111 (2018) 252–266.

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

[7] A. Raith, M. Schmidt, A. Schöbel, L. Thom, Multi-objective minmax robust combinatorial optimization with cardinality-constrained uncertainty, European Journal of Operational Research 267 (2) (2018) 628–642.

Table 3: Results of exact algorithms

							Lagrangian-based B&C								
		Benders		basic B&C			full-cut			C&B			adaptive		
instance	δ	time	#opt	time	#opt	gap	time	#opt	gap	time	#opt	gap	time	#opt	gap
A0540	0.10	1.15	5	1.14	5	0.0%	2.28	5	0.0%	0.78	5	0.0%	0.32	5	0.0%
	0.25	484.01	5	59.38	5	0.0%	420.10	4	0.5%	86.73	5	0.0%	50.83	5	0.0%
	0.50	t.l.	0	t.l.	0	11.3%	t.l.	0	9.3%	t.l.	0	9.3%	t.l.	0	9.3%
A0580	0.10	205.50	5	10.79	5	0.0%	114.57	5	0.0%	35.87	5	0.0%	16.65	5	0.0%
	0.25	t.l.	0	t.l.	0	20.1%	t.l.	0	11.6%	t.l.	0	11.6%	t.l.	0	11.6%
	0.50	t.l.	0	t.l.	0	58.4%	t.l.	0	35.8%	t.l.	0	35.8%	t.l.	0	35.8%
A1040	0.10	2.83	5	3.31	5	0.0%	12.29	5	0.0%	4.99	5	0.0%	1.04	5	0.0%
	0.25	1668.20	3	1192.33	4	0.3%	196.80	2	0.5%	1122.51	4	0.0%	926.58	5	0.0%
	0.50	t.l.	0	t.l.	0	18.8%	t.l.	0	11.9%	t.l.	0	11.9%	t.l.	0	11.9%
A1080	0.10	1523.17	3	1379.73	4	1.7%	184.69	3	1.7%	1531.68	3	1.7%	1470.31	3	1.7%
	0.25	t.l.	0	t.l.	0	31.3%	t.l.	0	17.0%	t.l.	0	17.0%	t.l.	0	17.0%
	0.50	t.l.	0	t.l.	0	71.2%	t.l.	0	41.5%	t.l.	0	41.5%	t.l.	0	41.5%
B0540	0.10	1.83	5	29.58	5	0.0%	213.82	5	0.0%	5.53	5	0.0%	6.47	5	0.0%
	0.25	49.03	5	167.13	5	0.0%	1580.94	4	0.0%	47.48	5	0.0%	13.95	5	0.0%
	0.50	2169.11	4	2152.68	3	4.3%	757.16	1	0.0%	1597.97	4	0.0%	804.76	5	0.0%
B0580	0.10	131.46	5	946.69	4	0.0%	1286.12	2	0.0%	212.03	5	0.0%	80.60	5	0.0%
	0.25	t.l.	0	t.l.	0	40.9%	t.l.	0	14.8%	t.l.	0	14.8%	t.l.	0	14.8%
	0.50	t.l.	0	t.l.	0	81.0%	t.l.	0	40.5%	t.l.	0	40.5%	t.l.	0	40.5%
B1040	0.10	6.95	5	428.18	5	0.0%	228.66	4	0.0%	15.42	5	0.0%	9.66	5	0.0%
	0.25	825.52	4	1653.36	3	4.0%	t.l.	0	0.9%	868.61	4	0.6%	822.07	4	0.6%
	0.50	3562.28	1	t.l.	0	14.4%	t.l.	0	4.0%	t.l.	0	4.0%	t.l.	0	4.0%
B1080	0.10	463.96	5	2688.55	2	242.4%	696.53	1	1.8%	1706.35	3	1.0%	1597.24	3	1.0%
	0.25	t.l.	0	t.l.	0	93.3%	t.l.	0	19.8%	t.l.	0	19.8%	t.l.	0	19.8%
	0.50	t.l.	0	t.l.	0		0.00	0	45.6%	t.l.	0	45.6%	t.l.	0	45.6%
C0540	0.10	2.59	5	20.02	5	0.0%	153.92	5	0.0%	6.36	5	0.0%	5.43	5	0.0%
	0.25	63.28	5	217.78	5	0.0%	1010.41	3	0.0%	64.89	5	0.0%	22.58	5	0.0%
	0.50	2538.39	2	2411.97	2	4.2%	t.l.	0	2.0%	2173.81	3	2.0%	1913.15	4	2.0%
C0580	0.10	95.58	5	429.68	5	0.0%	928.19	1	0.0%	205.20	5	0.0%	75.58	5	0.0%
	0.25	t.l.	0	t.l.	0	33.5%	t.l.	0	12.7%	t.l.	0	12.7%	t.l.	0	12.7%
	0.50	t.l.	0	t.l.	0	101.5%	t.l.	0	45.0%	t.l.	0	45.0%	t.l.	0	45.0%
C1040	0.10	9.62	5	1072.70	5	0.0%	1111.19	5	0.0%	13.76	5	0.0%	14.67	5	0.0%
	0.25	86.96	5	2992.29	1	11.6%	t.l.	0	1.4%	174.79	5	0.0%	136.73	5	0.0%
	0.50	2883.70	2	t.l.	0	44.1%	t.l.	0	4.5%	t.l.	0	4.5%	t.l.	0	4.5%
C1080	0.10	711.59	5	t.l.	0	451.2%	t.l.	0	2.1%	1721.21	3	1.5%	1664.48	3	1.5%
	0.25	t.l.	0	t.l.	0	203.5%	t.l.	0	16.7%	t.l.	0	16.7%	t.l.	0	16.7%
	0.50	t.l.	0	t.l.	0		t.l.	0	44.4%	t.l.	0	44.4%	t.l.	0	44.4%
E0540	0.10	54.86	5	386.90	5	0.0%	1712.01	3	0.4%	49.49	5	0.0%	19.50	5	0.0%
	0.25	t.l.	0	t.l.	0	10.9%	t.l.	0	0.0%	2507.06	5	0.0%	2706.21	3	0.0%
	0.50	t.l.	0	t.l.	0	38.7%	t.l.	0	22.6%	t.l.	0	22.6%	t.l.	0	22.6%
E0580	0.10	t.l.	0	t.l.	0	74.2%	t.l.	0	22.6%	3281.98	1	22.6%	3121.63	1	22.6%
	0.25	t.l.	0	t.l.	0	115.4%	t.l.	0	53.7%	t.l.	0	53.7%	t.l.	0	53.7%
	0.50	t.l.	0	t.l.	0	126.6%	t.l.	0	63.5%	t.l.	0	63.6%	t.l.	0	63.6%
E1040	0.10	2110.68	3	t.l.	0	125.2%	t.l.	0	5.7%	2248.42	2	4.6%	3399.44	1	5.0%
	0.25	t.l.	0	t.l.	0	102.7%	t.l.	0	18.6%	t.l.	0	18.6%	t.l.	0	18.6%
	0.50	t.l.	0	t.l.	0	84.0%	t.l.	0	27.0%	t.l.	0	27.0%	t.l.	0	27.0%
E1080	0.10	t.l.	0	t.l.	0		t.l.	0	47.0%	t.l.	0	47.3%	t.l.	0	47.0%
	0.25	t.l.	0	t.l.	0		t.l.	0	61.5%	t.l.	0	61.7%	t.l.	0	61.9%
	0.50	t.l.	0	t.l.	0		t.l.	0	65.5%	t.l.	0	65.5%	t.l.	0	66.1%

Table 4: Results of exact algorithms

Type	Benders	basic B&C	Lagrangian-based B&C
A	0 (26)	7 (28)	22 (28)
B	15 (34)	0 (27)	20 (32)
C	21 (34)	0 (23)	13 (32)
E	3 (8)	0 (5)	9 (10)

[8] E. Craparo, M. Karatas, E. I. Singham, A robust optimization approach to hybrid microgrid operation using ensemble weather forecasts, Applied Energy 201 (2017) 135–147.

[9] B. Zhang, Q. Li, L. Wang, W. Feng, Robust optimization for energy transactions in multi-microgrids under uncertainty, Applied Energy 217 (2018) 346–360.

[10] L. Marla, V. Vaze, C. Barnhart, Robust optimization: Lessons learned from aircraft routing, Computers & Operations Research 98 (2018) 165–184.

[11] R. Montemanni, J. Barta, M. Mastrolilli, L. M. Gambardella, The robust traveling salesman problem with interval data, Transportation Science 41 (2011) 366–381.

[12] J. Pereira, I. Averbakh, Exact and heuristic algorithms for the interval data robust assignment problem, Computers & Operations Research 38 (2011) 1153–1163.

[13] E. Conde, A. Candia, Minimax regret spanning arborescences under uncertain costs, European Journal of Operational Research 182 (2007) 561–577.

[14] H. Yaman, O. E. Karasan, M. C. Pinar, The robust spanning tree problem with interval data, Operations Research Letters 40 (2001) 29–31.

[15] I. Averbakh, The minmax regret permutation flow-shop problem with two jobs, European Journal of Operational Research 169 (2006) 761–766.

[16] A. Kasperski, P. Zielinski, An approximation algorithm for interval data minmax regret combinatorial optimization problems, Information Processing Letters 97 (2006) 177–180.

[17] O. E. Karasan, M. C. Pinar, H. Yaman, The robust shortest path problem with interval data, Tech. Rep., Bilkent University, Ankara, Turkey, available from the web site of Optimization Online (http://www.optimization-online.org/, accesed on May 3, 2016), 2001.

[18] T. Assavapokee, J. Realff, J. C. Ammons, Min-max regret robust optimization approach on interval data uncertainty, Journal of Optimization Theory and Applications 137 (2008) 297–316.

[19] I. Averbakh, Minmax regret linear resource allocation problems, Operations Research Letters 32 (2004) 174–180.

[20] J. Pereira, I. Averbakh, The robust set covering problem with interval data, Annals of Operations Research 207 (2013) 217–235.

[21] F. Furini, M. Iori, S. Martello, M. Yagiura, Heuristic and exact algorithms for the interval min-max regret knapsack problem, INFORMS Journal on Computing 27 (2015) 392–405.

[22] N. Ayvaz-Cavdaroglu, S. Kachani, C. Maglaras, Revenue management with minimax regret negotiations, Omega 63 (2016) 11–22.

[23] J. Pereira, The robust (minmax regret) assembly line worker assignment and balancing problem, Computers & Operations Research 93 (2018) 27–40.

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

[25] R. Montemanni, L. Gambardella, A branch and bound algorithm for the robust spanning tree with interval data, European Journal of Operational Research 161 (2005) 771–779.

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

[27] A. Kasperski, Discrete Optimization with Interval Data, Springer, Berlin, 2008.

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

[29] A. Candia-Véjar, E. Álvarez-Miranda, N. Maculan, Min-max regret combinatorial optimization problems: An algorithmic perspective, RAIRO - Operations Research 45 (2011) 101–129.

[30] S. Sahni, T. Gonzalez, P-complete approximation problems, Journal of the Association for Computing Machinery 23 (1976) 555–565.

[31] M. L. Fisher, R. Jaikumar, A generalized assignment heuristic for vehicle routing, Networks 11 (1981) 109–124.

[32] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Chichester, 1990.

[33] K. S. Ruland, A model for aeromedical routing and scheduling, International Transactions in Operational Research 6 (1999) 57–73.

[34] W. Wu, M. Yagiura, T. Ibaraki, Generalized Assignment Problem, in: T. F. Gonzalez (Ed.), Handbook of Approximation Algorithms and Metaheuristics: Methologies and Traditional Applications, vol. 1, chap. 40, Chapman & Hall/CRC, 2 edn., 713–736, 2018.

[35] W. Wu, M. Iori, M. Martello, M. Yagiura, Algorithms for the min-max generalized assignment problem with interval data, in: IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 734–738, 2014.

[36] J. F. Benders, Partitioning procedures for solving mixed variables programming problems, Numerische Mathematik 4 (1962) 238–252.

[37] M. J. Feizollahi, I. Averbakh, The robust (minmax regret) quadratic assignment problem with interval flows, INFORMS Journal on Computing 26 (2014) 321–335.

[38] R. Montemanni, L. Gambardella, The robust shortest path problem with interval data via Benders decomposition, 4OR 3 (2005) 315–328.

[39] S. Siddiqui, S. Azarm, S. Gabriel, A modified Benders decomposition method for efficient robust optimization under interval uncertainty, Structural & Multidisciplinary Optimization 44 (2011) 259–275.

[40] Y. C. Eldar, A. Ben-Tal, A. Nemirovski, Linear minimax regret estimation of deterministic parameters with bounded data uncertainties, IEEE Transactions on Signal Processing 52 (2004) 2177–2187.

[41] M. Posta, J. A. Ferland, P. Michelon, An exact method with variable fixing for solving the generalized assignment problem, Computational Optimization and Applications 52 (2012) 629–644.
[42] M. Yagiura, T. Ibaraki, F. Glover, An ejection chain approach for the generalized assignment problem, INFORMS Journal on Computing 16 (2004) 133–151.
[43] M. Yagiura, T. Ibaraki, F. Glover, A path relinking approach with ejection chains for the generalized assignment problem, European Journal of Operational Research 169 (2006) 548–569.
[44] P. C. Chu, J. E. Beasley, A genetic algorithm for the generalized assignment problem, Computers & Operations Research 24 (1997) 17–23.
[45] M. Laguna, J. P. Kelly, J. González-Velarde, F. Glover, Tabu search for the multilevel generalized assignment problem, European Journal of Operational Research 82 (1995) 176–189.