IPSJ SIG Technical Report
Vol.2015-AL-151 No.2
2015/1/13

Min-Max Regret基準の一般化割当問題に対する解法

呉 偉[1,a) Manuel Iori[2,b) Silvano Martello[3,c) 柳浦 睦憲[1,d)

Abstract: Many real life optimization problems do not have accurate estimates of the problem parameters at the optimization phase. For this reason, the min-max regret criteria are widely used to obtain robust solutions. In this paper we consider the *generalized assignment problem* (GAP) with min-max regret criterion under interval costs. We show that the decision version of this problem is Σ_2^p-complete. We present two heuristic methods: a fixed-scenario approach and a dual substitution algorithm. For the fixed-scenario approach, we show that solving the classical GAP under a median-cost scenario leads to a solution of the min-max regret GAP whose objective function value is within twice the optimal value. We also propose exact algorithms, including a Benders' decomposition approach and branch-and-cut methods which incorporate various methodologies, including Lagrangian relaxation and variable fixing. The resulting Lagrangian-based branch-and-cut algorithm performs satisfactorily on benchmark instances.

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 and robust optimization are two common approaches for the solution of optimization problems under uncertainty. The min-max regret criterion is one of the typical approaches for robust optimization. The *regret* is defined as the difference between the actual cost and the optimal cost that would have been obtained if a different solution had been chosen. The min-max regret approach is to minimize the worst-case regret. This criterion is not as pessimistic as the ordinary min-max approach, which looks for a solution with the best worst-case value across all scenarios.

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 [9] having many applications (see [4], [7], and [8]). The *interval min-max regret generalized assignment problem* (MMR-GAP) is a generalization of the GAP to the case where the cost coefficients are uncertain. In real life applications the costs are often affected by many factors, and can be unknown at the optimization stage. 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 prove that the decision version of MMR-GAP is Σ_2^p-complete. We propose 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), and we show that the median cost scenario leads to a solution of the MMR-GAP whose objective function value is within twice the optimal value. We also propose 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 propose exact algorithmic approaches that iteratively solve the problem by only including a subset of scenarios. The first approach is based on Benders' decomposition: it solves a MIP with incomplete scenarios, and iteratively supplements the scenarios corresponding to violated constraints. We then introduce a basic branch-and-cut algorithm, and enhance it through: (i) 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 compare the introduced algorithms through computational experiments on different benchmarks.

2. Problem Description

2.1 Interval Min-Max Regret Generalized Assignment Problem

The *generalized assignment problem* (GAP) is defined as follows. Given n jobs $J = \{1, \dots, n\}$ and m agents $I = \{1, \dots, m\}$, we undertake to determine 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 of 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 $x_{ij} = 1$:

1 Nagoya University, Japan
2 University of Modena and Reggio Emilia, Italy
3 University of Bologna, Italy
a) goi@co.cm.is.nagoya-u.ac.jp
b) manuel.iori@unimore.it
c) silvano.martello@unibo.it
d) yagiura@nagoya-u.jp

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

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

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

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

For convenience, we define X_0 to be the set of all feasible solutions of GAP: $X_0 = \{x \mid x \text{ satisfies constraints (2)–(4)}\}$.

In many real-life situations the cost c_{ij} is affected by a number of factors, and can be unknown at the optimization stage. In this paper we assume that the cost c_{ij} varies within a given range $[c^-_{ij}, c^+_{ij}]$, i.e., the job-agent assignment cost can take any value in this range. A vector of costs c^s_{ij} satisfying $c^s_{ij} \in [c^-_{ij}, c^+_{ij}]$ is called a *scenario* and is denoted by s. We define $z^s(x)$ to be the objective function value of solution x under scenario s: $z^s(x) = \sum_{i=1}^m \sum_{j=1}^n c^s_{ij}x_{ij}$. 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: $r^s(x) = z^s(x) - z^s_*$.

Let S denote the set of all possible scenarios, i.e., $S = \{s \mid c^s_{ij} \in [c^-_{ij}, c^+_{ij}]\}$. The *maximum regret* of a solution x is then the maximum $r^s(x)$ value over all scenarios: $r_{\max}(x) = \max_{s\in S} r^s(x)$. The MMR-GAP is to find a feasible solution x such that the maximum regret is minimized:

$$\begin{aligned}\min_{x\in X_0} r_{\max}(x) &= \min_{x\in X_0}\max_{s\in S} r^s(x)\\ &= \min_{x\in X_0}\max_{\substack{y\in X_0\\ s\in S}}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c^s_{ij}x_{ij} - \sum_{i=1}^{m}\sum_{j=1}^{n} c^s_{ij}y_{ij}\right\}.\end{aligned}$$

This formulation can be re-written using a classical result for min-max regret problems with interval costs [1]:

Lemma 2.1. *For every solution $x \in X_0$, the regret is maximized under the following scenario $\sigma(x)$:*

$$c^{\sigma(x)}_{ij} = \begin{cases} c^+_{ij} & \textit{if } x_{ij} = 1\\ c^-_{ij} & \textit{otherwise}\end{cases} \qquad \forall i \in I,\ \forall j \in J. \tag{5}$$

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.

From Lemma 2.1, the MMR-GAP can be rewritten as

$$\begin{aligned}\min_{x\in X_0} r_{\max}(x) = \min_{x\in X_0}&\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c^+_{ij}x_{ij}\right.\\ &\left. - \min_{y\in X_0}\sum_{i=1}^{m}\sum_{j=1}^{n}(c^-_{ij} + (c^+_{ij} - c^-_{ij})x_{ij})y_{ij}\right\}.\end{aligned} \tag{6}$$

We assume in the following, without loss of generality, that all input data are integers.

2.2 Complexity of MMR-GAP

In this section, we show that the decision version of the MMR-GAP is Σ^p_2-complete. The proofs are omitted due to space limitations.

It is not hard to see that the MMR-GAP is at least as hard as the classical GAP, because in the special case where $c^+_{ij} = c^-_{ij}$ for all $i \in I$ and $j \in J$, there only exists a single scenario, and the MMR-GAP becomes the classical GAP. It is known that the GAP is NP-hard in the strong sense [9], as it can model, without introducing large numbers, the *bin packing problem*, which is known to be NP-hard in the strong sense [5]. Accordingly, we have the following lemma.

Lemma 2.2. *The MMR-GAP is NP-hard in the strong sense.*

Using a reduction from the interval min-max regret knapsack problem, which is known to be Σ^p_2-hard [3], we then obtain the following.

Lemma 2.3. *The MMR-GAP is Σ^p_2-hard.*

The decision version of the MMR-GAP satisfies the condition of Theorem 7.4 in [5] for $k = 2$, which implies the following.

Lemma 2.4. *The decision version of the MMR-GAP lies in Σ^p_2.*

By combining Lemma 2.3 and 2.4, we can conclude with the following property.

Property 2.1. *The decision version of the MMR-GAP is Σ^p_2-complete.*

3. Heuristic Algorithms

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 (6).

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

For the median-cost scenario, the following result holds (proof omitted).

Lemma 3.1. *Let $\widetilde{s}$ be the median-cost scenario, i.e., $c^{\widetilde{s}}_{ij} = (c^-_{ij} + c^+_{ij})/2$, $\forall i \in I$, $\forall j \in J$, 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 $x \in X_0$.*

3.2 Dual Substitution Heuristic

The dual substitution heuristic introduced in this section generally gives better solutions compared to the fixed-scenario heuristic. The algorithm is based on a *mixed integer programming* (MIP) formulation in which some of the constraints are replaced by their dual counterpart in the linear relaxation of the problem.

The minimization problem over y in (6),

$$\min_{y\in X_0}\sum_{i=1}^{m}\sum_{j=1}^{n}(c^-_{ij} + (c^+_{ij} - c^-_{ij})x_{ij})y_{ij}, \tag{7}$$

for every fixed x is a GAP that can be expressed as $\min_{y\in X_0}\sum_{i=1}^m\sum_{j=1}^n c^{\sigma(x)}_{ij}y_{ij}$. We consider the corresponding linear program obtained by replacing the integrality constraint $y_{ij} \in \{0,1\}$ with the weaker requirement $y_{ij} \ge 0$ for all $i \in I$ and $j \in J$.

We introduce two types of dual variables, u_i ($i \in I$) for constraints (2) and v_j ($j \in J$) for constraints (3). By embedding the dual of the linear relaxation of (7) into (6), we obtain the following dual substitution model (D-MMR-GAP):

$$\begin{aligned}\min\ & \sum_{i=1}^{m}\sum_{j=1}^{n} c^{+}_{ij}x_{ij} + \sum_{i=1}^{m} b_i u_i + \sum_{j=1}^{n} v_j \\ \text{s.t.}\ & -a_{ij}u_i - v_j \le c^{-}_{ij} + (c^{+}_{ij} - c^{-}_{ij})x_{ij}, && \forall i \in I,\ \forall j \in J \\ & u_i \ge 0, && \forall i \in I \\ & x \in X_0.\end{aligned}$$

Property 3.1. *The optimal value of D-MMR-GAP is an upper bound on the optimal value of MMR-GAP.*

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

Property 3.2. *The bound obtained by evaluating the maximum regret of any optimal solution of D-MMR-GAP is at least as good as the optimal value of D-MMR-GAP.*

4. Exact Algorithms

The two exact algorithms proposed 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 $\varphi \le z^s_*$, $\forall s \in S$, the MMR-GAP can be expressed by the following MIP model (MIP-MMR-GAP):

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c^{+}_{ij}x_{ij} - \varphi \tag{8}$$

$$\text{s.t. } \varphi \le \sum_{i=1}^{m}\sum_{j=1}^{n}(c^{-}_{ij} + (c^{+}_{ij} - c^{-}_{ij})x_{ij})y_{ij}, \qquad \forall y \in X_0 \tag{9}$$

$$x \in X_0. \tag{10}$$

4.1 Benders-Like Decomposition

Model (8)–(10) is hard to handle due to the exponential number of constraints (9). Let us define a *master problem* $P(X)$ as the relaxation of the MIP-MMR-GAP in which set X_0 in constraints (9) is replaced by a subset X of X_0. We label the constraints in (9) as Benders' constraints. For an optimal solution (x^*, φ^*) to the current master problem $P(X)$, we define a *slave problem* $Q(x^*)$ as: $\min_{y\in X_0} \sum_{i=1}^{m}\sum_{j=1}^{n}(c^{-}_{ij} + (c^{+}_{ij} - c^{-}_{ij})x^*_{ij})y_{ij}$. 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 (9) 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 solves the updated $P(X)$. The process is iterated until the algorithm finds a solution (x^*, φ^*) for which no violated constraints exist.

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 the final solution, which does not violate any constraint (9), 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. At the initial state, we start with the 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 a unique Benders' cut, the optimal solution of $P(X)$ has some interesting properties, namely:

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

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

4.2 A Branch-and-Cut Algorithm

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. If its optimal value, a lower bound for the corresponding partial problem, is not smaller than the incumbent solution value, then we prune the current node. Otherwise, we look for a violated constraint (9) 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 x^* to the current set X, and solve the updated $\bar{P}(X)$. The process continues until no violated constraint (9) 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.

5. Lagrangian-Based Branch-and-Cut Algorithm

To improve the performance of the basic branch-and-cut algorithm of the previous section, we propose a Lagrangian-based branch-and-cut algorithm, which features a stronger lower bound, an efficient variable fixing method, and an effective solution of the Lagrangian subproblems through dynamic programming.

5.1 Lagrangian-Based Lower Bound

In this section we propose an improved lower bound computation based on Lagrangian relaxation. By embedding constraints (3) and (9) in the objective function (8) through Lagrangian multipliers λ_j and β^s, respectively, we get the following Lagrangian relaxation $L(X, \lambda, \beta)$:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} \hat{c}_{ij}x_{ij} + \sum_{j=1}^{n}\lambda_j - \sum_{y^s\in X}\left(\beta^s \sum_{i=1}^{m}\sum_{j=1}^{n} c^{-}_{ij}y^s_{ij}\right) \tag{11}$$

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

where

$$\hat{c}_{ij} = c^{+}_{ij} - \lambda_j - \sum_{y^s\in X}\left(c^{+}_{ij} - c^{-}_{ij}\right)y^s_{ij}\beta^s. \tag{12}$$

Note that $L(X, \lambda, \beta)$ 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 5.3.

It is not hard to show that $L(X, \lambda, \beta)$ provides a lower bound at least as good as the LP lower bound of Section 4.2 when we use the values of λ and β 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 $\overline{P}(X)$. If it is lower than the incumbent solution value, then a round of Benders' cut additions is performed. When no violated constraint (9) exists, we solve $L(X, \lambda, \beta)$ by setting the λ and β values to those in an optimal solution to $\overline{D}(X)$. If the resulting lower bound is not smaller than the incumbent solution value, the node is fathomed.

5.2 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 Δ the difference between U and the optimal solution value of $L(X, \lambda, \beta)$ at the current node.

Let $\check{x}$ be an optimal solution of $L(X, \lambda, \beta)$ and Ξ_{ij} as the lower bound increase when we force x_{ij} to take value $1 - \check{x}_{ij}$.

The basic variable fixing rules can then be stated as follows:

- Rule 1: if $\check{x}_{ij} = 0$ and $\Xi_{ij} \geq \Delta$, then x_{ij} can be fixed to 0.
- Rule 2: if $\check{x}_{ij} = 1$ and $\Xi_{ij} \geq \Delta$, then x_{ij} can be fixed to 1 and x_{kj} to 0 for all $k \neq i$.

Other rules embedded in the algorithm are omitted due to space limitations.

5.3 Dynamic Programming Approach

As all input data are integers, the knapsack problems needed to compute (11) can be solved through the following dynamic programming approach.

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 $f_i(j, k)$ recursively by dynamic programming as

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

where $f_i(j, k) = +\infty$ is assumed for convenience for all $k < 0$. The computation can be implemented by 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. We call this dynamic programing a *head-to-tail* approach.

In order to efficiently compute the 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. In this way each Ξ_{ij} can be obtained from the two DP tables. (Details are omitted due to space limitations.) 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.

5.4 Benders' Cuts Management

In our Lagrangian-based branch-and-cut approach, it takes only polynomial or pseudo-polynomial time to solve the LP relaxation $\overline{P}(X)$ or the Lagrangian relaxation problems, while the slave problem $Q(\cdot)$ is NP-complete in the strong sense. We handled such difficulty by reducing the computation time needed to solve each $Q(\cdot)$, and by limiting the number of times we solve it. The latter objective was implemented through methods that add cuts either only at the root, or at every node, or adaptively. For solving the slave problem $Q(\cdot)$, we used two efficient heuristic approaches: the ejection chain approach in [10] and the path relinking approach with ejection chains in [11].

6. Computational Experiments

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 well-known GAP benchmark instances called Types A, B, C, D and E (see [2] and [6]). We produced a group of GAP instances for each type, with $m \in \{5, 10\}$ and $n \in \{40, 80\}$. Finally, we obtained MMR-GAP instances by randomly generating the boundaries of cost interval ranges c^-_{ij} and c^+_{ij} from ranges $[(1-\delta)c_{ij}, c_{ij}]$ and $[c_{ij}, (1+\delta)c_{ij}]$, respectively, with $\delta \in \{0.1, 0.2\}$, thus obtaining 8 instances per type.

We performed computational experiments for all the algorithms of Sections 3, 4, and 5. The average gaps, from the best known lower bound values, of the solution values obtained by the two heuristic algorithms are reported in Table 1. The dual substitution heuristic (DS in the Table) obtained better upper bounds than the fixed-scenario heuristic for the instances of Types A, C, D and E.

Table 1 Average optimality gap of heuristic algorithms

	Fixed-Scenario			
Type	c^+	$(c^+ + c^-)/2$	c^-	DS
A	14.20%	0.68%	5.21%	0.00%
B	22.45%	10.02%	8.61%	9.31%
C	15.71%	6.10%	11.19%	5.62%
D	55.79%	41.78%	46.50%	40.09%
E	26.70%	19.39%	22.72%	18.81%

Table 2 Results of exact algorithms

Type	Benders'	basic B&C	Lagrangian B&C
A	0(7)	0(3)	7(7)
B	5(6)	0(0)	1(6)
C	6(7)	0(0)	1(7)
D	0(0)	0(0)	0(0)
E	0(3)	0(0)	4(4)

Table 2 provides the results for the introduced exact algorithms. For each algorithm and type, the entry in the table provides 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 had the best results for instances of Types A and E. In total, it solved to optimality 24 out of 32 instances of Types A, B, C, and E. No additional instance was solved to proven optimality by the Benders' decomposition, nor by the basic branch-and-cut.

7. Conclusions

We studied a robust version of the GAP called the min-max regret GAP (MMR-GAP). We proved that this problem is Σ_2^p-complete, and we presented and compared heuristic and exact methods.

We proposed a fixed-scenario heuristic, for which three scenarios were considered. We proved that the median-cost scenario provides a solution whose objective value is within twice optimal. We also presented a dual substitution heuristic that uses a mixed integer programming formulation obtained by replacing a sub-problem with its dual. We observed that this heuristic method in most cases provides better upper bounds than the fixed-scenario heuristic, obtaining, e.g., exact optimal solutions to all type A instances.

We proposed three exact methods: a Benders' 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, for which we showed that, using a two-direction dynamic programming table, the time complexity can be reduced to $O(nb)$ for $b = \sum_{i=1}^{m} b_i$. Compared with Benders' decomposition, the Lagrangian-based branch-and-cut had better performance for instances of types A and E, and it exactly solved 24 out of 32 instances of Types A, B, C and E with m up to 10 and n up to 80.

References

[1] 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.
[2] P.C. Chu, J.E. Beasley, "A genetic algorithm for the generalized assignment problem," Computers & Operations Research, 24 (1997) 17–23.
[3] V.G. Deineko, G.J. Woeginger, "Pinpointing the complexity of the interval min-max regret knapsack problem," Discrete Optimization, 7 (2010) 191–196.
[4] M.L. Fisher, R. Jaikumar, "A generalized assignment heuristic for vehicle routing," Networks, 11 (1981) 109–124.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman, San Francisco, 1979.
[6] M. Laguna, J.P. Kelly, J.L. Gonzalez-Velarde, F. Glover, "Tabu search for the multilevel generalized assignment problem," European Journal of Operational Research, 82 (1995) 176–189.
[7] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Chichester, New York, 1990.
[8] K.S. Ruland, "A model for aeromedical routing and scheduling," International Transactions in Operational Research, 6 (1999) 57–73.
[9] S. Sahni, T. Gonzalez, "P-complete approximation problems," Journal of the Association for Computing Machinery, 23 (1976) 555–565.
[10] M. Yagiura, T. Ibaraki, F. Glover, "An ejection chain approach for the generalized assignment problem," INFORMS Journal on Computing, 16 (2004) 133–151.
[11] 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.