O P E R A T I O N S R E S E A R C H A N D D E C I S I O N S
No. 2 2012
DOI: 10.5277/ord120203

Bogusz PRZYBYSŁAWSKI*
Adam KASPERSKI*

A COMPUTATIONAL STUDY OF APPROXIMATION ALGORITHMS FOR A MINMAX RESOURCE ALLOCATION PROBLEM

A basic resource allocation problem with uncertain costs has been discussed. The problem is to minimize the total cost of choosing exactly p items out of n available. The uncertain item costs are specified as a discrete scenario set and the minmax criterion is used to choose a solution. This problem is known to be NP-hard, but several approximation algorithms exist. The aim of this paper is to investigate the quality of the solutions returned by these approximation algorithms. According to the results obtained, the randomized algorithms described are fast and output solutions of good quality, even if the problem size is large.

Keywords: *discrete optimization, robust optimization, resource allocation, approximation algorithms*

1. Introduction

In the paper, we consider the following problem: we are given a finite set of items $\mathbf{E} = \{e_1, ..., e_n\}$. In the deterministic case, each item $e_i \in \mathbf{E}$ has a nonnegative cost c_i and we wish to choose a subset $\mathbf{X}$ of exactly p items, i.e. $|\mathbf{X}| = p$, to minimize the total cost $\mathbf{F}(\mathbf{X}) = \sum_{ei\in \mathbf{X}} c_i$. This is one of the simplest combinatorial optimization problems and it is easy to check that the optimum solution is to choose p items with the smallest costs. Furthermore, this optimum solution can be constructed in $O(n)$ time. The problem can be seen as a basic resource allocation problem [4] or a special case, solvable in polynomial time, of the 0-1 knapsack problem [5, 9], where the costs of all the items are equal to 1.

*Institute of Organisation and Management, Wrocław University of Technology, ul. Smoluchowskiego 25, 50-372 Wrocław, e-mail addresses: bogusz.przybyslawskii@pwr.wroc.pl, adam.kasperski@pwr.wroc.pl

In this paper, we discuss the case when the item costs are uncertain. Namely a *scenario set* $\mathbf{\Gamma}$ is specified, which contains all the possible vectors of the item costs. These vectors are called *scenarios*. No probability distribution on $\mathbf{\Gamma}$ is given. For a given cost scenario, $S = (c_1(S), ..., c_n(S))$, we define the cost of $\mathbf{X}$ under S as $\mathbf{F}(\mathbf{X}, S) = \sum_{e_i \in \mathbf{X}} c_i(S)$. In order to choose a solution, we apply the *minmax criterion*, namely we seek a solution which minimizes the largest cost over all scenarios. Let $\text{cost}(\mathbf{X}) = \max_{S \in \mathbf{\Gamma}} F(\mathbf{X}, S)$. We thus consider the following optimization problem:

$$\text{OPT} = \min_{\mathbf{X} \in \mathbf{\Phi}} \text{cost}(\mathbf{X}) \tag{1}$$

where $\mathbf{\Phi} = \{\mathbf{X}: |\mathbf{X}| = p\}$ is the set of all feasible solutions. Problem (1) belongs to the class of *robust discrete optimization problems*, where we seek a solution maximizing performance in the worst case. A detailed description of the robust approach and various robust criteria can be found in the book by Kouvelis and Yu [9]. We also refer the reader to the paper by Aissi et al. [1] and the book by Kasperski [6] for a survey of the recent results in this area. In general, the robust approach can be applied if there is no additional information provided with the scenario set (for example a probability distribution) and decision makers are interested in maximizing performance of the explored system in the worst case scenario.

There are several methods of defining the scenario set $\mathbf{\Gamma}$ which have been proposed in the existing literature. Using the *interval uncertainty representation*, each item cost c_i is known to belong to a closed interval $[c_i^-, c_i^+]$, and the scenario set $\mathbf{\Gamma}$ is the Cartesian product of all the uncertainty intervals. However, in this case the minmax problem (1) is trivial, because it is enough to find an optimum solution for the pessimistic scenario $(c_1^+, ..., c_n^+)$. For the interval uncertainty representation, the problem with the *minmax regret* criterion, which is based on the largest deviation from the optimum over all scenarios, is more interesting [2, 3, 8].

In the *discrete scenario representation*, the scenario set $\mathbf{\Gamma}$ contains $K \geq 1$, explicitly given cost vectors, i.e. $\mathbf{\Gamma} = \{S_1, ..., S_K\}$. In this case, problem (1) is much harder to solve. Namely, it is NP-hard even for 2 scenarios [2]. Furthermore, it becomes strongly NP-hard and not approximable within any constant factor, if the number of scenarios is a part of the input, unless $\mathbf{P} = \mathbf{NP}$ [7]. So, there is no hope of constructing a fast algorithm, which returns an optimum solution, if the problem size is large.

There are several approximation algorithms known for the problem. If the number of scenarios K is fixed, then there exists a fully polynomial time approximation scheme [1] which is however exponential in K and can be applied in practice only if the number of scenarios is small. This problem also has a simple deterministic K-approximation algorithm [1], and a randomized $O(\log K)$-approximation algorithm [7]. We will give a clear description of these algorithms in the next section.

The aim of this paper is to explore the quality of the solutions returned by these approximation algorithms. In Section 2, we describe in detail the exact and approximation algorithms known for the problem. In Section 3, we present the results of computational tests, which characterize the efficiency of the exact method and the quality of the solutions returned by the approximation algorithms.

2. Exact and approximation algorithms for the problem

We first recall an exact method of solving the problem, which is based on mixed integer programming. Let us introduce the binary variable $x_i \in \{0, 1\}$ such that $x_i = 1$ if and only if the item e_i is chosen. We can then obtain an optimum solution to problem (1) by solving the following mixed integer linear programming problem:

$$\begin{aligned} &\min T \\ &x_1 + x_2 + \ldots + x_n = p \\ &x_1 c_1(S) + x_2 c_2(S) + \ldots + x_n c_n(S) \le T \qquad \text{for all } S \in \mathbf{\Gamma} \\ &x_i \in \{0, 1\} \qquad \text{for all } i = 1, \ldots, n \\ &T \ge 0 \end{aligned} \tag{2}$$

Of course, solving (2) is **NP**-hard, thus this approach is efficient only when the problem size is not very large. In the next section we will investigate how large problems can be solved efficiently using formulation (2). Notice that if we replace the binary constraints $x_i \in \{0, 1\}$ by $0 \le x_i \le 1$ for all $i = 1, ..., n$, then we get a linear programming relaxation of (2). This relaxation is solvable in polynomial time and gives us a lower bound on OPT.

Let $c'_i = (1/K)\sum_{S \in \mathbf{\Gamma}} c_i(S)$ be the average item cost over all scenarios and let $\mathbf{X}'$ be an optimum solution for the average costs. It can be easily shown that the maximum cost of the solution $\mathbf{X}'$ is at most K times larger than the maximum cost of an optimum solution, i.e. $\text{cost}(\mathbf{X}') \le K \times \text{OPT}$. The easy proof proceeds as follows (see also Aissi et al. [1]). For any solution $\mathbf{X}$, we have:

$$\begin{aligned} \text{cost}(\mathbf{X}) &= \max_{S \in \mathbf{\Gamma}} \mathbf{F}(\mathbf{X}, S) \ge \frac{1}{K} \sum_{S \in \mathbf{\Gamma}} \mathbf{F}(\mathbf{X}, S) \\ &\ge \frac{1}{K} \sum_{S \in \mathbf{\Gamma}} \mathbf{F}(\mathbf{X}', S) \ge \frac{1}{K} \max_{S \in \mathbf{\Gamma}} \mathbf{F}(\mathbf{X}', S) = \frac{1}{K} \text{cost}(\mathbf{X}') \end{aligned}$$

In consequence, if $\text{cost}(\mathbf{X}) = \text{OPT}$, i.e. $\mathbf{X}$ is an optimum minmax solution, then $\text{cost}(\mathbf{X}') \le K \times \text{OPT}$. We thus get a polynomial K-approximation algorithm for the problem. We will denote this simple approximation algorithm as Average.

The bound for the algorithm Average is tight, which is demonstrated in Table 1. We have a problem with $2K$ elements, K scenarios and $p = K$. Each of the $2K$ items has the same average cost, so the algorithm Average may return the solution $\mathbf{X}' = \{e_1, e_2, ..., e_K\}$ such that cost($\mathbf{X}'$) = K. But the optimum minmax solution is composed of the items = $\{e_{K+1}, e_{K+2}, ..., e_{2K}\}$ and its maximum cost is equal to 1. Thus, for this case we obtain cost($\mathbf{X}'$) = K×OPT.

Table 1. An instance of input data for which the algorithm Average returns a bad solution

	S_1	S_2	...	S_K
e_1	1	0	...	0
e_2	1	0	...	0
...	1	0	...	0
e_K	1	0	...	0
e_{K+1}	1	0	...	0
e_{K+2}	0	1	...	0
...	...	...	...	...
e_{2K}	0	0	...	1

We now recall an approximation algorithm with a better worst case ratio, which was constructed in the paper by Kasperski and Zieliński [7]. Let $\mathbf{E}(T)$ be the subset of the items such that $e_i \in \mathbf{E}(T)$ if and only if $c_i(S) \leq T$ for all scenarios $S \in \mathbf{\Gamma}$. In other words $\mathbf{E}(T)$ contains all the items whose costs do not exceed T under any scenario. Consider the following system of linear constraints:

$$\begin{aligned} & x_1 + x_2 + \ldots + x_n = p \\ & x_1 c_1(S) + x_2 c_2(S) + \ldots + x_n c_n(S) \leq T \quad \text{for all } S \in \mathbf{\Gamma} \\ & x_i = 0 \quad \text{for all } e_i \notin \mathbf{E}(T) \\ & 0 \leq x_i \leq 1 \quad \text{for all } i = 1, \ldots, n \\ & T \geq 0 \end{aligned} \tag{3}$$

Using binary search, in polynomial time we can find the minimum value of T, denoted by T^*, for which the system (3) is feasible. Clearly T^* is a lower bound on OPT, i.e. OPT $\geq T^*$. Let $x_1^*, ..., x_n^*$ be a feasible solution to (3) for T^*. Consider now the following algorithm:

Randomized

Step 1. $\mathbf{X} := \emptyset$, $k := 0$, $z := \infty$

Step 2. $k := k + 1$, $\mathbf{Y} := \emptyset$, for each $e_i \in \mathbf{E}$ add e_i to $\mathbf{Y}$ with probability x_i^*

Step 3. If $|\mathbf{Y}| \geq p$ and cost($\mathbf{Y}$) $\leq z$ then $\mathbf{X} := \mathbf{Y}$ and $z :=$ cost($\mathbf{Y}$)
Step 4. If $k < \log n/(\log 2n - \log(n + 2))$ then go to Step 2
Step 5. If $|\mathbf{X}| \geq p$ then output any p items from $\mathbf{X}$ otherwise output $\varnothing$

The algorithm Randomized is a randomized algorithm for problem (1). In Step 2, it constructs a set $\mathbf{Y}$ by performing a randomized rounding of the solution $x_1^*, \ldots, x_n^*$ of the linear relaxation. This step is repeated a certain number of times and the best solution found is returned. Notice that this algorithm may fail to return a feasible solution. This bad event happens if the cardinalities of all the sets $\mathbf{Y}$ constructed in Step 2 are less than p. In this case, the empty set is returned. However, the probability of this bad event is very small. The following theorem characterizes the performance of the algorithm Randomized:

Theorem 1. If $n \geq 11$ and $K \leq n^7$, then the algorithm Randomized returns a feasible solution $\mathbf{X}$, such that cost($\mathbf{X}$) = $O(\log K)$ OPT with the probability at least $1 - 1/n$ (Kasperski and Zieliński [7]).

Theorem 1 states that the algorithm returns a feasible solution of maximum cost at most $O(\log K)$ OPT with a high probability. The assumptions of the theorem are technical, but not restrictive. We thus can expect that the algorithm Randomized will return better solutions than the simple K-approximation algorithm Average, which returns an optimum solution for the average costs. We will verify this conjecture in the next section.

3. Results of the computational tests

In this section, we present the results of computational tests. The aim of the tests was to explore the quality of the solutions returned by the approximation algorithms. Let (n, K, U) be an instance of the problem, where n is the number of items, K is the number of scenarios and the cost of each item under any scenario is a number chosen uniformly at random from the interval $[0, U]$. We assume that $p = n/2$, which gives us the largest solution space. We fixed n = 100, 200 ..., 1000, K = 2, 4, ..., 10 and U = 10, 1000. For each combination of n, K and U we randomly generated 10 instances and performed the tests using them.

We first computed the optimum solutions by using the mixed integer programming formulation, see problem (2). In order to solve the problem, we used the CPLEX 12.1 solver and a computer equipped with an Intel Core2 Duo E6550 2.33 GHz processor with 2 GB RAM and a 64-bit operational system. Unfortunately, we were unable to solve some instances with $K \geq 8$ in reasonable time. In this case, we only computed

lower bounds on OPT by relaxing the binary constraints in formulation (2), i.e. by replacing the constraints $x_i \in \{0, 1\}$ by $0 \le x_i \le 1$ for all $i = 1, ..., n$. The average running times required to solve the problems are presented in Table 2.

Table 2. The average running times [s] required by CPLEX to solve the problem for various n and K for $U = 1000$*

n	K				
	2	4	6	8	10
100	0.24	0.45	1.00	3.15	8.92
200	0.31	0.72	12.83	111.70	
300	0.26	2.73	41.44		
400	0.29	2.54	114.89		
500	0.28	4.06			
600	0.30	3.12			
700	0.33	3.11			
800	0.31	3.47			
900	0.34	2.59			
1000	0.39	2.91			

*Empty boxes denote that the authors were unable to solve the problem due to memory limitations.

As we can see in Table 1, the time required by the CPLEX solver to compute an optimum solution grows quickly with the number of scenarios. If the number of items exceeds 400, then an optimum solution can be quickly computed for up to 6 scenarios, otherwise we get an "out of memory" message from the solver.

Table 3. The minimum, average and maximum percentage deviations from the optimum reported for $K = 2$

n	$U = 10$						$U = 1000$					
	Average			Randomized			Average			Randomized		
	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.
100	0.51	5.46	11.01	0.00	1.38	4.16	0.00	6.89	18.31	0.00	0.53	1.36
200	1.12	5.41	13.87	0.02	0.99	2.28	0.18	3.95	10.20	0.00	0.50	1.50
300	0.86	3.84	8.82	0.07	0.28	0.86	0.54	3.45	8.78	0.02	0.66	1.60
400	0.73	3.58	6.74	0.00	0.21	0.53	0.51	3.31	5.59	0.00	0.32	0.99
500	0.44	3.67	6.61	0.00	0.32	0.97	0.54	2.34	7.40	0.02	0.34	0.89
600	0.18	2.71	6.18	0.04	0.23	0.70	0.33	2.95	7.97	0.00	0.17	0.55
700	0.25	2.50	4.39	0.00	0.17	0.43	0.56	2.91	5.79	0.02	0.12	0.24
800	0.97	3.08	7.11	0.00	0.12	0.37	0.06	2.86	7.00	0.01	0.14	0.37
900	0.20	2.31	4.26	0.00	0.11	0.42	0.43	1.93	6.02	0.00	0.10	0.29
1000	0.65	1.68	2.55	0.00	0.11	0.32	0.60	1.97	3.31	0.00	0.12	0.31

We next computed approximate solutions by applying the two approximation algorithms, Average and Randomized, described in the previous sections. For each particular instance, we reported the minimum, average and maximum percentage deviation from the optimum (or the lower bound for larger K). The results obtained are shown in Tables 3–7.

Table 4. The minimum, average and maximum percentage deviations from the optimum reported for $K = 4$

n	$U = 10$						$U = 1000$					
	Average			Randomized			Average			Randomized		
	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.
100	4.71	9.36	16.38	0.21	1.43	3.15	5.57	10.05	17.18	0.62	1.96	4.01
200	6.60	9.12	12.65	0.26	1.18	2.42	4.76	7.37	12.08	0.19	0.78	1.80
300	2.57	7.30	11.97	0.04	0.98	1.97	2.41	6.36	11.58	0.18	0.48	1.07
400	1.18	5.15	9.79	0.02	0.55	1.55	2.86	5.90	11.79	0.13	0.24	0.54
500	0.95	4.95	10.45	0.12	0.40	0.76	2.31	6.12	11.17	0.14	0.49	0.94
600	1.50	4.19	8.26	0.08	0.30	0.69	2.41	5.48	8.23	0.04	0.42	1.23
700	1.77	3.97	7.11	0.05	0.18	0.32	2.18	4.76	8.48	0.07	0.29	0.70
800	1.63	3.80	5.22	0.10	0.35	0.86	1.81	4.18	8.29	0.06	0.28	0.70
900	2.27	3.44	5.05	0.00	0.19	0.38	1.26	3.91	8.07	0.12	0.28	0.49
1000	1.49	2.93	5.66	0.02	0.16	0.34	2.19	3.94	7.63	0.03	0.19	0.39

Table 5. The minimum, average and maximum percentage deviations from the optimum (lower bound) reported for $K = 6$

n	$U = 10$						$U = 1000$					
	Average			Randomized			Average			Randomized		
	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.
100	2.57	9.88	22.02	0.40	2.89	6.10	5.78	10.74	15.41	0.34	2.29	6.28
200	2.62	7.11	10.37	0.47	1.84	4.79	2.25	8.20	13.71	0.31	1.15	1.98
300	1.98	6.81	13.36	0.38	0.99	1.67	2.33	6.59	11.69	0.38	0.99	2.05
400	3.12	6.14	10.65	0.33	0.64	1.22	4.18	5.84	9.70	0.13	0.37	0.89
500	(2.94)	(5.36)	(9.29)	(0.20)	(0.60)	(0.93)	(2.10)	(4.68)	(7.94)	(0.11)	(0.54)	(1.52)
600	(2.29)	(4.89)	(8.25)	(0.24)	(0.63)	(1.21)	(2.52)	(5.19)	(10.99)	(0.23)	(0.52)	(0.77)
700	(3.11)	(4.93)	(7.28)	(0.05)	(0.21)	(0.33)	(3.33)	(5.05)	(7.69)	(0.11)	(0.31)	(0.66)
800	(1.91)	(3.91)	(7.84)	(0.14)	(0.37)	(0.83)	(2.11)	(5.05)	(8.97)	(0.18)	(0.45)	(1.08)
900	(2.03)	(4.11)	(6.46)	(0.07)	(0.23)	(0.51)	(3.14)	(4.76)	(10.53)	(0.23)	(0.34)	(0.57)
1000	(2.60)	(4.65)	(7.82)	(0.06)	(0.31)	(0.78)	(1.40)	(3.17)	(5.18)	(0.15)	(0.19)	(0.78)

Table 6. The minimum, average and maximum percentage deviations from the optimum (lower bound) reported for $K = 8$

n	$U = 10$						$U = 1000$					
	Average			Randomized			Average			Randomized		
	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.
100	2.57	9.88	22.02	0.21	1.43	3.15	6.25	10.66	15.77	0.60	2.54	4.23
200	2.62	7.11	10.37	0.26	1.18	2.42	4.69	10.01	15.30	0.44	2.28	5.63
300	(1.98)	(6.81)	(13.36)	(0.04)	(0.98)	(1.97)	(4.55)	(8.98)	(15.26)	(0.39)	(1.27)	(2.60)
400	(3.12)	(6.14)	(10.65)	(0.02)	(0.55)	(1.55)	(4.66)	(7.02)	(12.06)	(0.39)	(1.00)	(2.18)
500	(2.94)	(5.36)	(9.29)	(0.20)	(0.60)	(0.93)	(4.80)	(6.40)	(7.67)	(0.13)	(0.62)	(1.03)
600	(2.29)	(4.89)	(8.25)	(0.24)	(0.63)	(1.21)	(3.97)	(5.49)	(7.57)	(0.35)	(0.57)	(1.02)
700	(3.11)	(4.93)	(7.28)	(0.05)	(0.21)	(0.33)	(2.13)	(5.25)	(7.79)	(0.31)	(0.54)	(0.91)
800	(1.91)	(3.91)	(7.84)	(0.14)	(0.37)	(0.83)	(1.21)	(4.47)	(7.41)	(0.21)	(0.40)	(0.90)
900	(2.03)	(4.11)	(6.46)	(0.07)	(0.23)	(0.51)	(2.55)	(4.25)	(6.40)	(0.10)	(0.37)	(0.84)
1000	(2.60)	(4.65)	(7.82)	(0.06)	(0.31)	(0.78)	(2.60)	(4.28)	(6.57)	(0.11)	(0.46)	(0.49)

Table 7. The minimum, average and maximum percentage deviations from the optimum (lower bound) reported for $K = 10$

n	$U = 10$						$U = 1000$					
	Average			Randomized			Average			Randomized		
	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.	Max.
100	7.40	13.59	21.27	0.73	3.20	5.46	6.92	11.57	17.46	0.90	3.60	7.38
200	(5.00)	(9.63)	(13.81)	(0.97)	(2.14)	(3.44)	(3.15)	(7.03)	(12.57)	(1.10)	(2.61)	(4.71)
300	(3.50)	(7.13)	(13.01)	(0.60)	(1.79)	(4.60)	(3.89)	(8.39)	(14.63)	(0.59)	(1.38)	(2.42)
400	(3.90)	(8.33)	(17.07)	(0.36)	(1.00)	(2.35)	(3.78)	(6.71)	(9.93)	(0.38)	(1.07)	(2.46)
500	(2.92)	(6.95)	(9.40)	(0.36)	(0.81)	(1.75)	(4.47)	(6.57)	(10.03)	(0.35)	(1.06)	(1.65)
600	(3.08)	(5.49)	(11.11)	(0.18)	(0.53)	(1.11)	(4.61)	(6.65)	(10.35)	(0.49)	(0.97)	(1.77)
700	(3.01)	(5.15)	(8.83)	(0.43)	(0.73)	(1.47)	(2.58)	(5.41)	(8.86)	(0.20)	(0.94)	(1.89)
800	(3.41)	(5.01)	(7.57)	(0.18)	(0.40)	(0.78)	(3.23)	(5.52)	(7.60)	(0.31)	(0.59)	(1.18)
900	(2.93)	(5.30)	(9.01)	(0.22)	(0.56)	(1.55)	(2.79)	(4.71)	(8.32)	(0.18)	(0.48)	(1.54)
1000	(3.08)	(5.65)	(8.52)	(0.18)	(0.25)	(1.04)	(3.01)	(4.11)	(5.74)	(0.16)	(0.23)	(0.59)

The computational results obtained demonstrate that the randomized algorithm returns better results than the simple K-approximation algorithm Average. Moreover, when the number of items increases, the performance of the algorithm Randomized becomes better and the deviations from the optimum are smaller. According to these tests, one can say that the number of scenarios has the greatest impact on the performance of the approximation algorithms. The limit on the cost, denoted by U, seems to have no influence on the quality of the results.

3. Summary

The resource allocation problem with the discrete scenario uncertainty representation has been discussed. The minmax criterion was adopted to choose an optimum solution. We described three algorithms: one exact, a K-approximation and a randomized $O(\ln K)$ – approximation and compared their performance by performing some computational tests. The exact algorithm, based on the mixed integer programming formulation, seems to be inefficient for problems with more than 400 items and 6 scenarios. The randomized algorithms described are fast and output solutions of good quality. Thus, they are attractive if the size of the problem is large.

References

[1] AISSI H., BAZGAN C., VANDERPOOTEN D., *Min-max and min-max regret versions of combinatorial optimization problems: A survey*, European Journal of Operational Research, 2009, 197, 427–438.

[2] AVERBAKH I., *On the complexity of a class of combinatorial optimization problems with uncertainty*, Mathematical Programming, 2001, 90, 263–272.

[3] CONDE E., *An improved algorithm for selecting p items with uncertain returns according to the min-max regret criterion*, Mathematical Programming, 2004, 100, 345–353.

[4] IBARAKI T., KATOH N., *Resource allocation problems*, MIT Press, 1988.

[5] IIDA H., *A note on the max-min 0-1 knapsack problem*, Journal of Combinatorial Optimization, 2004, 3, 89–94.

[6] KASPERSKI A., *Discrete optimization with interval data. Minmax regret and fuzzy approach*, Studies in Fuzziness and Soft Computing, Vol. 228, Springer, 2008.

[7] KASPERSKI A., ZIELIŃSKI P., *A randomized algorithm for the min-max selecting items problem with uncertain weights*, Annals of Operations Research, 2009, 172, 221–230.

[8] KASPERSKI A., KURPISZ A., ZIELIŃSKI P., *Approximating the minmax (regret) selecting items problem*, Information Processing Letters, 2013, 113, 23–29.

[9] KOUVELIS P., YU G., *Robust discrete optimization and its applications*, Kluwer Academic Publishers, 1997.