A dual approach to solving a multi-objective assignment problem

O A Medvedeva and S N Medvedev
Department of Applied Mathematics, Informatics and Mechanics, Voronezh State University, 1 Universitetskaya pl., Voronezh, 394006, Russia
E-mail: s_n_medvedev@mail.ru

Abstract
Two approximation algorithms applied for solving the multi-objective assignment problem (MOAP) are proposed. The reduction to the single-objective problem is realized and dual Uzawa method is applied. The differences between algorithms are the Lagrangian construction technics, the application technics of Hungarian method or independent column minimization in each iteration. The results of the comparative computational experiment are presented.

1. Introduction
The assignment problem (AP) is one of the most-studied, well-known and important problem of discrete optimization. The classical linear AP has been well studied and many algorithms have been produced to solve it, e.g. Hungarian method.

The classical AP can be stated as: Let a number \( n \) of jobs be given that must be performed by \( n \) workers, where the costs depend on the specific assignments. Each job must be assigned to one and only one worker and each worker has to perform one and only one job. Let \( c_{ij} \) be the cost associated with worker \( i \) (\( i = \overline{1,n} \)) who has performed job \( j \) (\( i = \overline{1,n} \)). The problem is to find such an assignment that the total costs, i.e., the sum of the single assignments, become a minimum.

The mathematical model of the classical AP has the form:

\[
L(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}x_{ij} \rightarrow \min
\]

\[
\sum_{i=1}^{n} x_{ij} = 1, \; j = \overline{1,n},
\]

\[
\sum_{j=1}^{n} x_{ij} = 1, \; i = \overline{1,n},
\]

\[
x_{ij} = \{0,1\}, \; i, j = \overline{1,n},
\]

\( x_{ij} \) denotes that job \( j \) is to be assigned to the worker \( i \).
Combinatorial optimization problems are usually considered in single-objective formulation. Mathematical optimization problems, having feasible solutions set corresponding to the classical AP and involving more than one objective function to be optimized simultaneously, are still not fully understood, but important for practical applications. MOAP [1, 2] is one of such problems.

MOAP can be formalized as classical AP in more than one factories simultaneously. The mathematical model of the MOAP has the form:

\[ L_i(X_i) = \sum_{j=1}^{n_i} \sum_{k=1}^{K} c_{ij}^k x_{ij}^k \rightarrow \min \]

\[ \ldots \]

\[ L_K(X_K) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^k x_{ij}^k \rightarrow \min \]

\[ \sum_{j=1}^{n} x_{ij}^k = 1, \quad k = \overline{1,K}, \quad j = \overline{1,n_i}, \]

\[ \sum_{j=1}^{n} x_{ij}^k \leq 1, \quad i = \overline{1,m}, \quad k = \overline{1,K}, \]

\[ \sum_{k=1}^{K} \sum_{j=1}^{n} x_{ij}^k \leq 1, \quad i = \overline{1,m}, \]

\[ x_{ij}^k = \{0,1\}, \quad i = \overline{1,m}, \quad k = \overline{1,K}, \quad j = \overline{1,n_i}, \]

where \( C^k = \{c_{ij}^k\}_{m \times n_i}, \quad k = \overline{1,K} \) are the cost matrices; value \( c_{ij}^k \) represents the cost associated with worker \( i \) who has performed job \( j \) in the factory \( k \).

Multi-objective optimization methods for such problems can be conventionally classified in two categories. Methods of the first category transform the MOAP into a single-objective AP by using a convolution of the vector of individual objective functions into supercriterion (i.e., some generalized scalar objective function) [2–5]. The second category includes such methods as compromise method, step-by-step method, hierarchy analysis technique [5–7], etc.

In this paper, we propose the approach, which uses the convolution of the vector of objectives. The main results in this area are related to the studying the possibility of application of linear convolution algorithms to find the set of Pareto-optimal solutions (Pareto optimal set) or its subsets [8, 9].

In [10, 11] it has been proved that the MOAP is NP-hard problem if its objective vector contains at least two functions of the following type

\[ L_i(X_i) = \sum_{j=1}^{n} \sum_{i=1}^{m} c_{ij} x_{ij} \rightarrow \min. \]

Works [12, 13] propose polynomial algorithms to solve the AP with two objective functions

\[ L_1(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \rightarrow \min, \]

\[ L_2(X) = \max_{i=1}^{m} \max_{j=1}^{n} c_{ij} x_{ij} \rightarrow \min. \]

Using the linear convolution of \( L_1(X) \) and \( L_2(X) \), this problem can be transformed to the single-objective AP, which are solvable by well-studied techniques, e.g. Hungarian method.
Paper [14] proposes the MOAP solving algorithm based on the Hungarian method and Hopcroft-Karp algorithm [15].

The algorithm for finding the Pareto optimal set presented in [16] differs from the previous one presented in [14]: in [16] the Kachichko algorithm [17] is used in some certain cases instead of the Hopcroft-Karp algorithm.

In [7] it has been shown that if the objective functions are positive, the following non-linear convolution can be used for solving the MOAP

\[ L(X) = \max_{\nu \in \{1, \ldots, N\}} \{ \lambda_{\nu} L_{\nu}(X) \}. \]

Thus, in order to solve the MOAP, the main researches are related to the application of algorithm of linear convolution (ALC). However, this is not the unique approach, and use of other types of convolution is a promising direction. Further development of this research area is related not only to the application but also to the modification and addition of well-known vector optimization techniques, taking into account the problem specific character.

It is the purpose of this paper to develop MOAP-solving approximation algorithms based on transforming into a single-objective AP and further applying Uzawa method [18].

2. Materials and methods

Consider the mathematical model of MOAP (1)–(6).

Constraints (3) and (4) are standard MOAP restrictions for each fixed factory; the constraint (5) means that one and the same worker cannot simultaneously perform job in more than one factory. Let us remark here that (4) is a consequence of (5). However, constraint (4) is not removed from the mathematical model for the convenience of further algorithmization.

**Statement 1.** MOAP (1)–(6) has a solution on condition that \( m \geq \sum_{k=1}^{K} n_k \).

**Proof.**

Summing up all inequalities in (5) with respect to \( i = 1, m \) yields

\[
\sum_{i=1}^{m} \sum_{k=1}^{K} \sum_{j=1}^{n_k} x_{ij}^k \leq m.
\]

Summing up all equalities in (3) with respect to \( j = 1, n_k \) for each \( k = 1, K \) yields

\[
\sum_{j=1}^{n_k} \sum_{i=1}^{m} x_{ij}^k = n_k, \quad k = 1, K.
\]

Finally, summing up the latest equalities with respect to \( k = 1, K \) yields

\[
\sum_{k=1}^{K} \sum_{j=1}^{n_k} \sum_{i=1}^{m} x_{ij}^k = \sum_{k=1}^{K} n_k.
\]

Thus, \( m \geq \sum_{k=1}^{K} \sum_{j=1}^{n_k} \sum_{i=1}^{m} x_{ij}^k = \sum_{k=1}^{K} n_k \).

The statement is proved.

We provide two MOAP-solving algorithms based on transforming into a single-objective AP and further applying Uzawa method [18].

One of these algorithms is generalization of two-objective AP solving method proposed in [19, 20], whereby two-objective AP transforms into the classical AP at every stage. In the paper under consideration we propose different way for construction the Lagrangian that simplifies very much the solving algorithm.

Consider an application of Uzawa method for MOAP (1)-(6).
As we aim to minimize each of the objective functions, consider the objective function of following type [20]

$$\min \left\{ \max \left( L_1(X_1), L_2(X_2), \ldots, L_K(X_K) \right) \right\}.$$ 

Let us set \( \mu = \max \left( L_1(X), L_2(X), \ldots, L_K(X) \right) \). Note that \( L_i(X), \ i = 1, K \) take on the nonnegative values \( (c_{ij}^k \geq 0, \ x_{ij}^k \in \{0,1\}, \ i = 1, m, \ k = 1, K, \ j = 1, n_k) \). Therefore, it can be argued that \( \mu \geq 0 \).

MOAP (1)–(6) can be restated in equivalent form:

$$\mu \rightarrow \min$$

\[
L_i(X_i) - \mu \leq 0,
\]

$$\ldots$$

\[
L_K(X_K) - \mu \leq 0,
\]

\[
\sum_{k=1}^{K} \sum_{j=1}^{n_k} x_{ij}^k \leq 1, \ i = 1, m,
\]

\[
x_{ij}^k \in S, \ i = 1, m, \ k = 1, K, \ j = 1, n_k,
\]

\[
\mu \geq 0.
\]

The set \( S \) will be described further.

The Lagrangian for this problem can be written as

$$\Phi(x, \mu, u, w) = \mu + u_i \left( \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij}^1 x_{ij}^1 - \mu \right) + \ldots + u_K \left( \sum_{i=1}^{m} \sum_{j=1}^{n_k} c_{ij}^K x_{ij}^K - \mu \right) + \sum_{i=1}^{m} w_i \left( \sum_{k=1}^{K} \sum_{j=1}^{n_k} x_{ij}^k - 1 \right), \ u, w \geq 0, \ x \in S, \ \mu \geq 0.$$ 

Using the Lagrangian, the MOAP (1)–(6) and its dual problem can be respectively written as

$$\min \max_{x \in S \atop u, w \geq 0} \Phi(x, \mu, u, w); \ \max_{u, w \geq 0} \min_{x \in S \atop \mu \geq 0} \Phi(x, \mu, u, w).$$

We call \( \omega(u, w) = \min_{x \in S \atop \mu \geq 0} \Phi(x, \mu, u, w) \) the dual function. The dual problem \( \max_{u, w \geq 0} \omega(u, w) \) is more simply than the primal. The dual problem is continual \( (u \text{ and } w \text{ take on any real values}), \) whereas the primal problem is discrete.

Let us remark here that \( \omega(u, w) \) is a convex function [18]; this makes it possible to maximize it using iterative subgradient method.

The values of the dual variables \( u \) and \( w \) are calculated in each iteration by formula

$$\begin{pmatrix} u \\ w \end{pmatrix}^{N+1} = \begin{pmatrix} u \\ w \end{pmatrix}^{N} + \left( \begin{pmatrix} \alpha_N \\ \beta_N \end{pmatrix} \right) \tilde{\nabla} \omega(u^N, w^N),$$

where \( \tilde{\nabla} \omega \) is the corresponding value of subgradient of the dual function \( \omega(\cdot) \); \( \alpha_N \) and \( \beta_N \) are the step sizes.
Thus, the dual Uzawa method for the problem under consideration is summarized as follows:

**Step 0. Initialization.** Enter initial data $u^0 \geq 0$, $w^0 \geq 0$, $N = 0$.

**Step 1.** Find a solution of the problem

$$\Phi(x, \mu, u^N, w^N) \rightarrow \min_{x \in S, \mu \geq 0}.$$  

**Step 2.** STOP-test. If the stopping condition is fulfilled, we obtain the optimal solution $X_k^* = X_k^{N+1}$, $k = 1, K$. Otherwise, we have to calculate $\mu^{N+1}$.

**Step 3.** Calculate the values of dual variables $u^N_k, w^N_i$.

The main stopping criterion is fulfillment of constraints (5). The additional stopping conditions are: algorithm execution time, total number of iterations.

To implement the computing circuit, the Lagrangian is converted as follows:

$$\Phi(x, \mu, u, w) = \sum_{i=1}^{m} \sum_{j=1}^{n} (u_k^N c_{ij}^N + w_i^N) x_{ij}^N + \sum_{i=1}^{m} \sum_{j=1}^{n} (u_k c_{ij}^k + w_i) x_{ij}^k + \mu \left(1 - \sum_{k=1}^{K} u_k^N \right)$$

for $u, w \geq 0$, $x \in S$, $\mu \geq 0$.

As a result, Step 1 of the algorithm solves the following problems:

1. \[ \sum_{i=1}^{m} \sum_{j=1}^{n} (u_k^N c_{ij}^N + w_i^N) x_{ij}^N \rightarrow \min_{x \in S}, \quad k = 1, K, \]  
2. \[ \mu \left(1 - \sum_{k=1}^{K} u_k^N \right) \rightarrow \min_{\mu \geq 0}. \]  

Note that there are several alternatives of defining the structure of set $S$ for solving the problem (7).

1. If $S$ is defined as a set of $\{x_{ij}^N\}$ that satisfy the constraints (3), (4) and (6), the following APs will be solved in each iteration by Hungarian method holding dual variables $u_k$, $k = 1, K$ and $w_i$, $i = 1, m$ constant

$$\sum_{i=1}^{m} \sum_{j=1}^{n} (u_k^N c_{ij}^N + w_i^N) x_{ij}^N \rightarrow \min_{x \in S}, \quad k = 1, K.$$

2. If $S$ is defined as a set of $\{x_{ij}^k\}$ that satisfy the constraints (3) and (6), the following problem will be solved in each iteration

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{K} (u_k c_{ij}^k + w_i) x_{ij}^k \rightarrow \min$$

$$\sum_{i=1}^{m} x_{ij}^k = 1,$$

$$x_{ij}^k \in \{0, 1\}, \quad i = 1, m, \quad j = 1, n, \quad k = 1, K.$$

It is the problem of independent column minimization in the given cost matrix.

In view of definition of $\mu$, the following restrictions are valid

$$0 \leq \mu \leq \max(L_1(X_1^N), ..., L_K(X_K^N)).$$

Then problem (9) is solved as follows:
The computational experiment has shown that in case of \( \sum_{k=1}^{K} u_k = 1 \), it is necessary to assume \( \mu^N = \mu^{N-1} \).

Finally, the MOAP-solving algorithm takes the following form.

2.1. MOAP-solving algorithm

Step 1. Enter initial data \( u^0 \geq 0, \ w^0 \geq 0, \ N = 0, \ \alpha^N, \beta^N, \ N_{\max} \).

Step 2. Solve \( K \) assignment problems (9) or the problem of independent column minimization (10)–(12).

Step 3. Check: \( \sum_{i=1}^{n_k} \sum_{j=1}^{n_i} x_{ij}^k \leq 1, \ i = \overline{1,m} \) ? If yes, then the optimal solution \( X_k^N, \ k = \overline{1,K} \) is obtained and algorithm is finished. If no, then proceed to step 4.

Step 4. Check: \( N < N_{\max} \) ? If yes, then proceed to step 5. If no, then approximate solution \( X_k^N, \ k = \overline{1,K} \) is obtained and algorithm is finished.

Step 5. Solve the problem (8) for \( \mu \).

Step 6. Recalculate the values of dual variables by formulas

\[
 u_k^{N+1} = u_k^N + \alpha^N \left( \sum_{i=1}^{n_k} \sum_{j=1}^{n_i} c_{ij}^k x_{ij}^{kN} - \mu^N \right), \ k = \overline{1,K},
\]

\[
 w_i^{N+1} = w_i^N + \beta^N \left( \sum_{k=1}^{K} \sum_{j=1}^{n_i} x_{ij}^{kN} - 1 \right), \ i = \overline{1,m},
\]

where \( [x]^+ = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise}. \end{cases} \)

Assume \( N = N + 1 \). Proceed to step 2.

If the inequality \( N < N_{\max} \) is failed, it is possible to get an infeasible point. In this case, we have to correct the obtained solution by reducing the dimension of the cost matrices \( C_k = \{ c_{ij}^k \}, \ k = \overline{1,K} \). If the stopping condition \( \sum_{k=1}^{K} \sum_{j=1}^{n_k} x_{ij}^k = 1 \) is reached on the element \( x_{ij}^b = 1 \) for a given \( i^*, \) then row \( i^* \) is removed from all matrices and column \( l^* \) is removed from matrix \( C_b \).

This leads to reducing the problem size. Thereafter, the MOAP-solving algorithm is restarted for the obtained matrices.

The general block scheme of the solving algorithms for the problem (1)–(6) is presented in Figure 1. The difference between algorithms is in the implementation of the unit highlighted in black (see the Figure 1). The implementation of this unit is determined by the method of defining the set \( S \).
Statement 2 [21]. The computational complexity of single iteration of the algorithm using the Hungarian method is equal to $O(Km^3)$.

Statement 3 [21]. The computational complexity of single iteration of the algorithm using the independent column minimization is equal to $O(mn)$, where $n = \sum_{k=1}^{\xi} n_k$.

3. Results and discussion
A software complex to test and analyze the algorithms proposed in this paper has been developed in the Borland Delphi 7.0. Using this software we have conducted a computational experiment which made it possible to elaborate recommendations for using the algorithms according to problem sizes. The algorithms were tested on the input matrices of different dimension. The number of iterations needed to complete all stops and the algorithm execution time were evaluated.

Further, we introduce the notations:
Algorithm I – algorithm using the Hungarian method;
Algorithm II – algorithm using the independent column minimization.

Table 1 shows the results of the MOAP-solving Algorithm I in each iteration. 100 experiments were carried out for each problem size. The input matrices were filled with random elements between 1 and 100. Each table cell shows the number of these problems that have been solved at the corresponding number of iterations.
Table 1. Number of problems solved by Algorithm I.

| Problem size, $K \times m \times n$ | Number of iterations | Average execution time for one problem (sec) |
|-----------------------------------|----------------------|--------------------------------------------|
|                                   | Less than 50 to 100   | From 100 to 200 | From 200 to 500 | From 500 to 100 | More than 1000 |
| 2×6×3                             | 83                   | 9              | 3              | 2              | 0              | 0.01            |
| 2×10×5                            | 72                   | 10             | 8              | 2              | 2              | 0.05            |
| 2×20×10                           | 55                   | 12             | 11             | 9              | 6              | 0.20            |
| 2×30×15                           | 20                   | 5              | 13             | 15             | 31             | 0.55            |
| 3×9×3                             | 79                   | 11             | 7              | 2              | 1              | 0.02            |
| 3×15×5                            | 45                   | 12             | 8              | 5              | 10             | 0.10            |
| 3×30×10                           | 23                   | 6              | 9              | 13             | 36             | 1.30            |
| 4×12×3                            | 36                   | 14             | 25             | 15             | 7              | 0.07            |
| 4×20×5                            | 27                   | 2              | 3              | 16             | 23             | 0.25            |

Comment on the Table 1. For example, consider the second line corresponding to the problem size 2×10×5: 72 problems (of hundred) are solved at less than 50 iterations, the average time for solving one problem is about 0.05 seconds.

As seen from Table 1, the algorithm has a great convergence rate for problems with $m < 10$ in most cases. In addition, only 2 problems are not solved at 1000 iterations. If $10 < m < 20$ about 90% of problems are solved at 1000 iterations.

When parameter $m$ increases, the number of problems unsolved at 1000 iterations increases.

The similar conclusions can be drawn for a parameter $K$.

Consider the results of the MOAP-solving Algorithm II in each iteration.

As seen from Table 2, the algorithm has a great convergence rate only for smaller sized problems in most cases. More than 10% of problems are not solved at 1000 iterations if $m = 15$.

Table 2. Number of problems solved by Algorithm II.

| Problem size, $K \times m \times n$ | Number of iterations | Average execution time for one problem (sec) |
|-----------------------------------|----------------------|--------------------------------------------|
|                                   | Less than 50 to 100   | From 100 to 200 | From 200 to 500 | From 500 to 100 | More than 1000 |
| 2×6×3                             | 78                   | 13             | 2              | 5              | 0              | 2              | <0.001          |
| 2×10×5                            | 47                   | 12             | 8              | 14             | 5              | 4              | 0.005           |
| 2×20×10                           | 2                    | 5              | 19             | 18             | 23             | 30             | 0.015           |
| 2×26×13                           | 0                    | 23             | 14             | 12             | 6              | 30             | 0.02            |
| 3×9×3                             | 57                   | 2              | 5              | 0              | 6              | 10             | <0.001          |
| 3×15×5                            | 3                    | 56             | 9              | 13             | 3              | 16             | 0.01            |
| 3×21×7                            | 0                    | 4              | 13             | 42             | 12             | 29             | 0.02            |
| 4×12×3                            | 22                   | 41             | 10             | 13             | 9              | 5              | 0.01            |
| 4×20×5                            | 0                    | 29             | 13             | 8              | 18             | 32             | 0.04            |

Evaluate now the performance of the both methods by solving the larger sized problems. As seen from Tables 1 and 2, the number of iterations and average execution time increase in this case.

In this connection, the stopping criterion is algorithm execution time or total number of iterations (depending on the solved problem). In this situation, we obtain interim solution for which the constraints (5) are violated.
The average number of unfulfilled constraints (5) in the first and last iterations for both algorithms is presented in Tables 3 and 4. The algorithm execution time is 60.0 seconds. 500 experiments were carried out for each problem size. The input matrices were filled with random elements between 1 and 100.

### Table 3. The average number of unfulfilled constraints (5) for Algorithm I.

| Problem size, $K \times m \times n$ | Average number of constraints violated in the first iteration | Average number of constraints violated in the last iteration |
|-----------------------------------|------------------------------------------------------------|------------------------------------------------------------|
| $2 \times 200 \times 100$         | 51.7                                                      | 23.9                                                      |
| $3 \times 150 \times 50$         | 39.9                                                      | 22.0                                                      |
| $4 \times 200 \times 50$         | 51.2                                                      | 32.5                                                      |

### Table 4. The average number of unfulfilled constraints (5) for Algorithm II.

| Problem size, $K \times m \times n$ | Average number of constraints violated in the first iteration | Average number of constraints violated in the last iteration |
|-----------------------------------|------------------------------------------------------------|------------------------------------------------------------|
| $2 \times 200 \times 100$         | 53.2                                                      | 15.6                                                      |
| $3 \times 150 \times 50$         | 40.0                                                      | 5.3                                                       |
| $4 \times 200 \times 50$         | 47.9                                                      | 17.9                                                      |

As seen from Tables 1 and 2, the Algorithm I performs more stops and has a greater convergence rate at 1000 iterations versus the Algorithm II. However, the computational complexity of one iteration of Algorithm I is higher than the computational complexity of one iteration of Algorithm II, and as consequence, the Algorithm I execution time is greater than Algorithm II running time. Therefore, Algorithm II performs a greater number of iterations and stops than the Algorithm I at the same time. Tables 3 and 4 confirm this fact.

Note that the convergence of the subgradient method is essentially dependent on the type of step-size rules for $\alpha^N$ and $\beta^N$. It is known [18] that the subgradient method converges if any step sizes satisfy:

$$\sum_{N=1}^{\infty} \alpha^N = +\infty,$$

where $\alpha^N \to 0$, $N \to \infty$.

The computational experiment showed that if $\alpha^N = 1/(N + 1)$, the algorithms perform efficiently in any sized problems. As for $\beta^N$, step-size rule $\beta^N = \max_{i,j,k} c^k_{ij}/(N + 1)$ gives great convergence rate of algorithms only if $m < 20$.

Consider the different types of step-size rules for $\beta^N$ for MOAP $3 \times 30 \times 10$. The results of the computational experiment are presented in Table 5. Each table cell shows the number of problems that have been solved at the corresponding number of iterations.

As seen from Table 5, it is more efficient to use the following step-size rule for considered problem:

$$\beta^N = \begin{cases} \max_{i,j,k} c^k_{ij}/(N + 1), & N < \max_{i,j,k} c^k_{ij}, \\ 1 - N/(N_{\max} + 1), & \text{otherwise.} \end{cases}$$
effective ways to specify algorithm parameters for the more efficient than the Algorithm I

dual Uzawa method has been applied. The approximate MOAP solution has been obtained about 90% of problems are solved at 1000 iterations by both algorithms; the main stopping criterion in this case is constraints (5). Increasing the size problem leads to the decreasing the number of exactly solved problems; the stopping criterion in this case is number of iterations.

3. The Algorithm I is more efficient than the Algorithm II when stopping criterion is number of iterations.

4. The Algorithm II is more efficient than the Algorithm I when stopping criterion is algorithm execution time.

5. The presented algorithms are polynomial (see the Statements 2 and 3).

4. Conclusion

The considered multi-objective assignment problem is NP-hard and the pertinent question is to develop the heuristic solving algorithm for it. The approximate MOAP solution has been obtained using properties of the primal and dual problems. The dual Uzawa method has been applied.

The proposed algorithms are based on the dual problem solving with the help of Lagrangian relaxation of some constraints of the primal problem. The structure of the set formed by these constraints radically influences on the solving algorithms.

The computational experiment has made it possible to analyse the convergence rate of algorithms. The effective ways to specify algorithm parameters for the given problem sizes are obtained. The suggestions for opting the solving algorithm according to the problem size are developed.

References

[1] Larichev O I 2000 Theory and Methods of Decision-Making, and also the Chronicle of Events in the Enchanted Countries (Moscow: Logos) (in Russian)

[2] Nikonov O Ya, Podolyaka O A, Podolyaka A N and Skakalina E V 2011 Bulletin of KhNAHU 55 pp 103-112

[3] Baeva N B and Bondarenko Yu V 2003 Theoretical Framework Computational Schemes of Vector Optimization (Voronezh: Publishing House of VSU) (in Russian)

[4] Shvare D T 2013 Science and education. (Moscow: BMSTU) (2013)003 (in Russian)

[5] Steuer R E 1986 Multiple Criteria Optimization: Theory, Computation, and Application (New York: Wiley)

[6] Kashirina I L and Semenov B A 2007 Information Technology 5 pp 62-68 (in Russian)
[7] Podinovskij V V and Nogin V D 2007 Pareto-Optimal Solutions of Multicriteria Problems (Moscow: Nauka) (in Russian)
[8] Volkonskij V A, Eganyan G K and Pomanskij A B 1983 Siberian Mathematical Journal 24(2) pp 9-17
[9] Brucker P 1972 Operat. Res. 16(5) pp 189-197
[10] Emelichev V A and Perepelica V A 1988 Bulletin of the Georgian SSR Academy of Sciences 131(3) pp 501-504 (in Russian)
[11] Perepelica V A 1989 Multicriteria Problems of Graph Theory. Algorithmic Approach (Kiev: UMK VO) (in Russian)
[12] Sergienko I V and Perepelica V A 1987 Cybernetics 5 pp 85-93 (in Russian)
[13] Aneja P and Nair K R 1979 Manag. Sci. 25(1) pp 73-78
[14] Emelichev V A and Perepelica V A 1989 Computational Mathematics and Mathematical Physics 29(2) pp 171-183 (in Russian)
[15] Papadimitriou C H and Steiglitz K 1998 Combinatorial Optimization. Algorithms and Complexity (New York: Dover)
[16] Perepelitza V A 1989 14-th IFIP Conf. on Syst. Mod. and Opt. (Leipzig, GDR) pp 13-14
[17] Kakhichko A A 1986 Methods for Solving Nonlinear Problems and Data Processing: Collection (Dnipropetrovsk: Leningrad State University) pp 41-44 (in Russian)
[18] Minoux M 1990 Mathematical Programming: Theory and Methods (Moscow: Nauka) (in Russian)
[19] Medvedeva O A 2013 Bulletin of St. Petersburg University: Applied Mathematics. Computer Science. Control Processes 10(1) pp 85-94 (in Russian)
[20] Medvedeva O A, Medvedev S N and Tchernyshova G D 2013 Bulletin of Voronezh State University. Series: Systems analysis and information technologies 2 pp 38-41 (in Russian)
[21] Medvedeva O A 2013 Models and algorithms of solving assignment problems with additional constraints: PhD dissertation (Mathematics) (Voronezh: Voronezh State University) p 138