Kovalova K. O.
Candidate of Technical Sciences, Associate Professor
Simon Kuznets Kharkiv National University of Economics

Misiura Ie. Iu.
Candidate of Technical Sciences, Associate Professor
Simon Kuznets Kharkiv National University of Economics

Pivavar I. V.
Candidate of Economic Sciences, Associate Professor
Simon Kuznets Kharkiv National University of Economics

STUDY OF TRANSPORTATION PROBLEM WITH FICTITIOUS COSTS USING A GENETIC ALGORITHM IN MATLAB

Summary
A transportation problem with fictitious costs is a linear programming problem with a discontinuous objective function. The chapter provides the possibility of transposing this problem to the problem of integer linear programming. The authors' starting point was Balinski's method. However, this approach allows solving problems only by approximate methods. In this connection, the authors propose to solve the problem using the genetic algorithm of the software environment MATLAB. At the same time, they were supposed to develop their unique functions of crossing, mutation and population. In the current chapter, the situational economic transportation problem is presented through an example where the authors show the basic aspects of its solution. Namely, a mathematical model was developed and a numerical experiment was performed in MATLAB, which shows a fairly good result.

Introduction

Almost all large companies have departments of logistics. As it is well known, logistics solves such problems as reducing supply chain transportation costs, choosing the shortest path problems, the improved transit times, simplifying the delivery schemes, reducing additional transportation costs. Transportation costs can be significant parts of a company's overall logistics spend [1]. Minimizing transportation routes [2] and improving transit times [3; 4] are two other ways to improve the economic efficiency of companies' transportation. Simplifying the delivery schemes means to develop models for calculating comparable combined internal and external costs of intermodal and road freight transport networks. Wherein, it is usually considered that internal costs consist of the operational-private costs borne by the transport and intermodal terminal operators, and the time costs of goods tied in transit. And the external costs include the costs of the impacts of both networks on society and the environment such as local and global air pollution, congestion, noise pollution, and traffic accidents [5; 6].

For all the above problems, a mathematical tool − what is called transportation problem − has been developed [7]. The transportation problem is one of the subclasses of linear programming problem where the objective is to transport various quantities of a single homogeneous product that are initially stored at various origins, to different destinations in such a way that the total transportation is minimum. The classical transportation problem belongs to the class of T-problems and its solution does not cause any difficulty. However, the modern growth of large companies' economic activity has generated a lot of new transportation problems, mathematical modelling and software implementation of which still causes embarrassments in their solutions. These tasks include: transportation problems with limited capacity (Td-task), problems with heterogeneous cargo, multi-index problems, transportation problems by the time criterion, etc. A transportation problem with fictitious costs is the most difficult of them [8]. This is due to the fact that this is a problem of integer linear programming with a discontinuous cost function.

Theoretically, any general mixed integer programming solution methods can be used to solve this kind of problem, for example, branch and bound method. However, these methods are generally inefficient and computationally expensive for the problems with large size. According to research [9−11], such problems of integer linear programming with discontinuous cost functions can be investigated and solved using genetic algorithms. Many prior works have been done only on the applications of recommender systems, but there are a lot of works that give practical decisions of these problems. For example, Huang et al (2015) used genetic algorithms to solve carpool service problems in cloud computing. They utilized the concept of Genetic-based Carpool Route and Matching Algorithm (GCRMA). Mesbah et al (2011) used a genetic algorithm to optimize transit priority in transportation problem. They offered a parallel genetic algorithm, which has considerably shorter execution time. Lau et al. (2010) used genetic algorithms to solve the multidepot vehicle routing problem. They proposed to use a stochastic search technique called Fuzzy Logic Guided Genetic Algorithms (FLGA). The short review of the literature shows that the use of genetic algorithms is individual: the genetic approach for optimization does not have the character of universality in general. The main processes of genetic algorithms (mutation, crossover, and selection) are created as functions or are modified in accordance with the specific formulation of the problem by many scientists. According to [12], different problems usually have different data structures or genetic representations. Others scientists (Solomon, Thompson, Psaraftis, Russell, Antes, Derigs, Cordone, Wolfer-Calvo, Caseau, Laburthe, Bräysy, Rochat, Taillard, Chiang, Cordeau, Thangiah, Homberger, Gehring, Mester, Barkaoui, Csiszár, Van Henlenryck and others) agreed that new economics transportation problem with a discontinuous cost function needs new modification in existing algorithms [13−15].

In connection with the foregoing, the study of transportation problems with fictitious costs as problems of integer programming with discontinuous cost functions is still *relevant*. The need to solve such transportation problems by minimizing the transportation costs is determined by the great economic effect since this clearly increases the profit of the enterprise. Exact methods for solving transportation problems enable us to find solutions for only problems of small dimension. To solve problems of large dimension, exact methods are not effective due to their large time costs. However, right now, effective algorithms for solving large-scale problems are required since currently globalization processes in the economy have been observed. This gives a rise to the search for solutions in the field of the possibility of applying genetic algorithms for optimization problems.

The *subject* of this monograph is transportation problem with fictitious costs as a transportation problem of large dimension with a discontinuous cost function.

The paper focuses on the mathematical and digital models of the transportation problem. For this type of problem, there are given general content and mathematical statements, a specific example and its solution.

The *purpose* of the study is to develop a modification of the main functions of the genetic algorithm (selection, crossover, and mutation) in accordance with the specifics of the problem. This work presents software-algorithmic modules for solving the transportation problem based on the genetic algorithm of the built-in function of the MATLAB software environment. This presents an enormous *practical value* of this work and can be further usable for graduate students, young scientists and specialists in the field of genetic algorithms.

Part 1. Description of the problem

A transportation problem with fictitious costs means a problem with a discontinuous cost function. As in the classical transportation problem, let $i=\overline{1,m}$ denote sources and let $j=\overline{1,n}$ denote destinations. Let the quantities be given:

a_i − the amount of supply available at source *i*;

b_j − the demand required at destination *j*.

We need to find x_{ij} (the quantity transported from source *i* to destination *j*), which satisfies natural transport bounds:

$$x_{ij} \geq 0,\ \sum_{j=1}^{n} x_{ij} = a_i,\ \sum_{i=1}^{m} x_{ij} = b_j,\ i=\overline{1,m},\ j=\overline{1,n}, \tag{1}$$

and minimizes the function

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\left(x_{ij}\right) \quad . \qquad (2)$$

Each $c_{ij}\left(x_{ij}\right)$ in (2) has the form

$$c_{ij}\left(x_{ij}\right)=\begin{cases} 0,\, x_{ij}=0 \\ c_{ij}x_{ij}+d_{ij},\, x_{ij}>0 \end{cases} \quad . \qquad (3)$$

There c_{ij} is the cost of transporting one unit between source *i* and destination *j*. The numbers d_{ij} can be interpreted as the rent of transport (do not depend on the loading of vehicles) or the cost of building a highway from the source *i* to the destination *j* (does not depend on future units).

The problem (1) − (3) is called a transportation problem with fictitious costs, or an inhomogeneous transportation problem. It is clear that if all $d_{ij}=0$ then the problem turns into a classical transportation problem. Otherwise, this problem, due to the discontinuity of each term (3) at zero, drops out of the framework of linear programming altogether. However, by introducing additional integer variables, it can be reduced to a partially integer linear programming problem.

The method of such information was indicated by M. L. Balinski [16]. It consists of the following. Let us find the quantities

$$M_{ij}=\min\{a_i, b_j\},\ i=\overline{1,m},\ j=\overline{1,n}. \qquad (4)$$

We consider the minimization problem

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}+d_{ij}y_{ij} \qquad (5)$$

taking into account conditions (1) and the additional condition

$$y_{ij}=\begin{cases}0, \\ 1,\end{cases} x_{ij}\le M_{ij}y_{ij} \quad . \qquad (6)$$

The partially integer-valued problem (1), (5), (6) turns out to be equivalent to the original problem (1) − (3). Indeed, for $y_{ij}=0$ by (6) $x_{ij}=0$ will be automatically, and for $y_{ij}=1$ the inequalities (6) become redundant, since in any admissible plan of the transportation problem the conditions $x_{ij}\le M_{ij}$ are always satisfied. On the contrary, if $x_{ij}=0$ in some optimal plan of problem (1), (5), (6) then the corresponding y_{ij} must be zero (if it turns out to be positive, the plan will not be optimal, since by decreasing this y_{ij} we do not break bounds and give the objective function (5) a smaller value). Finally, if $x_{ij} > 0$, then, by conditions (6), y_{ij} can be only 1.

Thus, the integer variables y_{ij} in this formulation match the construction of highways between *i* and *j* (or, depending on the interpretation, the rent of transport for this highway) and movement between source *i* and destination *j*.

Based on this information, an approximate method of solving the transport problem with fictitious costs is described in [16].

We now consider a more general model with an inhomogeneous discontinuous objective function. The authors formulated it in terms close to the problem of mixtures, although such an interpretation is not the only possible one. We denote the components of the mixture by $i = \overline{1, n}$. The components of this mixture we denote by $j = \overline{1, m}$. Let the quantities be given:

a_{ij} – the components of the mixture;

b_i – lower bounds of each element of the mixture;

c_j – *j*-component purchase cost;

d_{ij} – fixed price of the *j*-component, which not depend on the quantity ordered.

It is required to compile the cheapest mixture that satisfying the given bounds. We denote the required quantities of components by x_j. Then the problem can be represented as the minimization problem

$$\sum_{j=1}^{n} c_j\left(x_j\right) \to \min \tag{7}$$

under conditions

$$x_j > 0,\ \sum_{j=1}^{n} a_{ij} x_j \geq b_i,\ i = \overline{1, m} \quad , \tag{8}$$

where

$$c_j\left(x_j\right) = \begin{cases} 0, & x_j = 0 \\ c_j x_j + d_j, & x_j > 0 \end{cases} \quad . \tag{9}$$

The authors want to remind that the requirement of integer-valuedness on the variables x_j is not imposed.

This problem can be reduced to a general partially integer linear programming problem. Let us suppose, in addition to the conditions (8), the upper bounds for the variables are also given:

$$x_j \le k_j,\ j = \overline{1, n}. \tag{10}$$

We consider the minimization problem

$$\sum_{j=1}^{n} c_j x_j + d_j y_j \tag{11}$$

with the conditions (8) and the additional condition

$$y_j = \begin{cases} 0, \\ 1, \end{cases} 0 \le x_j \le k_j y_j\ . \tag{12}$$

The equivalence of the partially integer-valued problem (11), (8), (12), and the original problem are established by arguments completely analogous to those in [10, 13, 16] they can be left to the reader.

We note that the possibility of reducing tasks with a discontinuous objective function to partially integer ones is based on the presence of upper bounds for variables. This technique is a general characteristic of integer programming. In the transportation problem, these boundaries are determined internally (see (4)); in the general problem, they must be postulated or defined in a special way. In some cases, the boundaries are calculated simply; for example, if all $a_{ij} \ge 0$ we can take k_j which is describing by the formula:

$$k_j = \min \frac{b_i}{a_{ij}},\ j = \overline{1, n}\ , \tag{13}$$

where the minimum is taken for all *i f*or which $a_{ij} > 0$.

Balinski's method gives an approximate value for each of the variables, and in addition, it gives an upper and lower bound for the optimal value of the objective function. Unfortunately, it gives no indication of how the approximate solution is close to the optimal solution, and one has no way of predicting how the upper and lower bounds are close to the optimal solution. For that reason, the authors find it expedient to apply the genetic algorithm to the solution of transportation problem with fictitious costs, similar to a number of works [8−13]. However, the authors plan to use built-in MATLAB function − gamultiobj that uses to create a set of points on the Pareto front. Gamultiobj uses a controlled, elitist genetic algorithm (a variant of NSGA-II [17]). An elitist genetic algorithm always favours individuals with better fitness value (rank). A controlled elitist genetic algorithm also favours individuals that can help increase the diversity of the population even if they have a lower fitness value. However, based on the specifics of the mathematical formulation of this problem, the authors see it necessary to develop unique algorithms for the three main processes of the genetic algorithm: crossing, selection, and mutation.

Part 2. Genetic algorithms in MATLAB

The first main task of this section is to expand each parameter of the genetic algorithms that are customizable in MATLAB. The authors give an explanation about the conditions of setting options for the genetic algorithm of Matlab. In order to obtain the best results, as a rule, calculations are usually made with different options. The choice of the best type of option values is based on trial and error. In this section, there are certain methods for selecting options in order to improve the results obtained in accordance with the specificity of the transportation problem.

As it is well known, genetic algorithms are a method for solving optimization problems based on the biological principles of natural selection and evolution. Each genetic algorithm repeats a certain number of times the procedure for modifying a population (a set of individual solutions), seeking to obtain new sets of solutions (new populations). At the same time, at each step, the “parents” are selected from the population. The joint modification of these decisions (crossing) leads to the formation of a new individual in the next generation. The genetic algorithm uses three types of rules, on the basis of which one forms a new generation: the rules for selection, crossing, and mutation. The mutation allows, by making changes to the new generation, to avoid falling into the local minima of the optimized function.

The mechanism of working with genetic algorithms in the MATLAB environment is implemented in two ways:

1. Calling the function of genetic algorithms
2. Using the Genetic Algorithm Tool

Both methods are supplied as a standard set of MATLAB functions and modules. According to the opinion of the authors, the second option with genetic algorithms in MATLAB connected with the use of the Genetic Algorithm Tool is much more convenient and vivid. We will consider it in more details.

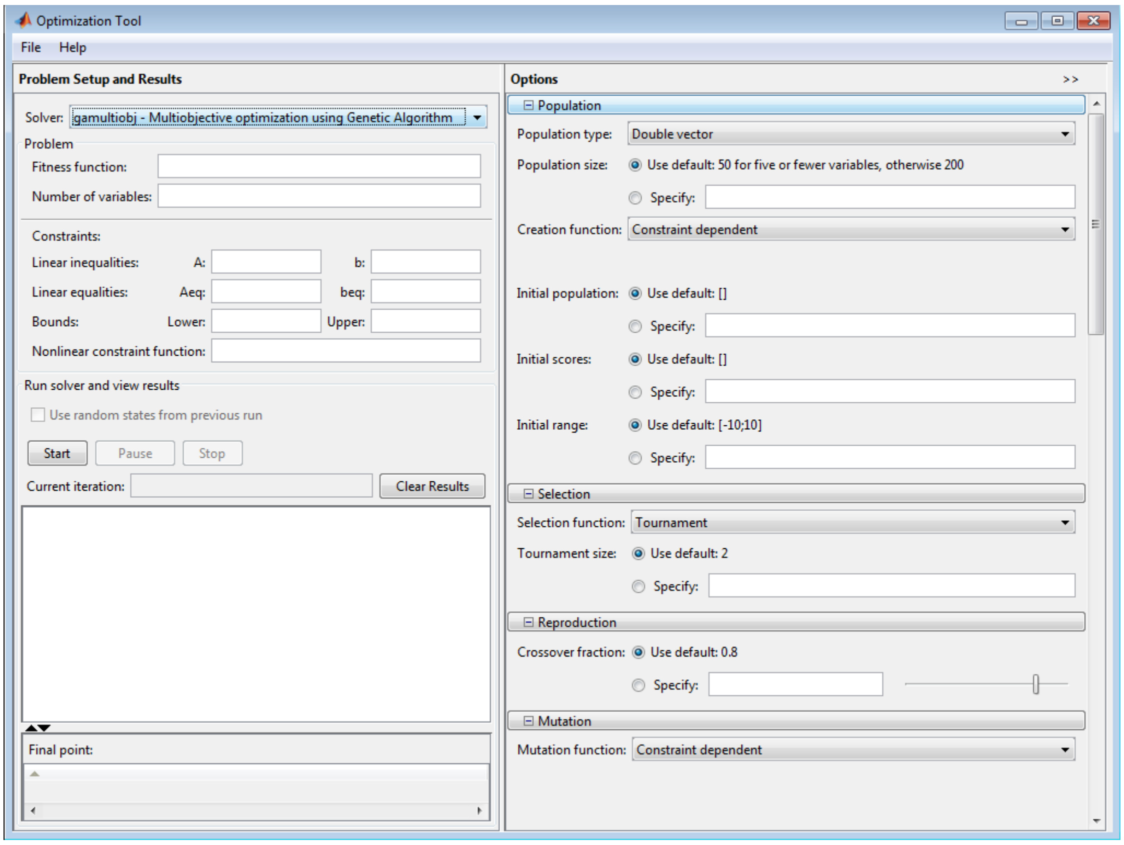
To run the Genetic Algorithm Tool on the MATLAB command line, run the command “optimtool”. After that, a package of genetic algorithms begins working and the main window of the utility will appear on the screen (Fig. 1).

The *Solver* allows selecting the built-in MATLAB function, which implements one of the optimization methods.

The *Fitness function* field specifies an optimizable function in the form @fitnessfun where fitnessfun.m is the name of the M-file, in which the function to be optimized must be described first.

The *Number of variables* field specifies the length of the input vector of the optimized function.

In the *Constraints* panel, we can specify constraints or a limiting non-linear function. In the *Linear inequalities* field, a linear constraint is defined by an inequality of the form: $A * x \leq b$. In the *Linear equalities* field of this panel, linear constraints are defined by the equality: $A * x = b$. In both cases, A is some matrix, b is a vector.



Solver: gamultiobj - Multiobjective optimization using Genetic Algorithm

Problem
Fitness function:
Number of variables:

Constraints:
Linear inequalities: A: b:
Linear equalities: Aeq: beq:
Bounds: Lower: Upper:
Nonlinear constraint function:

Run solver and view results
☐ Use random states from previous run
Current iteration:

Population
Population type: Double vector
Population size: [✓ Use default: 50 for five or fewer variables, otherwise 200] Specify:
Creation function: Constraint dependent
Initial population: [✓ Use default: []] Specify:
Initial scores: [✓ Use default: []] Specify:
Initial range: [✓ Use default: [-10;10]] Specify:

Selection
Selection function: Tournament
Tournament size: [✓ Use default: 2] Specify:

Reproduction
Crossover fraction: [✓ Use default: 0.8] Specify:

Mutation
Mutation function: Constraint dependent

Fig. 1. Optimization Tool utility window

In the *Bounds* field, the lower and upper bounds of variables are specified, and in the *Nonlinear constraint function* field can be specified an arbitrary nonlinear constraint function.

If a specific problem does not require constraints, all fields in the *Constraints* panel should be left blank.

The *Run Solver panel* contains control elements (Start, Pause, and Stop buttons to start, temporarily and completely stop the work of the genetic algorithm). It also contains the *Status and results* fields, which display the current results of the running genetic algorithm, and Final point, which displays the value of the end point of the algorithm, the best value of the function being optimized (the desired value).

On the right side of the main window of the Optimization Tool utility is the *Options* (Fig. 1). It allows setting various settings for the operation of genetic algorithms. The main configurable parameters in Optimization Tool are:

– population (Population tab);
– selection operator (Selection tab);
– reproduction operator (Reproduction tab);
– mutation operator (Mutation tab);
– operator crossing (Crossover tab);
– transfer of individuals between populations (Migration tab);
– special parameters of the algorithm (Multiobjective problem settings tab);

− hybrid function specification (Hybrid function tab);

− setting the criterion for stopping the algorithm (Stopping criteria tab);

− output of various additional information on the course of the genetic algorithm (Plot Functions tab);

− output of the results of the algorithm in the form of a new function (Output function tab);

− adjust a set of information for output to the command window (Display to command window tab);

− a method of calculating the values of optimized and bounding functions (User function evaluation tab).

Let's take a closer look at all the above tabs and the elements that they contain.

In the *Population* tab, the user can select the type of mathematical objects, to which the individuals of all populations will belong (double vector, bit string or custom type). It should be taken into account that the use of bit string and user types imposes restrictions on the list of permissible operators for creating, mutating, and crossing individuals. Also, the population tab allows adjusting the size of the population (how many individuals will consist of each generation) and how the initial generation will be created (*Uniform* − if there are no restrictions imposed, otherwise − *Feasible population*). In addition, it is possible to set the initial generation (using the Initial population item) or a part of it manually, the initial rating of individuals (Initial scores item), and enter the restrictive numerical range, to which the initial population should belong.

The *Selection* tab allows selecting the parental selection operator based on the data from the scaling function.

The *Reproduction* tab specifies how the creation of new individuals takes place. Elite count allows specifying the number of individuals that are guaranteed to move into the next generation. The Crossover fraction indicates the proportion of individuals that are created by crossing. The rest is created by mutation.

In the *Mutation* tab, the mutation operator type is selected.

The *Crossover* tab allows selecting the type of cross operator (single point, two point, intermediate, heuristic, arithmetic or scattered, which generates a random binary parent match vector). There is also the option of specifying a custom cross function.

In the *Migration* tab, rules can be configured according to which individuals will move between subpopulations within the same population. Subpopulations are created when a vector is specified as the size of the population and not as a natural value. In this tab we can specify the direction of migration (forward − to the next subpopulation, both − to the previous and next), the proportion of migrating individuals and the frequency of migration (how many generations pass between migrations). If the creation of subpopulations is not required, this tab should always be left unchanged.

The *Multiobjective problem settings* tab allows customizing distance measure function and Pareto front population function. A Pareto front is a set of points in parameter space (the space of decision variables) that have non-inferior fitness function values. In other words, for each point on the Pareto front, we can improve one fitness function only by degrading another.

The *Hybrid function* tab allows specifying one more minimization function that will be used after the algorithm finishes.

In the *Stopping criteria* tab, you specify the situations in which the algorithm makes a stop.

The *Plot Functions* tab allows selecting various information that is displayed in the course of the algorithm's operation and shows both the rightness of its operation and the specific results achieved by the algorithm.

The *Output function* tab allows displaying the history of the algorithm in a separate window with the specified interval of generations (the History to new window flag and the Interval field, respectively) and also allows specifying and outputting any arbitrary output function specified in the Custom field function.

The *User function evaluation* tab describes, in which order the values of the optimized and bounding functions are calculated (separately, in parallel in one call or simultaneously).

Finally, the *Display to command window* tab allows configuring the information that is displayed in the main MATLAB command window while the algorithm is running.

During the writing stage of this section, the authors used their personal experience and Help of the MATLAB environment. In the next section, the authors demonstrate the aforementioned mechanism through a specific example of a transportation problem with fictitious costs.

Part 3. The solution of the transportation problem with fictitious costs

Substantive statement of the transportation problem with fictitious costs. In general, this problem is formulated as follows. Let j denote transportation roads, so it is necessary to perform $b_j,\ j=\overline{1,n}$ deliveries. Let i denote vehicles in quantity $a_i, i=\overline{1,m}$. In this case, the required time for transporting goods by i vehicle on j way is t_{ij}, and transportation costs are c_{ij}. In addition, it is necessary to take into account previous transportation works, which consume time t_{ij}^{+} and money c_{ij}^{+}. It is necessary to optimally distribute existing vehicles $a_i, i=\overline{1,m}$ along roads $b_j,\ j=\overline{1,n}$.

Mathematical formulation of the transportation problem with fictitious costs. Let X_{ij} denote the number of flights that i vehicle must perform on j way. Then the mathematical model of the transportation problem with fictitious costs has the form:

$$F=\sum_{i=1}^{m}\sum_{j=1}^{n}s_{ij}\left(x_{ij}\right)\to\min, \tag{14}$$

with the following restrictions:

$$\sum_{j=1}^{n} h_{ij}\left(x_{ij}\right) \le a_i, \quad i = \overline{1,m}, \tag{15}$$

$$\sum_{i=1}^{m} x_{ij} = b_j \ , \quad j = \overline{1,n}, \tag{16}$$
$$i$$

$$x_{ij} \ge 0, \quad i = \overline{1,m},\ j = \overline{1,n}, \tag{17}$$

$$x_{ij} = \text{int}, \quad i = \overline{1,m},\ j = \overline{1,n}. \tag{18}$$

In this case, the functions included in the formulas (1) and (2) are described by the following dependencies:

$$s_{ij}\left(x_{ij}\right) = \begin{cases} 0, & x_{ij} = 0 \\ c_{ij}x_{ij} + c_{ij}^{+}, & x_{ij} > 0; \end{cases} \tag{19}$$

$$h_{ij}\left(x_{ij}\right) = \begin{cases} 0, & x_{ij} = 0 \\ t_{ij}x_{ij} + t_{ij}^{+}, & x_{ij} > 0. \end{cases} \tag{20}$$

Thus, the quantities x_{ij} (the number of flights) that satisfy the transportation constraints (15) − (18) and minimize the function (14) are being sought. Dependencies (19) and (20) are interpreted as costs (transportation and time) for transporting goods by i vehicle on j way.

An example of a solution to a situational economic transportation problem with fictitious costs. A company has three production lines: soda water production, juice production, and beverage production. On the way to these productions, it is necessary to perform $\begin{bmatrix}11 & 7 & 9\end{bmatrix}$ flights, using three types of vehicles. The useful time of vehicles of each type is $\begin{bmatrix}100 & 130 & 250\end{bmatrix}$. Time and transportation costs for the transport vehicle are described by the following matrices: $T = \begin{pmatrix} 4 & 3 & 3 \\ 2 & 2 & 3 \\ 2 & 2 & 2 \end{pmatrix}$, $C = \begin{pmatrix} 5 & 5 & 4 \\ 3 & 3 & 5 \\ 4 & 3 & 4 \end{pmatrix}$. The time t_{ij}^{+} which spends on previous transportation works and the transportation costs c_{ij}^{+} for carrying out these works are given according to the matrices: $T^{+} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 3 & 2 \end{pmatrix}$, $C^{+} = \begin{pmatrix} 1 & 2 & 2 \\ 1 & 1 & 2 \\ 2 & 1 & 1 \end{pmatrix}$. It is necessary to optimally distribute existing vehicles $a_i, i = \overline{1,3}$ by roads $b_j,\ j = \overline{1,3}$.

The mathematical model of this problem, taking into account the notation adopted for the general model (14) − (20), will have the form:

$$F = s_{11} + s_{12} + s_{13} + s_{21} + s_{22} + s_{23} + s_{31} + s_{32} + s_{33} \to \min, \tag{21}$$

where

$$\begin{aligned} &f_1 = h_{11} + h_{12} + h_{13} \le 100, \\ &f_2 = h_{21} + h_{22} + h_{23} \le 130, \\ &f_3 = h_{31} + h_{32} + h_{33} \le 250, \\ &f_4 = x_{11} + x_{21} + x_{31} = 11, \\ &f_5 = x_{12} + x_{22} + x_{32} = 7, \\ &f_6 = x_{13} + x_{23} + x_{33} = 9, \\ &x_{ij} \ge 0, i = \overline{1,3},\ j = \overline{1,3},\ x_{ij} = \text{int}. \end{aligned} \qquad (22)$$

$$s_{11} = \begin{cases} 0, & x_{11} = 0, \\ 5x_{11} + 1, & x_{11} > 0, \end{cases} \quad s_{12} = \begin{cases} 0, & x_{12} = 0, \\ 5x_{12} + 2, & x_{12} > 0, \end{cases}$$

$$s_{13} = \begin{cases} 0, & x_{13} = 0, \\ 4x_{13} + 2, & x_{13} > 0, \end{cases} \quad s_{21} = \begin{cases} 0, & x_{21} = 0, \\ 3x_{21} + 1, & x_{21} > 0, \end{cases}$$

$$s_{22} = \begin{cases} 0, & x_{22} = 0, \\ 3x_{22} + 1, & x_{22} > 0, \end{cases} \quad s_{23} = \begin{cases} 0, & x_{23} = 0, \\ 5x_{23} + 2, & x_{23} > 0, \end{cases} \qquad (23)$$

$$s_{31} = \begin{cases} 0, & x_{31} = 0, \\ 4x_{31} + 2, & x_{31} > 0, \end{cases} \quad s_{32} = \begin{cases} 0, & x_{32} = 0, \\ 3x_{32} + 1, & x_{32} > 0, \end{cases}$$

$$s_{33} = \begin{cases} 0, & x_{33} = 0, \\ 4x_{32} + 1, & x_{33} > 0, \end{cases}$$

$$h_{11} = \begin{cases} 0, & x_{11} = 0, \\ 4x_{11} + 1, & x_{11} > 0, \end{cases} \quad h_{12} = \begin{cases} 0, & x_{12} = 0, \\ 3x_{12} + 2, & x_{12} > 0, \end{cases}$$

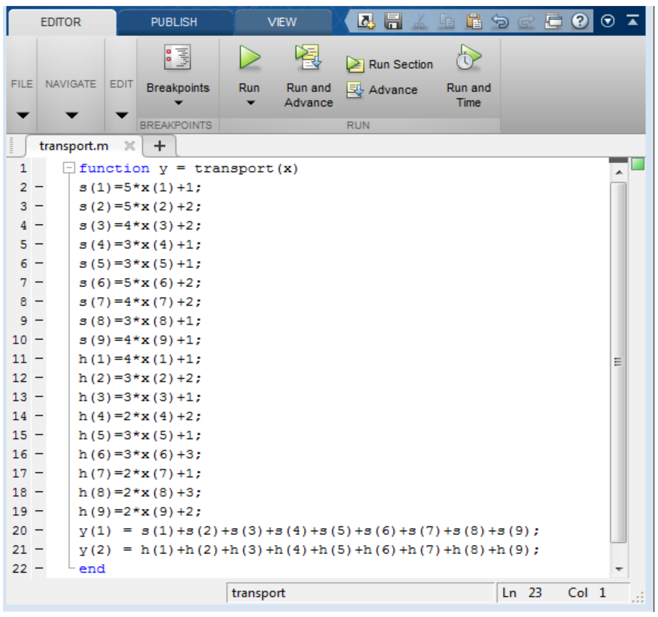
$$h_{13} = \begin{cases} 0, & x_{13} = 0, \\ 3x_{13} + 1, & x_{13} > 0, \end{cases} \quad h_{21} = \begin{cases} 0, & x_{21} = 0, \\ 2x_{21} + 2, & x_{21} > 0, \end{cases}$$

$$h_{22} = \begin{cases} 0, & x_{22} = 0, \\ 3x_{22} + 1, & x_{22} > 0, \end{cases} \quad h_{23} = \begin{cases} 0, & x_{23} = 0, \\ 3x_{23} + 3, & x_{23} > 0, \end{cases} \qquad (24)$$

$$h_{31} = \begin{cases} 0, & x_{31} = 0, \\ 2x_{31} + 1, & x_{31} > 0, \end{cases} \quad h_{32} = \begin{cases} 0, & x_{32} = 0, \\ 2x_{32} + 3, & x_{32} > 0, \end{cases}$$

$$h_{33} = \begin{cases} 0, & x_{33} = 0, \\ 2x_{32} + 2, & x_{33} > 0. \end{cases}$$

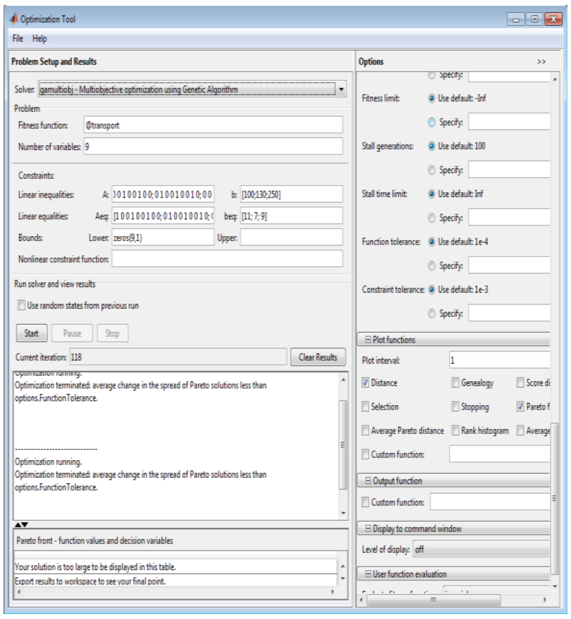
Digital model and solution of the transportation problem with fictitious costs in MATLAB software environment. This example has a fitness function $f(x)$, where x is matrix 3 by 3. Let us create an M-file for this function before proceeding (Fig. 2).



```
function y = transport(x)
s(1)=5*x(1)+1;
s(2)=5*x(2)+2;
s(3)=4*x(3)+2;
s(4)=3*x(4)+1;
s(5)=3*x(5)+1;
s(6)=5*x(6)+2;
s(7)=4*x(7)+2;
s(8)=3*x(8)+1;
s(9)=4*x(9)+1;
h(1)=4*x(1)+1;
h(2)=3*x(2)+2;
h(3)=3*x(3)+1;
h(4)=2*x(4)+2;
h(5)=3*x(5)+1;
h(6)=3*x(6)+3;
h(7)=2*x(7)+1;
h(8)=2*x(8)+3;
h(9)=2*x(9)+2;
y(1)  = s(1)+s(2)+s(3)+s(4)+s(5)+s(6)+s(7)+s(8)+s(9);
y(2)  = h(1)+h(2)+h(3)+h(4)+h(5)+h(6)+h(7)+h(8)+h(9);
end
```

transport   Ln 23   Col 1

Fig. 2. M-file with fitness-function



Solver: gamultiobj - Multiobjective optimization using Genetic Algorithm

Problem
Fitness function: @transport
Number of variables: 9

Constraints:
Linear inequalities: A:)0100100;010010010;00   b: [100;130;250]
Linear equalities: Aeq: [100100100;010010010;(beq: [11; 7; 9]
Bounds: Lower: zeros(9,1)   Upper:
Nonlinear constraint function:

Run solver and view results
☐ Use random states from previous run
Current iteration: 118

```
Optimization terminated: average change in the spread of Pareto solutions less than
options.FunctionTolerance.

-----------------------------
Optimization running.
Optimization terminated: average change in the spread of Pareto solutions less than
options.FunctionTolerance.
```

Pareto front - function values and decision variables
Your solution is too large to be displayed in this table.
Export results to workspace to see your final point.

Fitness limit: ● Use default: -Inf ○ Specify:
Stall generations: ● Use default: 100 ○ Specify:
Stall time limit: ● Use default: Inf ○ Specify:
Function tolerance: ● Use default: 1e-4 ○ Specify:
Constraint tolerance: ● Use default: 1e-3 ○ Specify:

Plot interval: 1
✓ Distance ☐ Genealogy ☐ Score di
☐ Selection ☐ Stopping ✓ Pareto f
☐ Average Pareto distance ☐ Rank histogram ☐ Average
☐ Custom function:

☐ Custom function:

Level of display: off

Fig. 3. Performing the Optimization with Optimization Tool

Table 1 enumerates unique programs that implement the three main stages of the genetic algorithm: population, mutation, and crossing.

Table 1

The code user-defined functions

The name of parameter in the Matlab environment	Software implementation of the current parameter by the authors
Population	`function Population = schedule_create(GenomeLength, FitnessFcn, options) totalpopulation = sum(options.PopulationSize); range = options.PopInitRange; lower = range(1,:); span = range(2,:) - lower; Population = repmat(lower,totalpopulation,1) + round(repmat(span,totalpopulation,1) .* rand(totalpopulation,GenomeLength));end`
Mutation	`function mutationChildren = schedule_mutate(parents, options, GenomeLength, FitnessFcn, state, thisScore, thisPopulation) lb = repmat(options.LinearConstr.lb', length(parents), 1); ub = repmat(options.LinearConstr.ub', length(parents), 1); RandChange = round(2.5*randn(length(parents), GenomeLength) - 0.05); mutationChildren = min(max(lb, thisPopulation(parents,:) + RandChange), ub);end`

Ending of Table 1

Crossover	`function xoverKids = schedule_xover(parents, options, nvars, FitnessFcn, unused, thisPopulation)`
	`p1 = thisPopulation(parents(1:2:end),:);`
	`p2 = thisPopulation(parents(2:2:end),:);`
	`decide = rand(size(p1))<0.6;`
	`xoverKids = decide .* min(p1,p2) + ~decide .* max(p1,p2);end`

To define the optimization problem, let us start the Optimization Tool and set it as Figure 3. Then, according to Fig. 3, set the options for the problem. Run the optimization by clicking Start under Run solver and view results. A plot appears in a window as in Figure 4. These plots show the tradeoff between the two components of *f*. It is plotted in objective function space and the average distance between them. The results of the optimization appear below.

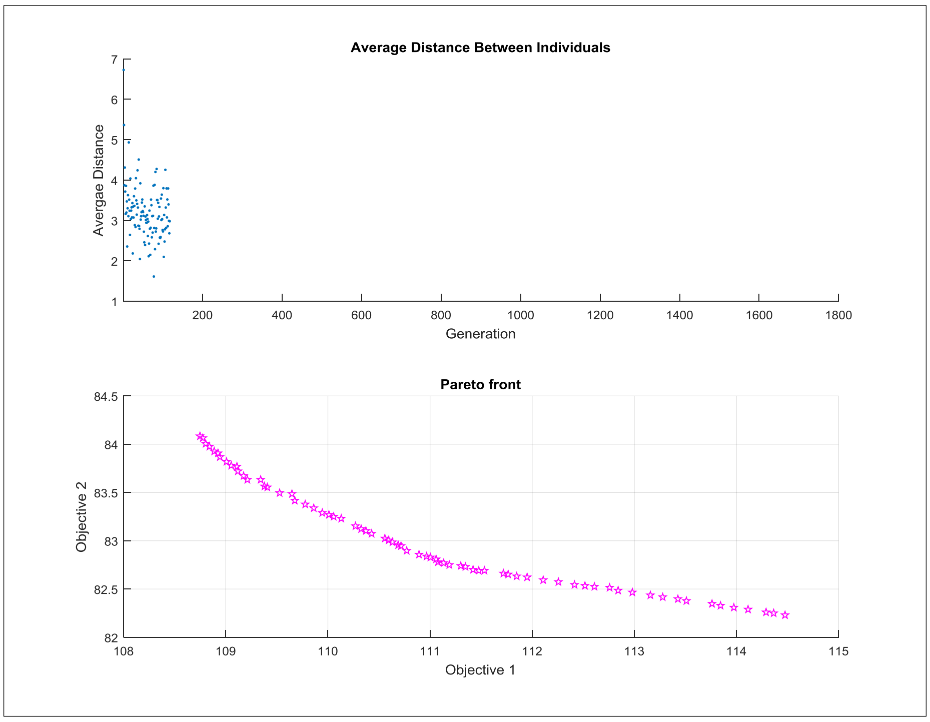


Fig. 4. The average distance between individuals and Pareto front

The optimal solution of the problem is the matrix of values of variables $X = \begin{pmatrix} 0 & 0 & 6 \\ 11 & 4 & 0 \\ 0 & 3 & 3 \end{pmatrix}$, which provides the minimum value of the objective function $F^*_{opt} = 96$.

Conclusions

In this chapter, the situational transportation problem with fictitious costs is successfully solved. As a result of the problem, all the main aspects of mathematical and computer modelling of problems with a discontinuous objective function are presented. The authors specifically chose a small-dimensional problem to simplify the presentation of the main material and for greater clarity. In the subsequent study, the authors set the task of testing programs from Table 1 for large-dimensional problems.

Transport problems with a discontinuous objective function are complex tasks in terms of computer programming. The analysis of published sources shows that genetic algorithms are successfully used in solving problems with discontinuous objective function. Based on the analysis of the literature, the authors also identified three main stages of the genetic algorithm: crossing, selection, and formation of a new generation (population). For all the main stages, the authors develop their unique m-functions (see Table 1) and write them into an article with open program code, which can give the chapter an enormous practical value.

The authors chose the MATLAB software environment to write the program code. MATLAB is a powerful mathematical package that makes it possible to simplify the process of task preparation, its solution, and analysis of results as much as possible.

References:

1. Masoud S. A. Assessing the Cost Impact of Multiple Transportation Modes to Enhance Sustainability in an Integrated, Two Stage, Automotive Supply Chain / S. A. Masoud, S. J. Mason // Informatics. – Multidisciplinary Digital Publishing Institute, 2017. – T. 4. – №. 4. – P. 34.
2. Sallam G. Shortest Path and Maximum Flow Problems Under Service Function Chaining Constraints / G. Sallam, et al. // arXiv preprint arXiv:1801.05795. – 2018.
3. Diab E. I. Bus transit service reliability and improvement strategies: Integrating the perspectives of passengers and transit agencies in North America / E. I. Diab, M. G. Badami, A. M. El-Geneidy // Transport Reviews. – 2015. – T. 35. – №. 3. – P. 292-328.
4. Diab E. Recommending transit: Disentangling users’ willingness to recommend transit and their intended continued use / E. Diab, D. van Lierop, A. El-Geneidy // Travel Behaviour and Society. – 2017. – T. 6. – P. 1-9.
5. Eng-Larsson F. Modal shift for greener logistics – the shipper's perspective / F. Eng-Larsson, C. Kohn // International journal of physical distribution & logistics management. – 2012. – T. 42. – №. 1. – P. 36-59.
6. Luthra S. The impacts of critical success factors for implementing green supply chain management towards sustainability: an empirical investigation of Indian automobile industry / S. Luthra, D. Garg, A. Haleem // Journal of Cleaner Production. – 2016. – T. 121. – P. 142-158.
7. Kantorovich L. On the translocation of masses / L. Kantorovich // C. R. (Doklady) Acad. Sci. URSS (N. S.) (37), 1942. – P. 199-201.
8. Belotti P. Mixed-integer nonlinear optimization / P. Belotti, et al. // Acta Numerica. – 2013. – T. 22. – P. 1-131.
9. Wang S. Optimization of Location-Routing Problem for Cold Chain Logistics Considering Carbon Footprint / S. Wang, F. Tao, Y. Shi // International journal of environmental research and public health. – 2018. – T. 15. – №. 1. – P. 86.
10. Wang S. Optimization of vehicle routing problem with time windows for cold chain logistics based on carbon tax / S. Wang, et al. // Sustainability. – 2017. – T. 9. – №. 5. – P. 694.

11. Andreica A. Best-order crossover for permutation-based evolutionary algorithms / A. Andreica, C. Chira // Applied Intelligence. – 2015. – T. 42. – №. 4. – P. 751-776.
12. Gen M. Improved genetic algorithm for generalized transportation problem / M. Gen, J. Choi, K. Ida // Artificial Life and Robotics. – 2000. – T. 4. – №. 2. – P. 96-102.
13. Csiszar S. Optimization Approaches for Logistic Problems – Vehicle Routing Problem with Time Windows / S. Csiszar // PhD Dissertation. – Budapest, Hungary, submitted: August 2006. – 108 p.
14. Cordeau J.-F. The VRP with Time Windows. Chapter 7 / J.-F. Cordeau, G. Desaulniers, J. Gesrosiers, M. Solomon, F. Soumis // Paolo Toth and Daniel Vigo (eds), SIAM, Monographs on Discrete Mathematics and Applications. 2001.
15. Babb T. Pickup and Delivery Problem with Time Windows, Coordinated Transportation Systems: The State of the Art / T. Babb // Department of Computer Science University of Central Florida Orlando, Florida. 2005.
16. Balinski M. L. Fixed-Cost Transportation Problems / M. L. Balinski // Naval Res. Logist. Quart. – 1961. – Vol. 8. – No. 1. – P. 41-54.
17. Kalyanmoy D. Multi-Objective Optimization using Evolutionary Algorithms / D. Kalyanmoy // John Wiley & Sons, Ltd, Chichester, England, 2001. – 518 p.

Pozdniakov S. V.
Candidate of Economic Sciences,
Public Organization "Platform for Public Dialogues"
Kuzmin O. V.
Candidate of Technical Sciences,
National University of Food Technologies
Kiiko V. V.
Candidate of Technical Sciences, Associate Professor,
National University of Food Technologies
Korenets Y. M.
Donetsk National University of Economics and Trade
named after Mykhailo Tugan-Baranovsky

DEFINITION OF THE ROLE OF BUSINESS MODELLING IN THE BUILDING OF A MANAGEMENT INFORMATION SYSTEM

Summary
The complex method of introduction of management information systems is proposed on the basis of creation of a complete business model of the company, which is excellent in that it allows for efficient designing of management information systems in accordance with its concept, provides further automation of management information systems based on the organizational and methodological approach of project management, and allows increasing the effectiveness of the process of implementing the management systems of the new technology.