LogForum

> Scientific Journal of Logistics <

2020, 16 (3), 397-408

http://doi.org/10.17270/J.LOG.2020.493

http://www.logforum.net p-ISSN 1895-2038 e-ISSN 1734-459X

ORIGINAL PAPER

SOLVING FOUR-INDEX TRANSPORTATION PROBLEM WITH THE USE OF A GENETIC ALGORITHM

Volodymyr Skitsko, Mykola Voinikov

Kyiv National Economic University named after Vadym Hetman, Kyiv, **Ukraine**

ABSTRACT. **Background:** Under conditions of digital transformation, the effective decision-making process should involve the usage of different mathematical models and methods, one of which is the transportation problem. The transportation problem, as the problem of resource allocation, is applicable in such domains as manufacturing, information technologies, etc. To get more precise solutions, the multi-index transportation problem can be applied, which allows taking into account several variables.
Methods: This paper develops an approach for applying the genetic algorithm for solving four-index transportation problems.
Results: The steps of the genetic algorithm for solving four-index transportation problems are outlined. The research has proved the steps of the genetic algorithm to be the same for all four-index transportation problem types, except for the first step (initialization), which is described for every type of transportation problem separately.
Based on the theoretical results, the program implementation of the genetic algorithm for solving four-index symmetric transportation problems has been developed with the open-source programming language typescript.
Conclusions: The paper promotes the application of the genetic algorithm for solving multi-index transportation problems. The investigated problem requires comprehensive studies, specifically, on the influence of change different parameters of the genetic algorithm (population size, the mutation, and crossover rates, etc.) on the efficiency of the algorithm in solving four-index transportation problems.

Key words: four-index transportation problem, symmetric transportation problem, genetic algorithm, program implementation.

INTRODUCTION

Under conditions of digital transformation, effective decision-making in management is possible to achieve with the use of different economic models and methods, which can be based either on classic approaches and tools or cutting-edge ones, for example, algorithms of collective intelligence, evolutionary algorithms, etc. The transportation problem as the problem of resource allocation in different domains such as manufacturing, information technologies, for example, when building communication networks or hardware and software resources sharing in cloud computing stays relevant until today.

The omnipresent usage of digital technologies is enabling the collection of large amounts of data, which can make the decision-making process more efficient. Besides, the increasing complexity of mathematical methods and models, on the one hand, is allowing using large amounts of data to make decisions more precise, and on the other hand, the number of computation increases as well and can affect the time needed to implement them. However, the evolution of computing hardware lets to solve complicated problems in an adequately short time. The problems, which

Citation: *Skitsko V., Voinikov M., 2020. Solving four-index transportation problem with the use of a genetic algorithm. LogForum 16 (3), 397-408, http://doi.org/10.17270/J.LOG.2020.493*
Received: 11.05.2020, Accepted: 09.06.2020, on-line: 30.06.2020.

were known earlier, but required numerous computations, got a second wind, one of which is the multi-index transportation problem.

LITERATURE REVIEW

Multi-index transportation problems are the extension of classic two-index transportation problems. Diverse researches have been devoted to multi-index transportation problems. One of the present complete reviews of multi-index transportation problems and their extensions is done in work [Singh et al. 2016].

The most common extensions of the multi-index transportation problem are taking into account fuzzy parameters and multiple objectives. For instance, multi-index transportation problems with fuzzy parameters are studied in [Kumar, Yadav 2012, Senapati 2018]; multi-objective multi-index transportation problems have been researched by [El-Shorbagy et al. 2020, Singh et al. 2018]; in work [Javadi et al. 2014] the research on handling several objectives in solving logistics problems is performed.

The application of a genetic algorithm in solving logistics problems, in particular, transportation and distribution ones, has been studied in the following studies [Dimov and Lukyanov 2016, El-Shorbagy et al. 2020, Indra et al. 2020, Javadi et al. 2014, Kaedure Bakhuet 2016, Karthy, Ganesan 2018, Thu Huyen et al. 2013, Yun et al. 2020].

The authors [Karthy, Ganesan 2018] suggest initializing the population with Vogel's approximation method. Additionally, the special mutation operator is introduced, which is applied after each crossover and serves a function of returning the chromosome to the feasible region. [Dimov, Lukyanov 2016, Kaedure Bakhuet 2016] propose the algorithm for initializing the initial population. In [Thu Huyen et al. 2013], described the solving of classical transportation problem with the use of a genetic algorithm.

The authors [Ritha, Vinotha 2012] propose the heuristic method for solving the triaxial transportation problem and describe the steps of the solving process.

In those works, different approaches to solving the transportation problems are proposed, in particular, with the use of a genetic algorithm. However, some aspects of solving multi-index transportation problems with the use of a genetic algorithm are not studied enough in modern literature. Additionally, to get practical results, it is essential to implement the algorithm, for example, programmatically, which can reveal new information to the researcher. The paper aims to describe the overall steps for solving four-index problems of different types with a genetic algorithm and to implement the algorithm programmatically.

FOUR-INDEX TRANSPORTATION PROBLEMS

In this paper, the following markings will be used:

$i \in I = \overline{1,n}$ is the index of the manufacturer; n is the number of manufacturers;

$j \in J = \overline{1,m}$ is the index of the good type; m is the number of types of goods;

$k \in K = \overline{1,p}$ is the index of the vehicle; p is the number of vehicles;

$l \in L = \overline{1,q}$ is the index of the consumer; q is the number of consumers;

$(ijkl) \in E = I \times J \times K \times L$ is a component part of the transportation problem;

f_{ijkl} is the cost of transportation of the good j, which is transported from the manufacturer i to the consumer l on the vehicle k;

x_{ijkl} is the quantity of the available good j, which is planned for transportation from the manufacturer i to the consumer l on the vehicle k;

a_{ijk} is the overall quantity of the good j, which is planned for transportation from the manufacturer j using the vehicle k;

b_{ikl} is the overall quantity of goods, that are planned for transportation from the manufacturer i to the consumer l using the vehicle k;

c_{ijl} is the quantity of the good *j*, which is planned for transportation from the manufacturer *i* to the consumer *l*;
d_{ikl} is the quantity of the good *j*, which is planned for transportation to the consumer *l* on the vehicle *k*;
a_{ij} is the quantity of the good *j*, which is offered by the manufacturer *i*;
b_{ik} is the quantity of goods, which are transported from the manufacturer *i* on the vehicle *k*;
c_{il} is the quantity of goods, which are transported from the manufacturer *i* to the consumer *l*;
d_{jk} is the quantity of the good *j*, which is transported on the vehicle *k*;
g_{kl} is the overall quantity of goods, which are transported on the vehicle *k* to the consumer *l*;
e_{jl} is the quantity of the good *j*, which is transported from the consumer *l*;
a_i is the overall quantity of goods, which are transported from the manufacturer *i*;
b_j is the overall quantity of good *j*;
c_k is the overall quantity of goods, which are transported on the vehicle *k*;
d_l is the overall quantity of goods, which are transported to the consumer *l*.

Objective function [Kaedure Bakhuet 2016, Raskin and Kirichenko 1982, Tuyet-Hoa and Philippe 2013]:

$$\text{Min } L(X) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ijkl} f_{ijkl} \quad (1)$$

Depending on the real economical requirements, agreements, and contracts, the x_{ijkl} variable can be constrained differently.

The imposition of restrictions allows the researcher to add real economical requirements to the mathematical model. Depending on the type of the imposed restrictions, the transportation problems can be either symmetric or asymmetric. Among the symmetric transportation problems, the following can be highlighted [Raskin, Kirichenko, 1982]: tetraspace, hexaplanar, and tetraaxial.

The four-index tetraspace transportation problem. The tetraspace transportation problem may have the following economical interpretation: a manufacturer refers to a carrier to transport several types of goods from factories (manufacturers, in the context of the described earlier problem) to distribution places (consumers). The goods can be picked up on any factory if they are available at a particular place; the main condition is to pick up all goods from factories and to satisfy the demand for distribution places. Since there is no special requirement for vehicles, carriers can choose which vehicle to use on their own, guided by their maximal gainings.

For the described problem only the restrictions a_i, d_l, c_k and b_j are defined [Raskin and Kirichenko 1982]:

$$\sum_{j=1}^{m}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ijkl} = a_i, \forall i \in I \quad (2)$$

$$\sum_{i=1}^{n}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ijkl} = b_j, \forall j \in J \quad (3)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{q} x_{ijkl} = c_k, \forall k \in K \quad (4)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} x_{ijkl} = d_l, \forall l \in L \quad (5)$$

$$x_{ijkl} \geq 0 \quad (6)$$

The meaning of the markings in the formulas refers the earlier described one.

Accordingly, the formalized four-index tetraspace transportation problem is presented by formulas (1)-(6).

The four-index hexaplanar transportation problem. The hexaplanar transportation problem might have, in particular, the following economical interpretation: a manufacturer refers to a carrier to transfer several types of raw materials from manufacturer's suppliers (manufacturers, in the context of the described earlier problem) to the manufacturers (consumers). Additionally, the following requirements have to be satisfied [formulas (7)-(12)]:

- the quantity of the picked-up good of the particular type from the manufacturer is restricted by the availability of the good;
- the quantity of the good of the particular type is restricted by the type of vehicle;

- the quantity of goods that are transported from the particular manufacturer on the particular vehicle type is fixed;
- the overall quantity of goods is restricted, for example, by contracts, etc.;
- the quantity of transported good of the particular type to the particular consumer is restricted by the demand;
- the quantity of goods that is transported to the particular consumer on the particular vehicle type is restricted.

The mathematical formalization of the described restrictions [Raskin and Kirichenko 1982]:

$$\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ijkl} = a_{ij}, \forall i \in I, \forall j \in J \quad (7)$$

$$\sum_{j=1}^{m}\sum_{l=1}^{q} x_{ijkl} = b_{ik}, \forall i \in I, \forall k \in K \quad (8)$$

$$\sum_{j=1}^{m}\sum_{k=1}^{p} x_{ijkl} = c_{il}, \forall i \in I, \forall l \in L \quad (9)$$

$$\sum_{i=1}^{n}\sum_{l=1}^{q} x_{ijkl} = d_{jk}, \forall j \in J, \forall k \in K \quad (10)$$

$$\sum_{i=1}^{n}\sum_{k=1}^{p} x_{ijkl} = e_{jl}, \forall j \in J, \forall l \in L \quad (11)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ijkl} = g_{kl}, \forall k \in K, \forall l \in L \quad (12)$$

$$x_{ijkl} \geq 0 \quad (13)$$

Thus, the formalized four-index hexaplanar transportation problem is presented by the formulas (1), (7)-(13).

The four-index tetraaxial transportation problem. The tetraaxial transportation problem has the hardest restrictions. This type of transportation problem is used in case of having transportations plans, restricted payload between warehouses, in particular, the transportation of the huge types of goods, which take all the payload of the vehicle.

In the tetraaxial transportation problem, additionally, the following requirements have to be satisfied [formulas (14)-(17)]:

- the quantity of the particular type of goods transported from the manufacturer on the particular type of vehicle is restricted;
- the quantity of goods which is planned for transportation from the manufacturer to the consumer on the particular vehicle is restricted;
- the quantity of particular type of goods from the manufacturer to the consumer is restricted;
- the quantity of the particular type of goods that is transported to the consumer on the particular type of vehicle is restricted;

The mathematical formalization of restrictions [Raskin and Kirichenko 1982]:

$$\sum_{l=1}^{q} x_{ijkl} = a_{ijk}. \forall i \in I, \forall j \in J, \forall k \in K \quad (14)$$

$$\sum_{j=1}^{m} x_{ijkl} = b_{ikl}, \forall i \in I, \forall k \in K, \forall l \in L \quad (15)$$

$$\sum_{k=1}^{p} x_{ijkl} = c_{ijl}, \forall i \in I, \forall j \in J, \forall l \in L \quad (16)$$

$$\sum_{i=1}^{n} x_{ijkl} = d_{jkl}, \forall j \in J, \forall k \in K, \forall l \in L \quad (17)$$

$$x_{ijkl} \geq 0 \quad (18)$$

Thus, the formalized four-index tetraaxial transportation problem is defined by the formulas (1), (14)-(18).

TOOLS FOR SOLVING FOUR-INDEX TRANSPORTATION PROBLEMS

The more indexed the problem is, the more time is required to get the optimal solution. The problem's non-linear increasing complexity does not allow us to solve it for a reasonable time, using classical optimization methods. Therefore, there is a need for search of such optimization methods, which let us get the suboptimal solution (or even optimal) for a reasonable time. One of those methods for solving four-index transportation problems is a genetic algorithm.

Let us outline the fundamental aspects of a genetic algorithm (based on [Goldberg 1988, Luke 2013]).

A genetic algorithm is a search evolutionary algorithm that is used for solving optimization and modeling problems by randomly creating,

combining, and variation of the searched parameters with the use of the mechanisms, which resemble biological evolution.

The fundamental concepts of genetic algorithms are derived from genetics, in particular, population, chromosome, and gene. The population is the set of genotypes of the particular generation. The chromosome (or individual) is the ordered sequence of genes, which represents the decoded solution to the problem. The gene is the atomic element of the chromosome. The mating pool consists of the chromosomes, which is selected in a predefined way using selection function, to which in the future the genetic operator will be applied (for example crossover, mutation), which have the random nature.

The mechanism of genetic algorithms gives it specialness: a generic algorithm works with several potential solutions (chromosomes) on each generation. That allows getting rid of the possibility of getting into local extrema of an objective function and reducing the work time. The fitness function helps the selection function to perform a selection for the creation of a mating pool.

The classic generic algorithm consists of the following steps: 1) the initialization of the initial population; 2) calculating fitness scores of the chromosomes, based on the fitness function; 3) checking the generic algorithm's stop criteria; 4) selection of chromosomes; 5) applying the genetic operators; 6) forming the population for the next generation; 7) choosing the fittest chromosome.

The steps from step 2 to step 7 are repeated until the stop criterion is satisfied. If the stop criterion (step 3) is satisfied, the genetic algorithm executing goes to step 7; otherwise – to step 4.

SOLVING FOUR-INDEX TRANSPORTATION PROBLEMS WITH A GENETIC ALGORITHM

To solve four-index transportation problems with the use of a generic algorithm, we will define genetic algorithm concepts in the context of the transportation problem:

- a component part (x_{ijkl}) of the transportation problem is the gene of the chromosome in the genetic algorithm;
- a feasible solution is a chromosome;
- an objective function in the transportation problem is a fitness function of the chromosome.

Let us define the steps of a genetic algorithm for solving symmetric four-index transportation problems.

The four-index tetraspace transportation problem

Step 1. Initializing the initial population. The process of initializing the initial population connotes the generating of chromosomes for the initial population. For the clearness, let us make an illustrative example. Supposed, we have three manufacturers, two types of goods, two types of vehicles, and three consumers. Then, $n = 3$, $m = 2$, $p = 2$ and $q = 3$. In that case, the visualization of the chromosome can take the form of the four-index array (fig. 1). To get the value of the needed gene, the indexes of the manufacturer, type of good, type of vehicle, and the consumer should be entered.

The size of the population does not change over the generations. The researchers choose the number of chromosomes in population on their own. Nonetheless, the number of chromosomes should satisfy the diversity of genetic material, since the lack of diversity may lead to inefficiency. Not only the small size can impact the performance negatively, but also the large one. A too-large population will consume more time on calculations, consequently leading to decreased efficiency. Thus, there should be a compromise decision.

As mentioned before, for high performance, the genetic material should have high diversity. Furthermore, the genes have to satisfy the restrictions of the transportation problem (2)-(6). To initialize the chromosomes with random values of genes, which satisfies the restrictions, the approach for initializing the initial population of chromosomes for the four-

index transportation problem is adapted [Dimov and Lukyanov 2016, Kaedure Bakhuet 2016, Skitsko and Voinikov 2018]:

1. the upper bound value is calculated for the x_{ijkl} variable. The upper bound equals to the lowest value among all the restrictions: $u_{ijkl} = min\{a_i; b_j; c_k; d_l\}$, where u_{ijkl} is the upper bound for the x_{ijkl} variable, $i \in I = \overline{1,n}$, $j \in J = \overline{1,m}$, $k \in K = \overline{1,p}$, $l \in L = \overline{1,q}$;
2. the value of the variable is assigned; the entered value is chosen randomly in the range from zero the upper bound u_{ijkl}: $x_{ijkl} = Rand(0; u_{ijkl})$;
3. the move to the subsequent variable is performed. The entered value for the current variable has to be taken into account when determining the values for the next variables. To achieve that, from the related restrictions, we subtract the ultimate value. Let us mark the subsequent consumer, vehicle, type of good and manufacturer as l^*, k^*, j^* and i^*:
   $x_{ijkl^*} = min\{a_i; b_j; c_k; d_l - x_{ijkl}\}$,
   $x_{ijk^*l} = min\{a_i; b_j; c_k - x_{ijkl}; d_l\}$,
   $x_{ij^*kl} = min\{a_i; b_j - x_{ijkl}; c_k; d_l\}$,
   $x_{i^*jkl} = min\{a_i - x_{ijkl}; b_j; c_k; d_l\}$.
   If the variable i, j, k or l is the last one for the 'manufacturer', 'type of good', 'vehicle' or 'consumer' and equals to *n, m, p* or *q,* then the value for the chromosome with that index is entered as the upper bound u_{ijkl}.

The generated chromosome is checked on getting into the feasible region. If the created chromosome is in the feasible region, it is taken to the initial population; otherwise - the chromosome is destroyed and the new one is created.

Step 2. Calculating fitness scores of the chromosomes. The fitness function for the transportation problem is the objective function.

The fitness function:

$$L(X(ch_q)) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ijkl}(ch_q) f_{ijkl}, \quad (19)$$

where:

q is the index of the chromosome in population; the number of chromosomes is determined by the researcher;

ch_q is the chromosome q in population (for each generation);

$x_{ijkl}(ch_q)$ is the quantity of the good j, which is transported from the manufacturer j to the consumer l on the vehicle k; $x_{ijkl}(ch_q)$ is the gene of the chromosome ch_q;

$X(ch_q)$ is the transportation problem potential solution, which is defined by the genes of the chromosomes ch_q;

Step 3. Checking the genetic algorithm's stop criteria. The stop criteria for four-index transportation problem can be [Luke 2013]: 1) the number of generations; 2) the time of genetic algorithm functioning; 3) the reach of the approximate value for fitness function;

Step 4. The selection of the chromosomes, which used for creation of the new generation. The method of selection can be chosen from the existing ones [Luke 2013]: 1) fitness proportionate selection (or roulette-wheel selection); 2) tournament selection; 3) rank selection; 4) elitism selection, etc.

Step 5. Applying the genetic operators, such as crossover, mutation. The crossover includes combining the genetic material of two chromosomes in a predefined way. For the transportation problem, the use of multi-point crossover is efficient, in which the recombined genes have the same real-world essence.

The mutation alters one or more gene values in a particular chromosome, not relying on other chromosomes. For example, the mutation can be performed as follows:

1. choose the gene, which is to mutate. To choose the gene, the indexes are randomly generated
   $i = Rand(0; n)$,
   $j = Rand(0; m)$,
   $k = Rand(0; p)$,
   $l = Rand(0; q)$ for the variable x_{ijkl};
2. the value of the gene is changed is a particular way, for example $x_{ijkl}(ch_q) = Rand(0; min\{a_i, b_j, c_k, d_l\})$.

Because of applying the genetic operators, the created (new) chromosomes can go out of the feasible region. To return them, the procedure for returning a chromosome to the feasible region can be applied. For example, the procedure for a three-index transportation problem [Skitsko and Voinikov 2018] can be adapted for four-index.

Step 6. Forming the population for the new generation. From the chromosomes, formed as a result of applying genetic operators to the selected pairs, after the check on the subject of being in the feasible region, the population for the next generation is created.

Step 7 same as step 2.

Step 8. Choosing the fittest chromosome, which is considered the final solution. If the genetic algorithm stops, among the chromosomes of the current population, the best one is chosen by the fitness function's value. For the transportation problem, the fittest chromosome would be the one with the lowest fitness function.

In different types of transportation problems, the essence of genes may vary, as well as the restrictions applied to them may also change.

That is why all the steps of the investigated problems will be the same, except for the first one - the initializing of the initial population, which we will cover later.

The four-index hexaplanar transportation problem

Step 1. Initializing the initial population. The procedure of generating the initial population for the four-index hexaplanar problem will vary in the way of calculating the upper bound for the variable (gene). In the four-index hexaplanar transportation problem, the upper bound for the variable x_{ijkl} is calculated as:

$$u_{ijkl} = \min\{a_{ij}; b_{ik}; c_{il}; d_{jk}; e_{jl}; g_{kl}\}.$$

Moreover, all the variables, which is related to the particular restriction should be taken into account, when calculating the upper bound for the subsequent variables:

$x_{ijkl^*} = \min\{a_{ij}; b_{ik}; c_{il} - x_{ijkl}; d_{jk}; e_{jl} - x_{ijkl}; g_{kl} - x_{ijkl};\}$,

$x_{ijk^*l} = \min\{a_{ij}; b_{ik} - x_{ijkl}; c_{il}; d_{jk} - x_{ijkl}; e_{jl}; g_{kl} - x_{ijkl}\}$,

$x_{ij^*kl} = \min\{a_{ij} - x_{ijkl}; b_{ik}; c_{il}; d_{jk} - x_{ijkl}; e_{jl} - x_{ijkl}; g_{kl}\}$,

$x_{i^*jkl} = \min\{a_{ij} - x_{ijkl}; b_{ik} - x_{ijkl}; c_{il} - x_{ijkl}; d_{jk}; e_{jl}; g_{kl}\}$.

The four-index tetraaxial transportation problem

Step 1. Initializing the initial population. In the four-index hexaplanar transportation problem, the upper bound for the variable x_{ijkl} is calculated as:

$$u_{ijkl} = \min\{a_{ijk}; b_{ikl}; c_{ijl}; d_{jkl}\}.$$

Moreover, all the variables, which is related to the particular restriction should be taken into account, when calculating the upper bound for the subsequent variables:

$x_{ijkl^*} = min\{a_{ijk}; b_{ikl} - x_{ijkl}; c_{ijl} - x_{ijkl}; d_{jkl} - x_{ijkl}\}$,

$x_{ijk^*l} = min\{a_{ijk} - x_{ijkl}; b_{ikl} - x_{ijkl}; c_{ijl}; d_{jkl} - x_{ijkl}\}$,

$x_{ij^*kl} = min\{a_{ijk} - x_{ijkl}; b_{ikl}; c_{ijl} - x_{ijkl}; d_{jkl} - x_{ijkl}\}$,

$x_{i^*jkl} = min\{a_{ijk} - x_{ijkl}; b_{ikl} - x_{ijkl}; c_{ijl} - x_{ijkl}; d_{jkl}\}$.

THE PROGRAMMATIC IMPLEMENTATION OF THE GENERIC ALGORITHM FOR THE TRANSPORTATION PROBLEM

Based on the material, we developed program implementation for solving the transportation problem with the genetic algorithm, using the open-source programming language TypeScript [TypeScript programming language 2020]. All the methods described in this chapter are the authors' intellectual property and do not relate to any other library.

General information about the program. The program is developed using the TypeScript language in an object-oriented paradigm. The class called GeneticAlgorithm (all the class and method names have been given by authors) provides the interface to a researcher to solve the problem. To instantiate the class GeneticAlgorithm, the researcher must pass the information related to the current transportation problem to the class constructor: the payoff matrix, the restrictions, and the dimensions of the problem.

Besides the information about the current transportation problem, the researcher may also change the general settings of the program, which include the population size, the number of iterations of the genetic algorithm, the crossover rate, and the mutation rate. The default settings are as follows: population size - 100; number of iterations - 1000; crossover rate - 0.95; mutation rate - 0.05.

Let us describe the steps of the genetic algorithm for the program implementation.

Step 0. Entering data. At this step, the user enters the dimensions of the problem, fills in the pay-off matrix, and adds the restrictions.

The pay-off matrix is presented by a four-dimensional array; the property dimensions consists of four properties (*n, m, p, q*), which relate to the dimensions of the problem.

The property restrictions consist of the array of the class Restriction's instances. The class Restriction checks whether the chromosome is in the feasible region of the problem (the method isChromosomeValid), and allows calculating the upper bounds for genes' value while initializing the initial population (the method calculateAvailable Capacities).

The mechanism of those functions can differ, depending on the type of restrictions. Therefore, for the instantiation of the class, the user must pass the custom function calculateCapacityUsage as an argument for the class's constructor. That function takes a chromosome object as an argument and returns boolean value whether the chromosome is in the feasible region.

Step 1. The initialization of the initial population. For generating the initial population, the class Population is used, which has properties such as the current generation, the population size, and the number of generations (the last two taken from the configuration file, described before), and private method called initializePopulation.

The method initializePopulation starts the initialization of the initial population. That method creates a chromosome, using the algorithm described earlier in the paper: for each gene gets value is chosen randomly from zero to the calculated upper bound. If the gene is the last for one of the restrictions, the value of the gene is chosen as the current upper bound. After the creation, the chromosome is checked on getting into the feasible region; if the chromosome is not in the feasible region, the new one is created instead.

Pseudocode of the function initializePopulation:

procedure start
  for *populationSize* times
    generate: for all the genes of the chromosome
      upperBound = *min*(restrictions' available capacities)
        if the gene is the last one for any *restriction*
          then the gene equals to the *upperBound*
        else the gene equals *rand*(0, *upperBound*)

    if the chromosome is in the feasible region
      then add chromosome to the population
    else continue: generate
procedure end

Step 2. Calculating fitness scores of the chromosomes. For calculating the fitness function of a chromosome, the protected method called calculateFitness is used, which iterates all the genes, multiplying them by the corresponding value of the pay-off matrix, then sums all the multiplications.

Pseudocode of the private method calculateFitness:

```
procedure start
        for each chromosome
        fitness = 0
        for each gene of the chromosome
                fitness = fitness + current gene value * current pay-
off matrix value
            chromosome's fitness = fitness
procedure end
```

Step 3. Checking the genetic algorithm's stop criteria. In the program implementation, the stop criterion is the number of generations. For checking the stop criterion, the method checkCriteria is used. If the method returns the truthy value, the execution is continued at step 8, which we will describe later.

Step 4. The selection of the chromosomes, which will be used for the creation of the new generation. For the selection function, the tournament method is chosen, where we randomly choose two chromosomes of the current generation, and the better one gets included in the mating pool. The number of 'tours' is the same as the population size.

Pseudocode of the function selectChromosomes:

```
input: void
output: nextGeneration
procedure start
   nextGeneration = empty array

   for populationSize times
          select1 = random chromosome from current population
          select2 = random chromosome from current population
          winner = max fitness function(select1, select2)
          push winner to nextGeneration array
   return nextGeneration
procedure end
```

Step 5. Applying the genetic operators, such as crossover, mutation. For the crossover, we use the function called crossoverCromosome. The probability of crossover correlates to the value in the configuration file.

Pseudocode of the function crossoverChromosome:

```
input: parent1, parent2
output: child1, child2
procedure start
    for all the genes
    if the gene index is odd
        child1's gene with given index = parent1's
corresponding gene
        child2's gene with given index = parent2's
corresponding gene
      else
        child2's gene with given index = parent1's
corresponding gene
        child1's gene with given index = parent2's
corresponding gene
    return array of child1 and child2
procedure end
```

In the other case, the mutation is applied (method mutateChromosome). In the program implementation, the genes, which are going to be mutated, are chosen randomly (by generating all the indexes). The value of the gene is mutated from zero the upper bound, which is calculated as the minimum restrictions' capacity.

Pseudocode of the function mutateChromosome:

```
input: chromosome
output: void
procedure start
        i = rand(0, n)
        j = rand(0, m)
        k = rand(0, p)
        l = rand(0, q)
        upperBound = min(restrictions' capacities)
        chromosome[i][j][k][l] = rand(0, upperBound)
procedure end
```

Step 6. Forming the next generation's population. Applying genetic operators can cause the chromosome to go out of its feasible region. For returning it, we use the function returnToAllowableRange.

Step 7 same as step 2.

Step 8. Choosing the fittest chromosome, which is considered to be the final solution. When the specified number of iterations have been performed, for the final solution, the elite chromosome is taken. Displaying of the solution is done with the showResult function.

Described earlier functions, in particular, the function for generating the initial population (initializePopulation), the function of calculating the fitness function

(calculateFitness), the function for performing selection (selectChromosomes), the function for mutation and crossover (mutate Chromosome and crossoverChromosome) are written by authors.

The program implementation of the genetic algorithm for solving four-index transportation problems allows solving four-index symmetric transportation problems with different types of restriction (passing custom Restriction instance to the genetic algorithm constructor). The code of the program can be adapted for particular use cases by method overriding.

CONCLUSIONS

The multi-index transportation problems are getting popularity since they allow taking into account more variables from the real world, compared to the classic one. Earlier, multi-index transportation problems did not get much attention because of the absence of adequate tools to solve them. The evolution of computing hardware made it possible to use such tools as a genetic algorithm.

In the paper, we elucidate the theoretical part of the four-index transportation problems problem. The solving process of the four-index transportation problems is proposed: the fundamental transportation problem concepts described in the context of the genetic algorithm; the steps of the genetic algorithm adapted for the problem.

Based on the material, we implemented the genetic algorithm for solving four-index transportation problems programmatically, using the programming language typescript. The program implementation allows solving four-index symmetric transportation problems.

Since the transportation problem can be observed as the problem of allocating resources, the material of the paper can be used for solving specific problems in the area of manufacturing, logistic systems, information technologies, etc.

For future work, we consider useful exploring the efficiency of a genetic algorithm, in particular, the influence of the population size, the dimensions, the crossover and mutation rates on the final solution; the usage of other types of selection (roulette-wheel, ranking method, etc.), crossover.

ACKNOWLEDGMENTS AND FUNDING SOURCE DECLARATION

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

Dimov Yu.S., Lukyanov N.D., 2016. Genetic algorithm application for solving a multi-index transportation problem. Proceedings of Irkutsk State Technical University, 7, 73–79. http://doi.org/10.21285/1814-3520-2016-7-73-79

El-Shorbagy M., Mousa A., ALoraby H., Abo-Kila T., 2020. Evolutionary algorithm for multi-objective multi-index transportation problem under fuzziness. Journal of Applied Research on Industrial Engineering, 7(1), 36-56. http://doi.org/0.22105/jarie.2020.214142.1119

Goldberg D. E., 1988. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing, Inc. 432 pages.

Indra Z., Chairunisah, Refisis N.R., 2020. Application Genetic Algorithm in solving three-level supply chain distribution problems. Journal of Physics: Conference Series. 1462. 012035. http://doi.org/10.1088/1742-6596/1462/1/012035

Javadi A., Tarokh M.J., Piroozfar S., 2014. Solving a multi-objective vehicle scheduling- routing of interurban transportation fleet with the purpose of minimizing delays by Using the Differential Evolutionary Algorithm. Uncertain Supply Chain Management, 2 (3), 125-136. http://doi.org/10.5267/j.uscm.2014.5.005

Kaedure Bakhuet A.-J., 2016. Solving bi-objective 4-dimensional transportation problem by using PSO. AGK Bakhat. Sci.Int (Lahor.), 28 (3), 2403-2410. http://www.sci-int.com/pdf/636551900738439387.pdf

Karthy T., Ganesan K., 2018. Multi Objective Transportation Problem - Genetic Algorithm Approach. International Journal of Pure and Applied Mathematics. Volume 119 No. 9, 343-350. https://acadpubl.eu/jsi/2018-119-9/articles/9/33.pdf

Kumar A., Yadav S.P., 2012. A Survey of Multi-index Transportation Problems and Its Variants with Crisp and Fuzzy Parameters. K. Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011. 919–932. http://doi.org/10.1007/978-81-322-0487-9_86

Luke S., 2013. Essentials of Metaheuristics. Lulu, second edition. 242 pages., Cite this document as: Sean Luke, 2009,Essentials of Metaheuristics, available at http://cs.gmu.edu/~sean/book/metaheuristics/

Raskin, L.G., Kirichenko, I.O., 1982. Mnogoindeksnye zadachi lineinogo programmirovaniya (Multi-index Problems of Linear Programming), Moscow: Radio i Svyaz'.

Ritha W., Vinotha. M.J., 2012. Heuristic algorithm for multi-index fixed charge fuzzy transportation problem. Elixir Comp. Sci. Engg. 46, 8346-8353. https://www.elixirpublishers.com/articles/1351162502_46%20(2012)%208346-8353.pdf

Senapati. S., 2018. Multi-Index Bi-Criterion Transportation Problem: A Fuzzy Approach. International Journal of Advanced Engineering, Management and Science, 4 (7), 550-556. http://doi.org/10.22161/ijaems.4.7.8

Singh S., Chauhan S.K., Kuldeep., 2018. A Bi-Criteria Multi-Index Bulk Transportation Problem. Annals of Pure and Applied Mathematics, 16 (2), 479-485. http://doi.org/10.22457/apam.v16n2a26

Singh S., Tuli R., Sarode D., 2016. A review on fuzzy and stochastic extensions of the Multi Index transportation problem. Yugoslav Journal of Operations Research, 27 (1), 3-29. http://doi.org/10.2298/YJOR150417007S

Skitsko V.I., Voinikov M. Yu., 2018. Solving a Three-Index Transportation Problem under Risk Conditions Using a Genetic Algorithm. The Problems of Economy, 3, 246–252. https://www.problecon.com/export_pdf/problems-of-economy-2018-3_0-pages-246_252.pdf

Thu Huyen N., Luong Sy Uoc, Rosaly B. Alday, 2013. Genetic Algorithm for Solving Balanced Transportation Problem. International Journal of Innovative Technology and Exploring Engineering. (IJITEE), 3(4), 24-27. http://www.ijitee.org/wp-content/uploads/papers/v3i4/D1163093413.pdf

Tuyet-Hoa P., Philippe D., 2013. An Exact Method for Solving the Four Index Transportation Problem and Industrial Application. American Journal of Operational Research, 3(2), 28-44. http://article.sapub.org/10.5923.j.ajor.20130302.02.html

TypeScript programming language, 2020. https://www.typescriptlang.org/

Yun Y., Chuluunsukh A., Gen M., 2020. Sustainable closed-loop supply chain design problem: A hybrid genetic algorithm approach. Mathematics, 8 (1), 84. http://doi.org/10.3390/math8010084

ROZWIĄZANIA CZTEROCZYNNIKOWEGO PROBLEM TRANSPORTOWEGO PRZY POMOCY ALGORYTMU GENETYCZNEGO

STRESZCZENIE. **Wstęp:** W warunkach komputerowej transformacji, efektywny proces podejmowania decyzji powinien obejmować wykorzystania modeli metod matematycznych. Przykładem takiej sytuacji jest problem transportowy, który jest problemem alokacji zasobów, występujący w takich obszarach jak produkcji, technologie informatyczne, itp. W celu uzyskania precyzyjniejszych rozwiązań, można zastosować wieloczynnikowy problem transportowy, który umożliwia uwzględnienie wielu zmiennych.

Metody: W pracy zastosowano algorytm genetyczny dla rozwiązania czteroczynnikowych problemów transportowych.
Wyniki: Wyszczególniono kroki algorytmu genetycznego dla czteroczynnikowego problem transportowego. Udowodnione, że kroki algorytmu genetycznego są takie same dla wszystkich typów czteroczynnikowych problemów transportowych, z wyjątkiem pierwszego kroku (inicjalizacji), który został opisany osobno dla każdego z typów problemu transportowego.
W oparciu o wyniki teoretyczne, utworzono programowanie dla algorytmu genetycznego dla rozwiązywania czteroczynnikowych problemów transportowych przy użyciu opensourcowego języka typescript.
Wnioski: W pracy zaproponowano zastosowanie algorytmu genetycznego dla rozwiązywania wieloczynnikowych problemów transportowych. Analizowany problem wymaga dalszych badań, szczególnie w zakresie wpływu zmian poszczególnych parametrów algorytmu genetycznego (wielkości populacji, mutacji, współczynnika podziału, itp.) na efektywność algorytmu w rozwiązywaniu czteroczynnikowych problemów transportowych.

Słowa kluczowe: czteroczynnikowy problem transportowy, symetryczny problem transportowy, algorytm genetyczny, wdrożenie programu

Volodymyr Skitsko ORCID ID: https://orcid.org/0000-0002-6290-9194
Kyiv National Economic University named after Vadym Hetman
Institute Information Technologies in Economics
Kyiv, **Ukraine**
e-mail: skitsko@kneu.edu.ua

Mykola Voinikov ORCID ID: https://orcid.org/0000-0001-7961-5312
Kyiv National Economic University named after Vadym Hetman
Institute Information Technologies in Economics
Kyiv, **Ukraine**
e-mail: qwoxa1@gmail.com