THE GENETIC ALGORITHM FOR SOLVING THE NON-LINEAR TRANSPORTATION PROBLEM

Tatiana PAȘA

Moldova State University, Chișinău, Republic of Moldova
(pasa.tatiana@yahoo.co)

DOI: 10.19062/1842-9238.2018.16.2.4

Abstract: *The paper examines the transportation problem on a network with concave cost functions and presents a genetic algorithm implemented in Wolfram Language which is able to solves this problem. We experimentally prove that the algorithm is stable, effective and converges to an optimum solution. The results are compared to the ones obtained by means of the standard functions of Wolfram Language.*

Keywords: *transportation problem, concave cost function, genetic algorithm, population, crossover operation, mutation operation.*

1. INTRODUCTION

The transport problem appeared when the need to solve a group of economic problems arose. These problems consist in finding the smallest transportation costs for the products as they travel towards their destination. A classical variant of the transportation problem is a linear programming problem, and a solution was proposed in 1947 by Dj. Dantzing [10] that solves it through the simplex method. It consists in determining the optimum transport plan for the required amount of a single product from a source to the destination, that minimizes the transportation cost. This model of the problem is used to optimize the supply of businesses with raw materials, the supply of stores with products from wholesalers and the design of telecommunication, water, gas or petrol networks. Several authors, D. R. Fulkerson [19], R. G. Busacker și P. J. Goven [3], J. Edmonds and R. M. Karp [14], N. Tomizawa [40], M. Klein [27], D. D. Sleator and R. E. Tarjan [37], A. V. Goldberg and R. E. Tarjan [20, 21], P. T. Sokkalingam, R. K. Ahuja and J. B. Orlin [38], I-L.Wang, S-J. Lin [41], L. Ciupală [8], J. M. Davis and D. P. Williamson [11], P. Kovacs [28], A. Sifaleras [36], S. Ding [13], M. Dawuni and K. F. Darkwah [12], N. A. El-Sherbeny [15], M. B. Cohen, A. Madry, P. Sankowski, and A. Vladu [9], A. M. P. Chandrasiri and D. M. Samarathunge [4], R. A. Maher and F. A. Abdula [3], J. Erickson, K. Fox and L. Lkamsuren [16], S. Abdi, F. Baroughi and B. Alizadeh [1] have proposed solutions to the network transportation problem with linear cost functions, and also with uncertain cost and/or capacities for each edge in the network. They have also presented a theoretical and practical analysis of the algorithms, fitted with conclusions and recommendations.

When the network has non-linear cost functions, the problem becomes more complicated. In this case, there are several concepts that are widely used to solve it: the optimality conditions Kuhn-Tucker and the Lagrange multipliers, the first order derivatives (gradients), the second order derivatives (Hessian matrix), and also the penalty functions [39], [30].

The usage of these techniques is much more difficult in networks with concave cost functions, because there are multiple local minima and it can lead to only getting a single local optima. This type of problems have been studied in detail by R. Horst and P. M. Pardalos [24], R. Horst and H. Tuy [25], Q. He, A. Shabbir and G. L. Nemhauser [22].

We are especially interested in problems that describe real life situations which are typical for a modern economy. As this kind of models are extremely complex, our preoccupation resulted in genetic algorithms proposed to solve them. They have been described for the first time under the leadership of J. Holland [23] at the University of Michigan. The name "genetic" derives from the notions surrounding them, e.g. population, chromosomes, genes, selection, mutation, crossover. These are stochastic and heuristic algorithms, which mean that the obtained solutions ~~is~~ are not always optimal, but they come close to the optima. In general, these algorithms are polynomial and are used to solve complex problems. A comparative analysis of the genetic algorithms applied to optimization problems can be found in [33] and an overview of the algorithms is given in [29].

These algorithms are recommended because: one doesn't have to know gradients and Hessian matrices; they can't get stuck on local optima and still work very well on big problems with ~~a~~ many variables. Several authors have used genetic algorithms to solve transport network problems: D.B.M.M. Fontes and J.F. Gonçalves [18], A. Sadegheih and P. R. Drake [35]. The implementation of the algorithm is often slowed down by the need to use auxiliary variables and the computation of the fitness function many times.

Genetic algorithms have been proposed to solve micro circulation problems. Some of them are: the usage of adjacent roads to reduce the traffic from main roads [6], the minimization of the time spent before a plane can land [26] and the coordination of several urban bus routes [42].

The genetic algorithm described in [5] can plan efficiently the high-speed train stations. In [44] is present an algorithm that minimizes costs and streamlines the metro activity and [7] describes an algorithm that improves significantly the security and efficiency of corridors with multiple-rail grade crossings. Another genetic algorithm proposed in [2] optimizes the routes of the airport buses by minimizing the transfer time of the passengers. The problem of planning the deposit spaces in ports is studied in [43] with the purpose of minimizing taxes.

2. PROBLEM FORMULATION

We consider the transport network problem described by the convex graph $G = (V, E)$, $|V| = n$, $|E| = m$. The real production and consumption function $q = V \to R$ is defined on the finite set of vertices V. The concave non-decreasing functions of cost $\varphi_e(x_e)$ are defined on the edges E. To solve this non-linear optimization problem we must find a flow x^* that minimizes the function $F(x) = \sum_{e \in E} \varphi_e(x_e)$, thus satisfying the conditions:

$$F(x^*) = \min_{x \in X} F(x)$$

$$\sum_{e \in E^+(v)} x(e) - \sum_{e \in E^-(v)} x(e) = \begin{cases} -p(v), & v = v_0 \\ 0, & V/\{v_0, v_t\} \\ p(v), & v = v_t \end{cases}$$

$$x(e) \geq 0, for\ every\ e \in E.$$

X is the set of possible solutions, that satisfies the system of equations and the positivity restrictions, $E^-(v) = \{(v,u)|(v,u) \in E\}$, $E^+(v) = \{(u,v)|(u,v) \in E\}$.

We will consider the problem in which any quantity can be transported through an edge, the costs being described by a concave function. We also consider that in no intermediary point can the quantity of goods increase or decrease, thus the flow conservation condition being satisfied. Which means that no intermediary points consume or produce flow. Then, the problem requires the minimization of transportation costs from the source to intermediary centers and then to the destination.

In such situations, in which we have to check all admissible solutions, which can only be computed in a long time, the genetic algorithm is a solution that can provide the answer in a reasonable amount of time. Genetic algorithms are based on the theorem of templates [34]. A template H is defined as a pattern that describes a subset of chromosomes with similar genetic sections. Schemes have two properties:

- the degree of a template H, denoted by $o(H)$ – the number of fixed positions in a template;
- the definition length of a template H, denoted by $\delta(H)$ – the distance between the first and last position of the string of genes.

The first step in using a genetic algorithm is deciding how to encode the problem, how to describe the chromosomes as admissible solutions. The most often used is binary encoding, but it can also be numerical, symbolic and character-based depending on the problem. A population consists of chromosomes, which is actually a set of admissible solutions.

When creating a population, we must keep in mind that:

– the chromosomes will have a constant length;

– the number of chromosomes in a population is constant;

– every population $P(i+1)$ is created from only the offspring of the population $P(i)$ or parents and offspring.

We will consider the elements of a population of size $4n$ (or $2n$) represented by numerical strings of length n constructed on the numerical alphabet $\mathcal{A} = \{0,1,2,\dots,n-1\}$. The population will evolve to better solutions using *selection*, *mutation* and *crossover*. The value of the fitness function will have smaller and smaller values which means that the chromosomes will be closer to the minimum solution. The elitist selection is preferred in a genetic algorithm, because it guarantees that promising chromosomes will not be lost.

There are several ways in which we can make sure that we keep the best solution in a population. When applying ***selection*** at a step k, we will decide which chromosomes will participate in creating a new population. There are several methods of selection:

- the probability of choosing a particular chromosome depends on the value of –its fitness function;

- chromosomes are sorted in ascending order based on their fitness function and the probability depends on their position in the sorted list;

- for each 2 randomly chosen chromosomes, we take the one with a smaller fitness function.

For our problem, we will sort the chromosomes in increasing order and take the first half, thus we will never lose a good solution, even if it appears in the first population. These chromosomes will form the first half of the next population and will be parents to the offspring. Even if we aren't guaranteed to get better solutions this way, the chance is better than when choosing random parents and the offspring will be at least as good as the parents.

The other half of the population will be obtained through the ***crossover*** of the previously selected chromosomes. A pair of offspring will be created by taking two adjacent chromosomes and making a random cut. The first offspring will be formed from the left part of the mother chromosome and the right part of the father chromosome; the second one will be formed from the left part of the father chromosome and the right part of the father chromosome.

So if we have:

mother $(a_1, a_2, \dots, a_n)$

father $(b_1, b_2, \dots, b_n)$

then the offspring will be:

offspring1: $(a_1, a_2, \dots, a_k, b_{k+1}, \dots, b_n)$

offspring2: $(b_1, b_2, \dots, b_k, a_{k+1}, \dots, a_n)$

We can also use more cuts or create offspring from several chromosomes.

The random modification of ~~a~~ the gene of a chromosome, also called ***mutation***, has a mixed effect, it can improve or worsen the solution. The mutation will take place at a small rate of ε, e.g. $\varepsilon = 0.001$, so that we may avoid losing ~~not lose~~ good solutions, but enough to produce new solutions in order to elude a local minima. Mutation will generate a random value to a randomly chosen gene.

We must also define the stop condition. Usually, we run the algorithm until k populations are created and return the best solution in it. But in this case we may get to a point where subsequent populations are the same. To avoid such a situation, we can stop breeding new populations when the condition $|f(i-1) - f(i)| \leq \varepsilon$ is satisfied for the best solutions in two consecutive populations $P(i-1)$ and $P(i)$.

Based on what we described, a genetic algorithm completes several steps and at the end an optimum solution, it results in [32], [17]:

Step 1. Generation of the initial population;

Step 2. Evaluation of the fitness function for each chromosome of the population;

Step 3. Selection of the chromosomes so that we don't lose any good solutions;

Step 4. Crossover of the selected chromosomes to create offspring with a fitness function at least as good as that of the parents;

Step 5. Mutation of a gene of a chromosome at a rate of ε;

Step 6. Test of the stop condition; if it is satisfied, then we STOP; if not, we go back to Step 2.

3. DESCRIPTION OF THE ALGORITHM

The genetic algorithm starts with a random population of chromosomes, each individual chromosome being an admissible solution to the transport network problem. Every chromosome will have length n. Using *selection*, *crossover* and *mutation* we will improve the population and obtain a smaller value of the fitness function.

The algorithm generates in each population chromosomes at least as good as in the previous population, because at every step we select only the chromosomes with the lowest objective function value and the rest of the population is filled with their offspring.

The genetic algorithm P1 proposed to solve the transport problem with non-linear concave functions consists of the following steps:

1. *Initialization.* The initial population is generated in the following way: a string of n random natural numbers will be generated, such that the first position will have a number nr_i between 1 and $n-1$ (from the first vertex called *source* there is at least one outgoing edge and at most $n-1$ edges), on every position $i = \overline{2, n-1}$ a number nr_i between 0 and $n-i$ will be generated, and the last position will have the number $nr_i = 0$ (there are no outgoing edges from the last vertex, the *destination*). This string will be one of the chromosomes in the population.

The population will have $4n$ chromosomes. We will also randomly generate a matrix $R = \{r_{ij} | i = \overline{1,n}, j = \overline{1,n}\}$ that shows how much of the flow from vertex i will go through edge (i,j). For this matrix $r_{ij} = 0$ only if the edge (i,j) doesn't exist and $\sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij} = 1$.

2. *Evaluation* of the chromosomes from the current population means evaluating the objective function of each individual.

3. *Selection* of the parent chromosomes that will participate in the crossover is done so that their objective functions are the smallest possible. The chromosomes will be sorted in the ascending order of the objective function value. The first half of the new population will be formed from these chromosomes.

4. *Crossover* of the chromosomes is realized between the previously selected chromosomes to form the second half of the population. We will cut randomly each parent in the same place, and combine these parts as described earlier to create two new offspring. This way, each pair will have two offspring and the size of the population will be constant.

5. *Mutation* of a single gene of a chromosome will be done at a rate of $\varepsilon = 0{,}01 - 0{,}0001$, by generating a new random value for a gene ~~of~~ in a chromosome.

6. *Testing the stop condition* can be done in a few ways:

a) after creating k populations;

b) a time restriction for very serious problems;

c) stopping the algorithm when the condition $|f(i-1) - f(i)| \leq \varepsilon$ is satisfied, for the best solutions in two consecutive populations $P(i-1)$ and $P(i)$.

In Step 2 we evaluate the chromosomes of a population. As we said earlier, each chromosome is a string of numbers in which position i represent the number nr_i of outgoing edges in the subgraph that contain only edges through which the flow passes. To evaluate a chromosome, we must first obtain the solution encoded in it, and then evaluate the objective function.

To decode the solution from a chromosome we will do the following:

I. Let Nr_i be the number of outgoing edges from a vertex i in the graph G that describes the transport network. Then we have the following two cases:

- $nr_i \geq Nr_i$, the graph that describes the admissible solution contains all outgoing edges from the vertex i of graph G;

- $nr_i < Nr_i$, the graph that describes the admissible solution has only a subset of size nr_i of outgoing Nr_i edges from vertex i of graph G. These edges will be selected randomly.

II. We know that we need to transport a quantity of flow $p(v)$ through the network from the description of the problem. This flow will be assigned to the edges based on the matrix $R = \{r_{ij} | i = \overline{1,n}, j = \overline{1,n}\}$ generated in step 1, that shows how much of the flow from vertex i will go through edge (i,j). The obtained value is the flow x_k assigned to the edge (i,j) whose value will be placed in position k of the admissible solution of the form $(x_1, x_2, \dots, x_m)$.

After constructing the admissible solution, we will compute the objective function. If the chromosome is deemed fit to go into the next population, this value will be stored so that in the next populations only the offspring will be evaluated. This allows us to decrease the execution time of the algorithm, which is very important, especially for big problems.

Theorem1: *The genetic algorithm P1* uses $O(n^2)$ memory.

Proof: The transport network is described by an adjacency list of size $m < n^2$, the matrix $R = \{r_{ij} | i = \overline{1,n}, j = \overline{1,n}\}$ is of size n^2 and a population of chromosomes is of size $4n * n = 4n^2$, and every new population will be changed in place, so no other memory is necessary. As a result, the algorithm needs $O(n^2)$ memory. □

Theorem2: *The genetic algorithm P1* is polynomial and has a $O(n^3)$ complexity.

Proof: To create the adjacency list that describes the graph $O(m)$ operations are needed. To create a population of size $4n$ with every chromosome of length n, $O(n^2)$ operations are needed. To evaluate all chromosomes in a population, we need $4n * m = 4nm$ operations, which creates a $O(nm)$ complexity.

The crossover has a $O(n^2)$ complexity, and the mutation – $O(n)$. As a result, the complexity of the algorithm is $O(n^3)$. □

Remark1: For sparse graphs, the complexity is $O(nm)$.

Remark2: *The genetic algorithm P1* is convergent and always converges to a good solution. If the algorithm is run several times (for example in parallel or sequential) and we choose the best solution, we can get one very close to the global optima.

4. PRACTICAL APPLICATION

The algorithm described above was implemented in the Wolfram Language and tested on several random examples of different sizes. The tests were done using two stop conditions:

1. the algorithm was stopped when the condition $|f(i-1) - f(i)| \leq \varepsilon$ was satisfied, for the best solutions in two consecutive populations $P(i-1)$ and $P(i)$;
2. the algorithm was stopped after k populations were made.

As we can see in the following table (*Table 1.*), the execution time increases much slower for condition 1 compared to condition 2.

This happens because after a number of steps, even of a optima is found it will be sent to the next population because the condition 2 will not be satisfied. From this data we can recommend the first stop condition for the algorithm.

Table 1. Execution time of GA (seconds)

Nr. of vertices	t_ε	t_k	***Nr. of vertices***	t_ε	t_k
10	0,0262	0,0950	**60**	6,3544	46,7701
15	0,0809	0,3066	**65**	4,9332	83,5933
20	0,2421	0,7972	**70**	3,5041	102,1030
25	0,3294	1,6901	**75**	7,4183	195,1190
30	0,5540	3,700	**80**	18,0235	193,3200
35	1,1752	7,1764	**85**	12,8842	323,9760
40	0,9351	9,0325	**90**	10,5738	311,3430
45	1,1236	18,3405	**95**	45,6272	511,6740
50	3,5589	24,2091	**100**	55,9386	529,4140
55	2,2681	46,7701	**120**	117,9100	1621,2100

Using the standard Wolfram Language function $Minimize[\{f, cons\}, \{x, y, ...\}]$ we can obtain the global minima for our problem.

Table 2. Execution time for *Minimiz* (seconds)

Nr. vertices	**Minimize**
4	0.17
6	3.08
8	345.61

The execution time for the function *Minimize* given in *Table 2.* increases starting from graphs with 8 vertices, i.e. the execution time on a graph with 8 vertices is 3 times longer than the genetic algorithm on a graph with 100 vertices.

By conducting these tests we could experimentally prove that the algorithm converges by computing the total objective function of a population and observing that it is always decreasing.

CONCLUSIONS

In this paper, we have discussed the transportation network problem with concave cost functions. To solve this problem, we proposed a genetic algorithm, which uses elements of graph theory, to transform a chromosome into an admissible solution.

1. The experimental results prove the correctness of the described algorithm, because we always get a good solution, and we can even get the global optima if we run it several times.
2. The algorithm is convergent, because the total fitness function of a population is always decreasing.
3. The execution time is much better comparing to standard Wolfram Language functions, which means that the algorithm is fast even for bigger networks.

REFERENCES

[1] Abdi, S., Baroughi, F., & Alizadeh, B., *The Minimum Cost Flow Problem of Uncertain Random Network*, Asia-Pacific Journal of Operational Research, vol. 35, no. 03, 2018.

[2] Bao, D., Gu, J., Di, Z., & Zhang, T., *Optimization of Airport Shuttle Bus Routes Based on Travel Time Reliability*, Mathematical Problems in Engineering, Article ID 2369350, https://doi.org/10.1155/2018/2369350 (vis. August 2018), 12 pages, 2018.

[3] Busacher, R. G., & Gowen, P. J., *A procedure for determining a family of minimum-cost network flow patterns.* Bethesda, MD: Technical Report ORO-TP-15, Operations Research Office, The John Hopkins University, 1960.

[4] Chandrasiri, A. M., & Samarathunge, D. M., *Application of Minimum Cost Flow Problem: A Case Study of Crown Distributors in Kegalle, Sri Lanka*, International Journal of Scientific & Engineering Research, vol. 8, no. 1, pp. 1850-1853, 2017.

[5] Chen, D., Ni, S., Xu, C., Lv, H., & Wang, S., *High-Speed Train Stop-Schedule Optimization Based on Passenger Travel Convenience*, Mathematical Problems in Engineering, Article ID 8763589, http://dx.doi.org/10.1155/2016/8763589, (vis. August 2018), 10 pages, 2016.

[6] Chen, Q., & Shi, F., *Model for Microcirculation Transportation Network Design.* Mathematical Problems in Engineering, Article ID 379867, doi:10.1155/2012/379867 (vis. August 2018), 11 pages, 2012.

[7] Chen, Y., & Rilett, L. R., *Signal Timing Optimization for Corridors with Multiple Highway-Rail Grade Crossings Using Genetic Algorithm*, Journal of Advanced Transportation, Article ID 9610430, https://doi.org/10.1155/2018/9610430 (vis. August 2018), 14 pages, 2018.

[8] Ciupală, L., *The minimum cost flow problem with surplus*, Bulletin of the Transylvania University of Braşov, Series III: Mathematics, Informatics, Physics , vol. 3, no. 52, pp. 177-182, 2010.

[9] Cohen, M. B., Madry, A., Sankowski, P., & Vladu, A., *Negative-weight shortest paths and unit capacity minimum cost flow in O(m[superscript 10/7]log W) time*, 28th Annual ACM-SIAM Symposium on Discrete Algorithm, 16-19 January 2017 (pp. 752-771). Barcelona, Spain: Association for Computing Machinery, http://hdl.handle.net/1721.1/113883, 2017.

[10] Dantzig, G. B., *Application of the Simplex Method to a Transportation Problem*, In T. C. Koopmans, Activity of Production and Allocation, New York: John Wiley and Sons, pp. 359-373, 1951.

[11] Davis, J. M., & Williamson, D. P., *A dual-fitting 3/2-aproximation algorithm for some minimum-cost graph problems*, 20th Annual European Symposium, no. 7501 in Lecture Notes in Computer Science, Springer, pp. 373-382, 2012.

[12] Dawuni, M., & Darkwah, K. F., *Maximum flow-minimum cost algorithm of a distribution company in Ghana: Case of 'NAAZO' Bottling Company*, Tamale, Tamale Metropolis, African Journal of Mathematics and Computer Science Research, vol. 8, no. 2, pp. 23-30, 2015.

[13] Ding, S., *Uncertain minimum cost flow problem*, Soft. Comput., vol. 18, pp. 2201-22017, 2014.

[14] Edmonds, J., & Karp, R., *Theoretical improvements in algorithmic efficiency for network flow problems*, Journal of the ACM, no. 19, pp. 248-264, 1972.

[15] El-Sherbeny, N. A., *Minimum Cost Flow Time-Windows Problem with Interval Bounds and Flows*, Theoretical Mathematics & Applications, vol. 6, no. 3, 2016.

[16] Erickson, J., Fox, K., & Lkhamsuren, L., *Holiest Minimum-Cost Paths and Flows in Surface Graphs*, Cornell University Library, arXiv:1804.01045, 2018.

[17] Eroglu, E., & Adiguzel, B., *A genetic algorithm based approach to the workload balancing problem*, 4th International Logistics and Supply Chain Management Congress, 2006.

[18] Fontes, D., & Goncalves, J. F., *Heuristic solutions for general concave minimum cost network flow problems,* Networks, An International Journal, https://doi.org/10.1002/net.20167 (vis.: August 2018), vol. 50, no. 1, pp. 67-76, 2007.
[19] Fulkerson, D. R., *An Out-of-Kilter Method for Minimal-Cost Flow Problems,* Journal of the Society for Industrial and Applied Mathematics, vol. 9, no. 1, pp. 18-27, 1961.
[20] Goldberg, A. V., & Tarjan, R. E., *Finding Minimum-Cost Circulations by Canceling Negative Cycles,* Journal of the Association for Computing Machinery, vol. 36, no. 4, pp. 873-886, 1989.
[21] Goldberg, A. V., & Tarjan, R. E., *Solving minimum-cost flow problems by successive approximations,* 19th ACM Symposium on Theory of Computing, STOC 87, New York: ACM Press, pp. 7-18, 1987.
[22] He, Q., Shabbir, A., & Nemhauser, G. L. *Minimum Concave Cost Flow Over a Grid Network,* Mathematical Programming, vol. 150, no. 1, pp. 79-98, 2015.
[23] Holland, J. H., *Genetic algorithms,* Scientific American , vol. 267, pp. 66-72, 1992.
[24] Horst, R., & Pardalos, P. M. *Handbook of Global Optimization,* Springer - Science + Business Media, 1st edition, 1995.
[25] Horst, R., & Tuy, H., *Global optimization, Deterministic Approaches.* New-York: Springer-Verlag, 3rd edition 1996.
[26] Jiang, Y., Xu, X., Zhang, H., & Luo, Y., *Taxiing Route Scheduling between Taxiway and Runway in Hub Airport.* Mathematical Problem in Engineering, Article ID 925139, http://dx.doi.org/10.1155/2015/925139 (vis. August 2018), 14 pages, 2015.
[27] Klein, M., *A primal method for minimal cost flows with applications to the assignment and transportation problems,* Management science, vol. 14, no. 3, pp. 205-220 1967.
[28] Kovacs, P., *Minimum-cost flow algorithms: An experimental evaluation,* Egervary Research Group, Technical reports, http://bolyai.cs.elte.hu/egres/ (vis. August 2018), 2013.
[29] Kudjo, P. K., & Ocquaye, E., *Review of Genetic Algorithm and Application in Software Testing,* International Journal of Computer Applications, vol. 160, no. 2, pp. 1-6, 2017.
[30] Luenberger, D. G., & Ye, Y., *Linear and nonlinear programming,* International Series in Operations Research and management science. Stanford: Springer, 2008.
[31] Maher, R. A., & Abdula, F. A., *An Algorithm for Cost-Minimizing in Transportation via Road Networks Problem,* International Journal of Mathematical and Computational Methods, http://www.iaras.org/iaras/journals/ijmcm, vol. 2, pp. 292-299, 2017.
[32] Moanță, D., *Principii privind algoritmii genetici pentru soluționarea unei probleme tridimensionale de transport,* Revista Informatica economică, no. 6, pp. 47-51, 1988.
[33] Osaba, E., Carballedo, R., Diaz, F., Onieva, E., Iglesia, I., & Perallos, A., *Crossover versus Mutation: A Comparative Analysis of the Evolutionary Strategy of Genetic Algorithms Applied to Combinatorial Optimization Problem,* The Scientific World Journal, Article ID 154676, http://dx.doi.org/10.1155/2014/154676, (vis. August 2018), 22 pages, 2014.
[34] Paiu, O. I., *Algoritmi genetici seriali și paraleli,* Informatică economică, no. 6, pp.52-58, 1998.
[35] Sadegheih, A., & Drake, P. R., *A novel experimental analysis of the minimum cost flow problem,* IJE Transactiona A: Basics, vol. 22, no. 3, pp. 251-268, 2009.
[36] Sifaleras, A., *Minimum cost network flows: problems, algorithms, and software,* Yugoslav Journal of Operations Research , vol. 23, no. 1, pp. 3-17, 2013.
[37] Sleator, D. D., & Tarjan, R. E., *A data structure for dynamic trees.* Journal of Computer and System Sciences , vol. 26 no. 3, pp. 362-391, 1983.
[38] Sokkalingam, P. T., Ahuja, R. K., & Orlin, J. B., *New polynomial-time cycle-canceling for minimum cost flows.* Networks , vol. 36, pp. 53-63, 2000.
[39] Sun, W., & Yuan, Y.-X., *Optimization theory and methods, Nonlinear Programming,* Springer Science + Business Media, LLC, vol. 1, 2006.
[40] Tomizawa, N., *On some techniques useful for solution of transportation network problems.* Networks , vol. 1, pp. 173-194, 1971,
[41] Wang, I.-L., & Lin, S.-J., *A network simplex algorithm for solving the minimum distribution cost problem.* Journal of industrial and management optimization , vol. 5 no. 4, pp. 929-950, 2009.
[42] Yang, Z., Wang, W., Chen, S., Ding, H., & Li, X., *Genetic Algorithm for Multiple Bus Line Coordination on Urban Arterial,* Computational Intelligence and Neuroscience, Article ID 868521, http://dx.doi.org/10.1155/2015/868521 (vis. August 2018), 7 pages, 2015.
[43] Zhang, E., Mei, Q., Liu, M., & Zheng, F., *Stowage Planning in Multiple Ports with Shifting Fee Minimization.* Scientific Programming, Article ID 3450726, https://doi.org/10.1155/2018/3450726 (vis. August 2018), 9 pages, 2018.
[44] Zhang, P., Sun, Z., & Liu, X., *Optimized Skip-Stop Metro Line Operation Using Smart Card Data.* Journal of Advanced Transportation, Article ID 3097681, https://doi.org/10.1155/2017/3097681, (vis. August 2018), 17 pages, 2017.