Computer Science Journal of Moldova, vol.**28**, no.2(83), 2020

Solving transportation problems with concave cost functions using genetic algorithms

Tatiana Paşa

Abstract

In this paper we propose a genetic algorithm for solving the non-linear transportation problem on a network with concave cost functions and the restriction that the flow must pass through all arcs of the network. We show that the algorithm can be used in solving large-scale problems. We prove that the complexity of a single iteration of the algorithm is $O(nm)$ and converges to an ϵ-optimum solution. We also present some implementation and testing examples of the algorithm using Wolfram Mathematica.

Keywords: genetic algorithm, population, minimum cost flow, non-linear transport problem, large-scale problem, concave function.

MSC 2010: 05C21, 90C06, 90C26, 90B15, 90B06, 90C59.

1 Introduction

Problems that describe real situations using concave functions are often too complex to be solved by polynomial algorithms; therefore, they can be solved only by checking each possible solution with brute force algorithms. Because such algorithms would be too time consuming, genetic algorithms are an alternative that can find the solution in a reasonable amount of time. These are stochastic and heuristic algorithms, which means that the obtained solutions are not always optimal, but they come close to the optima. The use of these algorithms in favour to other heuristic algorithms is recommended, because they do not need the gradient or Hessian information. They are also resistant to locks

in a local minimum and can be used to solve large-scale non-linear optimization problems.

There are several principles [1] that must be followed when designing and implementing a genetic algorithm. We must use several characteristic operations, i.e. selection, crossover and mutation, to improve the final solution. When codifying a solution, we must keep in mind to use as little memory as possible for the chromosomes. The complexity of the evaluation, crossover and mutation has to be of a low order. Each chromosome, mutation or crossover must correspond to an admissible solution, thus we can guarantee the correctness of the algorithm. Although it is possible to decode the chromosomes one-to-one (each chromosome corresponds to a single solution), one-to-n (each chromosome corresponds to n solutions) or n-to-one (n chromosomes correspond to a single solution), the one-to-one decoding is recommended. When searching for a solution to the problem, a balance must be kept between the exploration of as many of the admissible solutions as possible and the exploitation of the solution as close to the optimal solution as possible.

An overview and comparative analysis of the genetic algorithms is given in [3,4]. In [2] an original genetic algorithm with local search is presented.

In this paper we present a modification to the algorithm discussed in [5,6] that can solve the non-linear transport problem with concave cost functions. This new algorithm was tested in Wolfram Mathematica, and some results are presented.

2 Problem formulation. Main results

We consider the transportation problem on a network described by a connected acyclic graph. On the finite set of vertices V the real function of production and consumption $q(v)$ is defined. On the finite set of arcs E the concave non-decreasing piecewise linear functions of cost $\varphi_e(x_e)$ are defined. It is required to solve the non-linear optimization problem that consists in determining a flow x^* that minimizes the function:

$$F(x) = \sum_{e \in E} \varphi_e(x_e)$$

We must solve the non-linear problem:

$$F(x^*) = \min_{x \in X} F(x) \tag{1}$$

$$\sum_{e \in E^+} x_e - \sum_{e \in E^-} x_e = q(v) \tag{2}$$

$$x_e > 0, \forall(e), \tag{3}$$

where X is the set of admissible solutions which satisfies the conditions (2) – (3) of existence of flow in the network. An additional restriction is that there must be a flow passing through every arc of the network. This condition ensures that the algorithm is capable of arriving at the solution, otherwise it would be nearly impossible for it to find a solution with no flow through some arcs. Such a problem is the model of the transportation of a flow through water or electricity networks.

$$q(v) = \begin{cases} p(v) = \sum_{i=1}^{k} p(v_i), & v = v_0, v_i \in V_t, \forall i = 1, ..., k \\ 0, & V/V_t \{v_0\} \\ -p(v_i), & v_i \in V_t, \forall i = 1, ..., k \end{cases} \tag{4}$$

It is preferable to use an elitist model of the genetic algorithm that transfers the best chromosomes to the next population, in order to avoid losing solutions that cannot be restored later.

Definition 1. *The value of the total objective function of the population $F_T(x)$ is the sum of the values of the objective function of the solutions decoded from the chromosomes of the population.*

The value of the total objective function will be smaller (better) after each step, because new chromosomes will have better characteristics.

2.1 Genetic algorithm P2

The first step in the use of a genetic algorithm is the codification of the problem, that is, the description of the chromosomes that each represents the admissible solution of the problem. A population consists of chromosomes which, in fact, represents a set of admissible solutions.

In this algorithm every gene of a chromosome will be a matrix that contains the rate of the flow that passes through outgoing arcs of the vertex i. To save computer memory, the chromosomes will not be described by a weighted matrix of size $n \times n$ which is sparse, but by a table of lists of total size m. Each list i contains the rate of the flow for those arcs coming out of the vertex i.

Description of the genetic algorithm P2:

Step 1.

Initialization. The initial population of $4n$ chromosomes is generated as follows: every chromosome is described by a list of the form $L = \{\{l_{i1}, ..., l_{im_i}\} | \forall i = 1, \ldots, n\}$, m_i – the number of outgoing arcs of the vertex i. Every $l_{ij}, i = 1, \ldots, n, j = 1, \ldots, m_i$ shows the part of the flow that passes through the arc j that comes out of vertex i. For every vertex the condition $\sum_{j=1}^{m_i} l_{ij} = 1, i = 1, \ldots, n$ must be satisfied.

Step 2.

Decoding and Evaluation of the chromosome from the current population. The decoding is the association of each chromosome with an admissible solution based on how the problem is codified. Evaluation is the determination of the value of the objective function for each of the decoded solutions.

Step 3.

Selection of the parent chromosomes that will participate in the creation of the next population. The chromosomes will be sorted in order of increasing value of the objective function. The first $2n$ of them will be transferred to the next population and will participate in crossover and creation of the offsprings.

Step 4.

Crossover of the chromosomes will occur between chromosomes transferred from population $P(i-1)$ to $P(i)$. The parents will be

cut randomly at the same point. Then the chromosomes will be combined in the following way: the left half of the first chromosome with right half of the second chromosome will yield one offspring, then another offspring will be obtained by combining the right half of the first chromosome with the left half of the second one. Thus every pair of parents will have two offsprings, and the total size of the population will remain constant.

Step 5.

Mutation of a chromosome will take place at a rate of $\alpha \in [0.01, 0.1]$ and implies the mutation of a random gene. A new list $\{l_{i1}, ..., l_{im_i}\}$ will be generated for a random vertex i such that the condition $\sum_{j=1}^{m_i} l_{ij} = 1, i = 1, \ldots, n$ is satisfied.

Step 6.

Checking the stopping condition implies stopping the algorithm when the condition $\left|f(x_{P(i)}) - f(x_{P(i-1)})\right| \leq \epsilon$ is satisfied for the solutions associated with the first chromosome of two consecutive populations $P(i-1)$ and $P(i)$ sorted in order of increasing value of the objective function. The solution to the problem will be the solution that corresponds to the first chromosome from the last population. If the stopping condition is not satisfied, the algorithm returns to Step 2.

Observation 1. *One of the following conditions can also be used to stop the algorithm:*

- *Generation of k populations. During our testing there proved to be no advantages to this stopping condition in favor of the condition* $\left|f(x_{P(i)}) - f(x_{P(i-1)})\right| \leq \epsilon$*, except minor improvements in the final solution or rare major changes. It works better in small tests where the solution set isn't very large and mutation or crossover that leads to a much better solution can easily be found.*

- *Generation of populations until a time limit is especially desirable for large networks. This condition is useful for extremely large networks, where the execution time and the complexity of the problem is too large for the condition* $\left|f(x_{P(i)}) - f(x_{P(i-1)})\right| \leq \epsilon$*. It is preferable for the generation of k populations, because it is*

not possible to accurately judge the amount of time needed for an iteration.

Decoding a chromosome of the population to a solution associated with it is done as follows: it is known that a flow $p(v)$ must pass through the transportation network. This flow will be distributed over the arcs of the network using the list L generated in step 1, which contains the part of the total flow available in vertex i that passes through the arc (i, j), $\forall j = 1, \ldots, m_i$. The value obtained thus is the flow x_k associated with the arc (i, j). This value will be placed on the position k in the admissible solution of the problem, which has the form $x = (x_1, x_2, ..., x_m)$.

2.2 Theoretical results

The algorithm P2 described above can be applied when the network satisfies the following conditions:

- The graph that describes the network is acyclic;
- Every vertex of the graph, except the destination, has at least one outgoing arc;
- The destination vertex does not have any outgoing arcs.

Theorem 1. *The genetic algorithm P2 requires memory $O(nm)$.*

Proof. The transportation network is described by an adjacency list of size m. Every chromosome consists of a list L of size m. Thus a population of $4n$ chromosomes will be of size $4nm$. This population will be renewed at each iteration of the algorithm and no additional memory will be needed. Therefore, the algorithm requires memory $O(nm)$.□

Theorem 2. *The complexity of a single iteration of the genetic algorithm P2 is $O(nm)$.*

Proof. To fill in the adjacency list that describes the graph, $O(m)$ operations are necessary. The generation of a single chromosome has a

complexity of $O(m)$, thus the generation of a population requires $O(nm)$ operations. The evaluation of a solution associated to a chromosome requires $O(m)$ operations. As there are $4n$ chromosomes in a population, the total complexity is $O(nm)$. The crossover and mutation have a complexity of $O(m)$ for each chromosome and require $O(nm)$ operations in total. Therefore a single iteration of the algorithm has the complexity $O(nm)$.□

Definition 2. *An ϵ-optimum solution of the non-linear transportation problem on a network is the solution $x_{P(i)}$ generated by a population $P(i)$ that satisfies the condition $\left|f(x_{P(i)}) - f(x_{P(i-1)})\right| \leq \epsilon$.*

Theorem 3. *The genetic algorithm P2 always converges to an ϵ-optimum solution.*

Proof. The chromosomes whose objective function is the smallest will be transferred from the population $P(i-1)$ to $P(i)$. The value of the total objective function of each new population will be smaller than the previous population. Each new population $P(i)$ will generate a solution whose objective function will be smaller or equal to the solution generated by the population $P(i-1)$. Because the algorithm converges to a local minimum and satisfies the condition $\left|f(x_{P(i)}) - f(x_{P(i-1)})\right| \leq \epsilon$, we say that it converges to an ϵ-optimum solution.□

Observation 2. *Based on Theorem 2, it can be implied that the execution time of the genetic algorithm P2 is $O(Unm)$, where U is the number of necessary iterations to obtain an ϵ-optimum solution.*

2.3 Practical results

Below we will consider the application of the Genetic Algorithm P2 on a set of test cases.

Example 1. *Let the transportation network be described by a connected acyclic graph.*

The set of the vertices $\{1,2,3,4,5\}$ *is associated with the production and consumption function:* $q(v)=\begin{cases} 15 \text{ if } v=1 \\ 0 \text{ if } v=2,3,4 \\ -15 \text{ if } v=5 \end{cases}$.

Every arc from the set $\{e_1,e_2,e_3,e_4,e_5,e_6,e_7,e_8,e_9\}$ *is associated with a cost function as follows:*

– *the cost function* $\varphi_1(x)=\begin{cases} x \text{ if } x\leq 1 \\ 1 \text{ if } x>1 \end{cases}$ *will be associated with the arcs* $\{e_1,e_3,e_4,e_7\}$;

– *the cost function* $\varphi_2(x)=\begin{cases} 2x \text{ if } x\leq 2 \\ 4 \text{ if } x>2 \end{cases}$ *will be associated with the arcs* $\{e_2,e_5,e_6,e_8,e_9\}$.

The chromosome mutation will be done with a probability of 0.1.

Step 1. Initialization of a population of 20 chromosomes, each containing 5 lists. Each list i will have some numbers that describe the part of the flow that passes through each outgoing arc of that vertex.

The value of the objective function, calculated for the first generated chromosome $\{\{0.39,0.01,0.23,0.37\},\{0.48,0.52\},\{0.18,0.82\},\{1.00\}\}$ *associated with the solution* $x=\{3.4,5.9,5.7,0.11,1.6,1.8,6.1,1.3,12.\}$, *is F(x)=17. This value describes the initial cost of transporting 15 units of the product from vertex 1 to vertex 5.* $F_T(x)=340$ *units.*

Iteration 1 The chromosomes of the initial population are sorted in increasing order of the value of the objective function for the solution associated with the respective chromosomes. The first half of them is transferred to the next population, and it will be the parents of the second half of that population. The offspring is obtained through crossover and then mutated by choosing a random vertex i and generating a new list $\{l_{i1},...,l_{im_i}\}$. *The minimum value of the objective function for the generated population is: F(x)=10 and* $F_T(x)=310$ *units.*

Iteration 2 The chromosomes of the population 1 are sorted in increasing order of the value of the objective function for the solution associated with the respective chromosomes. The first half of them is transferred to the next population, and it will be the parents of the second half of that population. The offspring is obtained through

crossover and then mutated by choosing a random vertex i and generating a new list $\{l_{i1}, ..., l_{im_i}\}$. *The minimum value of the objective function for the generated population is F(x)=10 which is equal to the minimum value of the objective function of the previous population. Because the solutions of two consecutive populations coincide, then* $x = \{3.4, 0.27, 7.7, 3.6, 3.1, 0.37, 3.2, 0.09, 11.\}$ *will be the* ϵ*-optimum solution of the presented problem.* $F_T(x) = 290$ *units. STOP.*

The following examples will be of much larger dimensions (n – vertices, m – arcs), that is why we will present only the value of $F_T(x)$ obtained at each iteration and the value of the objective function for the ϵ-optimum solution *(Table 1.)*.

Table 1. Examples 2-4 of GA

Iteration	**Ex. 2 (u.c.)**	**Ex. 3 (u.c.)**	**Ex. 4 (u.c.)**
	n=30 / m=238	n=100 / m=2392	n=150 / m=5875
1	*74520*	*479731*	*683430*
2	*70750*	*463270*	*658387*
3	*68168*	*461585*	*640905*
4	*65941*	*454880*	*624014*
5	*63955*	*448784*	*608769*
6	*61794*	*442621*	*593624*
7	*59992*	*437814*	*579315*
8	*58840*	*433995*	*564279*
9	*57422*	*430823*	*550445*
10	*56276*	*428015*	*535530*
11	*56276*	-	*525713*
F(x)	430	1000	820

The tests were performed on an Intel i5-2500 machine with 4 Cores and 8GB DDR3 memory in the Wolfram Mathematica 12.

Through practical examples the convergence of the algorithm is evident. By calculating $F_T(x)$ we notice a decrease of this value from one iteration to another.

The algorithm described in Section 2.1 was implemented in Wolfram Language and tested on a set of examples of transportation networks of various dimensions in terms of number of arcs and vertices of the graph that describes the network.

Table 2. Execution time of the GA P2

vertices/ arcs	$t_\epsilon(sec.)$	$t_k(sec.)$	**vertices/ arcs**	$t_\epsilon(sec.)$	$t_k(sec.)$
10 / 33	*0.078*	*0.3125*	**50 / 647**	*9.9687*	*27,7813*
15 / 62	*0.3125*	*0.2300*	**70 / 1274**	*42,9375*	*78,9219*
20 / 105	*0.4843*	*1.7675*	**80 / 1764**	*45,7031*	*130,0320*
25 / 168	*0.9687*	*3.4843*	**90 / 2275**	*216,6410*	*194,5470*
30 / 231	*4.8595*	*5.4218*	**100 / 2572**	*354,4537*	*244,5780*
40 / 402	*7.3593*	*13.4219*	**120 / 4530**	*237,4530*	*430,5310*

The tests (*Table 2*) were performed using two stopping conditions:

1. the algorithm is stopped only when the condition $\left|f(x_{P(i)}) - f(x_{P(i-1)})\right| \leq \epsilon$ is satisfied for solutions corresponding to the value of the minimum objective function from two consecutive populations $P\,(i-1)$ and $P\,(i)$, i.e. an ϵ-optimum solution was found;

2. the algorithm is stopped after $k = 10$ iterations, and the final solution is the solution from the last population with minimum objective function.

3 Conclusions

Genetic algorithms are very good solutions to nonlinear transportation problems with concave cost functions, especially when there are additional restrictions that force the flow to pass through all arcs of the network. From the above we can state that:

- The genetic algorithm P2 is correctly codified because it respects the condition of existence of flow in the network and lets us solve large-scale problems in reasonable time;
- The proposed decoding algorithm always obtains an admissible solution;
- The value of the total objective cost function $F_T(x)$ decreases from one generation to another, and practical tests confirm the convergence to a local solution that is also an ϵ-optimum solution;
- It has been proven practically that the stopping condition $\left|f(x_{P(i)}) - f(x_{P(i-1)})\right| \leq \epsilon$ is correct and lets us obtain a good result much faster than constructing a fixed number of populations.

References

[1] M. Gen, "Multiobjective Genetic Algorithms," in *Network Models and Optimization, Multiobjective Genetic Algorithm Approach*, Girona, Spain: Springer-Verlag London Limited, 2008, pp. 1–44.

[2] B. Ghasemishabankareh, *et al.*, "A genetic algorithm with local search for solving single-source single-sink nonlinear non-convex minimum cost flow problems," *The Soft Comput*, [Online]. Available: https://doi.org/10.1007/s00500-019-03951-2, (vis. July 2019), 17 pages, 2019.

[3] P. K. Kudjo and E. Ocquaye, "Review of Genetic Algorithm and Application in Software Testing," *International Journal of Computer Applications*, vol. 160, no. 2, pp. 1–6, 2017.

[4] E. Osaba, *et al.*, "Crossover versus Mutation: A Comparative Analysis of the Evolutionary Strategy of Genetic Algorithms Applied to Combinatorial Optimization Problem," *The Scientific World Journal*, [Online]. Available:

http://dx.doi.org/10.1155/2014/154676, (vis. July 2019), 22 pages, 2014.

[5] T. Pașa, "The genetic algorithm for solving the non-linear transportation problem," in *Review of the Air Force Academy, The Scientific Informative Review*, SPSR 2018, 21th edition, Bucharest, 2018, vol. XVI, no. 2(37), pp. 37–44. [Online]. Available: http://www.afahc.ro/ro/revista/ 2018_2/4-TatianaPasa.pdf.

[6] T. Pașa, "Solving the large-scale non-linear transportation problem," in *Matematics and Information Tehnologies: Research and Education, MITRE-2019*, (CEP USM, Chișinău), 2019, p. 53.

Tatiana Paşa

Moldova State University
60 A. Mateevici, MD-2009, Chișinău
Republic of Moldova
E–mail: pasa.tatiana@yahoo.com

Received July 13, 2019
Revised December 24, 2019
Accepted May 4, 2020