SOLVING THE TRANSPORTATION PROBLEM WITH PIECEWISE-LINEAR CONCAVE COST FUNCTIONS

Tatiana PAȘA, Valeriu UNGUREANU

Universitatea de Stat, Chișinău, Moldova (pasa.tatiana@yahoo.com, v.a.ungureanu@gmail.com)

DOI: 10.19062/1842-9238.2017.15.2.6

Abstract: *In this work we expose the transportation problem on a network with piecewise-linear concave cost functions on edge flows. Some properties of the problem are highlighted. A series of results are presented, which concern the implementation of a method exposed in [1] in Wolfram Mathematica System. These results are compared with the results obtained by applying built-in Wolfram Language functions on a family of test problems. Tests are provided both on PC-s and a cluster.*

Keywords: *network transportation problem, optimal solution, flow in a network, concave function.*

1. INTRODUCTION

Transport have a very important role in the socio-economic development of a country because it facilitates the transportation of passengers and goods to the point of destination.

Air transportation is a civil or military aviation branch which deals with the transport of goods and passengers. This type of transport is preferred by those who have the primary purpose of moving as quickly as possible from one point to another, this being its essential feature, but also the precise schedule for any time of the year regardless of day or night. For this purpose both mixed-duty aircrafts for passengers and goods, as well as cargo aircrafts, are used. Airplanes can be used on a regular basis - regular traffic, or irregular traffic, which means contracts between airlines and different recipients who want to operate the rented airplanes for a specified period. The most important airports based on the number of passengers and quantity of goods are in London, Tokyo, New York, Los Angeles and Frankfurt.

If we are talking about the economic aspect of air transport, everything has a cost, therefore, for a quantity of cargo or for a certain number of passengers transported we have a price that varies if the quantity of the cargo or the number of passengers changes. This is why the price per passenger or per ton of product may vary. In this case, we can say that the cost of the transport can be described by a concave nonlinear function, i.e. the cost will depend nonlinearly on the number of passengers or on the weight of the cargo being transported.

The air transport network consists of paths and nodes where the connections between the nodes are made by means of air transport such as airplanes and helicopters.

The network that starts from a node (airport) and allows the transport of passengers and / or cargo to another node, but can also use other airports as transit nodes, may be formalized by graph theory means as a mathematical model of a standard single-source transport network shipping with one destination and intermediate points. A single-source transport network with several destinations describes the real situation when goods and passengers are transported to several nodes using nodes of transit.

2. PRELIMINARIES

First, let us expose notions that a necessary for studying the transportation problem with concave cost functions.

Transportation network: A transportation network [2] is an oriented graph $G = (V, E)$, without loops, which satisfies the following properties:

1. There is a vertex (source) $v_0 \in V$ which has only outgoing edges;

2. There is a vertex (destination, sink) $v_t \in V$ which has only incoming edges;

3. For each arc it is associated a value $c(e)$, for any $e \in E$, named capacity of the arc.

The cases with several sources and / or destinations can be easily modelled. To reduce such models to the standard transportation network we have to add a super-source connected to each of the original sources. Its capacity will be the total of the product amount from the original sources. We can also add a super-sink connected to the original destinations, its capacity will be equal to the total capacity of all destinations.

Generally, we can set for each edge a value that describes the maximal size or quantity of the goods which may be transported, distances between nodes, the time to cover the distance or the price to transport the cargo. In this paper, a piecewise function is assigned to each edge describing the transport costs and depending nonlinearly on the quantity of the goods being transported.

Flow: A flow in a network is a function $f: E \to \mathbb{R}$ which satisfies the following properties:

1. Capacity constraint: For all $e \in E$ the condition $f(e) \leq c(e)$ is satisfied;

2. Skew symmetry: For all $(v_1, v_2) \in V$ the condition $f(v_1, v_2) = -f(v_2, v_1)$ is satisfied;

3. Flow conservation: $\sum_{x \in E^+(v)} x(e) - \sum_{x \in E^-(v)} x(e) = 0$, where $E^+(t)$ is the set of edges that enter $t \in V$ and $E^-(s)$ - that exit $s \in V$.

Capacity constraints limit the quantity that can be transported along an arc. In the context of this paper, the capacities are equal to ∞, i.e. an unlimited quantity can be transported along an arc.

Skew symmetry assumes that for each node the quantity of product entering the node equals the quantity that exits it, i.e. their modulus (absolute values) are equal.

Flow conservation implies that the whole quantity of product that exits from the source reaches the destination, i.e. there are no losses along the arcs.

It is also impossible to have a surplus in the quantity when reaching the destination, because that would imply that the intermediate nodes also produce goods.

Concave function: A function f is concave on an interval, if for all x and y from this interval and for all $\alpha \in [0,1]$ the following is true:

$$f\big((1-\alpha)x + \alpha y\big) \geq (1-\alpha)f(x) + \alpha f(y).$$

In this paper, concave functions describe the cost of the product shipping along the arcs. These functions are non-increasing piecewise-linear functions defined on the interval [0; +∞] that initially describes a rise from 0 to some value which then become a constant value. This means that the transport cost raises with the increase of the product flow only up to a fixed value, then the expenses are the same for any product quantity being transported.

3. PROBLEM FORMULATION

Let us consider the network transportation problem described by the graph $G = (V, E)$, $|V| = n$, $|E| = m$ with the source $v_0 \in V$ and the sink $v_t \in V$. A real bounded function $q: V \to R$ is defined on the finite set of its vertices V and non–decreasing piecewise-linear concave cost functions $\varphi_e(x_e)$, defined for each arc $e \in E$, which depend on flow. We must find a flow x^* for which the function $F(x) = \sum_{e \in E} \varphi_e\big(x(e)\big)$ has the smallest value, i.e. $F(x^*) = \min_{x \in X} F(x)$, where X is the set of flows admissible in G, which means that it satisfies the system:

$$\begin{cases} \sum\limits_{x \in E^+(v)} x(e) - \sum\limits_{x \in E^-(v)} x(e) = q(v), & for\ all\ v \in V \\ x(e) \geq 0, & for\ all\ e \in E \end{cases}$$

where $E^+(v)$ is the set of edges that enter $v \in V$ and $E^-(v)$ - that exit $v \in V$.

1. In the formulated problem for every intermediary node the flow is conserved which means that:

$$q(v) = \begin{cases} -p, & v = v_0, \\ 0, & v \in V \backslash \{v_0, v_t\}, \\ p, & v = v_t, \end{cases}$$

where p is the net capacity.

Therefore, the transportation problem may be formulated as it follows:

$F(x^*) = \min\limits_{x \in X} F(x)$,

where X is defined by the function:

$$\begin{cases} \sum\limits_{x \in E^+(v)} x(e) - \sum\limits_{x \in E^-(v)} x(e) = \begin{cases} -p, & v = v_0 \\ 0, & v \in V \backslash \{v_0, v_t\}, \\ p, & v = v_t \end{cases} & for\ all\ v \in V, \\ x(e) \geq 0, & for\ all\ e \in E. \end{cases}$$

2. A particular case of this problem is the transportation network with a source and several destinations. The function will then be described by the system:

$$q(v) = \begin{cases} -\sum_{u \in V_t} p_u, & v = v_0 \\ 0, & v \in V/V_t \backslash \{v_0\} \\ p_v, & v \in V_t \end{cases}$$

where the set of vertices $V_t \subseteq V$ is the set of destinations. For every $v \in V_t$ it is known the necessity p_v of the point v.

The transportation problem may be formulated as it follows: $F(x^*) = \min_{x \in X} F(x)$, where X is defined by the system:

$$\begin{cases} \sum_{x \in E^+(v)} x(e) - \sum_{x \in E^-(v)} x(e) = q(v) = \begin{cases} -\sum_{u \in V_t} p_u, & v = v_0 \\ 0, & v \in V/V_t \backslash \{v_0\} \\ p_v, & v \in V_t \end{cases}, & for\ all\ v \in V \\ x(e) \geq 0, & for\ all\ e \in E \end{cases}$$

The formulated problem is NP-hard because the function is a sum of piecewise-linear concave functions, which means that there are no efficient (polynomial) algorithms to solve this problem. That's why our aim is to find an algorithm that will find a good solution as efficiently as possible.

4. WAYS OF SOLVING THE PROBLEM

4.1 Applying a known algorithm: In [1] an algorithm for solving the non-linear transport problem is described by reducing the solution to a problem of linear programming for which polynomial algorithms are known. Fig. 1 shows the block diagram describing the algorithm.

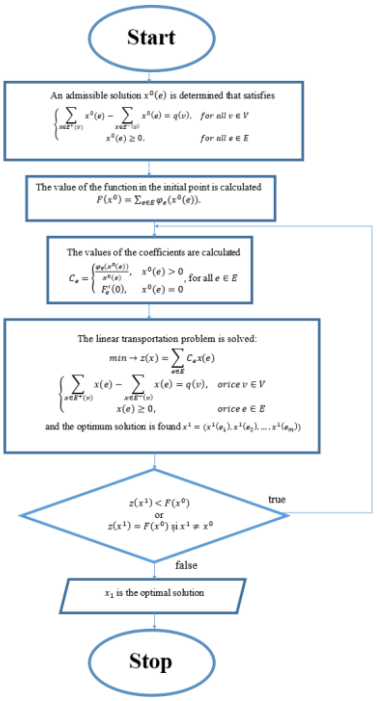


FIG.1. Block diagram for solving the TP

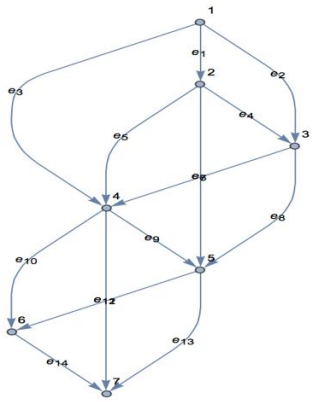
The Mathematica System and the Wolfram Language [3] make it much easier to implement the algorithm comparing with other languages and systems. The code is compact, easy-to-read, it is easy to define new variables and functions.

Next, we will explain how we applied the Wolfram Language to solve the problem based on the following example.

The description of the transport network is easily accomplished using the standard *Graph[]* function to which all pairs of connected vertices describing the direction of the edges are transmitted as parameters, see following Fig. 2 (a):

```
Graph[{Labeled[1 ↔ 2, "e1"], Labeled[1 ↔ 3, "e2"],
  Labeled[1 ↔ 4, "e3"],
  Labeled[2 ↔ 3, "e4"], Labeled[2 ↔ 4, "e5"],
  Labeled[2 ↔ 5, "e6"],
  Labeled[3 ↔ 4, "e7"], Labeled[3 ↔ 5, "e8"],
  Labeled[4 ↔ 5, "e9"], Labeled[4 ↔ 6, "e10"],
  Labeled[4 ↔ 7, "e11"],
  Labeled[5 ↔ 6, "e12"], Labeled[5 ↔ 7, "e13"],
  Labeled[6 ↔ 7, "e14"]}, VertexLabels → "Name"]
```

(a)



(b)

FIG. 2. Description of the transportation network

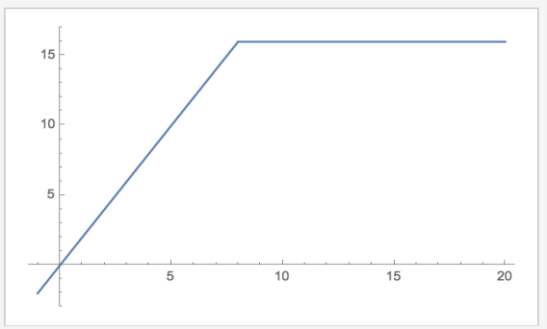
As a result of executing the *Graph[]* function we obtain the network represented in Fig. 2. (b), which is it graphical representation. Knowing the transportation network, the *IncidenceMatrix[]* function allows us to obtain the incidence matrix of the graph that describes the transportation network and permits to construct a system of constraints for the formulated problem.

As mentioned above, with each arc is associated a piecewise-linear concave function that is described generally as in Fig. 3 (a) and (b) using the Wolfram Language.

```
γ = {1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2};
δ = {1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2};

f[j_Integer, x_] := Piecewise[{{γ[[j]] x, x ≤ δ[[j]]},
                               {γ[[j]] δ[[j]], x > δ[[j]]}},
                              0, True]
```

(a)



(b)

FIG. 3. Description and graphic representation of the piecewise-linear function

In order to obtain an initial solution with which the program starts according to the block diagram of Fig. 1, we use the *FindInstace[]* standard function which solves the system of equations, provides the non-negative solutions, and as a result submits the set of possible solutions on the basis of which a linear objective function is obtained.

LinearPrograming[] is a standard function that solves the problem of linear programming that consists in minimizing the obtained linear objective function on the same system of constraints. In Fig. 4 we have the code of the implemented algorithm.

```
X1 = X0 = Values@FindInstance[{A.X == b[[All, 1]], X ≥ 0}, X][[1]];
F0 = Sum[f[j, X0[[j]]], {j, 1, m}];
Z1 = -∞;
i = 0;
While[Not[(Z1 > F0) || (Z1 == F0 && X0 == X1)],
 X0 = X1; F0 = Sum[f[j, X0[[j]]], {j, 1, m}];
 c = Table[fd[j, X0[[j]]], {j, 1, m}];
 X1 = LinearProgramming[c, A, b];
 Z1 = Sum[c[[j]] X1[[j]], {j, 1, m}];
 Print[++i];
 Print[Z1];
]
X1
Sum[f[j, X1[[j]]], {j, 1, m}]
```

FIG. 4. The algorithm code in the Wolfram Language

4.2 Using the standard Wolfram language functions: We can use the standard functions *Minimize[]* and *NMinimize[]*, that can find a global optimum, and *FindInstance[]*, for finding a local solution, in order to solve the formulated problem, which is in fact an optimization problem of nonlinear programming that consists in minimizing of an objective function that is described as a sum of concave piecewise-linear concave functions and satisfies the constraints described by a system of linear equations and constraints of non-negativity.

The prototype for the function *Minimize[{f, cons}, {x, y, ...}]* suggests that as input data we have the objective function and the system of linear constraints. It is used to obtain the exact global solution of the optimization problem that uses linear programming methods, the Lagrange multiplier method, integer programming methods and other symbolical and analytical methods, which involves obtaining exact solutions.

The prototype of the function *NMinimize [{f, cons}, {x, y, ...}]* suggests that, as input, we will have the objective function and the constraint system. It is used to obtain a global solution as a numerical value through the use of linear programming methods, Nelder-Mead, random search, i.e. numerical methods, which implies obtaining a result obtained by a series of approximations. We can use the attribute *Method* to select the method, e.g. *DifferentialEvolution*, *RandomSeach*, *SimulatedAnnealing*, *Neldermead*, by which the function *NMinimize[]* used to solve the problem.

The prototype of the function *FindMinimum{{f, cons}, {x, y, ...}]* suggests that as input it have the objective function and the system of constraints. It is used to obtain a local solution, that is a result that starts from a point in the region defined by the constraints. We can use the attribute *Method* to select the method, e.g. *PrincipalAxis*, *InteriorPoint*, *QuasiNewton*, *ConjugateGradient*, by which the function *FindMinimum[]* solves the problem.

Symbolic computation or, in other words, the use of computational algebra that is based on symbols that represent mathematical concepts, operates with polynomials, rational functions, trigonometric functions.

The results of symbolic algorithms are not affected by approximation errors that influence the result in numerical computation that operates with numbers.

Using the standard *Minimize[]* and *NMinimize[]* functions to solve the problem, the user does not know the method that was applied by the Wolfram Language. If we want to solve a series of problems and want to be sure which method was used then the *Method* option followed by the name of the requested method can be used. In Fig. 5 it is given the program that solves the problem using *Minimize[]* without specifying the method to be applied and as a result we have the solution and the value of the function at the point.

```
Minimize[
  {f1[x1] + f2[x2] + f3[x3] + f4[x4] + f5[x5] + f6[x6] +
    f7[x7] + f8[x8] + f9[x9] + f10[x10] + f11[x11] + f12[x12] +
    f13[x13] + f14[x14],
   -x1 - x2 - x3 == -10, (*1*)
   x1 - x4 - x5 - x6 == 0, (*2*)
   x2 + x4 - x7 - x8 == 0, (*3*)
   x3 + x5 + x7 - x9 - x10 - x11 == 0, (*4*)
   x6 + x8 + x9 - x12 - x13 == 0, (*5*)
   x10 + x12 - x14 == 0, (*6*)
   x11 + x13 + x14 == 10, (*7*)
   x1 ≥ 0, x2 ≥ 0, x3 ≥ 0, x4 ≥ 0, x5 ≥ 0, x6 ≥ 0, x7 ≥ 0,
   x8 ≥ 0, x9 ≥ 0, x10 ≥ 0, x11 ≥ 0, x12 ≥ 0, x13 ≥ 0, x14 ≥ 0},
  {x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14}] //
 AbsoluteTiming

{108.452,
 {4, {x1 → 0, x2 → 10, x3 → 0, x4 → 0, x5 → 0, x6 → 0, x7 → 10,
   x8 → 0, x9 → 10, x10 → 0, x11 → 0, x12 → 0, x13 → 10, x14 → 0}}}
```

FIG. 5. Application of the function *Minimize[]* to solve the TP in the Wolfram Language

In order to be able to control and compare the execution time of the algorithm, the *AbsoluteTiming[]* standard function, that returns the runtime of the program in seconds, is used.

We present a list of networks of different sizes and the computation time for each algorithm in Tab. 1.

Table 1. Computation time of the method (seconds)

Nr. Ord.	Nr. Nodes	Nr. Edges	Algorithm	Minimize	NMinimize	FindMinim
1	4	6	0.000279	0.17	0.2	0.18
2	6	10	0.000360	3.02	0.38	0.12
3	7	14	0.000492	108.45	0.61	0.19
4	8	16	0.000468	345.61	2.09	0.19
5	8	20	0.001135	10386.50	1.96	1.75

A substantial rise in the computation time can be observed when the function *Minimize[]* is used, though this method gives the best solution as can be seen in the diagram below.

FIG. 6. The number of optimal solutions for solving TP using different methods

CONCLUSIONS

We formulated a number of problems of different sizes, modelled on different networks with different number of edges and nodes. All were solved using the algorithm from the block scheme in Fig.1 and then using the standard functions *Minimize[]*, *NMinimize[]*, *FindMinimum[]*. The computation time and results of the programs has been compared for each of the used methods. With these data we can expose the following conclusions:

- The solutions obtained with the proposed algorithm is as good as *Minimize[]* or at least not worse than *NMinimize[]*;
- If we could choose another initial solution, the algorithm may give a better solution for the majority of problems;
- Half of the solutions obtained by applying the proposed algorithm are as good as the solutions from *Minimize[]*, while only 25% of the solutions obtained by *NMinimize[]* are as good as *Minimize[]*;
- The proposed algorithm's computation time is the fastest. It is smaller than that of *Minimize[]*, which needs more time for every new edge. E.g., for a network with 8 nodes and 20 edges it needs 3 hours, while our algorithm only about some milliseconds;
- *FindMinimum[]* returns a solution very fast, but it is rarely the optimal solution;
- If the problem is to get a good solution in a short time, our algorithm meets these conditions;
- This is an heuristic algorithm, it doesn't have a rigorous proof and must be improved to be used on a larger array of problems.

An improvement of this algorithm to give the optimum solution in all the cases depends on the initial solution generation, from which the algorithm starts, and may be realised efficiently through parallel computation. Applying an array of different initial solutions and taking the best one as the initial one, even if the computation time of the algorithm will increase it will increase considerably slower than using the function *Minimize[]*.

REFERENCES

[1] T. Pasha, D. Lozovanu. *An algorithm for solving the transport problem on network with concave cost functions on flow of edges.* Computer Science Journal of Moldova, vol. 10, no 3, Kishinev (2002), pp. 341-347.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts London, England, 2-nd edition, 2002.

[3] S. Wolfram, *An elementary introduction to the Wolfram Language.* Champaign, IL: Wolfram Media, Inc., 2016.