Bulletin of the *Transilvania* University of Braşov • Vol 2(51) - 2009
Series III: Mathematics, Informatics, Physics, 255-262

INTERACTIVE TOOL FOR THE SUCCESSIVE SHORTEST PATHS ALGORITHM IN SOLVING THE MINIMUM COST FLOW PROBLEM

Mircea PARPALEA[1]

Abstract

This article covers the minimum-cost flow problem and intends to offer an illustrative tool for efficient education in sciences, especially in the field of flows in networks. In the first part, the article concerns with the basic concepts of the problem itself. The second part describes the "Successive Shortest Path Algorithm" for finding a minimum cost flow in a network and in the final part, a set of two interactive applications of the problem are covered. The using of AWT (Java) in the developed applets brings the advantage of a sophisticated set of Graphical User Interface which allows step by step solving of both shortest path and minimum cost flow problems.

2000 *Mathematics Subject Classification:* 05C85, 11Y16, 05C12.

Key words: Dijkstra Algorithm, Successive Shortest Path Algorithm, Java applet, AWT (Abstract Window Toolkit).

1 Indroduction

Knowledge exchange is a very complex process. Although Internet makes the exchange of information possible at high speed rates, knowledge sharing and know-how broadcasting is still an open problem that is waiting for suitable solutions. The new paradigm of active learning can be fostered with the help of e-learning technologies, which take advantage of the familiarity of the students with computers. Dijkstra's Shortest Path algorithm and Minimum cost flow problem are easier to be understood in an interactive way, regarded as a two-way communication process. In order to point out some elements about the working of these algorithms, two applets, embedded in HTML pages, were developed. These applets are built in Java language using: Applet and AWT classes. AWT was a widget toolkit for Java and provides a sophisticated set of GUI (Graphical User Interface) components.

[1]National College, *Andrei Şaguna* of Braşov, Romania, e-mail: parpalea@gmail.com

2 Theoretical Aspects

2.1 Statement of the Problem

Let $G = (V, E)$ be a directed network defined by a set V of vertexes (nodes) and set E of edges (arcs). For each edge $(i, j) \in E$, two functions are assigned: the positive valued function **capacity** $u(i, j) : E \to \Re^+$ (u_{ij} denotes the maximum amount of flow on the edge (i, j)) and the positive valued function **cost** $b(i, j) : E \to \Re^+$ (b_{ij} denoting the cost per unit of flow on the edge (i, j)). For each vertex $i \in V$ a real valued function $v(i) : V \to \Re$ is assigned, representing the **supply/demand** of that vertex. If $v(i) > 0$, vertex i is a **supply node** and if $v(i) < 0$, vertex i is a **demand node** (its demand is equal to $-v(i)$). A vertex i is called **transshipment** if it has a null supply/demand value, $v(i) = 0$. If all the parameters are to be explicitly outlined, the digraph G is called a **transportation network** and it is written as $G = (V, E, u, b, v)$. Representing the flow on arc $(i, j) \in E$ by x_{ij}, the following optimization model for the minimum cost flow problem can be obtained:

$$\min\ z(x) = \sum_{(i,j)\in E} b_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = v(i) \quad for\ all \quad i \in V \tag{2}$$

$$0 \le x_{ij} \le u_{ij} \quad for\ all \quad (i, j) \in E. \tag{3}$$

The first constraint states that the total outflow of a node minus the total inflow of the node must be equal to mass balance (supply/demand value) of this node. This is known as the **mass balance constraints**. Next, the **flow bound constraints** model physical capacities or restrictions imposed on the flow's range. As it can be seen, this optimization model describes a typical relationship between warehouses and shops, for example, in a case where there is only one kind of product. It is desired to satisfy the demand of each shop by transferring goods from the subset of warehouses, while minimizing the expenses on transportation.

The article concentrates on solving the problem using some ideas related to network flow theory and reviews the necessary theoretical base for some basic algorithms used to solve the minimum cost flow problem.

2.2 Finding a solution

If $\delta = \sum_{i\in V} v(i) \neq 0$ the problem has no solution, because either the supply or the demand dominates in the network and the mass balance constraints come into play. However, this situation can be easily avoided if a special node r, with the supply/demand value, $v(r) = -\delta$ is added. If $\delta > 0$ (supply dominates) then for each node $i \in V$ with $v(i) > 0$ an arc (i, r) with infinite capacity and zero cost is added; otherwise (demand dominates), for each node $i \in V$ with $v(i) < 0$ an arc (r, i) with with the same properties is added. A network with $\sum_{i\in V\cup r} v(i) = 0$ is obtained and this new network has the same optimal value as the objective function.

Even if $\delta = 0$ it is not sure that the edge's capacities allow the transfer of enough flow from supply vertexes to demand ones. In order to establish if the network has a feasible flow, any transfer way what will satisfy all the problem's constraints is to be found. Of course, this feasible solution is not necessarily optimal, but if it is absent the problem cannot be solved. If the network includes more than one supply node (or demand node) a source node s and a sink node t are introduced. For each node $i \in V$ with $v(i) > 0$, a source arc (s, i) with capacity $v(i)$ and cost 0 is added to G. For each node $i \in V$ with $v(i) < 0$, a sink arc (i, t) with capacity $-v(i)$ and cost 0 is added to G. The new network is called a **transformed network**.

For a maximum flow problem from s to t, if the maximum flow saturates all the source and sink arcs, then the problem has a feasible solution; otherwise, it is infeasible. Having found a maximum flow, source, sink, and all adjacent arcs can be removed and a feasible flow in G is obtained.

From the theoretical point of view, for any minimum cost flow problem some necessary conditions for resolving the problem have to be checked:

- the supply/demand balance;

- the existence of a feasible solution;

- the non-existence of uncapacitated negative cycles (If the network contains a negative cost cycle of infinite capacity, supposing that the network has a feasible solution, the objective function will be unbounded).

From the practical point of view, the conditions can be checked while the solution is being found.

2.3 Assumptions

Although the problems could be solved without these assumptions which sometimes can lead to a loss of generality, in their absence the solutions would rapidly become too complex.

Assumption 1: *All data* $(u_{ij}, b_{ij}, v(i))$ *are integers.*

As the computer works with rational numbers, this assumption is not restrictive in practice while rational numbers can be converted to integers by multiplying by a suitable large number.

Assumption 2:*The network is directed.*

If the network were undirected it has to be transformed into a directed one with the requirement that the edge's cost to be nonnegative. To transform an undirected network to a directed one, each undirected edge connecting vertexes i and j is replaced by two directed arcs (i, j) and (j, i), both having the capacity and cost of the replaced edge. Each undirected edge $(i, j) \in E$ has an associated constraint $x_{ij} + x_{ji} \leq u_{ij}$ and the term $b_{ij}x_{ij} + b_{ij}x_{ji}$ in the objective function. Given that $b_{ij} \geq 0$ in some optimal solution either x_{ij} or x_{ji} will be zero. Such a solution is called non-overlapping and every non-overlapping flow in the original network has an associated flow in the transformed network with the same cost, and vise versa.

Assumption 3:*All costs associated with edges are nonnegative.*

This assumption imposes a loss of generality but however, one of the algorithms (cycle-canceling algorithm) is able to work without this assumption. For each vertex $i \in V$ it is associated a number denoted by p_i and called the **potential** of node i. The **reduced cost** b_{ij}^p of an edge $(i,j) \in E$ is defined as

$$b_{ij}^p = b_{ij} + p_i - p_j. \tag{4}$$

Denoting the reduced value by $z(x,p)$, if $p = 0$, then evidently

$$z(x,0) = \sum_{(i,j)\in E} b_{ij} x_{ij} = z(x). \tag{5}$$

For other values of p the following result is obtained:

$$z(x,p) = \sum_{(i,j)\in E} b_{ij}^p x_{ij} = z(x) + \sum_{(i,j)\in E} p_i x_{ij} - \sum_{(i,j)\in E} p_j x_{ij} =$$

$$z(x) + \sum_{i\in V} p_i \sum_{j:(i,j)\in E} x_{ij} - \sum_{j\in V} p_j \sum_{i:(i,j)\in E} x_{ij} =$$

$$z(x) + \sum_{i\in V} p_i \left(\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} \right) = z(x) + \sum_{i\in V} p_i v(i)$$

For a fixed p, the difference $z(x,p) - z(x)$ is constant. Therefore, a flow that minimizes $z(x,p)$ also minimizes $z(x)$ and vice versa.

Theorem 1. *For any node potential p, the minimum cost flow problems with edge costs b_{ij} or b_{ij}^p have the same optimal solutions. Moreover,*

$$z(x,p) = z(x) + \sum_{i\in V} p_i v(i). \tag{6}$$

Theorem 2. *Let G be a transportation network. Suppose P is a directed path from a vertex $a \in V$ to another vertex $b \in V$. Then for any node potential p,*

$$\sum_{(i,j)\in P} b_{ij}^p = \sum_{(i,j)\in P} b_{ij} + p_a - p_b. \tag{7}$$

For a directed cycle W and for any node potential p, holds that

$$\sum_{(i,j)\in W} b_{ij}^p = \sum_{(i,j)\in W} b_{ij}. \tag{8}$$

Introducing a vertex s, adding an arc (s,i) to G with some positive capacity and zero cost for each node $i \in V$ and supposing that for each node $i \in V$, the number p_i denotes the length of the shortest path from s to i with respect to cost function b, for each arc $(i, j) \in E$ the following shortest path optimality condition is satisfied:

$$p_j \leq p_i + b_{ij}. \tag{9}$$

Since, $b^p_{ij} = b_{ij} + p_i - p_j$ and $0 \leq p_i - p_j + b_{ij}$ yields $b^p_{ij} \geq 0$. Moreover, from Theorem 2 results that if G contains a negative cycle, it will be negative for any node potential p in the reduced network. So, if the transportation network has no negative cycle, the costs can be reduced and made positive by finding the shortest paths from the introduced vertex s.

In order to find the shortest path in a graph, Bellman-Ford (label-correcting) algorithm can be used to achieve this goal.

Assumption 4: *The supply/demand function satisfies the condition $\sum_{i \in V} v(i) = 0$ and the minimum cost flow problem has a feasible solution.*

If the network doesn't satisfy the first part of this assumption, it can either be said that the problem has no solution or corresponding transformation can be made according to the steps outlined above. If the second part of the assumption isn't met then the solution doesn't exist. However, many problems are often given in such a way which satisfies all the assumptions.

3 Algorithms

3.1 Working with Residual Networks

Let G be a network and x be a feasible solution of the minimum cost flow problem. Suppose that an edge (i, j) in E carries x_{ij} units of flow. The residual capacity of the edge (i, j) is defined as $r_{ij} = u_{ij} - x_{ij}$. This means that additional r_{ij} units of flow can be sent from vertex i to vertex j. The existing flow x_{ij} on the arc (i, j) can also be canceled by sending up x_{ij} units of flow from j to i over the arc (i, j).

Sending a unit of flow from i to j along the arc (i, j) increases the objective function by b_{ij}, while sending a unit of flow from j to i on the same arc decreases the flow cost by b_{ij}. Based on these ideas, for a transportation network $G = (V, E)$ and a feasible solution x, the residual network with respect to the given flow x is denoted by $G_x = (V, E_x)$, where E_x is the set of residual edges corresponding to the feasible solution x.

Each arc (i, j) in E is replaced by two arcs (i, j), (j, i): the arc (i, j) has cost b_{ij} and (residual) capacity $r_{ij} = u_{ij} - x_{ij}$, and the arc (j, i) has cost $-b_{ij}$ and (residual) capacity $r_{ji} = x_{ij}$.

The set E_x is constructed from the new edges with a positive residual capacity only.

3.2 Successive Shortest Path Algorithm

The successive shortest path algorithm searches for the maximum flow and optimizes the objective function simultaneously. Instead of searching for the maximum flow as usual, the flow from s to t is sent along the shortest path (with respect to arc costs). The residual network is then updated, another shortest path is found and the flow is augmented again, etc. The algorithm terminates when the residual network contains no path from s to t (the flow is maximal). Since the flow is maximal, it corresponds to a feasible solution of the original minimum cost flow problem. Moreover, this solution will be optimal.

The algorithm performs at most $O(nB)$ augmentations, where B is assigned to an upper bound on the largest supply of any node. Each augmentation strictly decreases the residual capacity of a source arc (which is equal to the supply of the corresponding node) by at least one unit. By using an $O(nm)$ algorithm for finding a shortest path, it is achieved an $O(n^2mB)$ complexity of the successive shortest path algorithm.

As exposed within assumption 3, all edge costs can be made nonnegative by using, for instance, Bellman-Ford's algorithm. Since working with residual costs doesn't change shortest paths, working with the transformed network and using Dijkstra's algorithm allows finding the successive shortest path more efficiently. In order to keep the edge costs nonnegative, node potentials and reduce costs are updated right after the shortest path has been found. For each i in V the potential p_i is equal to the length of the shortest paths from s to i. After having reduced the cost of each arc, all arcs along the shortest path from s to i will have zero cost while the arcs which lie out of any shortest path to any vertex will have a positive cost.

```
**Successive Shortest Path**;
 1. Transform network $G$ by adding source $s$ and sink $t$;
 2. Initial flow $x := 0$;
 3. Use Bellman-Ford's algorithm to establish potentials $p$;
 4. **Reduce Cost** ($p$);
 5. **While**($G_x$ contains a path from $s$ to $t$) **Do**
 6.     Use Dijkstra's algorithm for the shortest path $P$ from $s$ to $t$;
 7.     **Reduce Cost** ($p$);
 8.     Augment current flow $x$ along $P$;
 9.     Update $G_x$;
```

```
**Reduce Cost**($p$);
 1. **For** each $(i, j)$ in $E_x$ **Do**
 2.     $b^p_{ij} := b_{ij} + p_i - p_j$;
 3.     $b^p_{ji} := 0$;
```

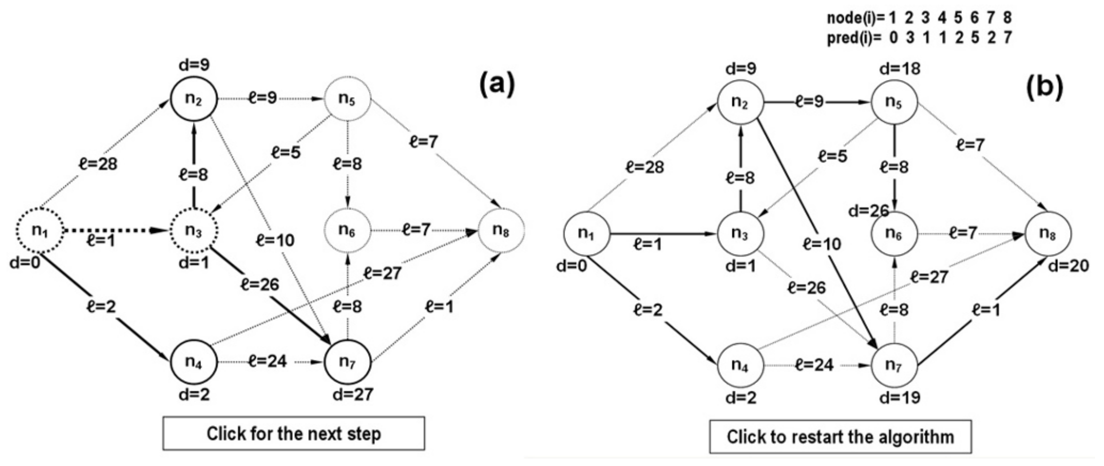
Before starting the cycle in line 5, node potentials are calculated and all costs are obtained to be nonnegative. In line 6, Dijkstra's algorithm is used to establish a shortest path with respect to the reduced costs.

Then the costs are reduced and the flow is augmented along the path. After the augmentation, all costs will remain nonnegative and in the next iteration Dijkstra's algorithm will work correctly. Bellman-Ford's algorithm is used only once to avoid negative costs on edges. It takes $O(nm)$ time. Then $O(nB)$ times is used Dijkstra algorithm, which takes either $O(n^2)$ (simple realization) or $O(m \log n)$ (heap realization for sparse network) time. Summing up results in $O(n^3B)$ estimate working time for simple realization and $O(nmB \log n)$ if using heap.

4 Applications

4.1 Dijkstra's shortest path application

The "Dijkstra's shortest path" application allows selecting different predefined networks and draws them in an applet. For each selected network the application is a step by step construction of the shortest path (with respect to arc costs) from the start node to the rest of nodes in the network. On each step, the candidate arcs are outlined as in Figure 1.



```
node(i)= 1 2 3 4 5 6 7 8
pred(i)= 0 3 1 1 2 5 2 7
```

Figure 1: (a) Selecting the minimum cost arc from the candidate arcs. (b) The shortest path.

Once a minimum path is found, it is marked by changing its colour and the application displays the distance from the start node to the rest of the nodes and the predecessor of the nodes on the shortest path. In the lower side of the applet a label indicates which will be the next action to be performed when clicking on the applet.

4.2 Minimum Cost Flow application

The Successive Shortest Path Algorithm for the Minimum Cost Flow application is written in Java and solves the problem in a step by step mode. The application consists in a set of two applets displayed simultaneously on the screen.

The first one presents the continuously updated residual network after the flow augmentation along the successive shortest paths (Figure 2.a) while the other one presents the total flow in each arc after successive augmentations (Figure 2.b). The application ends when the minimum cost flow is reached and the application displays its value.

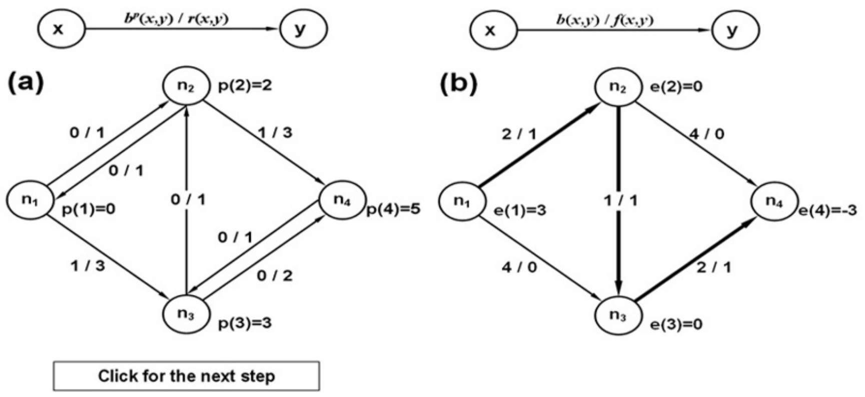


Figure 2: Updated residual network-(a) after flow augmentation-(b) for the "Minimum Cost Flow" application.

In the lower side of the first applet a label indicates which will be the next action to be performed when clicking on the applet. The first applet advances in a step by step mode while the second one, being a runnable application, permanently updates itself according to the changes of the values in the first one.

5 Conclusions

The idea of using applets as a teaching tool, both in face-to-face and online learning, is quite extended. The content of the lecture does not change, but the methods intend to improve students' attitude towards an active learning. The present paper can be extended for solving different network flows problems and used for illustrating their behaviour and operating mechanism. Applications developed in this manner can be successfully used in the e-Learning and distance Learning systems for the benefit of pupils and students. Being a Java based application it also can easily be integrated in web pages.

References

[1] Ciurea, E., Ciupală, L. *ALGORITMI Introducere in algoritmica fluxurilor in retele.*, MATRIX ROM, Bucureşti, 2006.

[2] Sângeorzan, L., Parpalea, M. *Interactive Demonstration of Harmonic Mechanical Oscillations and Elastic Waves Using Java Applets.*, Sixth International Conference Challenges in Higher Education and Research in 21st Century, June 4 - 7.Sozopol, Bulgaria (2008), 411-415.

[3] Sângeorzan, L., Parpalea, M., Nedelcu, A., et al. *Some Aspects in the Modeling of Physics Phenomena using Computer Graphics.*, MACMESE: Proceedings of the 10th Int. Conference on Mathematical and Computational Methods in Sciences and Engineering, Bucharest Romania, 2008, 518-523.