A Visualization Software for the Network Simplex Algorithm

Thanasis Baloukas
Dept. of Applied Informatics
University of Macedonia
thanasis@uom.gr

Konstantinos Paparrizos
Dept. of Applied Informatics
University of Macedonia
paparriz@uom.gr

Abstract

The teaching experiences of the network simplex algorithm to postgraduate students at our department, lead us to develop a visualization software which aims to support the instructor in his effort to explain the algorithm to the students. In addition we believe that the software will help students in their endeavor to understand the aforementioned algorithm which is quite complex, as it involves many concurrent mathematical and visual transformations.

1 Introduction

Algorithm visualization (AV) is a section of software visualization, which Price et al. [2003] define as "Algorithm visualization (or animation) is understood to be the visualization of a high-level description of a piece of software, which is in contrast to code or data visualization (which are collectively a kind of program visualization) where actual implemented code is visualized".

In order to improve the teaching quality, many educators are using AV software in their courses. In computer science education, the main focus of visualization tools available is the visualization of algorithms and data structures.

In this paper we will describe a visualization tool for the Network Simplex algorithm. The latter algorithm solves the well known optimization problem, the Minimum Cost Network Flow Problem (MCNFP). To the best of our knowledge it is the **first time** such a visualization is being made for the specific algorithm.

The proposed software is freely accessible and can be run as Java applet at http://eos.uom.gr/~thanasis/softvis06.htm. Moreover it can be executed locally as standalone Java application, after downloading a single executable file from the link: http://eos.uom.gr/~thanasis/softvis06.jar.

2 Problem and algorithm description

2.1 The Minimum Cost Network Flow Problem (MCNFP).

Let G = (N, A) be some transportation network. Each element in set N is called *node*. The set A consists of ordered pairs (i, j) where i, j ∈ N. The ordered pair (i, j) in set A is represented by an arrow from node i to node j and is called *arc*. The network has n nodes (n = |N|) and m arcs (m = |A|).

Some nodes (i) have a supply b(i)>0 for a commodity (*supply nodes*). Some other nodes have a demand b(i)<0 for a commodity (*demand nodes*) and finally some other nodes have neither a demand for nor a supply (b(i)=0) for a commodity (*transshipment nodes*). Moreover the total supply equals the total demand (*balanced problem*). In addition there is some cost c_{ij} to ship a unit amount along (i, j).

The MCNFP concerns finding the cheapest way to ship prescribed amounts of a commodity from supply nodes to demand nodes satisfying the total demand. The mathematical formulation of the balanced MCNFP is as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$s.t: \sum_{\{k:(i,k)\in A\}} x_{ik} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i),\ i \in N$$

$$x_{ij} \geq 0, \quad (i,j) \in A$$

$$\sum_{i=1}^{n} b(i) = 0$$

x_{ij} is the amount of a commodity that flows inside arc (i, j). Thus x_{ij} is called *flow* (or *decision variable*). Moreover for every node i we associate a variable w_i (*dual variable*) and for every arc (i, j) we associate a variable s_{ij} (*dual slack variable* or *reduced cost*). For s_{ij} we have: $s_{ij} = c_{ij} - w_i + w_j$.

2.2 The Network Simplex algorithm

The Network Simplex method starts with a feasible tree solution T. By feasible tree solution we mean that $x_{ij} = 0$ whenever (i, j) ∉ T. The arcs (i, j) ∈ T are called *basic arcs*. All other arcs (i, j) ∉ T are called *non-basic arcs*. The flows x_{ij} for the non-basic arcs are equal to 0 ($x_{ij} = 0$). The dual slack variables for the basic arcs are equal to 0 ($s_{ij} = 0$).

At each iteration the algorithm chooses a non-basic arc (g, h), such that $s_{gh} < 0$, and adds it to tree T. The latter arc is called *entering* arc. As a result, the tree T ∪ (g, h) has a unique cycle C. From all arcs in C, some have same direction with the entering arc while the remaining arcs have opposite direction. The former arcs are said to belong in C^+, while the latter in C^-.

Next, the algorithm chooses a basic arc (k, l) from C^-, that is called *leaving* arc, removing it from T. The leaving arc is chosen using the Cunningham's anticycling rule.

The removal of leaving arc from tree T splits it into two disjoint subtrees. One of these contains the root; the other, which it is computed by the algorithm and is denoted by T*, is necessary in order to properly update the variables w_i and s_{ij}.

Afterwards the algorithm updates the variables x_{ij}, w_i and s_{ij}, yielding the next tree T′. Each iteration is called *a pivot*.

Some nodes in T* form the so-called *pivot stem*. These nodes facilitate the update of data structures used to store the tree T′.

The algorithm terminates when either there not exist *non-basic arcs* (i, j) such that $s_{ij} < 0$, or when there are no arcs in C^- (that is $C^- = \varnothing$).

For a thorough description of the algorithm the reader is referred to [Ahuja et al. 1993; Chvatal 1983].

SOFTVIS 2006, Brighton, United Kingdom, September 04–05, 2006.
© 2006 ACM 1-59593-464-2/06/0009 $5.00

The project is co-funded by the European Social Fund & National Resources – EPEAEK II – PYTHAGORAS

3 A visualization example

After having created a directed graph the user clicks on menu choice *Solve a Problem* and after that on *Network Simplex*. Subsequently, he clicks the button "STEP FORWARD" once and the start feasible tree is displayed inside the graphical window (Figure 1).

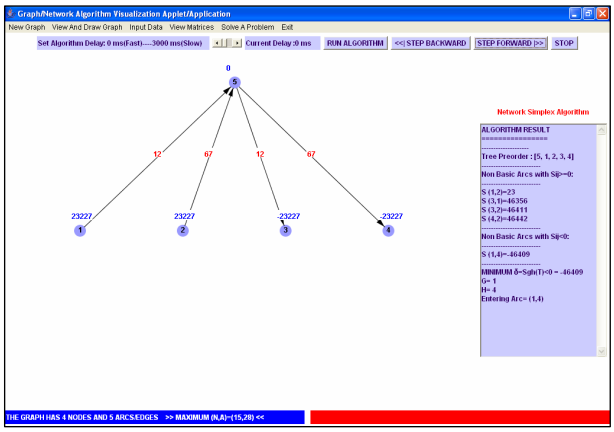

Figure 1: Algorithm visualization at step 1. The start feasible tree is displayed.

In Figures 2, 3 and 4 the visualization progress is shown, after the user has pressed the button "STEP FORWARD" two, three and four times correspondingly.

The *entering arc* is illustrated with the *green dotted curve*, the *leaving arc* with the two *red parallel lines*, the arcs belonging in C^- with the *character "– " in red color*, the arcs belonging in C^+ with the *character "+" in red color*, the nodes in T^* in *red color* and the nodes in *pivot stem* with an *extra blue outer circle.*

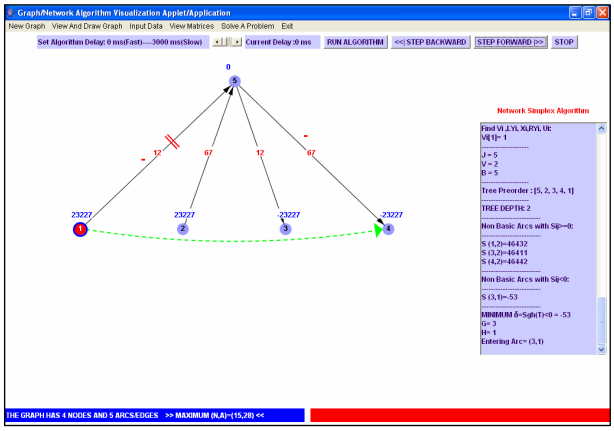

Figure 2: Algorithm visualization at step 2.

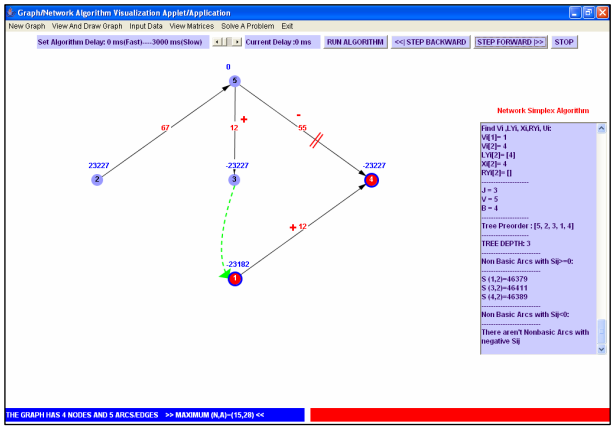

Figure 3: Algorithm visualization at step 3.

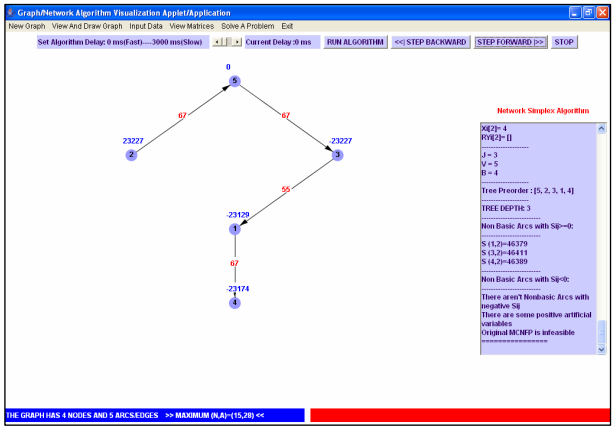

Figure 4: Algorithm visualization at step 4. The solution tree is displayed.

4 Avenues for future research

Several studies that investigate the educational efficacy of visualization have been already carried out. A number of these [Byrne et al. 1999; Stasko et al. 1993] have found that using visualizations to help teach algorithms, had less beneficial effects on students' learning outcomes than previously hoped.

Motivated by these studies we are going to empirically evaluate the learning outcomes of AV using the proposed software.

References

AHUJA, K. R., MAGNANTI, L. T., and ORLIN, B. J. 1993. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, Englewood Cliffs, NJ.

BYRNE, M.D., CATRAMBONE, R., and STASKO, J.T. 1999. Evaluating animations as student aids in learning computer algorithms, *Computers & Education 33*, 4, 253-278.

CHVATAL, V. 1983. *Linear Programming.* W. H. Freeman.

PRICE, B.A., BAECKER, R.M., and SMALL, I.S. 2003. A Principled Taxonomy of Software Visualization, *Journal of Visual Languages and Computing 4,* 3, 211-266.

STASKO, J., BADRE, A., and LEWIS, C. 1993. Do algorithm animations assist learning? An empirical study and analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, Amsterdam, The Netherlands, 61-66.