Yugoslav Journal of Operations Research
15 (2005), Number 1, 147-158

A PARAMETRIC VISUALIZATION SOFTWARE FOR THE ASSIGNMENT PROBLEM

Charalampos PAPAMANTHOU, Konstantinos PAPARRIZOS,
Nikolaos SAMARAS

Department of Applied Informatics
University of Macedonia, Thessaloniki, Greece
it0067@uom.gr, paparriz@uom.gr, samaras@uom.gr

Received: September 2003 / Accepted: September 2004

Abstract: In this paper we present a parametric visualization software used to assist the teaching of the Network Primal Simplex Algorithm for the assignment problem (AP). The assignment problem is a special case of the balanced transportation problem. The main functions of the algorithm and design techniques are also presented. Through this process, we aim to underline the importance and necessity of using such educational methods in order to improve the teaching of Computer Algorithms.

Keywords: Assignment problem, network simplex algorithm, visualization, computer algorithms.

1. INTRODUCTION

The AP is a complex combinatorial optimization problem. It is one of the most classic in the field of Computer Science and Operations Research. A lot of efficient algorithms have been and are constantly being developed for its solution [1], [2], [3], [4], [6], [9], [10], [11], [16], [19], [20] and [23]. The problem owes its name to the classic application of assigning a number of jobs to an equal number of persons at a minimal total cost, given the costs of the assignment of every job to every person. It is evident, that even if every person is capable of employing every job, the solution of the problem must assign only one job to each person. From a first point of view it looks really simple; one must just find all the possible combinations between the jobs and the persons, compare the respective costs, and choose the minimal one. However, taking into account that there are $n!$ possible permutations of n discrete objects, the approach just mentioned is computationally prohibitive, considering that the actual problems can be large. There are plenty of other applications of the assignment problem such as the optimal usage of n

CPUs for the parallel function of n peripheral devices (disks, printers) when we know that the assignment of the peripheral j to the CPU i, $1 \le i \le n$, requires R_{ij} units of some kind of resource (i.e. memory). Furthermore, applications of the AP can be found in computer networks design where we are looking for an optimal way to match network nodes with network concentrators [14].

The main purpose of this paper is to present the most often used algorithm; the Network Primal Simplex Algorithm (NPSA). This algorithm was invented in 1947 by G. B. Dantzig [15] and is the most widespread one due to its satisfactory computational efficiency. NPSA uses the Cunningham cycling pivoting rule [10]. Also, this algorithm was independently developed by Barr et al. [5]. The presentation will be achieved through a parametric visualization method, which can serve educational purposes and help the instructor explain the algorithm to his audience in a more efficient way [22], [26]. Using our software a user can solve any AP of size less or equal to twenty, unlike most other visualization software that solve a specific example by projecting sequential frames. Additionally, we hope that our software will help the students understand the nature of the algorithm and give them the ability to apply it themselves.

The paper is organized as follows. Following this introduction, in Section 2 we briefly present the NPSA for the AP. The algorithm visualization is presented in Section 3. An expansion of the software is presented in Section 4. Finally, Section 5 contains our conclusions.

2. THE ALGORITHM DESCRIPTION

Before explaining and presenting the computer program of the visualization of the NPSA we will refer to the algorithm itself and show how it can be applied with the use of pencil and paper. The AP is a special case of the balanced transportation problem [25]. In the AP we just set $m = n$ and $a_i = b_i = 1$, $i = 1, \ldots, n$. So, we say that we have to solve an AP of size n. The mathematical formulation of the AP with a nxn cost matrix is the following:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.t. } \sum_{i=1}^{n} x_{ij} = 1,\ j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,\ldots,n$$

$$x_{ij} \ge 0,\ 1 \le i, j \le n$$

The variables x_{ij} are called decision variables and can take the value one or zero. This means that we associate any assignment with these variables in a way that $x_{ij} = 1$ if person i is assigned to object j and $x_{ij} = 0$ otherwise.

During the execution of the algorithm, the assignments will be depicted with a rooted tree of special structure. The supply nodes of the tree will be presented with circles and will be called row nodes, whereas the demand nodes will be presented with squares and will be called column nodes. In this way, we can distinguish the supply from

the demand nodes. As the assignment problem comprises a linear programming problem, it is clear that we must find one initial feasible solution or one initial feasible tree. A feasible tree satisfies the basic constraints of the problem. Typically, the initial feasible tree has standard structure but it is theoretically obtained by applying the North-West Corner Method, as it is used to provide a starting feasible solution to the transportation problem [12]. The root of the tree will always be the column node 1. Consequently, the initial feasible tree will be a path from column node 1 to row node n. There will be $2n–1$ arcs connecting all the nodes of the tree. The arcs will always have a row-column direction. Each arc (i, j) will correspond to exactly one decision variable x_{ij} such as $x_{ij} = 0$ if (i, j) is directed from the root to the leaves of the tree (downward arc) and $x_{ij} = 1$ otherwise (upward arc).

Suppose that a starting feasible tree is available. This tree will be referred to, as T. Now we are ready to apply the algorithm in order to solve the AP. The input data will be an nxn matrix C. Let u_i, v_j, $i, j = 1, \ldots, n$ denote the dual variables of the supply and demand nodes respectively. If we set $u_1 = 0$ and $s_{ij} = c_{ij} – u_i – v_j = 0$, $\forall$ $(i, j) \in T$ (the reduced cost of a basic variable is always zero), we can solve a $(2n–1)$x$(2n–1)$ linear system and find the values of the variables u_i, v_j. Thus we produce an nxn matrix s where $\forall$ (i, j) $s_{ij} = c_{ij} – u_i – v_j$. This matrix is the most important data structure for the algorithm to work. The basic steps of the algorithm are the following:

- If $s_{ij} \geq 0$, $\forall$ (i, j) the tree T is optimal and the algorithm terminates, else it finds δ, g, h such as $\delta = s_{gh} = \min\{s_{ij} : s_{ij} < 0\}$ and adds the single arc (g, h) to the tree. The arc (g, h) is therefore the entering arc of the iteration and forms a discrete cycle.
- It discards the arc (k, l) of the cycle. This arc is therefore the leaving arc. It is proved that the entering and the leaving arc will always have one common node. To find out which arc will finally be the leaving one, we can apply the following rule: Let $P[T, g]$ denote the set of the nodes that belong to the single path from node g to the root of the tree. If $h \in P[T, g]$, then the sole upward arc (k, h) is discarded and we have $\varepsilon = x_{kh} = 1$, otherwise the sole downward arc (k, h) is discarded and we have $\varepsilon = x_{kh} = 0$. This rule is totally based on the Cunningham pivoting rule [10].

One of the most important points of the algorithm and the point that reveals the value of the educational software that will be presented is the update of the matrix s and the tree T. One special data structure that the algorithm uses is the tree T^* (Fig. 1). This tree contains the nodes that are “cut off” the initial tree when the leaving arc is discarded.

The update of the matrix s is achieved as follows: For all nodes $b \in T^*$, if b is a row node, we add to (or subtract from) the respective row of the matrix s the quantity $\delta <$ 0, otherwise (b is a column node) we subtract from (or add to) the respective column of the matrix s the quantity $\delta < 0$. To decide the operation (addition/subtraction) that will be performed at either the rows or the columns of the matrix s, we must be sure that after these operations take place, the reduced cost of the entering arc must become zero, as it has become basic. To achieve this, we check node h. If $h \in T^*$ we set $q = –\delta > 0$ else we set $q = \delta < 0$. Therefore, if $b \in T^*$ and s is the reduced costs matrix, we will have $s_{bj} = s_{bj} – q$, $j = 1,\ldots, n$, if and only if b is a row node and $s_{bj} = s_{bj} + q$, $j = 1,\ldots, n$, if b is a column node. This is one point of the algorithm that students find difficult to understand, so we

tried to emphasize it through the visualization software. Finally, the decision variables can be updated by setting $x_{ij} = 1$, if (i, j) is an upward arc, otherwise we set $x_{ij} = 0$.

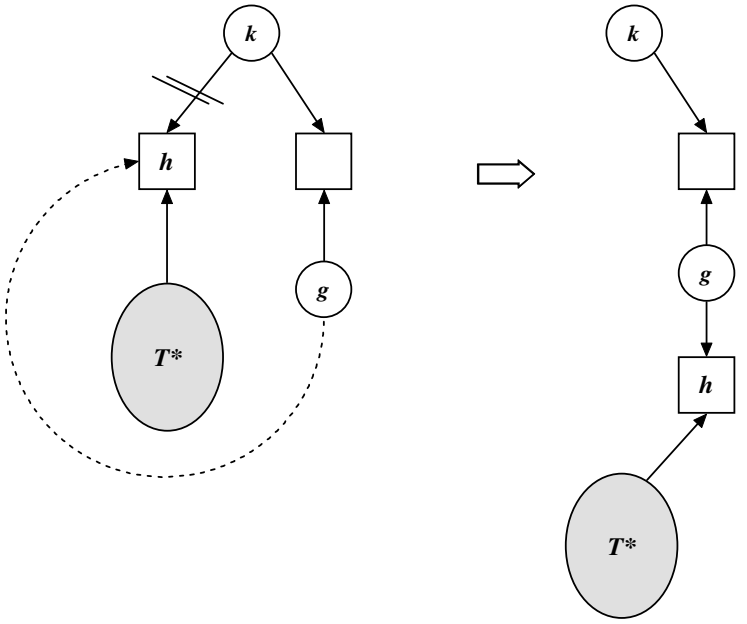


Figure 1: A pivot step of the NPSA. Arc (g, h) is the entering one, arc (k, h) is the leaving one

3. THE ALGORITHM VISUALIZATION

It is true that visualization processes in education comprise a necessary means of transmitting and enhancing knowledge on a scientific topic [11]. Software Visualization is the use of graphics, typography and animation techniques to show the execution of computer programs [24]. Especially in Computer Algorithms courses, it is one of the most widespread teaching methods in many universities all over the world and belongs to a promising field of scientific research.

Algorithm Visualization usually displays the state of the problem at each iteration of the algorithm execution and helps someone to understand the actions need to be made between two successive iterations [8] and [17]. It is obvious that one should not expect to learn the whole function of the algorithm through a visualization process [18]. It is crucial that one should have studied the algorithm [9] before running the visualization. The visualization software will definitely be of great help when the user applies the algorithm himself [26].

Therefore, a computer program has been developed in Java, an object-oriented high level programming language [28]. We chose Java, not only for its object-oriented

nature but also for the fact that it is a web-oriented language. Java programs can also run as internet applications (java applets), giving everyone the chance to run an application no matter the kind of platform is used (Linux, Windows, Mac), provided an internet connection and a web browser with the Java Virtual Machine (JVM) installed.

Our applet consists of four main forms (frames). The first two forms are used for the data input. These two forms can be seen at Fig. 2.The right form is derived from the left one by pressing the button **Data Input** after filling in the field labelled "Size of the assignment problem".

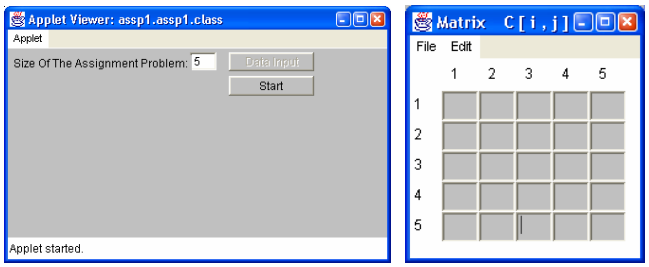


Figure 2: The forms used for the data input

The most important forms are the data control form and the tree display form. While the applet is running the user will interact with those two forms. From now on we will present and explain how our software will solve an AP of size 5, with

$$C = \begin{bmatrix} 0 & 0 & 34 & 74 & 10 \\ 47 & 4 & 0 & -3 & 89 \\ -44 & 35 & 96 & 99 & 87 \\ 20 & 7 & 81 & 0 & 28 \\ 71 & 86 & 99 & 21 & 0 \end{bmatrix}$$

The starting feasible tree as drawn by our software can be seen in Fig. 3.The data control form is referred to the arithmetic data the algorithm uses. This frame appears always beside the tree so that the user can have a general view of the algorithm (Fig. 3).

Data control frame displays the following arithmetic quantities:

- The reduced cost matrix s.
- Variable $\delta < 0$ which determines every time the entering arc.
- The entering arc (g, h).
- The set $P[T, g]$ that contains the column nodes included in the path from node g to the root of the tree.
- The leaving arc (k, l) that is chosen according to the elements of the set $P[T, g]$.
- The decision variable $\varepsilon \in \{0, 1\}$ of the leaving arc (k, l).
- The set containing the row nodes $b \in T^*$.
- The set containing the column nodes $d \in T^*$.
- The number of the iterations of the algorithm.

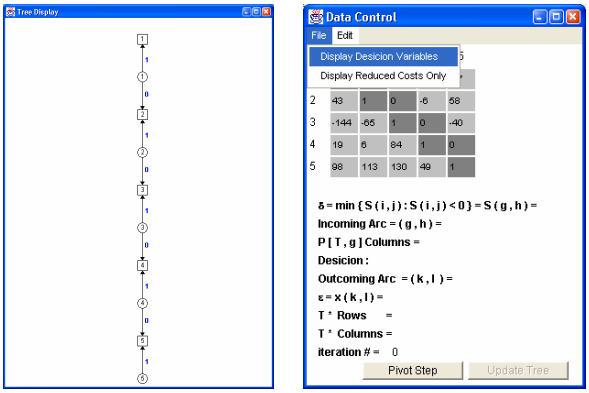
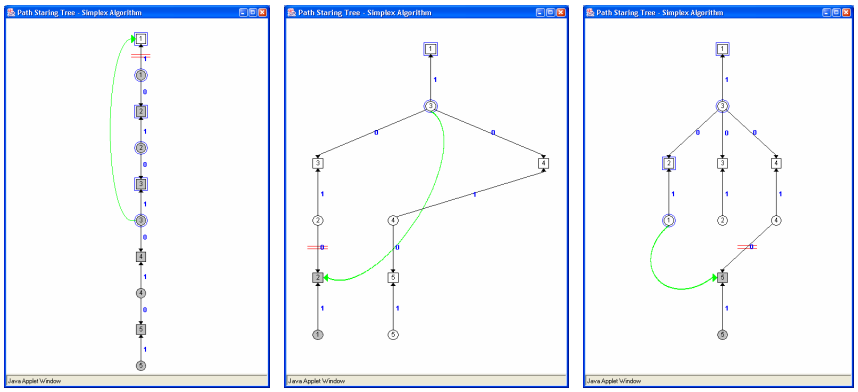
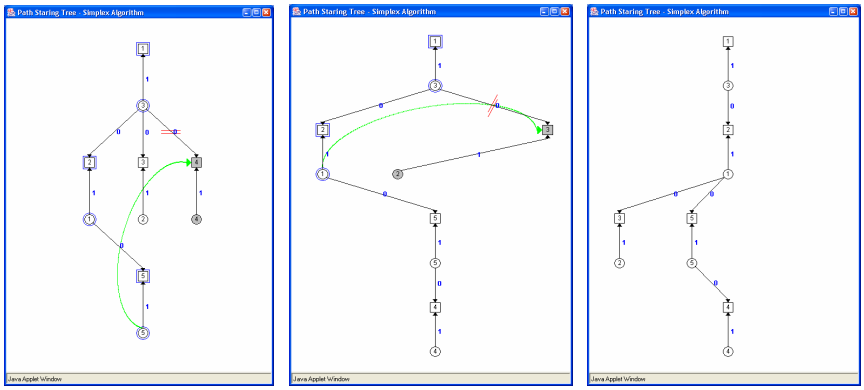


Figure 3: The starting feasible tree and the data control form

If we examine the data control form more carefully, we will see that the background colour of all the elements of the matrix *s* is not the same. If an arc (*a*, *b*) is a basic one, then the respective cell (*a*, *b*) of the matrix *s* appears in dark grey. Furthermore, if an arc (*a, b*) is a basic one, we can view either its reduced cost, which is always zero or the respective value x_{ij}, by using the **File** menu. Moreover, the cell of the entering arc always appears in green whereas the cell of the leaving arc appears in red. In this way, the user can have a general view of the pivot step and combine the information of the data control form with the frame depicting the tree of every iteration. Everything referred above proves how important the data control form is, giving the user the opportunity to understand how the arithmetic data of the algorithm and the current tree structure interact. The data control form can also be seen in Fig. 3. Consecutively, we will present the next iterations of the algorithm. The frames that are produced by the software can be seen below:



Iteration #1 **Iteration #2** **Iteration #3**



Iteration #4 **Iteration #5** **Optimal Solution**

Figure 4: Solving an assignment problem of size $n = 5$

All the frames above always lie beside the data control form of each iteration. As referred above, the data control form is a frame used by our software in order to provide the user with all the necessary information concerning the function of the algorithm. The data control form of the first iteration of our example is that of Fig. 5. By pressing the button labelled "**Pivot Step**", the first iteration begins. As we can see in Fig. 4 (Iteration #1), the entering arc is not just a black straight line but a green curve starting at node *g* and ending at node *h*. In this way we can distinguish it from the basic arcs of the tree.

The interior of the nodes belonging to the set T^* appears in dark grey, while the nodes included in the set $P[T, g]$ are rounded with blue. In this way a perfect visualization is achieved which helps the user understand the algorithm. Additionally, beside the rows and the columns of the matrix *s* that belong to T^* the quantity that must be added is displayed. In this way we can have an overview of the update of the matrix *s* and learn the method in which this update is achieved.

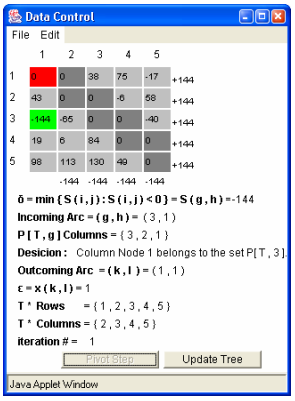


Figure 5: The data control form of the first iteration

After the entering and the leaving variables have been determined, the current tree has to be updated. This can be achieved by pressing the button labelled **Update Tree**. This is a very subtle point of our software. The button labelled **Update Tree** could have easily been omitted. We could only include the button labelled **Pivot Step** which would take over pivoting and updating the tree. However, in this way the discrete states of the tree would not have been clear. The user will definitely have to press less buttons to produce the solution but on the other hand would have missed the "concept" of the algorithm. So, having to deal with educationally oriented software development, we have added this button that will give the user the time to grasp the successive states of the tree. Therefore, if we are at iteration #1 and press the **Update Tree** button, the updated tree would look like that of Fig. 6.

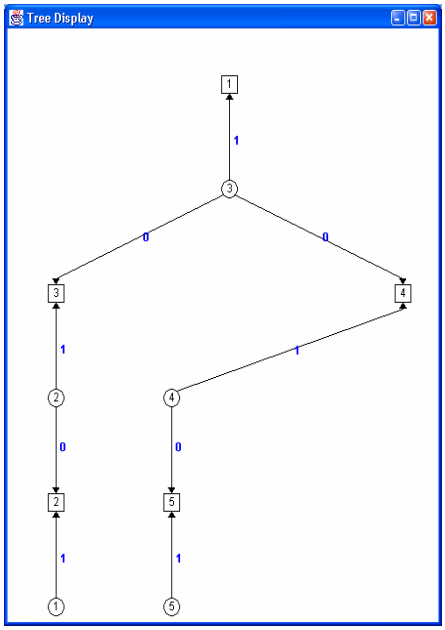


Figure 6: The state of the tree before the beginning of the iteration #2

Note that the algorithm we used to draw the nodes on the frame makes optimal use of the space. The drawing algorithm uses the depth of the tree to determine the levels that the frame is going to be divided. At each level, only nodes having the same depth appear. This is the reason that the distances between nodes of successive depths differ according the size of the assignment problem. To be more specific, the distances between the nodes are generally out of proportion with the size of the assignment problem. Theoretically, we can use this technique to depict the nodes of every assignment problem, no matter what its size is, but we restrict this number to 20 for aesthetic view's sake.

Finally, in order to see the function of the drawing algorithm for more complex assignment problems, we present some frames produced by our software when it solves

problems of greater size. Thus, at Fig. 7 we can see three consecutive frames that are produced when our software solves a problem of size n = 19.

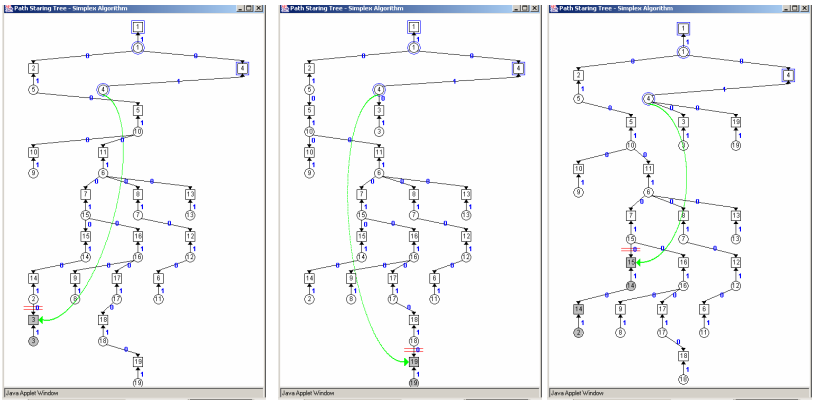


Figure 7: Solving a problem of greater size (n=19)

4. AN EXPANSION OF THE SOFTWARE

Our software has been expanded in order to solve the AP using other algorithms as well. The techniques used are exactly the same with the techniques used for the NPSA. So, the software has been expanded to solve the AP using the algorithms described in [1], [2] and [7]. Additionally, the software can use the algorithm described in [21] applied in the AP. Moreover, the software has been expanded in order to solve the transportation problem as well, using the algorithms described above. Below, we present some indicative frames produced by a modification of our program (Fig. 8 and Fig. 9).

5. CONCLUSION

After presenting a visualization approach of the NPSA for the AP and the algorithm itself as well as an expansion of the software, one can easily find out the value of such software, especially when it comes to deal with graph algorithms. This is happening because graph algorithms are usually difficult to teach, as the instructor always has to depict the successive states of the graph on board and present at the same time the respective arithmetic data that correspond to the current status of the graph. In this way, an analytical explanation of the function of the algorithm is usually omitted. Therefore, if our software is used, all the drawing and the mathematical transformations can be made automatically. Think about the tree T^* or the set $P[T, g]$. Instead of writing on board the whole sets or trying to provide some kind of visualization, the instructor could ask the students to explain themselves why things are the way the software presents them. Additionally, the instructor could challenge the students to propose new methods of visualization that, according their opinion, could lead to a better presentation of the

algorithm [27]. All the java applets are available at: http://eos.uom.gr/~samaras/gr/yujor/yujor.htm. Finally, we must say that Computer Algorithms courses should be escorted by visualization methods so that students could obtain a deeper knowledge in algorithmic issues, improving at the same time their ability to develop and apply algorithms as a means of confronting difficult problems.

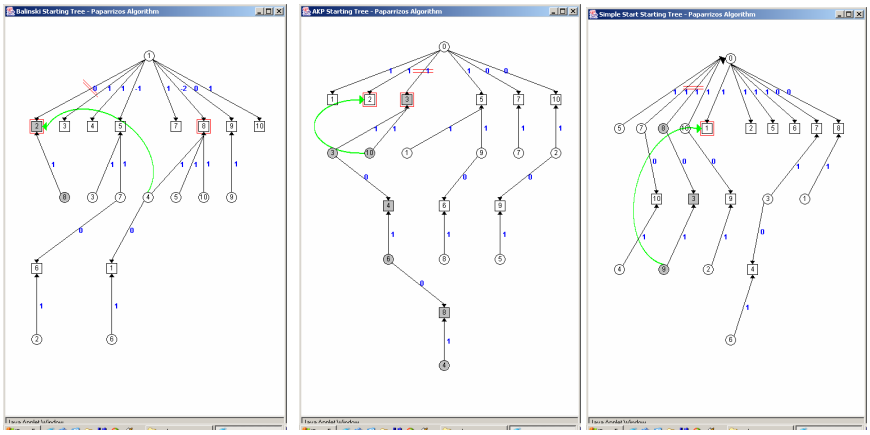


Figure 8: Solving APs of size n=10 using (a) An Infeasible Exterior Point Algorithm, (b) A Dual Forest Algorithm, (c) A Non Improving Simplex Algorithm

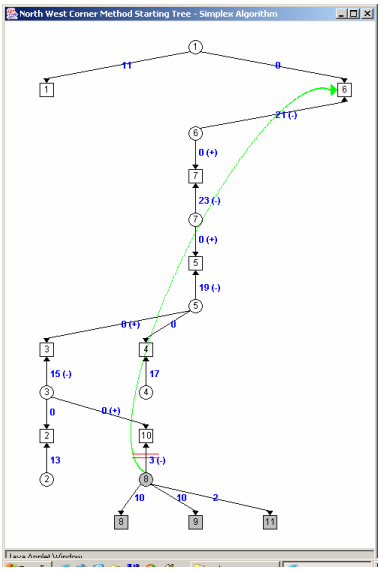


Figure 9: Solving a transportation problem of size 8x11 using the NPSA.

REFERENCES

[1] Achatz, H., Kleinschmidt, P., and Paparrizos, K., "A dual forest algorithm for the assignment problem", *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 4 (1991) 1-10.

[2] Achatz, H., Paparrizos, K., Samaras, N., and Tsiplidis, K., "A forest exterior point algorithm for the assignment problems", in: P.M. Pardalos, A. Migdalas and A. Buckard (eds.), *Combinatorial and Global Optimization*, Word Scientific Publishing Co., 2002, 1-10.

[3] Akgul, M., "A genuinely polynomial primal simplex algorithm for the assignment problem", Working Paper IEOR 87-07, Bilikent University, Ankara, Turkey, 1987.

[4] Balinski, M.L., and Gomory, R.E., "A primal method for the assignment and transportation problems", *Management Sciences*, 10 (1964) 578-598.

[5] Barr, R.S., Glover, F., and Klingman, D., "The alternating basis algorithm for assignment problems", *Mathematical Programming*, 13 (1977) 1-13.

[6] Bertsekas, D., "A new algorithm for the assignment problem", *Mathematical Programming*, 21 (1981) 152-171.

[7] Brown, M.H., and Hershberger, J., "Fundamental techniques for algorithm animation displays", in: Stasko, T.J., Dominique, J., Brown, H.M., and Price, A.B., (eds.), *Software Visualization: Programming as a Multimedia Experience*, MIT Press, 1998, 137-143.

[8] Byrne, D.M., Catrambone, R., and Stasko, T.J., "Do algorithm animations aid learning?", *7th Annual Winter Text Conference*, Jackson Hole, January, 1996.

[9] Cormen, T.H., Leiserson, C.E., and Rivest, R.L., *Introduction to Algorithms*, *2nd edition,* MIT Press, 2001.

[10] Cunningham, W.H., "A network simplex method", *Mathematical Programming*, 11, (1976) 105-116.

[11] Cunningham, W.H., and Marsh, B.A., "A primal algorithm for optimum matching", *Mathematical Programming Studies*, 8 (1978) 50-72.

[12] Dantzig, B.G., "Application of the simplex method to a transportation problem", in: Koopmans, T.C. (ed.), *Activity Analysis and Production and Allocation*, Willey, N.Y., 1951, 359-373.

[13] Dantzig, B.G., *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.

[14] Dosios, K., Paparrizos, K., Papatzikos, N., and Sifaleras, A., "LinPro, an educational informational system for linear programming", To be published in Proceedings of 15th Pan-Hellenic Conference of Greek Operations Research Society, Tripoli, 2002.

[15] http://ccwf.cc.utexas.edu/~sakshi/ Univ. of Texas , Austin.

[16] Hundhausen, D.C., Douglas, A.S., and Stasko, T.J., "A meta-study of algorithm visualization effectiveness", *Journal of Visual Languages and Computing*, 13(3) (2002) 259-290.

[17] Hung, M.S., and Rom, W.D., "Solving the assignment problem by relaxation", *Operations Research*, 28 (1980) 969-982.

[18] Jeffery, L.C., *Program Monitoring and Visualization*, Springer-Verlag, New York, 1999.

[19] Jensen, A. P., and Barnes, J. W., *Network Flow Programming*, Krieger Pub. Co., 1987.

[20] Kehoe, C., Stasko, T.J., and Taylor, A., "Rethinking the evaluation of algorithm animations as learning aids: an observational study", *International Journal of Human-Computer Studies*, 54 (2001) 265-284.

[21] Khuri, S., and Holzapfel, K., "EVEGA: An educational visualization environment for graph algorithms", *ACM SIGCSE Bulletin*, 33(3) (2001) 101-104.

[22] King, K. N., *Java Programming from the Beginning*, W.W. Norton & Co., New York, 2000.

[23] Kuhn, H. W. "The Hungarian method for the assignment problem", *Naval Research Logistics Quarterly*, 2 (1955) 83-97.

[24] Minieka, E., and Evans, R.J., *Optimization Algorithms for Networks and Graphs, 2nd edition*, Marcel Dekker, New York, 1992.

[25] Paparrizos, K., "A non improving simplex algorithm for transportation problems", *Operations Research*, 30 (1) (1996) 1-15.
[26] Paparrizos, K., "An infeasible (exterior point) simplex algorithm for assignment problems", *Mathematical Programming*, 51 (1991) 45-54.
[27] Rößling, G., Naps, T.L., "A tested for pedagogical requirements in algorithm visualizations", *7th annual Conference on Innovation and Technology in Computer Science Education ITICSE 2002*, Aarhus, Denmark, 2002, 96-100.
[28] Warland, J., *Communication Networks: A First Course*, McGraw-Hill, 1991.