CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

Research Report CCS 218

IMPROVED LABELING OF L. P. BASES IN NETWORKS

by

Fred Glover*
Darwin Klingman

August 1974
(Revised October 1975)

*Professor of Management Science, University of Colorado, Boulder, Colorado

This research was partly supported by the Navy Personnel Research and Development Laboratory Contract N00126-74-C-2275 with the Center for Cybernetic Studies, The University of Texas, Austin, Texas and the Alexander von Humboldt Stiftung in Bonn, Germany.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 512
The University of Texas
Austin, Texas 78712
(512) 471-1821

ABSTRACT

New labeling techniques are provided for accelerating the basis exchange step of specialized linear programming methods for network problems. These techniques substantially reduce the amount of computation involved in updating operations.

1.0 INTRODUCTION

In solving minimum cost flow network problems by specialized linear programming methods, an important question is: how can one update the spanning tree basis with the least amount of effort? A partial answer to this question is provided by special list structure techniques such as the API method [1] and the more recent ATI method [4], which have contributed dramatically to improving the efficiency of network algorithms (see, e.g., [2, 3, 6, 7, 8]). This paper addresses the issue of which supplemental techniques can be used to enable these list structures (and particularly the ATI method) to be implemented with greater efficiency.

As shown in [1], the major updating calculations of a basis exchange step can be restricted to just one of the two subtrees created by dropping the outgoing arc. Consequently, a natural goal is to identify the smaller of these two subtrees by means of a list t(x) that names the number of nodes in the subtree "headed by node x." A clever and rather intricate procedure for doing this was proposed by Srinivasan and Thompson [9]. However, unfortunately, this procedure requires sorting the nodes of the subtree by their distances from the root, and then further entails a full subtree update of both the distance values and the t(x) values at each basis exchange step. Because of the substantial amount of work required to update the t(x) list, its use has never been found practicable.

The purpose of this paper is to propose a new type of relabeling scheme that succeeds in updating t(x) without sorting. In fact, this scheme requires even less work than to update the distance values of [9]. The relabeling is based on "absorbing" t(x) into the updating calculations of the ATI method. Moreover, these calculations are carried out simultaneously with the procedures [4] for updating other changes introduced by the basis exchange step.

To achieve the integration of the ATI calculations and the update of $t(x)$, we introduce an index function $f(x)$ that names the last node in the subtree rooted at x. We show that $f(x)$ additionally makes it possible to streamline the ATI calculations. Finally, as a bonus, we show that $t(x)$ can accommodate all of the relevant functions filled by the distance values, and hence can replace these values. The net gains of all these advantages produce a substantially improved procedure for implementing the basis exchange operations.

2.0 NOTATION AND DEFINITIONS

We shall follow the notational conventions of [1], identifying the components of the basis exchange step as follows:

(p,q) = the arc leaving the basis, where p is currently the predecessor of q (via the "predecessor indexing" used with the ATI method).

(u,v) = the arc entering the basis, where u is the node whose "backward path" (consisting of all its ancestors under the predecessor indexing) contains arc (p,q).

T = the basis tree

$T(x)$ = the subtree of T that is rooted at node x (hence the subtree that includes x and all its descendants under the predecessor indexing)

$S(x)$ = the "thread successor" of x as defined in the ATI method [4]. (In particular $T(x) = \{s^0(x), s^1(x), \ldots, s^r(x)\}$ for some $r \geq 0$, where $s^0(x) = x$ and $s^{k+1}(x) = s(s^k(x))$.)

$f(x)$ = the "last node" ("$s^r(x)$") in $T(x)$

$t(x)$ = the number of nodes in $T(x)$

The basis exchange step may be visualized as consising of two components:

(1) Dropping arc (p,q) to create two independent subtrees: $T(q)$ and $T - T(q)$ (where the latter is the subtree of T that excludes $T(q)$ and all its nodes, and hence which excludes the "connecting arc" (p,q));

(2) Adding arc (u,v) to create a single new basis tree.

The subtrees $T(q)$ and $T - T(q)$ can be viewed as any two node-disjoint trees which are to be joined by an arc to create a new basis tree. We shall therefore first develop updating operations to make $T(q)$ and $T - T(q)$ into label "independent" trees in preparation for selecting which is to be the new "upper tree" (called T_1) rooted at x_1 and which is to be the new "lower" trees (called T_2) rooted at x_2. We may then assume that the root of T becomes the root of the new basis tree. Additionally, we let (y_1, y_2) be the arc that joins T_1, and T_2 where y_1 is a node of T_1 and y_2 is a node of T_2. Next we develop operations to re-root T_2 at y_2 in preparation for attaching T_2 to T_1 via arc

(y_1, y_2) to create the new basis tree. (There is no requirement that y_1 be distinct from x_1 or that y_2 be distinct from x_2.)

3.0 UPDATING OPERATIONS

Using the preceding definitions, we show how to find updated values s*(x), t*(x), f*(x) for s(x), t(x) and f(x) as follows.

I. Update s(x), t(x), f(x) to make the subtrees T(q) and T - T(q) independent

I.1. Update for T - T(q):

For s*(x): Identify the node y in T - T(q) such that s(y) = q. Then set s*(y) = s(f(q)). No other s(x) values are changed. (Note: while this step is obviously facilitated by the direct accessing f(q), the identification of y is further speeded by utilizing the f(x) functions as follows: First, let y^1 = p. Second, if $s(y^1) = q$, then $y = y^1$ and the process stops. Otherwise, let $y^1 = f(s(y^1))$ and repeat the second step.)

For t*(x): Set t*(x) = t(x) - t(q) for those nodes x on the "backward path" from p to the root of T. Due to cancellation effects of subsequent calculations, this step can be restricted to the partial backward path from p to the "intersection" node z that is the unique node of the basis loop that lies on the backward paths from both u and v, excluding node z itself from consideration. (Thus, possibly there may be no updating in this step.)

For $f^*(x)$: Set $f^*(x) = y$ for those nodes x on the predecessor from p to the root such that f(x) = f(q). (If f(p) ≠ f(q) no updating is done.)

I.2. Update for T(q):

For s*(x): Set s*(f(q)) = q.

No other updating of any s(x), t(x) or f(x) is required for T(q).

II. Decide which of T(q) and T - T(q) is to be T_1 and which is to be T_2.

If t(q) exceeds half the number of nodes in the network let T(q) be T_1, and let T - T(q) be T_2 (hence x_1 = q and x_2 = the root of T). Otherwise, let T - T(q) be T_1, and let T(q) be T_2.

The principal reason for re-rooting is to minimize the computational effort when the updating of the dual variables is integrated with the other updating operations. However, it is possible to separate these updating operations, carrying out the node potential update on the smaller subtree, and always carrying out the other updating operations on subtree T(q), thereby leaving the root unchanged. This latter case follows directly from the subsequent development.

III. Make y_2 the new root of T_2, reversing the predecessor orientation of the path from y_2 to x_2.

For s*(x): This portion of the updating of s(x) is carried out exactly as specified in [4].

For t*(x): Let $t^*(y_2) = t(x_2)$. (But if $T(q) = T_1$, and the restricted update of t(x) was carried out in step I.1 - which is computationally preferable - set $t^*(y_2) = t(x_2) - t(q)$.) Then for each x on the path from x_2 to y_2, excluding $x = y_2$ set $t^*(x) = t^*(y_2) - t(\bar{x})$, where $\bar{x}$ is the successor of x via the predecessor indexing before reversing the orientation of the path. (It is important to note here that t(x) and t*(x) must be kept distinct from each other; i.e., it is not legitimate to replace $t(\bar{x})$ by $t^*(\bar{x})$ before computing $t^*(x) = t^*(y_2) - t(\bar{x})$.)

For f*(x): The node of T_2 that is examined last in the process of updating the node potentials in T_2 via the ATI method is designated to be $f^*(x_2)$. In particular, if $x_2 = y_2$, or if $f(x_2) \neq f(\bar{x}_2)$, where $\bar{x}_2$ is the successor of x_2 before reversing the orientation of the path, then $f^*(x_2) = f(x_2)$ (i.e., no change occurs in $f(x_2)$.) Otherwise, $f^*(x_2) = y$ for the node y such that $s(y) = \bar{x}_2$. (This y may be found as in I.1, but it will be identified automatically at the conclusion of updating node potentials.) Thereupon, set $f^*(x) = f^*(x_2)$ for all x on the path from x_2 to y_2. (No other changes are made.)

IV. Attach T_2 to T_1 by adding arc (y_1, y_2) to create the new basis tree (where T_2 is now rooted at y_2 as a result of step III).

For s*(x): Set $s^*(f^*(y_2)) = s(y_1)$ and $s^*(y_1) = y_2$.

For $t^*(x)$: Set $t^*(x) = t(x) + t^*(y_2)$ for all x on the path from y_1 to x_1. (But if $T(q) = T_2$, and the restricted update of t(x) was applied in step I.1, then the current step should be restricted to those x on the path from y_1 to z, excluding z itself, for the "intersection" node z as identified in I.1.)

For $f^*(x)$: If $f(y_1) \neq y_1$, set $f^*(x) = f(y_2)$ for those nodes x on the predecessor from y_1 to x_2 such that $f(x) = f(y_1)$. (No changes are made if $f(y_1) = y_1$.)

The proposed procedure for updating t(x) clearly requires less effort than updating the distance function of [9], which minimally involves an addition for every node of the subtree T(q), and in the case of $T(q) = T_1$, requires an addition for every node of T. The fact that t(x) can replace the distance function is a direct consequence of the observation that if $t(x) \leq t(x^1)$, then x cannot be a descendant of x^1. Thus, t(x) can be used in essentially the same manner as the distance function to facilitate operations such as involved in identifying the loop created by adding arc (u,v) to the basis tree.

4.0 INITIALIZATION

It is left to characterize the procedure for establishing the initial values of t(x) and f(x). This occurs simultaneously with the initial determination of the s(x) values as follows.

Consider the step in which $s^{k+1}(x_o) = s(s^k(x_o))$ is identified ($K \geq 0$). If $s^k(x_o)$ is the predecessor of $s^{k+1}(x_o)$ (via the predecessor indexing), do nothing. Otherwise, for all nodes $s^i(x_o)$ on the backward path from $s^k(x_o)$ to the predecessor of $s^{k+1}(x_o)$, excluding the predecessor of $s^{k+1}(x_o)$ itself, set $t(s^i(x_o)) = k+1 - i$, and set $f(s^i(x_o)) = s^k(x_o)$.

When the last node $s^{n-1}(x_o)$ of the network is determined (where n = the total number of nodes in the network), set $t(s^i(x_o)) = n - i$ and set $f(s^i(x_o)) = s^{n-1}(x_o)$ for all $s^i(x_o)$ on the backward path from $s^{n-1}(x_o)$ to x_o.

To easily keep track of the index i for each node $s^i(x_o)$ that is to be considered on a given step, it is convenient to keep a list that consists precisely of the indexes i of the nodes $s^i(x_o)$ to x_o. Specifically, to begin the list contains the single index 0 (for $s^o(x_o)$). When $s^{k+1}(x_o)$ is created, the number k+1 is added to the end of the list. When a backward path from $s^k(x_o)$ is traced, consisting of r nodes (say) $s^i(x_o)$ whose values $t(s^i(x_o))$ and $f(s_i(x_o))$ are to be set, the indexes of these r nodes will be exactly the corresponding last r numbers on the list. By removing these numbers from the list just before adding the number k+1, the desired structure of the list is maintained.

5.0 CONCLUDING REMARKS

The computational advances in network algorithms afforded by the techniques to [1, 4, 5, 9],and the evident improvements in these techniques provided by the forgoing procedures, argues strongly in favor of the implementation value of these procedures. Moreover, we conjecture that coupling the ATI method with the use of t(x) and f(x) in the manner described, not only dominates all previous updating procedures, but closely approaches the "asymptotic limit" of effective trade-off between computational efficiency and memory requirement for a broad class of networks. Results of a computational study of these new procedures, imbedded in various alternative implementations of specialized simplex codes, will be reported in a subsequent paper.

References

1. Glover, F., D. Karney, and D. Klingman, "The Augmented Precedessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices In Distribution Problems." Transportation Science, 6, 171-180, (1972).

2. Glover, F., D. Karney, D. Klingman, and A. Napier, "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems." Management Science, 20, 5, 793-813, (1974).

3. Glover, F., D. Karney, and D. Klingman, "Implementation and Computational Study on Start Procedures and Basis Change Criteria for a Primal Network Code." Networks, 20, 191-212, (1974).

4. Glover, F., D. Klingman, and J. Stutz, "Augmented Threaded Index Method," INFOR, 12, 3, 293-298, (1974).

5. Johnson, Ellis, "Networks and Basic Solutions." Operations Research, 14, 4, 619-623, (1966).

6. Karney, D. and D. Klingman, "Implementation and Computational Study on an In-Core Out-of-Core Primal Network Code." CS 158, Center for Cybernetic Studies, University of Texas, Austin. To appear in Operations Research.

7. Klingman, D., A. Napier, and J. Stutz, "NETGEN - A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation and Minimum Cost Flow Network Problems." Management Science, 20, 5, 813-819, (1974).

8. Srinivasan, V. and G.L. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm", JACM, 20, 194-213, (1973).

9. Srinivasan, V. and G.L. Thompson, "Accelerated Algorithms for Labeling and Relabeling of Trees with Application for Distribution Problems," Journal of the Association for Computing Machinery, 19, 4, 712-726, (1972).

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
The University of Texas Center for Cybernetic Studies	Unclassified 2b. GROUP

3. REPORT TITLE

(6) Improved Labeling of L. P. Bases in Networks,

4. DESCRIPTIVE NOTE (*Type of report and inclusive dates*)

(9) Research Rept.

5. AUTHOR(S) (*First name, middle initial, last name*)

(10) Fred Glover
Darwin Klingman

(11) Oct 75

(12) 13 p.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August 1974(Revised October 1975)	8	9

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
(15) N00126-74-C-2275 NEW b. PROJECT NO (16) NR-047-021 c.	Center for Cybernetic Studies Research Report CS 218 9b. OTHER REPORT NO(S) (*Any other numbers that may be assigned this report*) (14) CCS-218-Rev d.

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Personnel Research & Development Center, San Diego, California

13. ABSTRACT

New labeling techniques are provided for accelerating the basis exchange step of specialized linear programming methods for network problems. These techniques substantially reduce the amount of computation involved in updating operations

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

406197

Unclassified

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Network						
Spanning Trees						
Linear Programming						
Link Lists						
Maximum Flaw						
Transportation						