A Pivot And Probe Algorithm(PARA) for Network Optimization

Moonsig Kang* · Young-Moon Kim*

ABSTRACT

This paper discusses a new algorithm, the PAPANET (Pivot And Probe Algorithm for NETwork optimization), for solving linear, capacitated linear network flow problem (NPs). PAPANET is a variation and specialization of the Pivot And Probe Algorithm (PAPA) developed by Sethi and Thompson, published in 1983-1984. PAPANET first solves an initial relaxed NP (RNP) with all the nodes from the original problem and a limited set of arcs (possibly all the artificial and slack arcs). From the arcs not considered in the current relaxation, we PROBE to identify candidate arcs that violate the current solution's dual constraints maximally. Candidate arcs are added to the RNP, and this new RNP is solved to optimality. This candidate pricing procedure and pivoting continue until all the candidate arcs price unfavorably and all of the dual constraints corresponding to the other, so-called noncandidate arcs, are satisfied. The implementation of PAPANET requires significantly fewer arcs and less solution CPU time than is required by the standard network simplex method implementation upon which it is based. Computational tests on randomly generated NPs indicate that our PAPANET implementation requires up to 40-50% fewer pivots and 30-40% less solution CPU time than is required by the comparable standard network simplex implementation from which it is derived.

I. Introduction

Because of its important applications, i.e., communication systems, inventory systems, traffic systems, and many other areas (e.g., see Aronson, 1989), a great deal of research on network theory and algorithmic development has been performed over the years (Aderohunmu and Aronson, 1993; Aronson and Chen, 1986; Balachandran and Thompson, 1975a, 1975b, 1975c, 1975d; Barr, Glover and Klingman,

* 계명대학교 경영대학 경영정보학과

1970; Bazaraa, Jarvis and Sherali, 1990; Fullkerson, 1961 ; Karney and Klingman, 1976; Kennington and Helgason, 1980; Srinivasan and Thompson, 1973, 1976, 1977). Among these, a few (e.g., Aderohunmu and Aronson, 1993; Aronson and Chen, 1986) have attempted to reduce the active problem size as a means of increasing algorithmic efficiency. For linear programming problems (LPs), see Karwan et al. (1983). In optimizing network flow problems, many arcs might never enter the basis, remaining nonbasic at zero flow throughout the entire optimization process. Therefore, identifying and removing such arcs from the problem would reduce problem size without affecting attainment of an optimum, and thus be an effective way to improve computational efficiency.

The Pivot And Probe Algorithm(PAPA) for solving Linear Programming(LP) problems was introduced by Sethi and Thompson (1983, 1984) and Sethi(1983). Later, Thompson and Sethi (1986) developed a specialized implementation of PAPA to solve constrained generalized transportation problems, and Sethi, Thompson and Hung (1990) applied PAPA to the LP dual, which may be specialized for network flow problems.

The basic ideas of PAPA is to reduce the active problem size by retaining only small number of constraints or dual variables that may potentially be included in an optimal solution to an LP. Sethi and Thompson(1983) defined a noncandidate (primal) constraint/(dual) variable as one that is never used during the entire optimization process of an LP. Similarly, a candidate (primal) constraint/(dual) variable was defined as one that is actually used at least once while optimizing. Keeping only candidate constraints/variables reduces the problem size, and reduces pricing and pivoting effort. Recently, Aronson and Kang(1993) introduced the general idea of applying the Pivot and Probe Algorithm to pure Network flow Problems (NPs). Here, this paper presents a more complete, detailed algorithmic description of the method as well as preliminary computational results.

We assume that the reader has a grasp of the definitions and methods of linear programming (e.g., see Bazaraa, Jarvis and Sherali, 1990; Simonnard, 1966) and network optimization (e.g., see Bazaraa, Jarvis and Sherali, 1990; Bertsekas, 1991; Evans and Minieka, 1992; Glover, Klingman and Philips, 1992; Kennington and Helgason, 1980; Murty, 1976).

The paper is organized as follows: A brief description of Pivot and Probe Algorithm for linear programming problems is presented in Section 2. In Section 3, this paper illustrates the PAPANET, a Pivot and Probe Algorithm for NETwork optimization. Preliminary computational results are discussed in Section 4. Section 5 contains our conclusions and potential further directions for the research.

II. The Pivot and Probe Algorithm for Linear Programming

2.1 Algorithm Description

Consider a linear program defined as follows:

$$\begin{aligned} \max\ z &= \mathbf{cx} \\ \text{s.t. } \mathbf{Ax} &= \mathbf{b} \\ \mathbf{x} &\geq 0 \end{aligned} \qquad (1)$$

where **A** is an m * n matrix , **b** is an m vector, and **c** and **x** are n vectors. Inequalities may be converted to equalities by including slack or surplus variables. PAPA first solves a relaxed linear program(RLP) that consists of initial candidate constraints. An initial candidate constraint is one for which, in the initial solution, there is a minimum ratio found for one of its row's elements and the variable, for which the minimum ratio occurs, prices favorably by a standard simplex method. To enforce finiteness of each RLP, a regularization constraint, $\mathbf{ex} \leq M$, where M is a large number and $\mathbf{e}=(1, 1, \ldots..1)$, is added to the RLP. Once an optimal solution to the RLP is found, the PAPA probes to find a set of most violated constraints from the noncandidate constraints (not in the RLP). The Probe step identifies the piercing points of violated constraints, if any, on the line segment between any feasible point of the original LP, initially $\mathbf{x}=0$, and the current RLP optimum which is usually infeasible to the original, primal problem. Noncandidate constraints are said to be violated if they are not satisfied by the current solution, i.e., $a_i\mathbf{x}^D > b_i$, where $\mathbf{x}^D$ is the current solution to RLP. Once the piercing points are identified, a set of constraints, called the most violated constraints, containing the piercing points closest to the feasible point, are added to the RLP, which may then be solved to optimality using dual simplex pivots. The most piercing point found while probing is a feasible point to the original LP and may be utilized by later probes. This procedure continues until the probe cannot find any violated noncandidate constraint.

2.2 PAPA Example Problem

Consider the LP maximization problem with 8 constraints and 2 variables shown graphically in Figure 1. PAPA starts with initial RLP consisting of only constraints 1 and 2 and a standard, primal simplex method obtains point A as its optimum. Probing from point A to the origin, **0**, identifies point B as the most piercing point and constraint 3 as the most violated constraint (note that point B is a feasible point). Once constraint 3 is added to the RLP, point C is found to be an optimum to the RLP. From point C, probing to the origin identifies constraint 4 as its most violated constraint, and probing to the previous most piercing point, B, identifies constraint 5. After adding constraints 4 and 5 to RLP, point E is found to be an optimum.

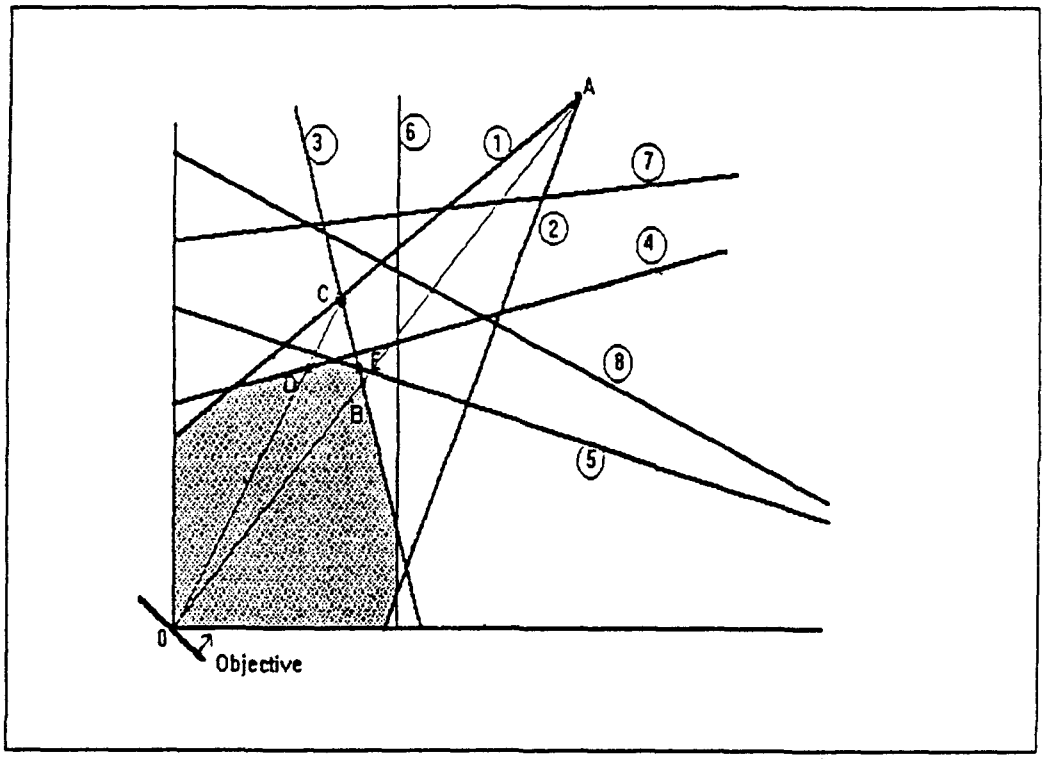

Figure 1. Example Problem

Point E is the optimal solution to the original LP because: 1) it is feasible point; and 2) the feasible region is convex. Thus, probing from point E to any other feasible point cannot pierce any violated constraint. Therefore, all noncandidate const- raints are satisfied by point E.

III. The Pivot and Probe Algorithm for Network Optimization

3.1 Algorithm Development

First we note that it is possible to apply the PAPA to the dual of an Network problem (NP), and in doing so, one may develop candidate lists of arcs (columns) to be considered in the primal problem. Then, a primal network simplex approach may be used to reoptimize the Relaxed NP (RNP). The capacitated, minimum cost network flow problem (NP) may be expressed as:

$$\text{Min} \sum C_{ij}X_{ij}, \ (i, j) \in \mathbf{A},$$

$$\text{s.t.,} \quad \sum_j x_{ij} - \sum_j x_{ij} = r_i, \quad (i, j), (j, i) \in \mathbf{A},$$

$$i \in \mathbf{N}, \qquad (2)$$

$$\mathbf{0} \leq x_{ij} \leq u_{ij} \quad (i, j) \in \mathbf{A},$$

where **A** denotes the set of arcs, **N** denotes the set of nodes, r_i denotes the

requirement of node i (positive for a supply node, negative for a demand node, and zero for a transshipment node), c_{ij} is the unit cost of arc (i, j), from node i to node j, and u_{ij} is the capacity of arc (i, j). We assume that lower bounds of all arcs are zero (otherwise, a simple transformation is required).

PAPANET first forms an initial relaxed network problem (RNP) with the entire node set and a set of few arcs, possibly all artificial and slack arcs. We omit the regularization constraint ($\mathbf{ex} \leq M$), because a capacitated network flow problem cannot be unbounded. In any iteration, let **I** be the index set of arcs that form the current RNP; **R** be the index set of arcs not considered; **w** be a dual feasible point; and **u** be the current optimal dual solution of the RNP. Let **H** denote the index set of the violated variables:

$$\mathbf{H} = \{(i, j) \mid u_i - u_j > c_{ij} \}, (i, j) \in \mathbf{R} \text{ and } i, j \in \mathbf{N} \quad (3)$$

and let **p** be any point between **w** and **u**, which may be written as

$$\mathbf{p} = (1 - k)\mathbf{w} + k\mathbf{u}, \; k \in [0, 1] \quad (4)$$

A probe is the line segment between **w** and **u**, i.e., the set of all such vectors **p** between a dual feasible point and the current RNP dual solution. Then the piercing point of the line segment (probe) and hyperplane h defined by dual constraint $h = (i, j) \in \mathbf{H}$ is determined by

$$k_h = (w_i - w_j - c_{ij}) / (w_i - w_j - u_i + u_j), \quad (i, j) \in \mathbf{H}. \quad (5)$$

A lower value of k_h indicates that the hyperplane is closer to the feasible point **w**. In the PROBE step, we identify the closest hyperplane to **w** and such a hyperplane is called the most violated dual constraint. Formally, a dual constraint $h^* \in \mathbf{H}$ said to be most violated if

$$k_{h^*} = \min \{ k_h \mid h \in \mathbf{H} \}. \quad (6)$$

If $\mathbf{w} = \mathbf{0}$, indicating that we probe to the origin, then (5) simplifies to

$$k_h = c_{ij} / (u_i - u_j), \quad (i, j) \in \mathbf{H}. \quad (7)$$

Then, the evaluation of k_{h^*} in (6) is equivalent to the standard simplex pricing formula to identify a most favorable, nonbasis arc to enter the basis of NP (2), that is, find $h^* = (i, j)^*$ such that

$$\max \{u_i - u_i - c_{ij}\}, \quad (8)$$

where (i, j) is nonbasic with zero flow. The piercing point, $\mathbf{p}^*$, of the hyperplane h^*, called the most piercing point, is feasible and is given by

$$\mathbf{p}^* = (1 - k_{h^*})\mathbf{w} + k\mathbf{u} \quad (9)$$

The arc, say $(i, j)^*$, whose corresponding dual constraint is the most violated one, is then added to the RNP and the index sets of **I** and **R** are updated as:

$\mathbf{I} = \mathbf{I} \cup (i, j)^*$

$\mathbf{R} = \mathbf{R} \sim (i, j)^*$

where ~ denotes set subtraction. Note that we need not probe arcs that are nonbasic at capacity because they must be in the RNP by definition.

Following each probe, the new RNP is solved by a standard, primal network simplex method. The arcs in set **R** (not in the RNP) cannot be involved in any pivoting activity. Therefore, noncandidate arcs in set **R** never influence the basis or the value of an optimal primal or dual solution of any RNP. Thus, we could manage the noncandidate arcs separately and consider only the candidate arcs of the RNP. Of course, one arc list with flags or pointers is generally sufficient in managing the arcs in both sets **I** and **R**. In the next section, this paper presents a detailed algorithmic statement of PAPANET.

3.2 Algorithm Statement

The Pivot and Probe Algorithm for Network optimization, PAPANET, may be stated as follows:

Step 1: Initialization

Use an all artificial/slack start. Calculate the current RNP dual solution **u**. Let **w=0** be the initial feasible solution to the original NP; **I** denote the current set of arcs in RNP, and **R** the remaining set of arcs.

Step 2: Probe

Find all violated primal variables (arcs = dual constraints), i.e., define set **H** from **R**. It is possible to identify a set of limited size. If **H** is empty, then the solution satisfies all dual constraints of set **R**.

Therefore, the current primal optimal solution is optimal to the original NP. Terminate. Otherwise, use (5) and (6) to identify the most violated dual constraint(s) and its (their) corresponding arc(s) in **H**.

Add a subset of the arc(s) of **I**; delete the arc(s) from **R**. Update feasible point **w** using (9) or retain several as $\mathbf{w}^1$, $\mathbf{w}^2$,...$\mathbf{w}^q$.

Step 3: Pivot

Solve the RNP and obtain a new dual solution **u**. Return to Step 2.

The algorithm converges as long as the network simplex method used in Step 3 converges, since, in the worst case, the RNP consists of all the arcs of the original NP.

IV. PAPNET Implementation

This paper developed and implemented the Pivot And Probe Algorithm for NETwork

optimization in Fortran (F77) on an IBM RS/6000 Model 340 workstation. The implementation was performed by modifying an efficient network simplex code, MINIC (Sun, 1986), by altering its pricing operation and adding a PROBE subroutine.

For RNP, PAPANET utilizes an all artificial start that includes the entire node set and all the artificial and slack arcs. PROBE is called to determine the candidate arcs that are to be added to RNP, which is solved again to optimality. The method iterates until PROBE cannot obtain any candidate arc. In large problems, we set PROBE to identify multiple arcs to enter RNP. Using PAPANET, we can solve NPs with a significantly reduced number of arcs and pivots, resulting in an substantial computational time savings over a standard, network simplex implementation.

We performed preliminary tests on randomly generated capacitated network flow problems by NETGEN (Klingman, Napier and Stutz, 1974). The preliminary test results are shown in Table 1. We solved square transportation problems, i.e., half supply nodes and half demand nodes. The capacities (ranging from 1 to 1000) and costs (ranging from 1 to 100) of all arcs were generated from integer uniform distributions. We initially set PROBE to find at most, as many candidate arcs as the number of nodes, and up to 300 arcs thereafter. Cyclic pricing was used in both MINIC and PAPANET. For both, the first favorably priced arc enters the basis. PAPANET has a flag, STATUS(j), indicating whether an arc j is candidate (STATUS(j)=1) or noncandidate (STATUS(j) =0).

In Table 1, the columns entitled "CPU Time" are the solution CPU times in seconds. The total pivot counts required to obtain an optimum are shown in columns "NO. Pivs". Under PAPANET, "NO. Arcs" indicates that the total number of arcs that are candidate (in RNP) at the optimum and "NO. Probe" is the number PROBEs taken. In Table 2, we show a comparison of the CPU times and pivot counts between PAPANET and MINIC. The first four columns are ratios; the last two indicate the savings of PAPANET over MINIC. The results indicate that PAPANET effectively reduces the problem size (fewer arcs than the original NP) and reduces the pivot count, which result in an overall savings of computational time. For example, in Problem 10 having 2000 nodes and 100,000 arcs, only 30% of the arcs are needed by PAPANET resulting in 46% of the pivots required by MINIC and an overall time savings of 38% over MINIC. Figures 2 and 3 graphically show the number of pivots and the solution CPU times, respectively, required by both implementations plotted by the number of arcs (effectively the problem density). Note that the efficiency of PAPANET increases as the problem size increases.

V. Conclusions and Future Research

This paper has presented a new algorithm, PAPANET, to solve minimum cost, capacitated, network flow problems. Our preliminary computational results demonstrate its efficiency over an equivalent, network simplex implementation, especially in solving larger problems.

Table 1. Computational Results of Solution CPU Times and Pivot Counts

	Problem size		MINIC		PAPANET			
Prob	Number of Nodes	Arcs	CPU Times	No. Pivs	CPU Time	No. Pivs	No. Arcs	No. Probe
1	2000	4,000	4.10	13048	3.10	9398	3354	8
2	2000	6,000	7.12	20191	5.26	13674	4072	9
3	2000	8,000	9.53	25666	5.95	15816	4632	10
4	2000	10,000	11.48	30165	7.86	18496	5298	9
5	2000	15,000	16.09	43097	10.38	24338	6766	9
6	2000	20,000	18.63	48513	12.28	27407	8190	10
7	2000	25,000	20.67	56974	12.01	29712	9377	9
8	2000	40,000	26.00	68042	15.78	33875	13182	9
9	2000	80,000	36.32	91536	22.98	43189	24586	11
10	2000	100,000	36.18	98678	23.58	45562	30277	11

Table 2. Comparison of The Computational Results (PAPANET vs. MINIC)

	MINIC/PAPANET		PAPANET/MINIC		SAVING OVER MINIC	
Prob	CPU Times	No. Pivs.	CPU Time	No Pivs.	CPU Times	No. Pivs.
1	1.323	1.388	0.756	0.720	24.39%	27.97%
2	1.354	1.477	0.739	0.677	26.12%	32.28%
3	1.602	1.623	0.624	0.616	37.57%	38.38%
4	1.461	1.631	0.685	0.613	31.53%	38.68%
5	1.550	1.771	0.645	0.565	35.49%	43.53%
6	1.517	1.770	0.659	0.565	34.08%	43.51%
7	1.721	1.918	0.581	0.522	41.90%	47.85%
8	1.646	2.009	0.607	0.498	39.31%	50.21%
9	1.581	2.119	0.633	0.472	36.73%	52.82%
10	1.619	2.166	0.618	0.462	38.24%	53.83%

One algorithmic and computational improvement we intend to pursue is the dropping of candidate arcs from the **R** set during the Probe step. See Sethi(1983) and Sethi and Thompson (1983, 1984) for LP implementation details. PAPANET is a good candidate for parallel implementations, because continuous, synchronous probing may be performed, even while pivoting. Our future work will focus on improved algorithms and

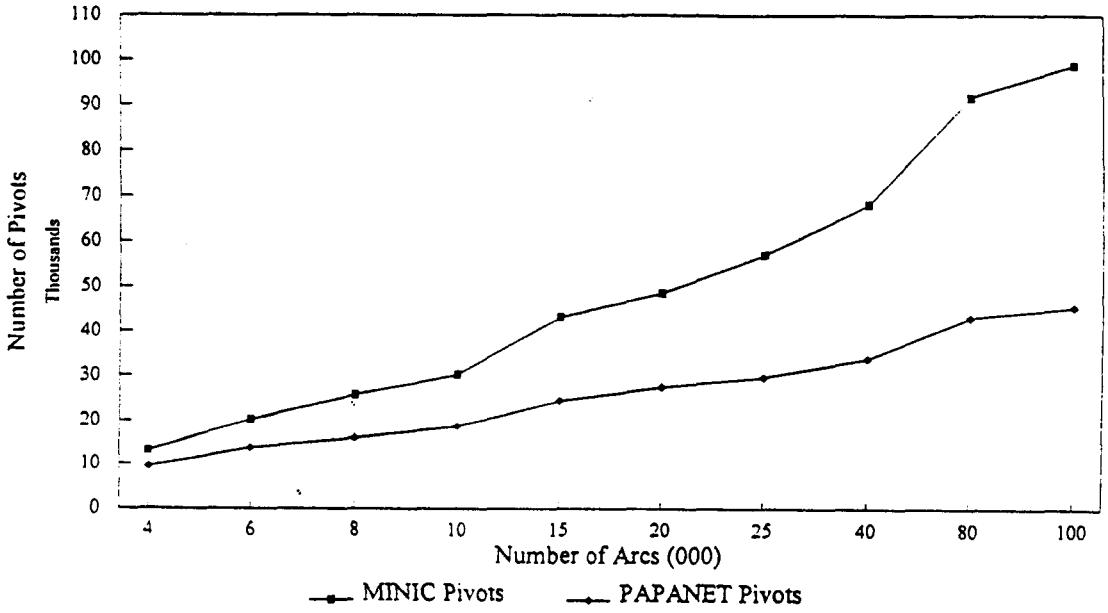

Figure 2. MINIC vs. PAPANET(number of pivots)

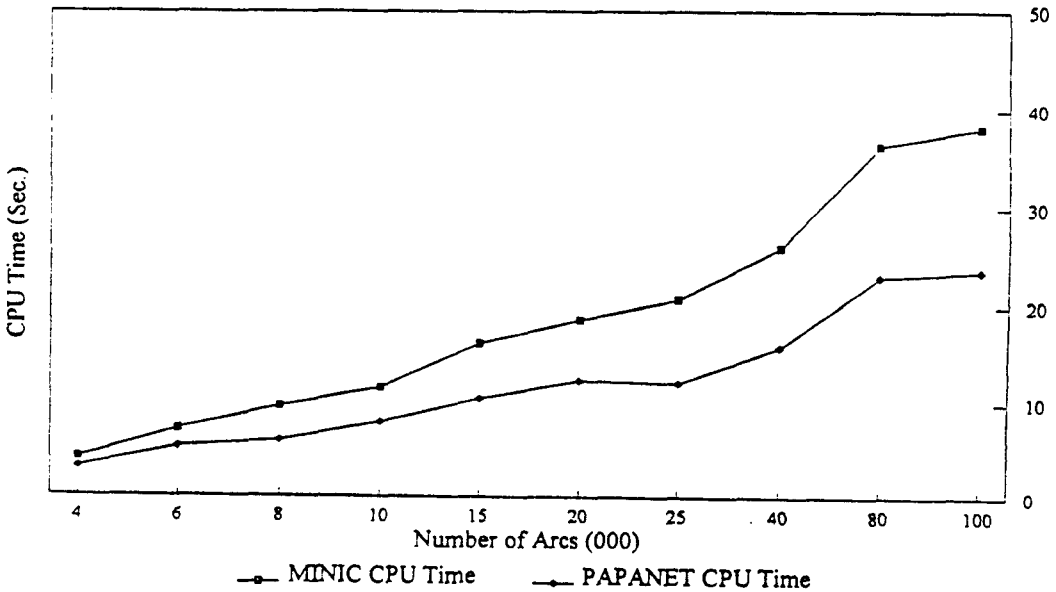

Figure 3. MINIC vs. PAPANET(CPU Times)

implementations, both serial and parallel.

Reference

[1] Aderohunmu, Rotimi S. and Jay E. Aronson, "The Solution of Multiperiod Network Flow Problems by Aggregation," *Management Science*, 39, 1(1993), pp. 54-71.

[2] Aronson, Jay E., "A Survey of Dynamic Network Flows," *Annuals of Operations Research*, 20(1989), pp. 1-66.

[3] Aronson, Jay E. and B. David Chen, "A Forward Network Simplex Algorithm for Solving Multiperiod Network Flow Problems," *Naval Research Logistics Quarterly*, 30(1986), pp. 445-467.

[4] Aronson, Jay E. and Moonsig Kang, "A Pivot And Probe Algorithm(PAPA) for Solving Network Flow Problems," Decision Sciences Institute, Second International Meeting, Seoul, Korea, June 14-16 (1993), Proceedings: pp. 220-223.

[5] Balachandran, V., and G. L. Thompson, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: 1. Basic Theory," *Naval Research Logisitics Quarterly*, 22, 1(1975a), pp. 79-100.

[6] Balachandran, V., and G. L. Thompson, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: II. Rim, Cost, and Bound Operators," *Naval Research Logisitics Quarterly*, 22, 1(1975b), pp. 101-126.

[7] Balachandran, V., and G. L. Thompson, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: III. Weight Operators," *Naval Research Logistics Quarterly*, 22, 2(1975c), pp. 297-316.

[8] Balachandran, V., and G. L. Thompson, "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: IV. Global Operators," *Naval Research Logistics Quarterly*, 22, 2(1975c), pp. 317-340.

[9] Barr, Richard S., Fred Glover and Darwin Klingman, "Enhancements of Spanning Tree Labelling Procedures for Network Optimization," *Infor.*, 17, 1 (Feb., 1970), pp. 16-34.

[10] Bazaraa, Mokhtar S., John J. Jarvis and Hanif D. Sherali, *Linear Programming and Network Flows*, 2nd ed., John Wiley & Sons, New York, NY, 1990.

[11] Bertsekas, Dimitri P., *Linear Network Optimization: Algorithms and Codes*, The MIT Press, Cambridge, MA, 1991.

[12] Evans, James R. and Edward Minieka, *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, Inc., New York, NY, 1992.

[13] Fulkerson, Delbert R., "An Out-of-Kilter Method for Minimal-Cost Flow Problems," *J. Society of Industrial and Applied Mathematics*, 9, 1(1961), pp. 18-27.

[14] Glover, Fred, Darwin Klingman and

Nancy V. Phillips, *Network Models in Optimization and their Applications in Practice*, John Wiley & Sons, New York, NY, 1992.

[15] Karney, David and Darwin Klingman, "Implementation and Computational Study on an In-Core, Out-of-Core Primal Network Code," *Operations Research*, 24, 6 (Nov.-Dec. 1976), pp. 1056-1077.

[16] Karwan, Mark, Vahid Lotfi, Jan Telgen and Stanley Zionts (eds.), *Redundancy in Mathematical Programming*, Springer-Verlag, Berlin, Germany, 1983.

[17] Kennington, Jeff L., and Richard V, Helgason, *Algorithms for Network Programming*, John Wiley and Sons, New York, NY, 1980.

[18] Klingman, Darwin, A. Napier and Joel Stutz, "NETGEN : A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimun Cost Flow Network Problems," *Management Science*, 20, 5(1974), pp. 814-821.

[19] Murty, Katta. G., *Linear and Combinatorial Programming*, John Wiley & Sons, New York, NY, 1976.

[20] Sethi, Awanti P., Algorithmic Enhancements of the Simplex Method, Unpublished Doctoral Dissertation, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, April, 1983.

[21] Sethi, Awanti P. and Gerald L. Thompson, "The Non-Candidate Constraint Method for Reducing the size of a Linear Program," in *Redundancy in Mathematical Programming*, Karwan, Mark, Vahid Lotfi, Jan Telgen and Stanley Zionts, eds., Springer-Verlag, Berlin, Germany, 1983.

[22] Sethi, Awanti P. and Gerald L. Thompson, "The Pivot and Probe Algorithm for Solving a Linear Program," *Mathematical Programming*, 29(1984), pp. 219-233.

[23] Sethi, Awanti P., and Gerald L. Thompson and Ming S. Hung, "An Efficient Simplex Pricing Procedure," Working Paper No. 1990-42, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, (Nov. 1990).

[24] Simmonnard, Michel. A., *Linear Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[25] Srinivasan, V. and Gerald L. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm, "*Journal of the Association for Computing Machinery*, 20(1973), pp. 194-213.

[26] Srinivasan, V. and Gerald L. Thompson, "Algorithms for Minimizing Total Cost,

Bottleneck Time and Bottleneck Shipment in Transportation Problems," *Naval Research Logistics Quarterly,* 23, 4(1976), pp. 567-596.

[27] Srinivasan, V. and Gerald L. Thompson, "Cost Operator Algorithms for the Transportation Problem, " *Mathematical Programming,* 12, 3(1977), pp. 372-391.

[28] Sun, Minghe, "MINIC: MINImum Cost Network Flow Code, "Department of Management Sciences and Information Technology, Terry College of Business Administration, The University of Georgia, Athens, GA (1986).

[29] Thompson, Gerald L. and Awanti P. Sethi, "Solution of Constrained Generalized Transportation Problems Using the Pivot and Probe Algorithm," *Comput. & Ops. Res.*, 13, 1(1986), pp. 1-9.