韓國經營科學會誌
第27卷 第2號
2002年 6月

네트워크 문제 해결에 있어서 효과적인 pricing 방법에 관한 연구*

강 문 식*

An Efficient Pricing Strategy(PAPANET) for Solving Network Flow Problems*

Moonsig Kang*

■ Abstract ■

In this paper, we present an efficient pricing strategy, the pivot and probe Algorithm for Network Flow Problems (PAPANET), specifically for solving capacitated, linear network flow problem (NPs). The PAPANET begins with an initial relaxed network problem (RNP), consisting of all the nodes and initial candidate arcs (possibly a few least cost arcs form the original problem and a set of all the artificial and slack arcs). After an initial solution to the RNP is derived by pivoting, the PROBE procedure identifies a set of most violated arcs from the noncandidate arcs that are not considered to be in the current RNP, and adds them to the RNP. The procedure also discards a set of least favorable, zero flow, nonbasic arcs from the RNP. The new RNP is solved to optimality and the procedure continues until all of the dual constraints of the noncandidate arcs are satisfied by the dual solution to the RNP. The PAPANET effectively reduces the problem size, time per pivot, and solution CPU time by eliminating noncandidate arcs. Computational tests on randomly generated problems indicate that PAPANET achieves an average savings of 50-80% of the solution CPU time of that of a comparable standard network simplex implementation.

Keyword : Network Flow Problems, Algorithms

논문접수일 : 2001년 12월 21일 논문게재확정일 : 2002년 6월 5일

* The present research has been conducted by the Bisa Research Grant of Keimyung University.

** Assistant Professor, Dept. of Management Information Systems, Keimyung University

1. Introduction

The network flow model describes an important class of optimization problems that has many applications in practice, e.g., production planning and scheduling, economic planning, communication systems, inventory systems, logistics systems, traffic systems, and many other areas that require a shipment of a commodity from supply points to demand points (see [2]). Because of the numerous important applications, much theoretical and algorithmic development research has been conducted on network flows [1, 3-6, 10, 15-17, 25, 26]. In this paper, we present a new method for improving the algorithmic efficiency of network flow algorithms. It is based on the Pivot And Probe Algorithm (PAPA) for solving linear programming(LP) problems developed by Sethi and Thompson [19, 20] and Sethi [18].

When using a simplex-based network optimization procedure, typically a large portion of arcs never enter the basis and at optimum, obviously, most of the arcs are nonbasic with zero flow (lower bound). Our tests on randomly generated capacitated network flow problems (NP) indicate that about 60~80% of all arcs are nonbasic with zero flow at the optimum, and only about 30-50% of all arcs are utilized once or more in the solution process (this varies with the problem characteristics). Similar results are described by Sethi [18] and Sethi and Thompson [20] in their Pivot And Probe Algorithm for solving the pure linear programming problem, a generalization of the pure network flow problem. We define a candidate arc as one that has a potential to enter the basis at least once. A noncandidate arc is one that never enters the basis, thus remaining nonbasic with zero flow throughout the optimization process and not affecting the optimum. Clearly, the solution of an NP can be obtained without the noncandidate arcs. It would reduce the problem size and be an effective way to improve computational efficiency.

PAPANET, the Pivot And Probe Algorithm for Network Flow Problems, starts with all the nodes and initial candidate arcs (a few least cost arcs from the original problem and all the artificial and slack arcs). A standard network simplex method is used to derive an initial primal solution and its corresponding dual solution to the relaxed network problem (RNP). Using the dual solution, the PROBE step identifies the most violated arcs from the noncandidate arcs that are not considered in the current relaxation. The most violated arcs become candidate and are added to the RNP. The PROBE step also discards the least favorable nonbasic zero flow arcs from the RNP. The new RNP is solved again and the process continues until all candidate arcs price unfavorably and all of the noncandidate arcs are satisfied by the dual solution to the new RNP.

We expect that the implementation of PAPANET would require significantly fewer arcs and fewer pivots, less time per pivot and less overall solution CPU time than is required by the comparable, standard network simplex implementation from which it is derived.

The paper is organized as follows: a brief description of the Pivot And Probe Algorithm for linear programming is presented in Section 2. In Section 3, we discuss the Pivot And Probe Algorithm for Network Flow Problems (PAPANET). Section 4 contains the implementation details of the PAPANET and introduces two coded versions of the new algorithm. Computational results are presented and analyzed in Section 5.

Section 6 is our summary and conclusions. We assume that the reader is familiar with the basic definitions, methods, and implementation of linear programming (e.g., see [7, 22]) and network programming (e.g., see [7-9, 11]).

2. The Pivot And Probe Algorithm for Linear Programming

The Pivot And Probe Algorithm (PAPA) for linear programming (LP) was developed by Sethi [18], and later by Sethi and Thompson [19, 20]. Thompson and Sethi [24] further applied the PAPA to solve constrained generalized transportation problems, and Sethi, Thompson and Hung [21] introduced its specialization to the LP dual. The conceptual foundation of PAPA was to reduce the active problem size by maintaining only a small number of constraints or variables that have the potential to be included in an optimal solution to an LP. The original idea involved determining which constraints to include and which to omit. Sethi and Thompson [19] defined a candidate constraint to be one that has a potential pivot element to enter the basis in at least one pivot step. Similarly, a noncandidate constraint is defined as one that never enters the basis during the course of solving an LP. Retaining only small number of constraints (primal) or variables (dual) reduces the active problem size, and reduces pricing and pivoting effort.

Consider a primal linear program (PLP) that may be defined as:

$$\max z = cx, \tag{1}$$

$$s.t. Ax \leq b, \tag{2}$$

$$x \geq 0, \tag{3}$$

where **A** is an mxn matrix, **b** is an m vector, and **c** and **x** are n vectors. Primal PAPA stars with a relaxed linear program (RLP) consisting of all initial candidate constraints. An initial candidate constraint is defined as one that contains a pivot element if any of the favorable variables ($z_j - c_j > 0$) is selected to enter the basis by the standard simplex algorithm. Geometrically, the initial candidate constraints consist of the collection of constraints that have an intercepting point on some coordinate axis that is closest to the origin. To enforce the finiteness of each RNP, a regularization constraint, $\mathbf{ex} \leq M$, where M is a large number and $\mathbf{e} = (1, 1, \cdots, 1)$, is added to the RLP. Once an optimal solution, $\mathbf{x}^*$, to the RLP is found, primal PAPA probes the noncandidate constraints (not considered in the current relaxation) to find the most violated one (or a set of several). The probe step identifies the piercing points (if any) of the line segment between any feasible point of the original LP (i.e., $\mathbf{x} = 0$) and $\mathbf{x}^*$, with violated noncandidate constraints. Noncandidate constraints are said to be violated if they are not satisfied by the current RLP solution, i.e., $a_i\mathbf{x}^* > b_i$. One the piercing points for the noncandidate constraints are identified, a set of constraints, called most violated constraints, containing piercing points closest to the feasible point, are added to the RLP. Also, it is possible to drop constraints that are loose in the current RLP solution in an analogous manner. The new RLP is solved to optimality by the dual simplex method. The most piercing point found by any probe is feasible to the original LP, and can thus be used as a feasible point by later probes. This procedure is repeated until an optimal solution to RLP is satisfied by

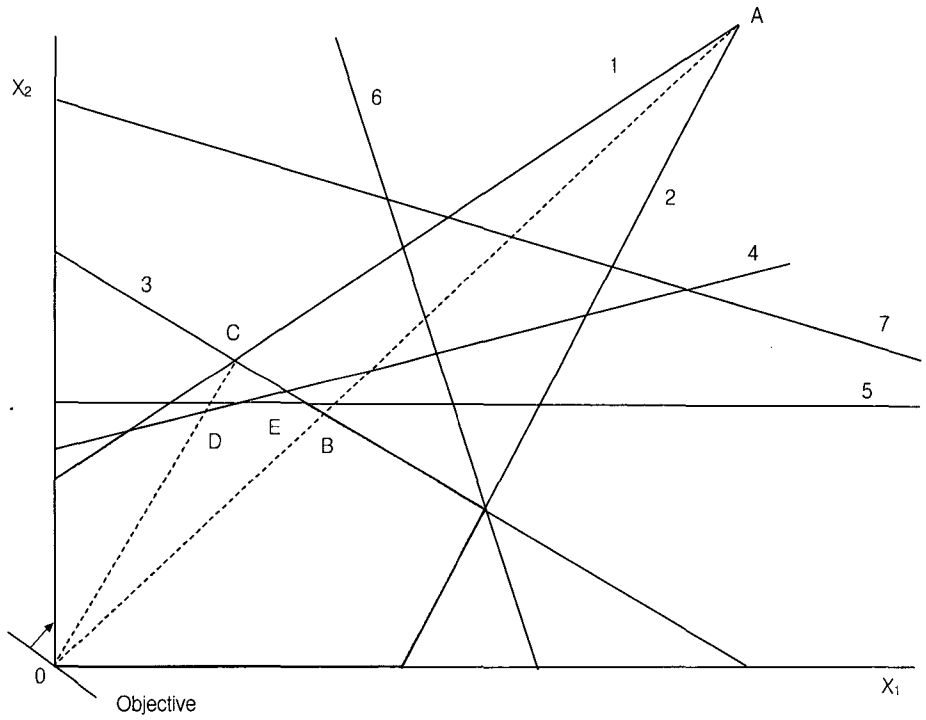
all the noncandidate constraints.

To clarify the algorithm, consider the LP maximization problem with 7 constraints and 2 variables shown graphically in [Figure 1].

Constraints 1 and 2 are initially candidate to enter RLP and two standard simplex pivots obtain point A as an optimum. From point A, probing to the origin, **0**, identifies constraint 3 as the most violated constraint and point B as the most piercing point (note that point B is feasible). After adding constraint 3 to RLP, point C is found to be an optimum to the new RLP. From point C, probing to the origin selects constraint 4 and probing to point B selects constraint 5 to be candidate constraints. Constraints 4 and 5 are added to the RLP and point E is found to be an optimal solution to the RLP. Since point E satisfies all noncandidate constraints, it is an optimum to the original LP.

The implementation of the Pivot And Probe Algorithm (PAPA) for linear programming (LP) on varying sizes ranging from 50×70 to 300×310 indicated that savings of 20~80% of the solution CPU time can be achieved [20]. PAPA required 80% of the solution CPU time of the standard simplex method when m = 50 but less than 20% when m = 300, implying that the PAPA is more effective when the problem size increases.

The main reason for the computational savings are the average size of RLP. The average size of the RLP starts at .33 for m = 50 and drops down to .1 when m = 300. It implies that the PAPA is less effective as the density decreases because the average size of RLP increases. Actually, Sethi and Thompson [20] showed that the PAPA method becomes somewhat less



[Figure 1] Example Problem

effective, apparently because the average size of RLP increases, as the density decreases. They also showed that the PAPA is less effective when the number of negative constraints increases, because of an increase in the average size of the RLP.

3. The Pivot and Probe Algorithm for Network Flow Problems (PAPANET)

3.1 Algorithm Development

The capacitated, linear, minimum cost network flow problem (NP) may be expressed as :

$$\text{Min} \sum c_{ij} x_{ij}, \quad (i,j) \in A, \tag{4}$$

$$\text{s.t.} \sum_j x_{ij} - \sum_j x_{ji} = r_i,$$

$$(i,j), \ (j,i) \in A, \ i \in N, \tag{5}$$

$$0 \le x_{ij} \le b_{ij}, \ (i,j) \in A. \tag{6}$$

The complementary slackness conditions corresponding to the NP may be stated as :

$$u_i - u_j \le c_{ij}, \text{for all} (i,j) \in A \text{ with } x_{ij} = 0, \tag{7}$$

$$u_i - u_j \le c_{ij}, \text{for all} (i,j) \in A \text{ with } x_{ij} = b_{ij}, \tag{8}$$

$$u_i - u_j \le c_{ij}, \text{for all} (i,j) \in A \text{ with } 0 < x_{ij} < b_{ij}, \tag{9}$$

where **A** denotes the set of arcs, **N** denotes the set of nodes, r_i is the requirement of node i (positive for a supply node, negative for a demand node, zero for a transshipment node), c_{ij} is the per unit cost coefficient of arc (i, j), x_{ij} is the flow of arc (i, j), b_{ij} is the upper bound on flow or capacity of arc (i, j) and u_i and u_j are node potentials (dual variables). We assume that the lower bound on the flow of each arc is zero. Otherwise, a simple transformation to NP is required. Furthermore, because of the special finiteness property of the capacitated network flow problem, a regularization constraint is redundant (if $b_{ij} = \infty$, generally a large finite value is used).

We next describe the application of the Pivot and Probe Algorithm concepts to the dual to the NP. Arcs are probed to determine if they should be included in a relaxed primal NP. PAPANET begins with an initial relaxed NP (RNP) consisting of the entire node set, a set of all the artificial and slack arcs, and few least cost arcs. An initial solution is derived by applying the standard network simplex method (alternatively, an advanced start could be used first). When each RNP has been solved to optimality, PAPANET PROBEs to determine a set of new, potential variables to enter the RNP from the noncandidate arcs (not in the current RNP). If the optimal solution to the RNP is feasible to the original problem, it is an optimal solution to the original problem and the method stops. Otherwise, in addition to identifying new arcs to enter the RNP, the PROBE can also release the least favorable nonbasic zero flow arcs from the RNP. In any iteration, let **C** be the index set of candidate arcs that form the current RNP ; **R** be the index set of noncandidate arcs; **w** be a dual feasible point; and **u*** be the optimal dual solution found to the RNP. Let H denotes the index set of noncandidate arcs that violate (7) :

$$H = \{(i,j) \mid u^* i - u^* j > c_{ij}\}, (i,j) \in R \tag{10}$$

Note that a nonbasic arc at upper bound cannot be noncandidate because all noncandidate arcs are not in RNP by definition. Therefore, equation (8) is irrelevant to the PROBE. Formally, a probe is the operation of identifying the piercing points (if any) of the line segment between **u*** and **w** and dual constraints of noncandidate arcs in H,

i.e., the set of all such vectors **p** between the current RNP solution and a dual feasible point. If we let **p** be any point on the line segment between **w** and **u***, then **p** can be defined as ;

$$p=(1-k)w+ku^*, \text{ for some } k\in[0,1]. \quad (11)$$

Let (i, j) ∈ **R**. Then the piercing point of the line segment (11) and hyperplane h defined by the dual constraint h = (i, j) ∈ **H** is obtained by solving the following for k_h :

$$p(e_i-e_j)=(1-k_h)w(e_i-e_j) + k_h u^*(e_i-e_j), \quad (12)$$

where e_i and e_j are unit vectors in E^m with 1's in the ith and jth positions respectively. Equation (12) can be reduced to

$$p_i-p_j=(1-k_h)(w_i-w_j)+k_h(u_i^*-u_j^*). \quad (13)$$

From (7) and (13), we obtain

$$k_h=\frac{(w_i-w_j-c_{ij})}{(w_i-w_j-u^*{}_i+u^*{}_j)}, \quad \text{where}(i,j)\in H. \quad (14)$$

In (14), a lower value of k_h indicates that the hyperplane is closer to the feasible point **w**. In the PROBE step, we identify the most violated dual constraint that contains the most piercing point, defined as the one closest to a feasible point. Formally, a dual constraint $h^*\in$**H** is said to be most violated if

$$k_{h^*}=\min\{k_h \mid h\in H\}. \quad (15)$$

The piercing point, p*, of the hyperplane h*, called the most piercing point, is dual feasible and is given by

$$p^*=(1-h_{h^*})w+k_{h^*}u^*. \quad (16)$$

The methodology is valid for adding any number of violated arcs, and the PROBE step is valid anytime, regardless of whether **u*** is an optimum to the RNP or not. As part of the method, unfavorable arcs with zero flow (slack dual constraints) may be removed from the primal RNP (dual to the RNP). To release an arc from an RNP, let **L** denote the index set of unfavorable nonbasic, zero flow arcs that are considered in the current relaxation :

$$L=\{(i,j)\mid u^*{}_i-u^*{}_j<c_{ij} \text{ and } x_{ij}=0\}, \quad (i,j)\in C, \text{and } i,j\in N. \quad (17)$$

Then, an arc l* = (i, j)* is said to be a most unfavorable arc if

$$l^*=\{(i,j)^* \mid \bar{c}_{(ij)}=\min u^*{}_i-u^*{}_j-c_{ij}, \text{ for all } (i,j)\in L\}. \quad (18)$$

The arc, say (q, r)*, which corresponds to the most violated dual constraint is added to the RNP and the most unfavorable arc, say (s, t)*, is removed from the RNP. The index set of **C** and **R** are updated as :

$$C=C\cup(q,r)^*\sim(s,t)^*, \quad (19)$$

$$R=R\sim(q,r)^*\cup(s,t)^*, \quad (20)$$

where ~ denotes a set subtraction. Again, note that several arcs may be added and removed. If **w** = **0** (we probe to the origin from **u**), then (14) and (16), respectively, are simplified to

$$k_h=\frac{cij}{(u^*{}_i-u^*{}_j)}, \quad \text{where } (i,j)\in H \text{ and} \quad (21)$$

$$p^*=k_{h^*}u^*. \quad (22)$$

Then, when **w** = **0**, the process of evaluating k_{h^*} in (15) is equivalent to the standard simplex

pricing operation to identify the most favorable nonbasic lower bounded arc enter the basis of NP, that is, find $h^* = (i, j)^*$ such that

$$h^* = \{(i,j)^* \mid \bar{e}_{(ij)*} = \max\{u^*_i - u^*_j - c_{ij}\}, \text{ for all}(i,j) \in R\}. \quad (23)$$

The new RNP with arc set **C** is solved to optimality and the PROBE step is again invoked. The process continues until all of the dual constraints relative to the arcs in **R** are satisfied by the dual solution to the current RNP, **u***.

3.2 Algorithm Statement

We now formally state the Pivot And Probe Algorithm for Network Flow Problems (PAPANET) :

Step 1 : Initialization
Form an initial RNP with a set of all artificial and slack arcs and few least cost arcs. Calculate its dual solution **u***. Let $\mathbf{w} = \mathbf{0}$ be the initial feasible solution to the original NP[1] ; **C** denote the set of arcs in the current RNP ; **R** be the remaining set of arcs. Continue with Step 2.

Step 2 : PROBE
Define H from **R** (10). If **H** is empty, then the current RNP solution is optimal to the original problem and stop. Otherwise, identify the most piercing point **p*** and most violated arc(s) from **H** using (14)~(16) and the most unfavorable arc from **L** using (18). Update the sets **C** and **R** by (19) and (20) respectively. Update **w** with **p*** or retain several as $\mathbf{w}^1$, $\mathbf{w}^2$, ⋯ $\mathbf{w}^q$. the first PROBE requires one feasible dual solution, the origin, and, thereafter, at least two feasible dual solutions : the origin and **p***. Continue with Step 3.

Step 3 : Pivot
Solve the RNP with **C** by the standard network simplex method and obtain a new dual solution **u***. Return to Step 2.

The Algorithm converges as long as the simplex method in Step 3 converges, because probing from **u** to **0** is equivalent to the pricing scheme of the network simplex method and, in the worst case, the RNP can evolve to become the original NP.

4. PAPANET Implementation

The PAPANET code was developed by modifying an efficient existing primal network simplex code, MINIC [23], to include a PROBE subroutine. In our implementation, PAPANET utilizes an all artificial start to form an initial RNP that include the entire node set and all the artificial and slack arcs. The number of least cost arcs that enter the RNP in the first iteration is set to equal the number of nodes. Once optimal

1) We used $\mathbf{w} = \mathbf{0}$ as an initial dual feasible solution. With an all artificial start, only artificial and slack arcs are basic and all the flows of the structural arcs are at their lower bound of 0. When $\mathbf{w} = \mathbf{0}$, $x_{ij} = 0$, and $c_{ij} > 0$ for all $(i,j) \in \mathbf{A}$, the primal constraint (5) may not be feasible but the dual constraint (7) is satisfied. Thus $\mathbf{w} = \mathbf{0}$ is initially feasible. When $c_{ij} < 0$, (7) is not satisfied by $\mathbf{w} = \mathbf{0}$. However, as mentioned earlier, probing from a point to the origin is equivalent to the standard simplex pricing procedure. Therefore, probing with $\mathbf{w} = \mathbf{0}$ always converges and the origin may be the best estimate of an initial dual feasible solution.

solution to the RNP is obtained with the network simplex method, PROBE is called to identify arcs to add to or remove from the RNP. Two feasible dual solutions are used to find entering (candidate) arcs for the RNP. The first PROBE step performs one probe, from the current solution to the origin, and, thereafter, PROBE performs two probes, one from the current solution to the origin and one from the solution to the most piercing point that was found in the previous iteration. Instead of a single arc, PAPANET identifies multiple arcs to enter or leave the RNP in the test problems. Half of the unfavorably priced arcs are removed from the RNP. The number of arcs entering the RNP varies, depending upon the size and the type of problem. Cyclic pricing was used by both MINIC and PAPANET. For both, the first favorably priced arc found enters the basis. Both the MINIC and PAPANET codes incorporate typical data structures. The node lists consist of the (1) node name, (2) requirement, (3) flow, (4) dual, (5) predecessor, (6) orientation, (7) arcid, and (8) next (thread). The arc lists consist of the (1) arc name, (2) from node, (3) to node, (4) cost, and (5) capacity.

In PAPANET, adding or removing arcs can easily be done by using a flag, STATUS(j), indicating that an arc j is noncandidate (STATUS(j) = 0), candidate and nonbasic (STATUS(j) = 1), or basis and candidate (STATUS(j) = 2). The computational results show that PAPANET can solve NPs with a significantly reduced number of arcs and pivots and, consequently, a substantial computational time savings over the standard network simplex code from which the implementation is derived. Detailed computational comparisons and analyses are provided in Section 5.

While testing th PAPANET, we observed the well-known optimization phenomenon called the long-tail-of-convergence (see [2, 12]), in obtaining an optimal solution to each RNP. Basically, as a method moves toward an optimum, the improvement in the objective function value per pivot tends to decrease. A second version of PAPANET was developed to limit the effects of this phenomenon. This version, PAPANET2, does not solve each RNP to optimality, but rather solves each one to a near optimum by limiting the number of pricing cycles (the number of times the RNP arc list is completely checked) to four before probing. Our preliminary tests indicated that the objective values of these solutions were typically within approximately 5% of the optimal objective value to the RNP. Then, a probe updates the arc list of the RNP. We shall refer to the original implementation that optimizes each RNP as PAPANET1 and to the modified coed that limits its optimization activity as PAPANET 2.

5. Computational Testing

PAPANET1 and PAPANET 2 were tested by solving medium-and large-scale randomly generated capacitated network flow problems constructed by NETGEN [14]. The codes were developed in C and all solution CPU times reported are on an IBM RS/6000 Model 590 POWERStation (Workstation). In testing, our primary interests were on measuring the solution CPU time (exclusive of problem generation, input and output), number of pivots, time per pivot, number of PROBES, and number of arcs that entered the RNP at least once.

5.1 Medium - Scale Problems

<Table 1> shows the parameters of the 50 medium-scale problems (Problem Set A) of a NETGEN problem suite developed by Klingman and Mote [13]. Set A contains both transportation (101~120) and transshipment problems (121~150) having 5000 nodes with different sets of total number of arc costs, capacity ranges, etc.

We selected a sample of three problems for problem Set B to test the sensitivity of the new algorithms with regard to different probe sizes. This problem set includes two transportation problems and one transshipment problem from Problem Set A. The probe size is the maximum number of arcs that may be found as candidate in one probe and is defined as the fraction of the total number of arcs. In a probe, the entire arc list is scanned, and, from the arcs not in the RNP, a set of best arcs of 'probe size' is added to RNP. The computational results of both PAPANET1 and PAPANET2 with a variety of probe sizes are listed in <Table 2>. The solution CPU times and PROBE times are shown in CPU seconds.

<Table 2> indicates that the number of PROBEs and the PROBE times decrease as the probe size increases. However, the overall solution times are more dependent on the number of pivots rather than on the PROBE times, because the PROBE CPU time is small compared to the total solution time. The overall solution CPU times and the pivot counts decrease up to a certain probe size (i.e., .04 or 4% of the arcs on Problem 101) and increase thereafter. The preliminary results indicate that. on average, the maximum differences between the best and the worst solution CPU times are about 15~20% of the best solution CPU times when we restrict the probe size to the range of .01~.05. The result implies that a probe size of .04 is best (or near best) for medium-sized transportation problems with 25000 arcs, .05 for medium-sized transshipment problems, and .01 for fairly large, medium-scale transportation or transshipment problems (75000 arcs or more). This result suggests that the probe size must be decreased as the problem size is increased. In terms of solution CPU times, PAPANET 2 is superior to PAPANET1 by an average of 10~15%.

<Table 3> shows the computational results for all 50 medium scale problems in Problem Set A solved by the 3 different codes, MINIC, PAPANET1 and PAPANET2. In both PAPANET1 and PAPANET2, the probe sizes are set to .04 for transportation problems and .05 for transshipment problems. In <Table 3>, the "Total Arcs" column contains the actual number of arcs of each problem generated by NETGEN. The "Time per Pivot" was calculated by taking the difference between the solution CPU time and PROBE Time (if applicable) and dividing the difference by the total number of pivots. For both PAPANET1 and PAPANET2. "Arcs Entered" indicates the total number of structural arcs that entered the RNP at least once and "AVG. Size" is the mean number of arcs, including artificial, slack and structural arcs, in the RNP throughout the solution process. For example, in Problem 108 having 5000 nodes and 50309 arcs, only 32% of the arcs are needed by PAPANET2 with an average RNP size of 9698 arcs (19%). Similarly, PAPANET2 utilized 40% of the arcs keeping an average RNP size of 13385 arcs (27%) and obtained an optimal solution in 14.99 CPU seconds while MINIC solved the problem in 69.03 CPU seconds (4.6 times slower than PAPANET2).

〈Table 1〉 NETGEN Problem Suite (Klingman and Mote 1987)

					Arc costs			Transshipment				Capacity			
Prob. No.	No. Nodes	No. Sources	No. Sinks	No. Arcs	Minimum	Maximum	Total Supply	Sources	Sinks	%High Cost	% Capacity	Minimum	Maximum	Random No.Seed	Objective Function
101	5,000	2,500	2,500	25,000	1	100	250,000	0	0	0	100	1	1,000	13,502,460	6,191,726
102	5,000	2,500	2,500	25,000	1	100	250,000	0	0	0	100	1	1,000	4,281,922	72,337,144
103	5,000	2,500	2,500	25,000	1	100	250,000	0	0	0	100	1	1,000	44,820,113	218,947,553
104	5,000	2,500	2,500	25,000	(100)*	(1)	250,000	0	0	0	100	1	1,000	13,450,451	(19,100,371)
105	5,000	2,500	2,500	25,000	101	200	250,000	0	0	0	100	1	1,000	14,719,436	31,192,578
106	5,000	2,500	2,500	12,500	1	100	125,000	0	0	0	100	1	1,000	17,365,786	4,314,276
107	5,000	2,500	2,500	37,500	1	100	375,000	0	0	0	100	1	1,000	19,540,113	7,393,769
108	5,000	2,500	2,500	50,000	1	100	500,000	0	0	0	100	1	1,000	19,560,313	8,405,738
109	5,000	2,500	2,500	75,000	1	100	750,000	0	0	0	100	1	1,000	2,403,309	9,190,300
110	5,000	2,500	2,500	12,500	1	100	250,000	0	0	0	100	1	1,000	92,480,414	8,975,048
111	5,000	2,500	2,500	37,500	1	100	250,000	0	0	0	100	1	1,000	4,230,140	4,747,532
112	5,000	2,500	2,500	50,000	1	100	250,000	0	0	0	100	1	1,000	10,032,490	4,012,671
113	5,000	2,500	2,500	75,000	1	100	250,000	0	0	0	100	1	1,000	17,307,474	2,979,725
114	5,000	500	4,500	25,000	1	100	250,000	0	0	0	100	1	1,000	4,925,114	5,821,181
115	5,000	1,500	3,500	25,000	1	100	250,000	0	0	0	100	1	1,000	19,842,704	6,353,310
116	5,000	2,500	2,500	25,000	1	100	250,000	0	0	0	0	1	1,000	88,392,060	5,915,426
117	5,000	2,500	2,500	12,500	1	100	125,000	0	0	0	0	1	1,000	12,904,407	4,420,560
118	5,000	2,500	2,500	37,500	1	100	375,000	0	0	0	0	1	1,000	11,811,811	7,045,842
119	5,000	2,500	2,500	50,000	1	100	500,000	0	0	0	0	1	1,000	90,023,393	7,724,179
120	5,000	2,500	2,500	75,000	1	100	750,000	0	0	0	0	1	1,000	93,028,922	8,455,200
121	5,000	50	50	25,000	1	100	250,000	50	50	0	100	1	1,000	72,707,401	66,366,360
122	5,000	250	250	25,000	1	100	250,000	250	250	0	100	1	1,000	93,040,771	30,997,529
123	5,000	500	500	25,000	1	100	250,000	500	500	0	100	1	1,000	70,220,611	23,388,777
124	5,000	1,500	1,500	25,000	1	100	250,000	1,000	1,000	0	100	1	1,000	52,774,811	17,803,443
125	5,000	1,500	1,500	25,000	1	100	250,000	1,500	1,500	0	100	1	1,000	22,492,311	14,119,622
126	5,000	500	500	12,500	1	100	125,000	500	500	0	100	1	1,000	35,269,337	18,802,218
127	5,000	500	500	37,500	1	100	375,000	500	500	0	100	1	1,000	30,140,502	27,674,647
128	5,000	500	500	50,000	1	100	500,000	500	500	0	100	1	1,000	49,205,455	30,906,194
129	5,000	500	500	75,000	1	100	750,000	500	500	0	100	1	1,000	42,958,341	40,905,209
130	5,000	500	500	12,500	1	100	250,000	500	500	0	100	1	1,000	25,440,925	38,939,608
131	5,000	500	500	37,500	1	100	250,000	500	500	0	100	1	1,000	75,294,924	16,752,978
132	5,000	500	500	50,000	1	100	250,000	500	500	0	100	1	1,000	4,463,965	13,302,951
133	5,000	500	500	75,000	1	100	250,000	500	500	0	100	1	1,000	13,390,427	9,830,268
134	1,000	500	500	25,000	1	100	250,000	500	500	0	100	1	1,000	95,250,971	3,804,874
135	2,500	500	500	25,000	1	100	250,000	500	500	0	100	1	1,000	54,830,522	11,729,616
136	7,500	500	500	25,000	1	100	250,000	500	500	0	100	1	1,000	520,593	33,318,101
137	10,000	500	500	25,000	1	100	250,000	500	500	0	100	1	1,000	52,900,925	46,426,030
138	5,000	500	500	25,000	1	100	250,000	500	500	0	100	1	50	22,603,395	60,710,879
139	5,000	500	500	25,000	1	100	250,000	500	500	0	100	1	250	55,253,099	32,729,682
140	5,000	500	500	25,000	1	100	250,000	500	500	0	100	1	500	75,357,001	27,183,831
141	5,000	500	500	25,000	1	100	250,000	500	500	0	100	1	2,500	10,072,459	19,963,286
142	5,000	500	500	25,000	1	100	250,000	500	500	0	100	1	5,000	55,728,492	20,243,457
143	5,000	500	500	25,000	1	100	250,000	500	500	0	0	1	1,000	593,043	18,586,777
144	5,000	500	500	25,000	1	10	250,000	500	500	0	100	1	1,000	94,236,571	2,504,597
145	5,000	500	500	25,000	1	1000	250,000	500	500	0	100	1	1,000	94,882,955	215,956,138
146	5,000	500	500	25,000	1	10000	250,000	500	500	0	100	1	1,000	48,489,922	2,253,113,811
147	5,000	500	500	25,000	(100)	(1)	250,000	500	500	0	100	1	1,000	75,578,374	(427,908,373)
148	5,000	500	500	25,000	(50)	49	250,000	500	500	0	100	1	1,000	44,821,152	(92,965,318)
149	5,000	500	500	25,000	101	200	250,000	500	500	0	100	1	1,000	45,224,103	86,051,224
150	5,000	500	500	25,000	1001	1100	250,000	500	500	0	100	1	1,000	63,491,741	619,314,919

* () indicates negative number

〈Table 2〉 Sensitivity Analysis of Probe on Problem Set B

Problem	Algorithms		Probe Size						
			0.01	0.02	0.03	0.04	0.05	0.1	1
101 (TP)** (25000 arcs)	PAPANET 1	CPU Time	13.06	12.93	12.89	10.40*	12.30	13.64	22.35
		No. Pivots	51059	49721	49030	42271	47208	50652	74087
		No. PROBE	25	19	17	12	11	8	6
		PROBE Time	0.38	0.27	0.25	0.16	0.15	0.10	0.07
	PAPANET 2	CPU Time	12.09	10.84	10.96	1049*	12.53	12.36	19.72
		No. Pivots	47482	43993	43613	43761	48224	67419	67419
		No. PROBE	30	21	18	18	16	12	11
		PROBE Time	0.45	0.31	0.24	0.22	0.19	0.14	0.10
123 (TS) (25000 arcs)	PAPANET 1	CPU Time	16.56	16.09	14.90	14.37	12.54*	14.06	16.32
		No. Pivots	73608	71822	67223	66055	60624	67781	76567
		No. PROBE	24	18	15	13	11	9	8
		PROBE Time	0.34	0.24	0.20	0.16	0.14	0.10	0.07
	PAPANET 2	CPU Time	14.68	14.29	14.16	12.12	11.15*	13.91	15.92
		No. Pivots	64509	64630	67691	59117	56305	688454	72768
		No. PROBE	28	21	20	16	14	12	12
		PROBE Time	0.38	0.25	0.24	0.19	0.15	0.12	0.10
120 (TP) (25000 arcs)	PAPANET 1	CPU Time	14.02*	14.19	15.49	16.19	17.62	23.14	43.82
		No. Pivots	45260	46388	50549	49636	55164	65469	114717
		No. PROBE	18	15	17	9	9	7	7
		PROBE Time	0.72	0.55	0.59	0.33	0.34	0.23	0.22
	PAPANET 2	CPU Time	12.45*	13.48	15.19	14.60	15.10	24.15	42.81
		No. Pivots	41752	44615	50549	48178	50290	69579	116271
		No. PROBE	21	16	17	13	13	12	12
		PROBE Time	0.79	0.57	0.60	0.44	0.42	0.35	0.34

* Best overall solution CPU time.

** TP = transportation, TS = transshipment.

The number of degenerate pivots required by PAPANET1 and PAPANET 2 are fewer than one half of the degenerate pivots required by MINIC. The average percentage of degenerate pivots performed out of the total number of pivots are 49%, 40%, and 41% by MINIC, PAPANET1 and PAPANET 2, respectively. Clearly, by reducing the working problem size, we maintain a "stronger" basis and tighter problem formulation than by carrying all arcs throughout the solution process.

Similar to the case of LPs, All three codes are relatively inefficient at solving transshipment problems with negative arc costs (Problems 147 and 148). We recognize that when all arc costs are negative, then, all arcs between transshipment nodes, and between transshipment nodes and demand nodes (either way) will initially price favorable, i.e.,

$$u_i - u_j - c_{ij} > 0 \text{ for } x_{ij} = 0, \qquad (24)$$

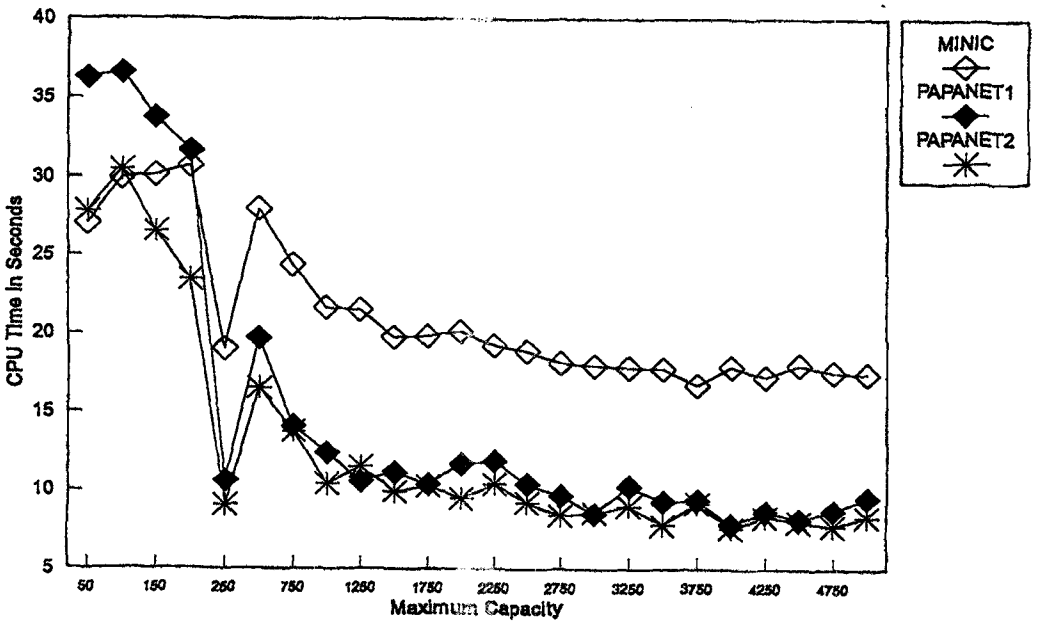
where $u_i = u_j = M$, and M is a very large, finite number. In cyclic pricing, most of the arcs enter the basis and much time is required to remove them. Not surprisingly, the computational results shown in <Table 3> indicate that a large portion (close to 100%) of the arcs have entered the RNPs during the solution process of PAPANET1 and PAPANET 2. For these problems, one may customize the method and the code by modifying the probe rule only to admit an arc if exactly one artificial arc is incident to either its from or to node.

〈Table 3〉 Computational Results on Problem Set A

Prob.	Tatal	MINIC					PAPANET1										PAPANET2									
		CPU	No.	Time	Deg.	Deg.	CPU	No.	Deg.	Deg	No.	PROB	Time	AVG.	Arcs	%Arcs	CPU	No.	Deg.	Deg	No.	PROB	Time	AVG.	Arcs	%Arcs
No.	Arcs	Time	Pivots	Per Pivot	Pivots	Pivots	Time	Pivots	Pivots	Pivot	ROB	Time	Per Pivot	Size	Entered	ntered	Time	Pivots	Pivots	Pivot	ROB	Time	Per Pivot	Size	Entered	ntered
101	25336	30.49	101744	0.000300	52537	52%	10.40	42272	14119	33%	12	0.16	0.000242	8092	12565	50%	10.5	43761	15833	36%	18	0.22	0.000235	9948	15966	61%
102	25387	40.06	114113	0.000351	18827	16%	18.29	72322	4695	6%	14	0.21	0.000250	9356	13751	54%	16.4	60494	4607	8%	18	0.22	0.000267	11428	15837	62%
103	25355	45.94	126300	0.000364	14785	12%	26.23	99063	4604	5%	14	0.23	0.000262	10291	15782	62%	27	96568	5283	5%	20	0.27	0.000276	13107	19209	76%
104	25344	30.88	104437	0.000296	55584	53%	17.21	55374	19979	36%	84	1.02	0.000292	6145	16629	47%	16.1	52289	19370	37%	92	1.15	0.000285	6646	18444	57%
105	25332	31.97	106167	0.000301	56402	53%	10.44	38714	13029	34%	14	0.17	0.000265	8379	11992	47%	9.24	36119	12329	34%	17	0.20	0.000250	9279	13383	53%
106	12870	13.61	55645	0.000245	32610	59%	9.93	41374	18506	45%	16	0.11	0.000237	7122	9644	75%	8.3	36078	16438	46%	18	0.11	0.000227	8032	10620	83%
107	37832	45.15	139424	0.000324	67527	48%	12.93	48612	13782	28%	10	0.19	0.000262	9526	14070	37%	11.4	44372	14201	32%	14	0.23	0.000252	11550	16294	43%
108	50309	69.03	164625	0.000419	66925	41%	15.32	53702	10355	19%	10	0.30	0.000280	9698	16012	32%	15	53233	11721	22%	16	0.39	0.000274	13385	20346	40%
109	75299	71.85	207708	0.000346	81381	39%	19.56	62323	12866	21%	8	0.30	0.000309	13193	19990	27%	17.3	59354	13362	23%	13	0.43	0.000284	16487	23509	31%
110	12825	14.28	53609	0.000266	25459	47%	9.64	37997	11978	32%	15	0.11	0.000251	7263	9737	76%	7.45	31272	10225	33%	17	0.11	0.000235	7792	10147	79%
111	37828	42.75	142100	0.000301	80255	56%	13.18	51094	18087	37%	12	0.23	0.000254	9829	15390	41%	11.2	45817	17566	38%	15	0.24	0.000239	11794	17026	45%
112	50325	50.55	163257	0.000310	90632	56%	15.07	53163	18585	35%	11	0.29	0.000278	10836	17856	35%	12.1	47579	16891	36%	14	0.33	0.000247	12996	19337	38%
113	75318	67.60	219832	0.000308	132917	60%	18.42	63119	23910	38%	10	0.36	0.000286	14802	23424	31%	14.9	53895	21177	39%	13	0.41	0.000268	16460	22416	30%
114	26514	19.44	82396	0.000236	42407	51%	6.80	34349	14167	41%	11	0.16	0.000193	8093	12621	48%	6.19	34244	15047	44%	16	0.20	0.000175	9461	14395	54%
115	25962	24.13	89588	0.000269	46705	52%	9.54	42201	15515	37%	13	0.17	0.000222	8122	12601	49%	8.53	39253	14595	37%	17	0.22	0.000212	9507	14545	56%
116	25304	24.41	77400	0.000315	33374	43%	11.91	42955	14361	33%	15	0.21	0.000273	8211	13811	55%	8.82	35044	11257	32%	15	0.20	0.000246	9341	14249	56%
117	12816	10.00	39369	0.000254	18059	46%	7.61	30446	12621	41%	16	0.10	0.000246	6696	9684	76%	7.83	31891	14014	44%	20	0.13	0.000241	7592	10861	85%
118	37797	35.20	109823	0.000321	47095	43%	11.57	41883	11687	28%	12	0.21	0.000271	9747	15675	41%	10.1	38032	10774	28%	13	0.23	0.000258	10972	17071	45%
119	50301	44.25	124018	0.000357	43278	35%	13.10	43979	8516	19%	10	0.25	0.000292	10374	15900	32%	12.3	42120	8444	20%	16	0.36	0.000282	13359	19807	39%
120	75330	59.55	162352	0.000367	53078	33%	16.19	49636	7810	16%	9	0.33	0.000320	11886	17839	24%	14.6	48178	7578	16%	13	0.44	0.000294	15283	24719	33%
121	25000	21.38	102306	0.000209	61033	60%	19.13	105817	64575	61%	15	0.16	0.000179	10491	15520	62%	14.9	85738	52675	61%	18	0.21	0.000171	11879	18737	75%
122	25000	20.34	93803	0.000217	58336	62%	11.72	58584	34337	59%	11	0.12	0.000198	9462	13010	52%	9.77	53722	33062	62%	14	0.16	0.000179	11120	15594	62%
123	25000	21.86	92641	0.000236	53587	58%	12.54	60624	31637	52%	11	0.13	0.000205	9302	13823	55%	10.7	56305	30532	54%	14	0.16	0.000187	11305	16666	67%
124	25000	26.18	91013	0.000288	42547	47%	15.73	62210	23914	38%	12	0.13	0.000251	9678	12111	48%	13.6	55808	22095	40%	17	0.19	0.000241	11280	15430	62%
125	25000	28.15	89590	0.000314	38398	43%	14.84	56333	16961	30%	12	0.15	0.000261	9162	12610	50%	12.9	51797	16849	33%	16	0.16	0.000246	11974	16492	66%
126	12500	9.75	40368	0.000242	23453	58%	8.07	36540	18486	51%	13	0.07	0.000219	7488	9530	76%	6.99	32322	17299	54%	16	0.09	0.000214	8014	10101	81%
127	37500	34.35	140692	0.000244	77113	55%	18.53	85833	42894	50%	11	0.19	0.000214	11805	17124	46%	15.1	72778	37327	51%	14	0.22	0.000204	14185	19848	53%
128	50000	42.77	170244	0.000251	88214	52%	24.36	101392	48702	48%	11	0.26	0.000238	12733	18777	38%	20.3	91469	45887	50%	15	0.31	0.000219	17644	24790	50%
129	75000	68.62	243572	0.000282	109303	45%	35.84	140348	57410	41%	10	0.35	0.000253	17749	28319	38%	31.2	126280	53367	42%	15	0.49	0.000213	23921	34384	46%
130	12500	10.87	41540	0.000262	20932	50%	9.33	39541	17818	45%	15	0.08	0.000234	7702	9820	79%	7.77	35385	16466	47%	17	0.09	0.000217	8240	10430	85%
131	37500	29.52	133041	0.000222	80479	60%	14.47	72387	40421	56%	11	0.20	0.000197	11320	16957	45%	12.2	63445	36890	58%	13	0.21	0.000189	13942	19689	53%
132	50000	34.40	162141	0.000212	101749	63%	17.33	84633	50630	60%	10	0.23	0.000202	12486	17827	36%	15.1	78403	47847	61%	14	0.32	0.000189	16413	24176	49%
133	75000	41.91	202353	0.000207	130895	65%	16.31	83151	50304	60%	9	0.32	0.000192	13180	19650	26%	16.3	84409	52438	62%	13	0.41	0.000188	20793	29278	39%
134	25000	1.59	33317	0.000048	3954	18%	0.97	15101	1016	7%	9	0.11	0.000057	3517	5125	21%	0.8	13912	1026	7%	12	0.13	0.000049	4809	6520	26%
135	25000	6.89	62645	0.000110	27038	43%	3.82	34121	13183	39%	9	0.10	0.000109	6283	9068	36%	3.42	32234	12855	40%	11	0.11	0.000103	7927	11379	45%
136	25000	34.84	100026	0.000348	63243	63%	23.94	78896	45576	58%	13	0.15	0.000302	12504	16919	68%	23.6	78311	45446	58%	17	0.18	0.000299	14396	19330	77%
137	25000	41.85	94274	0.000444	63247	67%	30.42	73922	46660	63%	14	0.15	0.000409	15217	18783	75%	27.5	68012	44173	65%	15	0.16	0.000402	16162	20333	81%
138	25000	27.10	98095	0.000276	31275	32%	36.05	140842	28895	21%	16	0.18	0.000255	12178	19462	78%	27.6	114040	25601	22%	19	0.20	0.000240	13909	21492	86%
139	25000	28.01	97192	0.000288	41682	43%	23.51	98983	32457	33%	15	0.17	0.000236	10516	15697	63%	19.3	51653	28788	35%	18	0.20	0.000234	12341	18006	72%
140	25000	26.04	97465	0.000267	48605	50%	17.48	76179	32392	43%	13	0.15	0.000227	9855	14340	57%	15.6	73000	31636	43%	18	0.20	0.000211	11963	17624	70%
141	25000	17.86	87688	0.000204	55728	64%	9.43	49650	30743	62%	12	0.14	0.000187	9101	12816	51%	8.04	44157	27960	63%	13	0.14	0.000179	10133	14353	57%
142	25000	16.41	83788	0.000196	54897	66%	8.37	46259	29978	65%	11	0.12	0.000178	9162	13222	53%	7.92	45795	29849	65%	15	0.18	0.000169	11019	15714	63%
143	25000	11.40	63980	0.000178	45360	71%	6.66	36887	26184	71%	13	0.15	0.000176	8821	12344	49%	5.76	34287	24609	72%	14	0.16	0.000163	9807	13948	56%
144	25000	19.67	85926	0.000229	50168	58%	9.11	47545	26629	56%	12	0.13	0.000189	9218	13242	53%	9.75	50309	28230	56%	16	0.16	0.000191	11093	15835	63%
145	25000	22.25	93862	0.000237	53933	57%	13.35	61244	32023	52%	11	0.14	0.000216	9637	13863	55%	12.4	59210	32359	55%	15	0.16	0.000207	11382	15885	64%
146	25000	22.54	95240	0.000237	55376	58%	12.74	60743	32175	53%	13	0.15	0.000207	9466	13543	54%	10.1	50885	27535	54%	15	0.17	0.000194	10561	15514	62%
147	25000	99.27	232576	0.000427	22706	10%	138.21	307094	25204	8%	312	6.96	0.000427	5458	24406	98%	107	234501	18665	8%	318	6.97	0.000425	5819	24928	100%
148	25000	74.80	179625	0.000416	27837	15%	88.28	213212	32138	15%	210	3.81	0.000396	5499	20983	84%	70.3	175080	24957	14%	216	3.85	0.000380	6049	24234	97%
149	25000	18.58	88270	0.000210	55515	63%	12.06	63193	38052	60%	12	0.15	0.000188	9605	14308	57%	11.4	61514	38083	62%	15	0.18	0.000183	11362	17222	69%
150	25000	18.12	89489	0.000202	57289	64%	12.23	64689	40514	63%	13	0.16	0.000187	9639	14107	56%	11.1	60221	38656	64%	16	0.18	0.000181	10991	17351	69%
High	75330	99.3	243572	0.000444	132917	71%	138.21	307094	64575	71%	312	6.96	0.000427	17749	28319	98%	107	234501	53367	72%	318	6.97	0.000425	23921	34384	100%
Low	12500	1.6	33317	0.000048	3951	10%	0.97	15101	1016	5%	8	0.07	0.000057	3517	5125	21%	0.8	13912	1026	5%	11	0.09	0.000049	4809	6520	26%
Mean	32668	33	113413	0.000275	54115	49%	18.76	69211	25640	40%	23.5	0.41	0.000242	9718	15046	52%	16.1	61892	24077	41%	27.1	0.45	0.000234	11696	17033	66%
Median	25000	28.1	97780	0.000268	53333	52%	13.27	57459	23912	40%	12	0.17	0.000240	9566	14208	51%	12.2	52761	20274	41%	15.5	0.2	0.000234	11334	17049	61%
STD	16966	20	49925	0.000075	27308	15%	21.13	47439	14678	17%	30.6	1.07	0.000064	2586	4140	17%	16.1	36902	13518	17%	51.2	1.07	0.000064	3334	5111	17%

When we exclude the outliers (Problems 147 and 148 ; both transshipment problems having arcs with negative costs), there is only one instance where the solution CPU time of MINIC is superior to that of PAPANET1 and PAPANET2. Problem 138 is tightly capacitated : the maximum capacity is limited to 50 which is very small compared to the problem's total supply (250,000). Tightly capacitated problems require many arcs to be nonbasic at their upper bounds in an optimal solution and, as a result, a large proportion of the arcs must enter the basis throughout the solution process. For Problem 138, 78% and 86% of the arcs are added to the RNP at least once by PAPANET 1 and PAPANET 2, respectively. [Figure 2] contains the plot of the solution CPU times versus different values of maximum capacities (all other parameters are constant) for runs of the three codes solving Problem 138. [Figure 2] indicates that both PAPANET 1 and PAPANET 2 are superior to MINIC when the maximum capacity is greater than or equal to 200 (0.08% of the total supply).

In <Table 4>, we summarize a comparison of the computational results of the 50 medium-scale problems. For example, PAPANET2 solved Problem 108 with 32% (1/3.09) of the pivots required by MINIC and an overall time saving of 78% ([1-(1/4.61)]) over MINIC. On average, PAPANET1 requires 1.85 times fewer pivots and 2.07 times less solution CPU time than is required by MINIC. By avoiding the long-tail-of-convergence, PAPANET 2 achieves a 10% savings in the number of pivots and a 15% savings in the solution CPU time over PAPANET1; on average, PAPANET 2 requires 43% of the solution CPU time, and 50% of the pivots, of MINIC. <Table 4> also indicates that reducing the working problem size decreases the CPU time per pivot. PAPANET 2 requires a slightly larger



[Figure 2] Plot of solution CPU time versus maximum capacity for problem 138

〈Table 4〉 Comparison of The Computational Results

Problem No	CPU Time			No. Prvots			Per Pivot time		
	MINIC/ PAPA. 1	MINIC/ PAPA. 2	PAPA.1/ PAPA. 2	MINIC/ PAPA. 1	MINIC/ PAPA. 2	PAPA.1/ PAPA. 2	MINIC/ PAPA. 1	MINIC/ PAPA. 2	PAPA.1/ PAPA. 2
101	2.93	2.91	0.99	2.41	2.32	0.97	0.81	0.78	0.97
102	2.19	2.45	1.12	1.58	1.89	1.20	0.71	0.76	1.07
103	1.75	1.70	0.97	1.27	1.32	1.03	0.72	0.76	1.05
104	1.79	1.92	1.07	1.89	2.00	1.06	0.99	0.96	0.97
105	3.06	3.46	1.13	2.74	2.94	1.07	0.88	0.83	0.94
106	1.37	1.64	1.20	1.34	1.54	1.15	0.97	0.93	0.96
107	3.49	3.96	1.13	2.87	3.14	1.10	0.81	0.78	0.96
108	4.51	4.61	1.02	3.07	3.09	1.01	0.67	0.65	0.98
109	3.67	4.16	1.13	3.33	3.50	1.05	0.89	0.82	0.92
110	1.48	1.92	1.29	1.41	1.72	1.22	0.94	0.88	0.94
111	3.24	3.83	1.18	2.78	3.10	1.12	0.84	0.79	0.94
112	3.35	4.18	1.25	3.07	3.43	1.12	0.90	0.80	0.89
113	3.67	4.55	1.24	3.48	4.08	1.17	0.93	0.87	0.94
114	2.86	3.14	1.10	2.40	2.41	1.00	0.82	0.74	0.90
115	2.53	2.83	1.12	2.12	2.28	1.08	0.82	0.79	0.95
116	2.04	2.77	1.35	1.80	2.21	1.23	0.87	0.78	0.90
117	1.31	1.28	0.97	1.29	1.23	0.95	0.97	0.95	0.98
118	3.04	3.50	1.15	2.62	2.89	1.10	0.85	0.81	0.95
119	3.38	3.61	1.07	2.82	2.94	1.04	0.82	0.79	0.97
120	3.68	4.08	1.11	3.27	3.37	1.03	0.87	0.80	0.92
121	1.12	1.44	1.29	0.97	1.29	1.23	0.86	0.82	0.95
122	1.73	2.08	1.20	1.60	1.75	1.09	0.91	0.82	0.90
123	1.74	2.04	1.17	1.53	1.65	1.08	0.87	0.79	0.92
124	1.66	1.92	1.15	1.46	1.63	1.11	0.87	0.84	0.96
125	1.90	2.18	1.15	1.59	1.73	1.09	0.83	0.78	0.94
126	1.21	1.39	1.15	1.10	1.25	1.13	0.91	0.88	0.98
127	1.85	2.27	1.23	1.64	1.93	1.18	0.87	0.84	0.96
128	1.76	2.10	1.20	1.68	1.86	1.11	0.95	0.87	0.92
129	1.91	2.20	1.15	1.74	1.93	1.11	0.90	0.86	0.96
130	1.17	1.40	1.20	1.05	1.17	1.12	0.89	0.83	0.93
131	2.04	2.41	1.18	1.84	2.10	1.14	0.89	0.85	0.96
132	1.99	2.28	1.15	1.92	2.07	1.08	0.95	0.89	0.93
133	2.57	2.58	1.00	2.43	2.40	0.99	0.93	0.91	0.98
134	1.64	1.98	1.21	2.21	2.39	1.09	1.19	1.02	0.85
135	1.81	2.02	1.12	1.84	1.94	1.06	0.99	0.93	0.94
136	1.46	1.48	1.01	1.27	2.28	1.01	0.87	0.86	0.99
137	1.38	1.52	1.11	1.68	1.39	1.09	0.92	0.91	0.98
138	0.75	0.98	1.31	0.70	0.86	1.24	0.92	0.87	0.94
139	1.19	1.45	1.22	0.98	1.19	1.21	0.82	0.81	0.99
140	1.49	1.67	1.12	1.28	1.34	1.04	0.85	0.79	0.93
141	1.89	2.22	1.17	1.77	1.99	1.12	0.92	0.88	0.96
142	1.96	2.07	1.06	1.81	1.83	1.01	0.91	0.86	0.95
143	1.71	1.98	1.16	1.73	1.87	1.08	0.99	0.92	0.93
144	2.16	2.02	0.94	1.81	1.71	0.95	0.83	0.83	1.01
145	1.67	1.79	1.08	1.53	1.59	1.03	0.91	0.87	0.96
146	1.77	2.24	1.27	1.57	1.87	1.19	0.88	0.82	0.94
147	0.72	0.93	1.30	0.76	0.99	1.31	1.00	1.00	0.99
148	0.85	1.06	1.26	0.84	1.03	1.22	0.95	0.91	0.96
149	1.54	1.63	1.06	1.40	1.43	1.03	0.90	0.87	0.97
150	1.48	1.63	1.10	1.38	1.49	1.07	0.92	0.89	0.97
High	4.51	4.61	1.35	3.48	4.08	1.31	1.19	1.02	1.07
Low	0.72	0.93	0.94	0.70	0.86	0.95	0.67	0.65	0.85
Mean	2.07	2.35	1.15	1.85	2.00	1.10	0.89	0.85	0.95
Median	1.80	2.08	1.15	1.71	1.87	1.09	0.89	0.84	0.95
STD	0.86	0.96	0.09	0.70	0.74	0.08	0.08	0.07	0.04
Averages									
Transportation(20)	2.77	3.13	1.13	2.38	2.57	1.08	0.85	0.81	0.95
Transshipment(30)	1.60	1.83	1.16	1.49	1.63	1.11	0.91	0.87	0.95
12,500 arcs(4)	1.48	1.72	1.16	1.39	1.53	1.10	0.94	0.89	0.95
25,000 arcs(31)	1.76	1.98	1.14	1.58	1.72	1.10	0.89	0.85	0.96
37,500 arcs(5)	2.73	3.20	1.17	2.35	2.63	1.13	0.85	0.81	0.95
50,000 arcs(5)	3.00	3.36	1.14	2.51	2.68	1.07	0.86	0.80	0.94
75,000 arcs(5)	3.10	3.51	1.13	2.85	3.05	1.07	0.90	0.85	0.94

number of arcs than PAPANET1 does. However, the time per pivot required by PAPANET 2 is slightly less than that required by PAPANET 1. Recall that as the basis evolves through pivoting, the basis tree becomes narrower and more vertical, and the basis tree requires more update time. When the improvement in the objective value is negligible for many pivots, the tree is relatively "fully grown" and requires much more time per pivot than is required in the early stage of the solution process.

<Table 4> also contains the average ratios of the code measures for each type of problem (transportation and transshipment) as well as those for each problem size. The results show that the new algorithm implementations, PAPANET1 and PAPANET 2, are more efficient on solving transportation problems than on solving transshipment problems; and that the efficiency increases as the problem size (number of arcs) increases. Obviously, the density of a problem increases as the number of arcs increases. The density of a transportation problem is higher than that of a transshipment problem when they have the same number of arcs. Therefore, the result implies that the computational efficiency of PAPANET increases as the density increases. Problems 112 and 132 have similar parameters, i.e., number of nodes, number of arcs, arc costs, total supply, and capacity, except for the types of the problems. PAPANET 1 and PAPANET 2 are 3.35 and 4.18 times, respectively, faster than MINIC in solving transportation Problem 112, while they are 1.99 and 2.28 times, respectively, faster than MINIC in solving transshipment Problem 132. For both PAPANET 1 and PAPANET 2, the efficiencies in solving transportation problems exceeds the efficiencies in solving transshipment problems.

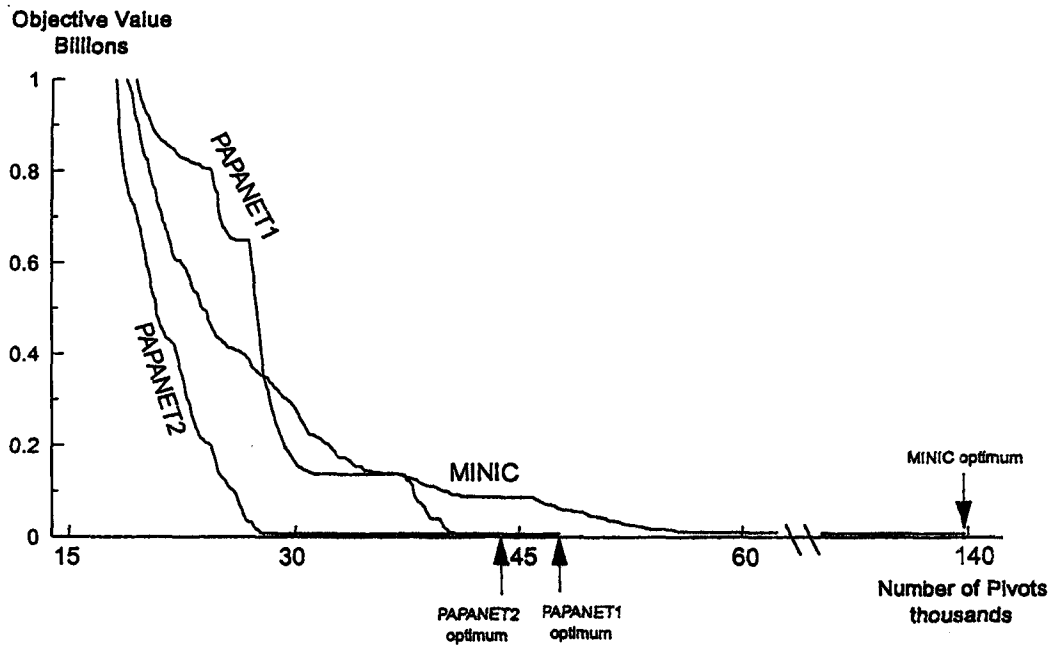
<Table 5> shows the statistical results of paired t-test for the three methods. For the solution CPU time, we can conclude that PAPANET1 is faster than MINIC and PAPANET 2 is faster than PAPANET 1. In the case of the number of pivots, we can conclude that PAPANET 1 requirs fewer pivots than is required by MINIC. However, there is no statistical evidence to conclude that the number of pivots required by PAPANET 1 and PAPANET 2 are different.

The convergence behavior of the three algorithms is shown in [Figure 3] by plotting the objective values versus the number of pivots for Problem 107. The plots clearly show the efficiencies of PAPANET 1 and PAPANET 2 over MINIC. Note that PAPANET 2 converges very quickly to a certain point (a near optimal solution to the original problem) and then moves slowly

<Table 5> Statistical Results

	DIFFERENCE	MEAN	STDDEV	DF	t	p-value (two-sided)
CPU Time	MINIC - PAPANET 1	14.21	16.26	49	6.177	0.000**
	MINIC - PAPANET 2	16.91	14.30	49	8.361	0.000**
	PAPANET 1 - PAPANET 2	2.70	5.02	49	3.807	0.000**
No. Pivots	MINIC - PAPANET 1	44202.76	44720.42	49	6.989	0.000**
	MINIC - PAPANET 2	45721.82	60205.60	49	5.370	0.000**
	PAPANET 1 - PAPANET 2	1519.06	42780.37	49	.251	0.803

** $p < 0.01$



[Figure 3] Plot of objective value versus pivots for problem 107

toward an optimum. For some large applications in which only an approximate solution is needed, PAPANET 2 may be applied effectively. Since PAPANET 2 does not solve the RNP to optimality before probing, the slope of PAPANET 2 does not change too rapidly. Therefore, it is not readily evident when the PROBEs are performed. However, for PAPANET 1, the PROBEs can be easily detected at the points where the slope changes dramatically. PAPANET 1 has a somewhat longer tail of convergence than PAPANET 2 in solving each RNP.

5.2 Large-Scale Problems

The Large-scale Problem Set C contains five randomly generated one million arc problems (two transportation and three transshipment problems) designed by Barr and Hickman [6]. Each problem was solved by MINIC, PAPANET1 and PAPANET2. The computational results along with the parameters of each problem are provided in <Table 6>.

The average improvements in solution CPU time of PAPANET1 and PAPANET 2 over MINIC are 5.51 and 6.01, respectively. As for the medium-scale problems, PAPANET 2 saves an average of 10% of the solution CPU time over PAPANET 1. For Problem 3, PAPANET 2 is 11.01 times faster and requires 84% fewer pivots than does MINIC. For the problem, PAPANET2 utilized 12.41% of the total arcs, spent 9.18 seconds (2.1% of the total solution CPU time) performing 18 PROBES (2 more than PAPANET 1), and obtained an optimum 13% faster than PAPANET 1. On four of the five problems, the comparison ratios of the CPU time far exceed the ratios of the number of pivots, which implies that the time per pivot

〈Table 6〉 Problem Set C Characteristics and Computational Results

Problem Characteristic*	Problem 1	2	3	4	5	Average
Type**	TP	TS	TP	TS	TP	
Nodes	10,000	20,000	20,000	50,000	50,000	30,000.00
Sources	5,000	4,000	10,000	10,000	25,000	10,800.00
Sinks	5,000	4,000	10,000	10,000	25,000	10,800.00
Arcs	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000.00
Supply	2,500,000	2,500,000	10,000,000	10,000,000	2,500,000	5,500,000.00
Cost Range	1-100	1-100	1-100	1-10000	1-100	1-2080
CapacityRange	1-1000	1-1000	1-1000	1 500	1-1000	1-900
%Capacitated	100	100	0	100	100	80
Seed	13,502,460	75,578,374	13,502,460	63,491,741	13,450,451	35,905,097.20
MINIC						
CPU Time	937.66	4,476.10	4,812.71	48,992.79	46,851.19	21,214.09
No. Pivots	1,059,019	2,618,272	1,613,179	7,030,478	5,687,674	3,601,724
Per Pivot Time	0.00089	0.00171	0.00298	0.00697	0.00824	0.00416
PAPANET 1						
CPU Time	198.28	727.61	492.48	19,999.27	10,503.78	6,384.28
No. Pivots	221,579	599,400	289,582	4,330,593	1,972,711	1,482,773
Per Pivot Time	0.00076	0.00120	0.00167	0.00461	0.00532	0.00271
No. PROBE	22	13	16	24	20	18.80
PROBE Time	30.21	6.84	8.60	13.93	13.31	14.58
Arcs Entered	60,361	96,044	109,615	240,764	190,112	139,379
PAPANET 2						
CPU Time	172.52	812.01	436.93	17,062.12	8,961.19	5,488.95
No. Pivots	201,915	606,988	264,124	3,370,484	1,612,814	1,211,265
Per Pivot Time	0.00070	0.00132	0.00162	0.00505	0.00555	0.00285
No. PROBE	27	19	18	39	29	26.45
PROBE Time	32.14	9.08	9.18	24.41	15.69	18.10
Arcs Entered	100,845	145,611	124,064	299,544	236,399	181,293
			Comparisons			
CPU Time						
MINIC/PAPA.1	4.73	6.15	9.77	2.45	4.46	5.51
MINIC/PAPA.2	5.45	5.51	11.01	2.87	5.23	6.01
PAPA1/PAPA2	1.15	0.90	1.13	1.17	1.17	1.10
No.Pivots						
MINIC/PAPA.1	4.78	4.37	5.57	1.62	2.88	3.84
MINIC/PAPA.2	5.25	4.31	6.11	2.09	3.53	4.26
PAPA1/PAPA2	1.10	0.99	1.10	1.28	1.22	1.14
Per Pivot Time						
MINIC/PAPA.1	1.17	1.42	1.79	1.51	1.55	1.49
MINIC/PAPA.2	1.27	1.29	1.84	1.38	1.49	1.45
PAPA1/PAPA2	1.09	0.91	1.03	0.91	0.96	0.98

* In all cases, the number of transshipmen sources and sinks, and percent high cost are 0.
** TP = transportation, TS = transshipment.

decreased substantially as the working problem size was reduced. On average, PAPANE1 used 14% of the total arcs and obtained a 33% saving of CPU time per pivot over MINIC. It is difficult to say that either PAPANET 1 or PAPANET 2 requires more or less time per pivot than the other. However, as far as the overall solution CPU time is concerned. PA-

PANET2 is generally superior to PAPANET1. In only one case, Problem 2, did PAPANET1 outperform PAPANET2, and only by 10% in terms of solution CPU time. Overall, as with the results for solving medium-scale problems <Table 6> also indicates that the efficiencies of the new algorithms increase with an increase in the problem size.

6. Summary and Conclusions

We have presented a new algorithm, the Pivot and Probe Algorithm, for solving minimum cost, capacitated, network flow problems. Two implementations, PAPANET1 and PAPANET2 were developed and tested. Computational experience on randomly generated network flow problems clearly indicates that the new method can substantially reduce the working problems size, the number of pivots, the CPU time per pivot, and the total solution CPU time required to solve a network flow problem. We also have shown that PAPANET 2 can avoid the long-tail-of-convergence effect, by probing when a near optimal solution to each relaxed network flow problem is obtained, and thereby obtain an extra 10~15% savings in solution CPU time. The test results also imply that the efficiency of the new method increases as the problem size increases. Any network simplex-based code can be enhanced by incorporating the Pivot and Probe Algorithm. In the future, we intend to apply the procedure to some specialized network flow problems and to explore additional sensitivity issues on medium- and large-scaled problems. To prove the effectiveness, it is essential to compare the method with the other effective methods, such as steepest edge rule.

REFERENCES

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

[2] Aronson, J.E., "A Survey of Dynamic Network Flows," Annals fo operations Research, 20(1989), pp.1-66.

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

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

[5] Barr, R.S., Fred Glover and D. Klingman, "Enhancements of Spanning Tree Labelling Procedures for Network Optimization," Infor., 17, 1(1970), pp.16-34.

[6] Barr, R.S. and B.A, Hickman, "Parallel Simplex for Large Pure Network Problem : Computational Testing and Sources of Speedup," Operations Research, 42, 1(1994), pp. 65-80.

[7] Bazaraa, M.S., J.J. Jarvis ans H.D. Sherali, Linear Programming and Network Flows, 2nd ed., John Wiley & Sons, New York, NY, 1990.

[8] Bertsekas, D.P., Linear Network Optimization : Algorithms and Codes, The MIT Press, Cambridge, MA, 1991.

[9] Evans, J.R. and E. Minieka, Optimization

Algorithms for Networks and Graphs, 2nd ed., Marcel Dekker, Inc., New York, NY, 1992.

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

[11] Glover, F., D. Klingman and N.V. Phillips, Network Models in Optimization and their Applications in Practice, John Wiley & Sons, New York, NY, 1992.

[12] Ho, J.K. and E. Loute, "Computational Experience with Advanced Implementation of Decomposition Algorithms," Mathematical Programming, 29, 3(1983), pp.283-290.

[13] Klingman, D. and J. Mote, "Computational Analysis of Large-Scale Pure Networks," Presented at the Joint National Meeting of ORSA/TIMS, October, 1987.

[14] Klingmanm, D., A. Napier and J. Stutz, "NETGEN : A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," Management Science, 20, 5(1974), pp.814-821.

[15] Mathies, S. and P. Mevert, "A Hybrid Algorithm for Solving Network Flow Problems with Side Constraints," Computers and Operations Research, 25, 9(1998), pp.745-756.

[16] Mateus, G.R., H.P. Luna and A.B. Sirihal, "Heuristics for Solving Distribution Network Design in Telecommunication," Journal of Heuristics, 6, 1(2000), pp.131-148.

[17] Melkote, S. and M.S. Daskin, "Capacitated Facility Location/Network Design Problems," European Journal of Operational Research, 129, 3(2001), pp.481-495.

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

[19] Sethi, A.P. and G.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.

[20] Sethi, A.P. and G.L. Thompson, "The Pivot and Probe Algorithm for Solving a Linear Program," Mathematical Programming, 29, 2(1984), pp.219-233.

[21] Sethi, A.P., G.L. Thompson and M.S. Hung, "An Efficient Simplex Pricing Procedure," Working Paper No.1990-42, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1990.

[22] Simonnard, M.A., Linear Programming, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[23] Sun, M., "MINIC : MINImum Cost Network Flow Code," Department of Management Sciences and Information Technology, Terry College of Business Administration, The University of Georgia, Athens, GA, 1990.

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

[25] Xiong, D., "A Three-Stage Computational Approach to Network Matching," Transportation Research, 8C, 1(2000), pp.71-89.

[26] Zhang, J. and Z. Liu, "A Further Study on Inverse Linear Programming Problems," Journal of Computational and Applied Mathematics, 106, 2(1999), pp.345-359.