

Technical Report 01-EMIS-02

**Generalized Networks: The Theory of  
Preprocessing and an Empirical Analysis**

by

Jeffery L. Kennington

and

Karen R. Lewis

{jlk,klewis}@engr.smu.edu

**Dept. of Engineering Management, Information, and Systems**

**School of Engineering**

**Southern Methodist University**

**Dallas, Texas 75275-0122**

March 2001

## **Abstract**

The idea of preprocessing a linear program with the goal of eliminating constraints and/or columns is well established. The default setting for CPLEX is to apply preprocessing prior to application of any of its four solvers. In this investigation, a set of preprocessing procedures is developed for the special case of a generalized network flow problem. Some procedures involve adding and/or replacing arcs in order to eliminate a node. These procedures all produce an optimal basic feasible solution and optimal dual solution to the original problem. In an empirical analysis, our software implementation resulted in a 20% improvement in solution time compared to the best results using CPLEX 6.5.3.

acknowledgement: This work was partially supported by the Office of Naval Research under Award No. N00014-96-1-0315.

note: The latest version of this technical report can be found at [www.engr.smu.edu/~jlk/publications/01-EMIS-02.ps](http://www.engr.smu.edu/~jlk/publications/01-EMIS-02.ps)

keywords: generalized networks, preprocessing, presolve

# 1 INTRODUCTION

Let  $P$  denote the linear program  $\min\{cx: Ax = r, l \leq x \leq u\}$  where  $c, l$ , and  $u$  are  $n$ -component vectors,  $A$  is an  $m \times n$  matrix,  $r$  is an  $m$ -component vector, and  $x$  is an  $n$ -component vector of unknowns. If every column of  $A$  has exactly two non-zero elements, a  $+1$  and a  $-1$ , then we call  $P$  a *pure network flow model*. If every column of  $A$  has at most two nonzero elements, then we call  $P$  a *generalized network flow model*. For both problems, there is an underlying network with nodes corresponding to the  $m$  rows of  $A$  and arcs corresponding to the  $n$  columns of  $A$ . The term generalized network is used because flow along these arcs may expand, contract, or remain constant. Hence, generalized network models are used for problems involving the time value of money (that increases over time), perishable goods (that decrease over time), and conversion of raw materials to finished goods, as well as many others. A discussion of applications may be found in Ahuja et al. [2].

The best-known algorithms for the generalized network flow problem are specializations of the simplex method (see [2, 7, 8]). For these procedures, the basis matrix is maintained as a special graphical structure and operations are performed directly on this structure. Additional exploitation of the generalized network structure should be possible by application of preprocessing techniques that attempt to reduce the size of the model prior to application of the simplex algorithm. The idea of preprocessing is to create a second

problem  $\bar{P}$ , say  $\min\{\bar{c}\bar{x} : \bar{A}\bar{x} = \bar{r}, \bar{l} \leq \bar{x} \leq \bar{u}\}$  which is smaller than  $P$ . Problem  $\bar{P}$  must be developed such that an optimal basic feasible solution  $([x_B|x_N])$  and optimal dual vector  $(\pi)$  for  $P$  can be constructed from the corresponding solutions  $([\bar{x}_B|\bar{x}_N], \bar{\pi})$  for  $\bar{P}$ . Preprocessing techniques for the generalized network problem is the subject of this investigation.

## 1.1 Reduction Methodology

In order to prove that a solution to a problem is optimal, it is sufficient to show that the primal solution is feasible, the dual solution is feasible, and the objective function values for the primal and dual solutions are equal. Given an optimal solution for  $\bar{P}$  in the form  $([\bar{x}_B|\bar{x}_N], \bar{\pi})$ , we will show how to construct a solution to  $P$   $([x_B|x_N], \pi)$  that satisfies the above three criteria. Generally, a proposition corresponding to the criteria and a figure illustrating the reduction are given for each reduction procedure. All proofs of the propositions may be found in [9].

## 1.2 Survey of the Literature

The investigations reported in this section are for the general LP. We know of no studies of the problem of preprocessing either pure or generalized network flow problems. In the mid 1970's, Brearley et al. [5] presented procedures to remove redundant constraints, replace constraints by simple bounds, re-

move or tighten bounds, and fix variables. Additional clarification may be found in Williams [15] and Tomlin and Welch [14], along with a technique for detecting duplicate rows. Klein and Holm [10] show how to identify redundant constraints by comparing constraints to identify those which must be nonbinding at optimality. Reductions for LPs with range constraints can be found in Bradley, Brown, and Graves [4]. A procedure involving row operations that attempts to reduce the density of the constraint matrix is proposed by Adler et al. [1]. Gondzio [6] has developed an alternative approach to creating a sparser matrix. A set of similar techniques may also be found in Lustig, Marsten, and Shanno [12]. Andersen and Andersen [3] provide implementation suggestions and postsolve procedures.

### 1.3 Objective of the Investigation

Preprocessing may be viewed as a set of independent procedures which can be applied to a problem in any order. Applied to a generalized network, a preprocessor attempts to eliminate nodes and/or arcs. If it is discovered that the flow on arc  $j$  can be fixed or modeled as a linear combination of other arcs in the problem, then arc  $j$  can be eliminated. If the network equivalent of variable substitution can be implemented, an arc and a node can be eliminated. If the preprocessing routines are fast and the second model is substantially smaller than the first, then the potential exists for

computational savings. In addition, it may be feasible to eliminate a node by adding an arc. Since the size of the basis depends on the number of nodes, this can also reduce solution time.

The objective of this investigation is to develop the methodology for preprocessing generalized network models and to empirically evaluate this methodology. The strategy is to exploit the network structure to produce algorithms superior to the generic linear programming methods which appear in the literature and those implemented in CPLEX.

## 2 REMOVING ARCS AND NODES

For a generalized network model, each node has a right-hand-side value  $r_i$  for  $i = 1, \dots, m$ . There are seven entities associated with each arc  $j$  where  $j = 1, \dots, n$ . They include the from node ( $f_j$ ), the to node ( $t_j$ ), the unit cost ( $c_j$ ), the lower bound ( $l_j$ ), the upper bound ( $u_j$ ), the multiplier at the from node ( $a_j$ ), and the multiplier at the to node ( $b_j$ ). We abuse notation by allowing  $l_j$  ( $u_j$ ) to be  $-\infty$  ( $\infty$ ), implying that an arc is unbounded below (above). Further, some arcs, called root arcs, are incident to a single node. They can be implemented by setting  $b_j$  to 0 and  $t_j$  to any node.

## 2.1 Adaptation for Networks

Many of the preprocessing techniques from the literature can be adapted to the special case of a generalized network model. Trivial reductions, such as equal or infeasible lower and upper bounds on an arc, and duplicate arcs can result in the elimination of an arc. Singleton and doubleton constraint methods can eliminate an arc and a node with degree one or two from the problem. Forcing constraints can be used to set arc flow values. An implied free arc can be removed from the problem by substitution, using the node constraint equation that frees its bounds. It is possible that after eliminating nodes and arcs, the remaining network may consist of a single node with multiple adjacent arcs. Another possibility is that an arc may be removed from the network, isolating a subnetwork with only one node. These single-node networks can be solved using a greedy approach for a single-constraint linear program. The details of these procedures can be found in [11].

## 2.2 Arcs Between Root Arcs

Suppose there exists an arc between two nodes, each of which is incident to a root arc. Label such an arc  $n$ , and the root arcs 1 and 2, as illustrated in Figure 1. If arcs 1 and 2 are unbounded above,  $a_n/a_1 > 0$ ,  $b_n/a_2 > 0$ , and  $c_n > c_1a_n/a_1 + c_2b_n/a_2$ , then arc  $n$  can be set to its lower bound and removed from the problem, as illustrated in Figure 1b. Let  $\bar{P}$  be constructed

as follows: 1) drop arc  $n$ , 2) set  $\bar{r}_1 = r_1 - a_n l_n$ , and 3) set  $\bar{r}_2 = r_2 - b_n l_n$ .

**Proposition 1** *In any optimum for  $P$ ,  $x_n = l_n$ .*

### 3 TECHNIQUES THAT ADD ARCS

In the remaining reduction techniques, an arc is added or arcs are replaced to eliminate a node. In the first method, an arc is added to remove a node of degree three and an adjacent arc. In the subsequent method, four new arcs replace all of the arcs adjacent to a node of degree four. An additional arc may be needed to eliminate that node.

#### 3.1 Nodes Having Degree Equal Three

If a node of degree three cannot be removed by substitution of an implied free variable, then a reduction may still be possible by adding an arc to the problem. Label this node  $m$ , and let the incidence arc from node  $m - 1$  be arc  $n$ , and the incidence arcs to nodes 1 and 2 be arcs 1 and 2, respectively, as illustrated in Figure 2. If each of the arcs is unbounded above, and the following conditions are satisfied

$$a_1, a_2 > 0, \quad b < 0, \quad l_n = 0, \quad r_m > 0, \quad \text{and} \quad a_1 l_1 + a_2 l_2 < r_m$$

then problem  $\bar{P}$  can be constructed as follows, as illustrated in Figure 3:

1) drop node  $m$ ; 2) drop arc  $n$ ; 3) add arc  $\bar{n}$  from node 1 to node 2, with

$\bar{a}_n = -a_2b_1$ ,  $\bar{b}_n = a_1b_2$ ,  $\bar{c}_n = c_2a_1 - c_1a_2$ ,  $\bar{l}_n = l_2/a_1$ , and  $\bar{u}_n = (r_m - a_1l_1)/a_1a_2$ ;  
 4) set  $\bar{r}_1 = r_1 - r_mb_1/a_1$ ; and 5) set  $\bar{a}_i = -aa_i/b$ ,  $\bar{c}_i = c_i - ca_i/b$ , and  $\bar{l}_i = 0$   
 for  $i = 1, 2$ . Let  $\bar{x} = [\bar{x}_1 | \bar{x}_2 | \hat{x} | \bar{x}_n]$  and  $\bar{\pi}$  be optimal for  $\bar{P}$ .

**Proposition 2** *If  $x = [r_m/a_1 + \bar{x}_1 - a_2\bar{x}_n | \bar{x}_2 + a_1\bar{x}_n | \hat{x} | -(a_1\bar{x}_1 + a_2\bar{x}_2)/b]$  and  $\pi = [\bar{\pi} | \pi_m] = [\bar{\pi} | (c_1 - b_1\bar{\pi}_1)/a_1]$ , then  $x$  is feasible for  $P$ ,  $\pi$  is feasible for the dual of  $P$ , and  $\pi r = cx$ .*

Construction of an optimal basis is not as straightforward as in previous reductions. Since there is one more node in  $P$  than in  $\bar{P}$ , there must be one additional arc in the basis of  $P$ . But in calculating the values of  $x_1$ ,  $x_2$ , and  $x_n$ , there may be more than one additional arc with flow between bounds. It is well known that the connected components of a basis for a generalized network are either one-trees or rooted trees [2, 8, 13]. For any cycle in a generalized network, the nodes and arc multipliers may be renamed as in Figure 4. Let  $w_i = -b_i/a_i$ , and let the cycle factor be defined as  $1 - w_1w_2 \cdots w_p$ . A set of arcs is linearly independent if and only if the cycle factor is not 0. In the reduction of a degree three node, the multipliers of arc  $n$  are defined in such a way that the cycle formed by arcs 1, 2, and  $n$  always has a cycle factor of 0. Therefore, these three arcs cannot all be basic in the reduced network. In this case, create a candidate list  $C$  of basic arcs for  $P$ . The candidate list contains arcs 1, 2, and  $n$ , as well as any other arcs that are basic in  $\bar{P}$ . There is a cycle of linearly dependent arcs in the candidate list.

By sending flow through the cycle in order to force an arc to a bound, an arc can be removed from the basis. Choose an arc closest to one of its bounds, and send flow through the cycle in the amount and direction appropriate to put this arc at that bound.

Figure 5 illustrates the flow for  $\bar{P}$  in a pure network that requires this technique. Figure 6 indicates the cycle formed in  $P$  using the equations for  $x$ . The arc (2,3) has flow value closest to its lower bound. By sending one unit of flow through the cycle in the direction opposite to that of the arc, new flow values are obtained. Arc (2,3) can be removed from the candidate list at its lower bound, as illustrated in Figure 7.

A similar node of degree three can also be eliminated. If each of the arcs is unbounded above, and the following conditions are satisfied

$$a_1, a_2 < 0, \quad b > 0, \quad l_n = 0, \quad r_m < 0, \quad \text{and} \quad a_1 l_1 + a_2 l_2 > r_m$$

then problem  $\bar{P}$  can be constructed as above, except for changes to the characteristics of the added arc. In this case, add arc  $\bar{n}$  from node 2 to node 1, with  $\bar{a}_n = -a_1 b_2$ ,  $\bar{b}_n = a_2 b_1$ ,  $\bar{c}_n = c_1 a_2 - c_2 a_1$ ,  $\bar{l}_n = -l_2/a_1$ , and  $\bar{u}_n = (a_1 l_1 - r_m)/a_1 a_2$ , as illustrated in Figure 8. Let  $\bar{x} = [\bar{x}_1 \mid \bar{x}_2 \mid \hat{x} \mid \bar{x}_n]$  and  $\bar{\pi}$  be optimal for  $\bar{P}$ .

**Proposition 3** *If  $x = [r_m/a_1 + \bar{x}_1 + a_2 \bar{x}_n \mid \bar{x}_2 - a_1 \bar{x}_n \mid \hat{x} \mid - (a_1 \bar{x}_1 + a_2 \bar{x}_2)/b]$  and  $\pi = [\bar{\pi} \mid \pi_m] = [\bar{\pi} \mid (c_1 - b_1 \bar{\pi}_1)/a_1]$ , then  $x$  is feasible for  $P$ ,  $\pi$  is feasible for the dual of  $P$ , and  $\pi r = cx$ .*

### 3.2 Nodes Having Degree Equal Four

Suppose there exists a node whose degree is four. Under specific conditions, this node can be removed. Without loss of generality, this node may be labeled  $m$ , as illustrated in Figure 9. Let the incident arcs associated with nodes 1, 2, 3, and 4 be  $x_1, x_2, x_3$ , and  $x_4$ , respectively.

Suppose that the following conditions hold:

$$b_1, b_2 < 0, \quad a_3, a_4 > 0, \quad r_m = 0, \quad l_i = 0, u_i = \infty \quad \forall \quad i = 1, \dots, 4$$

Then  $\bar{P}$  can be constructed as follows: 1) drop node  $m$ ; 2) drop  $x_1, \dots, x_4$ ; 3) add  $\bar{x}_1$  from node 1 to node 3 with  $\bar{a}_1 = -a_1/b_1$ ,  $\bar{b}_1 = b_3/a_3$ ,  $\bar{c}_1 = -c_1/b_1 + c_3/a_3$ ,  $\bar{l}_1 = 0$ , and  $\bar{u}_1 = \infty$ ; 4) add  $\bar{x}_2$  from node 1 to node 4 with  $\bar{a}_2 = -a_1/b_1$  and  $\bar{b}_2 = b_4/a_4$ , and  $\bar{c}_2 = -c_1/b_1 + c_4/a_4$ ,  $\bar{l}_2 = 0$ , and  $\bar{u}_2 = \infty$ ; 5) add  $\bar{x}_3$  from node 2 to node 3 with  $\bar{a}_3 = -a_2/b_2$ ,  $\bar{b}_4 = b_3/a_3$ ,  $\bar{c}_3 = -c_2/b_2 + c_3/a_3$ ,  $\bar{l}_3 = 0$ , and  $\bar{u}_3 = \infty$ ; and 6) add  $\bar{x}_4$  from node 2 to node 4 with  $\bar{a}_4 = -a_2/b_2$ ,  $\bar{b}_4 = b_4/a_4$ ,  $\bar{c}_4 = -c_2/b_2 + c_4/a_4$ ,  $\bar{l}_4 = 0$ , and  $\bar{u}_4 = \infty$ . This is illustrated in Figure 10. Let  $\bar{x}$  and  $\bar{\pi}$  be optimal for  $\bar{P}$ .

**Proposition 4** *If  $x = [-(\bar{x}_1 + \bar{x}_2)/b_1 \mid -(\bar{x}_3 + \bar{x}_4)/b_2 \mid (\bar{x}_1 + \bar{x}_3)/a_3 \mid (\bar{x}_2 + \bar{x}_4)/a_4 \mid \hat{x}]$  and  $\pi = [\bar{\pi} \mid \max\{(c_1 - a_1\pi_1)/b_1, (c_2 - a_2\pi_2)/b_2\}]$ , then  $x$  is feasible for  $P$ ,  $\pi$  is feasible for the dual of  $P$ , and  $\pi r = cx$ .*

In this way, a node with degree four can be eliminated. The four arcs incident to the node are replaced with four new arcs, and multipliers and

costs are adjusted. Construction of an optimal basis for  $P$  requires some additional analysis. Using the basis for  $\bar{P}$ , a set of candidate basic arcs,  $C$ , for  $P$  will be constructed. Place in  $C$  all arcs from  $P$  with corresponding basic arcs in  $\bar{P}$  whose arc number is greater than four. If  $\bar{x}_1$  is basic, then  $x_1$  and  $x_3$  are placed in  $C$ . If  $\bar{x}_2$  is basic, then  $x_1$  and  $x_4$  are placed in  $C$ . If  $\bar{x}_3$  is basic, then  $x_2$  and  $x_3$  are placed in  $C$ . If  $\bar{x}_4$  is basic, then  $x_2$  and  $x_4$  are placed in  $C$ . If  $|C| = m$ , then the arcs in  $C$  correspond to an optimal basis for  $P$ . If  $|C| = m + 1$ , then some additional work is required to determine the optimal basis. As in the degree three reductions, there will exist a cycle in the candidate list through which flow can be forced to remove one arc.

If the right-hand side of constraint  $m$  or a lower bound is nonzero, node  $m$  can be removed by adding an arc. Since a problem with  $m - 1$  nodes and  $n + 1$  arcs can generally be solved more quickly than a problem with  $m$  nodes and  $n$  arcs, this substitution, while not reducing the total number of nodes plus arcs, can still reduce the time required to solve the problem.

Again, refer to Figure 9 and suppose that the following conditions hold:

$$b_1, b_2 < 0, \quad a_3, a_4 > 0, \quad l_i = 0, u_i = \infty \quad \forall \quad i = 1, \dots, 4$$

If the lower bound of 0 is not satisfied for some  $i$ , an arc substitution can be performed such that  $\tilde{x}_i = x_i - l_i$ . The lower bound on  $\tilde{x}_i$  is then 0. The right-hand side of the constraint for the from node must be adjusted by  $-a_i l_i$ , and the right-hand side of the to-node constraint is adjusted by

$-b_i l_i$ . If the conditions above hold and  $r_m < 0$ , then  $\bar{P}$  can be constructed as follows: 1) drop node  $m$ ; 2) drop  $x_1, \dots, x_4$ ; 3) add  $\bar{x}_1$  from node 1 to node 3 with  $\bar{a}_1 = -a_1/b_1$ ,  $\bar{b}_1 = b_3/a_3$ ,  $\bar{c}_1 = -c_1/b_1 + c_3/a_3$ ,  $\bar{l}_1 = 0$ , and  $\bar{u}_1 = \infty$ ; 3) add  $\bar{x}_2$  from node 1 to node 4 with  $\bar{a}_2 = -a_1/b_1$ ,  $\bar{b}_2 = b_4/a_4$ ,  $\bar{c}_2 = -c_1/b_1 + c_4/a_4$ ,  $\bar{l}_2 = 0$ , and  $\bar{u}_2 = \infty$ ; 3) add  $\bar{x}_3$  from node 2 to node 3 with  $\bar{a}_3 = -a_2/b_2$ ,  $\bar{b}_3 = b_3/a_3$ ,  $\bar{c}_3 = -c_2/b_2 + c_3/a_3$ ,  $\bar{l}_3 = 0$ , and  $\bar{u}_3 = \infty$ ; 3) add  $\bar{x}_4$  from node 2 to node 4 with  $\bar{a}_4 = -a_2/b_2$ ,  $\bar{b}_4 = b_4/a_4$ ,  $\bar{c}_4 = -c_2/b_2 + c_4/a_4$ ,  $\bar{l}_4 = 0$ , and  $\bar{u}_4 = \infty$ ; and 3) add  $\bar{x}_{n+1}$  from node 2 to node 1 with  $\bar{a}_{n+1} = -a_2 b_1$ ,  $\bar{b}_{n+1} = a_1 b_2$ ,  $\bar{c}_{n+1} = c_1 b_2 - c_2 b_1$ ,  $\bar{l}_{n+1} = 0$ , and  $\bar{u}_{n+1} = -r_m/b_1 b_2$ ; and 3) set  $\bar{r}_1 = r_1 - r_m a_1/b_1$ . This is illustrated in Figure 11. Let  $\bar{x} = [\bar{x}_1 \mid \bar{x}_2 \mid \bar{x}_3 \mid \bar{x}_4 \mid \hat{x} \mid \bar{x}_{n+1}]$  and  $\bar{\pi}$  be optimal for  $\bar{P}$ .

**Proposition 5** *If  $x = [(r_m - \bar{x}_1 - \bar{x}_2)/b_1 + b_2 \bar{x}_{n+1} \mid -(\bar{x}_3 + \bar{x}_4)/b_2 - b_1 \bar{x}_{n+1} \mid (\bar{x}_1 + \bar{x}_3)/a_3 \mid (\bar{x}_2 + \bar{x}_4)/a_4 \mid \hat{x}]$  and  $\pi = [\bar{\pi} \mid (c_1 - a_1 \pi_1)/b_1]$ , then  $x$  is feasible for  $P$ ,  $\pi$  is feasible for the dual of  $P$ , and  $\pi r = cx$ .*

If the conditions above hold and  $r_m > 0$ , then  $\bar{P}$  can be constructed as follows: 1) drop node  $m$ ; 2) drop  $x_1, \dots, x_4$ ; 3) add  $\bar{x}_1$  from node 1 to node 3 with  $\bar{a}_1 = -a_1/b_1$ ,  $\bar{b}_1 = b_3/a_3$ ,  $\bar{c}_1 = -c_1/b_1 + c_3/a_3$ ,  $\bar{l}_1 = 0$ , and  $\bar{u}_1 = \infty$ ; 4) add  $\bar{x}_2$  from node 1 to node 4 with  $\bar{a}_2 = -a_1/b_1$ ,  $\bar{b}_2 = b_4/a_4$ ,  $\bar{c}_2 = -c_1/b_1 + c_4/a_4$ ,  $\bar{l}_2 = 0$ , and  $\bar{u}_2 = \infty$ ; 5) add  $\bar{x}_3$  from node 2 to node 3 with  $\bar{a}_3 = -a_2/b_2$ ,  $\bar{b}_3 = b_3/a_3$ ,  $\bar{c}_3 = -c_2/b_2 + c_3/a_3$ ,  $\bar{l}_3 = 0$ , and  $\bar{u}_3 = \infty$ ; 6) add  $\bar{x}_4$  from node 2 to node 4 with  $\bar{a}_4 = -a_2/b_2$ ,  $\bar{b}_4 = b_4/a_4$ ,  $\bar{c}_4 = -c_2/b_2 + c_4/a_4$ ,  $\bar{l}_4 = 0$ , and  $\bar{u}_4 = \infty$ ; and

7) add  $\bar{x}_{n+1}$  from node 3 to node 4 with  $\bar{a}_{n+1} = -a_4b_3$ ,  $\bar{b}_{n+1} = a_3b_4$ ,  $\bar{c}_{n+1} = c_4a_3 - c_3a_4$ ,  $\bar{l}_{n+1} = 0$ , and  $\bar{u}_{n+1} = r_m/a_3a_4$ ; and 8) set  $\bar{r}_3 = r_3 - r_mb_3/a_3$ . This is illustrated in Figure 12. Let  $\bar{x} = [\bar{x}_1 \mid \bar{x}_2 \mid \bar{x}_3 \mid \bar{x}_4 \mid \hat{x} \mid \bar{x}_{n+1}]$  and  $\bar{\pi}$  be optimal for  $\bar{P}$ .

**Proposition 6** *If  $x = [-(\bar{x}_1 + \bar{x}_2)/b_1 \mid -(\bar{x}_3 + \bar{x}_4)/b_2 \mid (r_m + \bar{x}_1 + \bar{x}_3)/a_3 - a_4\bar{x}_{n+1} \mid (\bar{x}_2 + \bar{x}_4)/a_4 + a_3\bar{x}_{n+1} \mid \hat{x}]$  and  $\pi = [\bar{\pi} \mid (c_3 - b_3\pi_3)/a_3]$ , then  $x$  is feasible for  $P$ ,  $\pi$  is feasible for the dual of  $P$ , and  $\pi r = cx$ .*

In this way, a supply/demand node with degree four can be replaced with an arc. The four arcs incident to the node are replaced with four new arcs, and multipliers and costs are adjusted. Additionally, the right-hand side of the constraint associated with one of the nodes is adjusted to accommodate the requirement at node  $m$ .

## 4 EMPIRICAL ANALYSIS

Each of the reduction techniques has been implemented in the Generalized Network Preprocessor (GNP), written in ANSI C. After each of the techniques has been applied, CPLEX 6.5 Callable Library routines are called by GNP to solve the reduced problem. The solution to the reduced problem is then used by GNP to derive the solution to the original problem. The CPLEX solvers are used so that GNP can be compared with CPLEX Aggre-

gator and CPLEX Presolve, without confounding the effect of the solver.

GNP has been compared with CPLEX Aggregator plus CPLEX Presolve on 32 different problems. Each problem is a connected generalized network with 100,000 nodes which has been randomly created using a generalized network generator. Five factors with two levels each have been used as inputs to the generator. The first factor is the average node density. The two levels of this factor are 3, resulting in 150,000 arcs, and 7, resulting in 350,000 arcs. The second factor is the number of nodes which either provide a supply or require a demand. The two levels are 2000 nodes and 50,000 nodes, with the number being divided equally between supply and demand. The third factor is the percentage of supply/demand nodes which are transshipment nodes. The low level is 0 and the high level is 50% of the supply/demand nodes. The fourth factor is percentage of arcs with a lower bound greater than zero. The two levels are 0 and 25%. The last factor is the percentage of arcs with a finite upper bound. The low level for this factor is 0; the high factor level is 50%. The problem set represents a full factorial combination of these factors. The specific characteristics of each problem are provided in Table 1.

After creating the problems, the preprocessor was used as the sixth factor, with one level being CPLEX Aggregator plus CPLEX Presolve, and the other level being GNP. This created a set of 64 problems which were solved together

as a block, with the run order of the problems randomized. The block was run six times, twice each for three solution methods. The solvers used for comparison are the network, dual, and primal solvers. The primal solver was used for comparison because of its popularity, even though it is not the preferred solution method for solving generalized network problems. The problems were run on a Compaq Alphaserwer DS20.

The number of nodes and arcs eliminated by the preprocessors is compared in Table 2. Since GNP eliminates and adds arcs, both of these quantities are shown, as well as the net reduction. Columns 8 and 9 display the number of nodes and arcs eliminated by CPLEX Aggregator plus CPLEX Presolve on the reduced problem generated by GNP. This is for comparison of the elimination of nodes and arcs only. Neither CPLEX Aggregator nor CPLEX Presolve were used when GNP called CPLEX routines to solve the problems.

CPLEX always eliminated more arcs in these problems. Many of these are arcs with one non-zero multiplier. Such an arc can be eliminated in a linear program, effectively changing an equality constraint to an inequality. But the network structure is not retained, and it may not effectively decrease the solution time. The number of nodes eliminated is a more significant number, since the size of the basis depends on the number of nodes in the problem. In some cases, CPLEX is able to reduce more nodes, but as the average node

degree is increased, GNP outperforms in this metric.

A more important measure to a person using a network solver, however, is the amount of time required to obtain an optimal feasible solution. Table 3 gives an idea of the utility of a preprocessor, by comparing the times required to solve each of these problems using no preprocessor, CPLEX Aggregator plus CPLEX Presolve, and GNP using the network solver. The solution times are averages of the run times in the two blocks that used the network solver. When using the dual (primal) solver, the time savings was 15.3% (17.3%) for the CPLEX preprocessors and 28.3% (22.1%) for the GNP. Complete tables with these results may be found in [9]. These tables show that CPLEX Aggregator plus CPLEX Presolve may not be effective when solving generalized network problems with a large average node degree using the network solver or the dual solver. It is effective with the primal solver, but this solver does not furnish a competitive solution time.

A breakdown of the total time into presolve and solution/postsolve times is given in Table 4 for the network solver. Detailed comparison of times for the dual and primal solvers may be found in [9]. These details indicate that while the time to solve the reduced problem is sometimes shorter after using CPLEX Aggregator plus CPLEX Presolve, the time spent in reducing the problem results in a longer total time than when using GNP. This difference averages 23.3% using the network solver, 24.8% using the dual solver, and

5.8% using the primal solver.

An ANOVA analysis of the data indicates the main factors and interactions. The effect of the preprocessor and the interactions between the preprocessor and each of the other factors helps compare the preprocessors. The other main effects identify the factors with the greatest impact on solution time. From this analysis, we can conclude that the improved time of the GNP over the preprocessors of CPLEX is statistically significant. The only significant interaction is that between the preprocessor and the average node degree. From this, we can conclude that as the average node degree increases, the difference between the effectiveness of the GNP and the preprocessors of CPLEX becomes more pronounced. This is important, because the effect of the average node degree is the most significant of these factors in determining how much time is required to solve a problem. Regardless of the effect of an individual factor on solution times, the GNP yields faster average solution times for both levels of the factor.

Therefore, although CPLEX may eliminate more network components, the time spent implementing those techniques may cost more time than can be saved in solving a generalized network problem. GNP, however, is able to produce effective reductions very quickly to provide a solution to the original problem in a relatively short time.

## 5 SUMMARY AND CONCLUSIONS

This investigation addresses preprocessing a generalized network flow problem. The purpose of preprocessing is to reduce the size of a model in such a way that an optimal feasible solution to the original problem can be derived from an optimal feasible solution to the reduced problem. In order to be practical, the preprocessing and postprocessing must be efficient enough and the reduced problem small enough that the total time required to solve the original problem is reduced. Although other investigations have addressed the problem of preprocessing linear programs, they have not specifically addressed preprocessing networks.

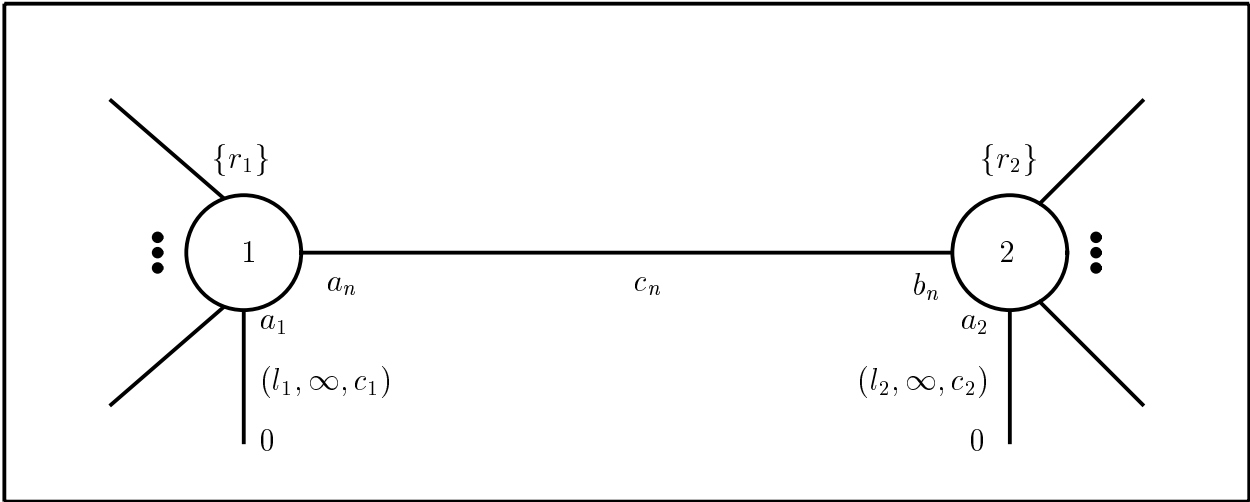
We have developed new reduction techniques and shown how an optimal solution to the original problem can be determined from an optimal feasible solution to the reduced problem. We have presented some cases in which the addition of an arc makes the elimination of a node possible.

The elimination of an arc between two root arcs applies only to generalized networks, since it requires two arcs with only one nonzero multiplier each. The remaining reduction processes apply equally well to pure networks. Each of these processes stipulate at most the sign of one of the two multipliers of each arc.

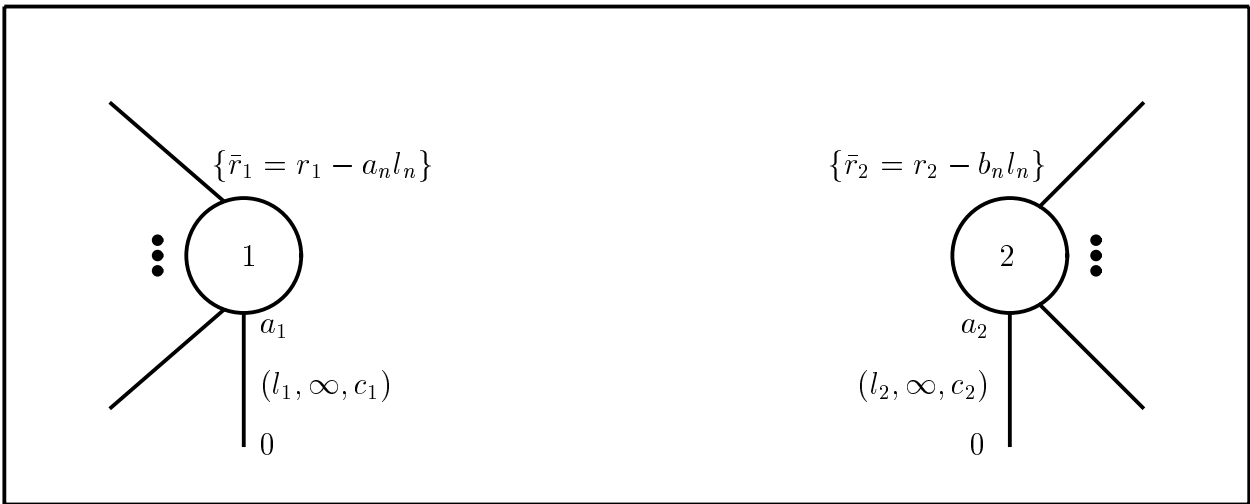
Each of these reduction techniques have been implemented in the Generalized Network Preprocessor (GNP), which preprocesses a network model,

calls routines from the CPLEX Callable Library to solve the reduced problem, then provides postprocessing to obtain an optimal feasible solution to the original problem. Three codes - GNP, CPLEX with Aggregator and Presolve, and CPLEX without Aggregator and Presolve - have been compared using 32 test problems. The problems in the experiment were designed using two levels each of five factors, including average node degree, number of supply/demand nodes, number of transshipment supply/demand nodes, number of arcs with a lower bound greater than zero, and number of arcs with a finite upper bound. Each program was used with each of three CPLEX solvers - the primal simplex, the dual simplex, and the network simplex. Each combination of program, problem, and solution method was run twice. The run order for each program and problem combination was randomized.

Each run measured the size of the reduced problem and the time to solve the original problem. Results indicate that CPLEX can reduce more arcs for any of the problems in this set. For some of the smaller problems, CPLEX also eliminated more nodes, but GNP reduced more nodes for the majority of problems. The GNP code was the fastest overall program, regardless of problem factor levels or solution method. It was on average 23.3% faster than CPLEX with Aggregator and Presolve using the network solver, 24.8% faster using the dual solver, and 5.8% faster using the primal solver. The key to its faster time is the speed of the preprocessing.



a. Arc  $n$  is Not Cost Effective



b. Removal of Arc

Figure 1: Removal of Costly Arc

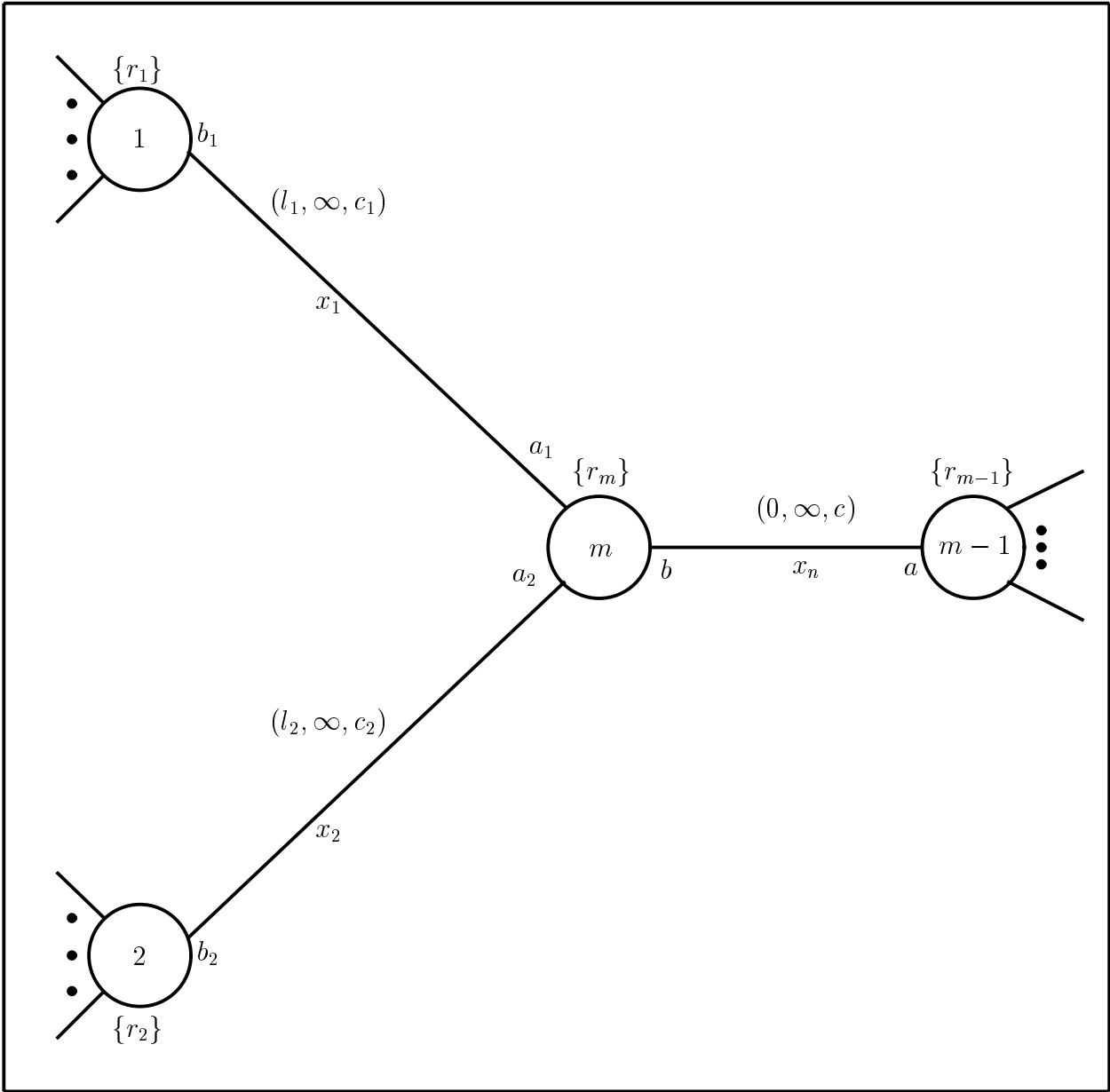


Figure 2: A Network with Node of Degree 3

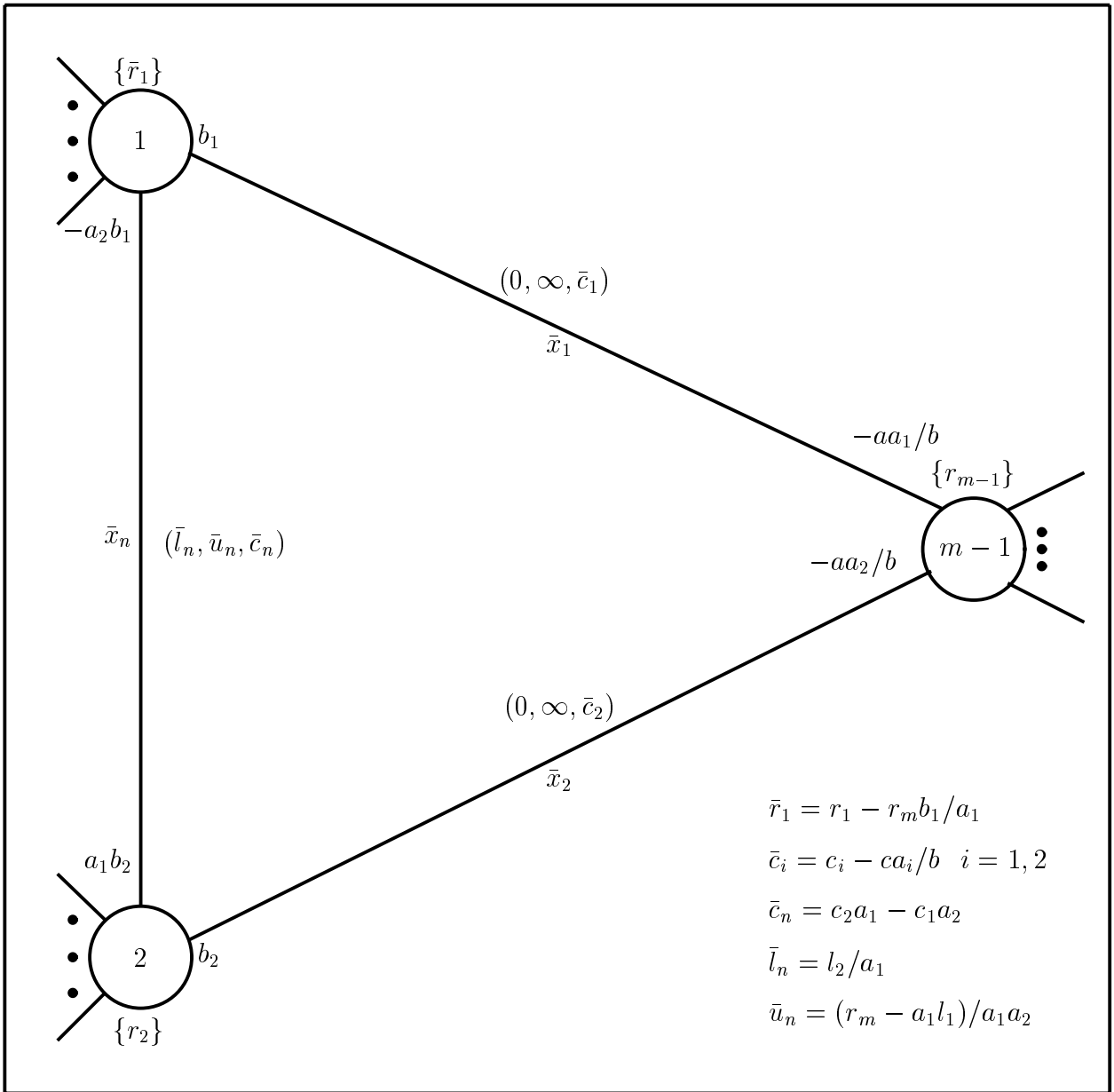


Figure 3: Replacement for Node of Degree 3 with  $r_m > 0$

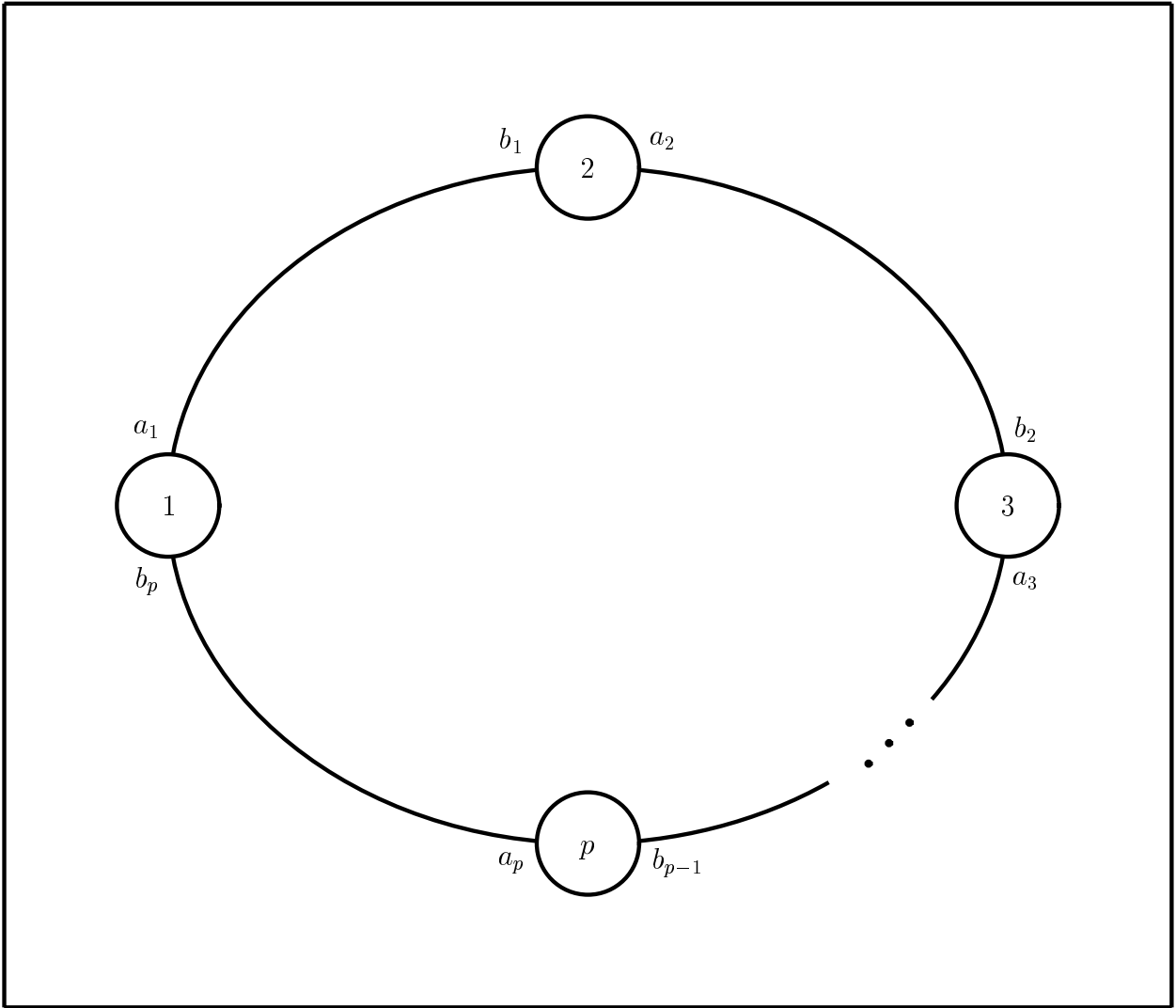


Figure 4: Renaming Components of a Generalized Network Cycle

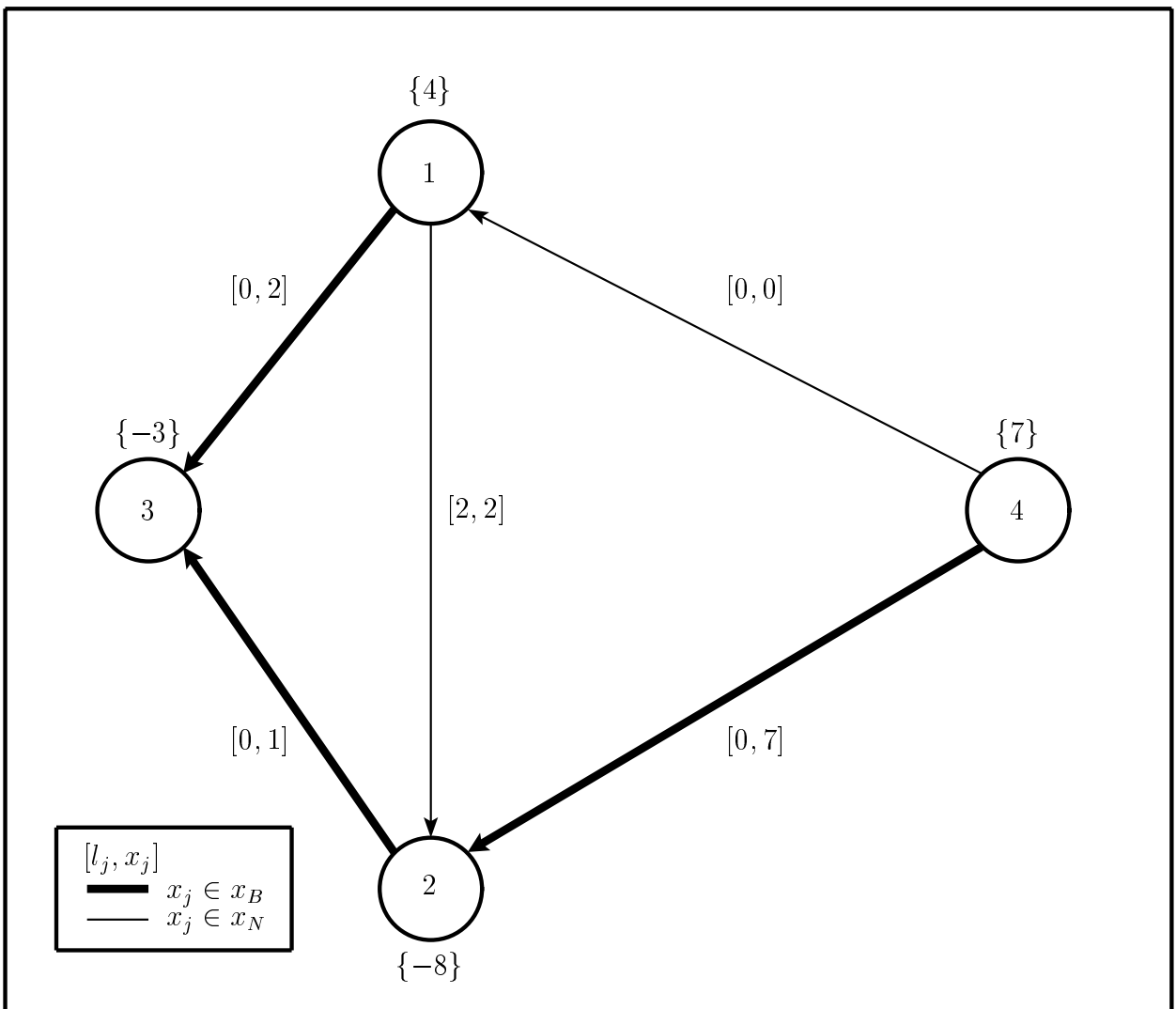


Figure 5: A Basis with Node of Degree 3 Eliminated with Arcs Unbounded

Above

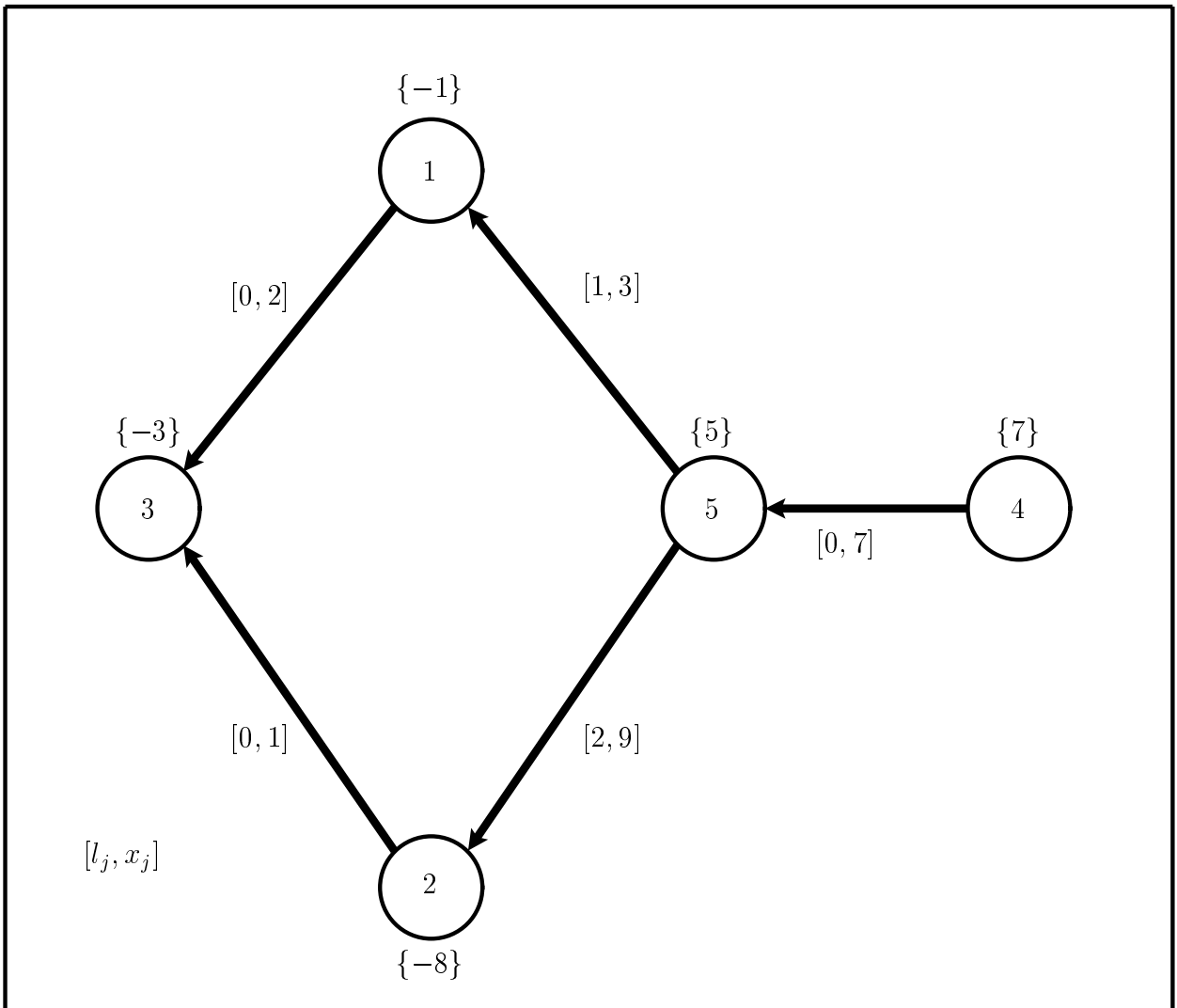


Figure 6: A Corresponding Graph for  $P$  with Arcs Unbounded Above

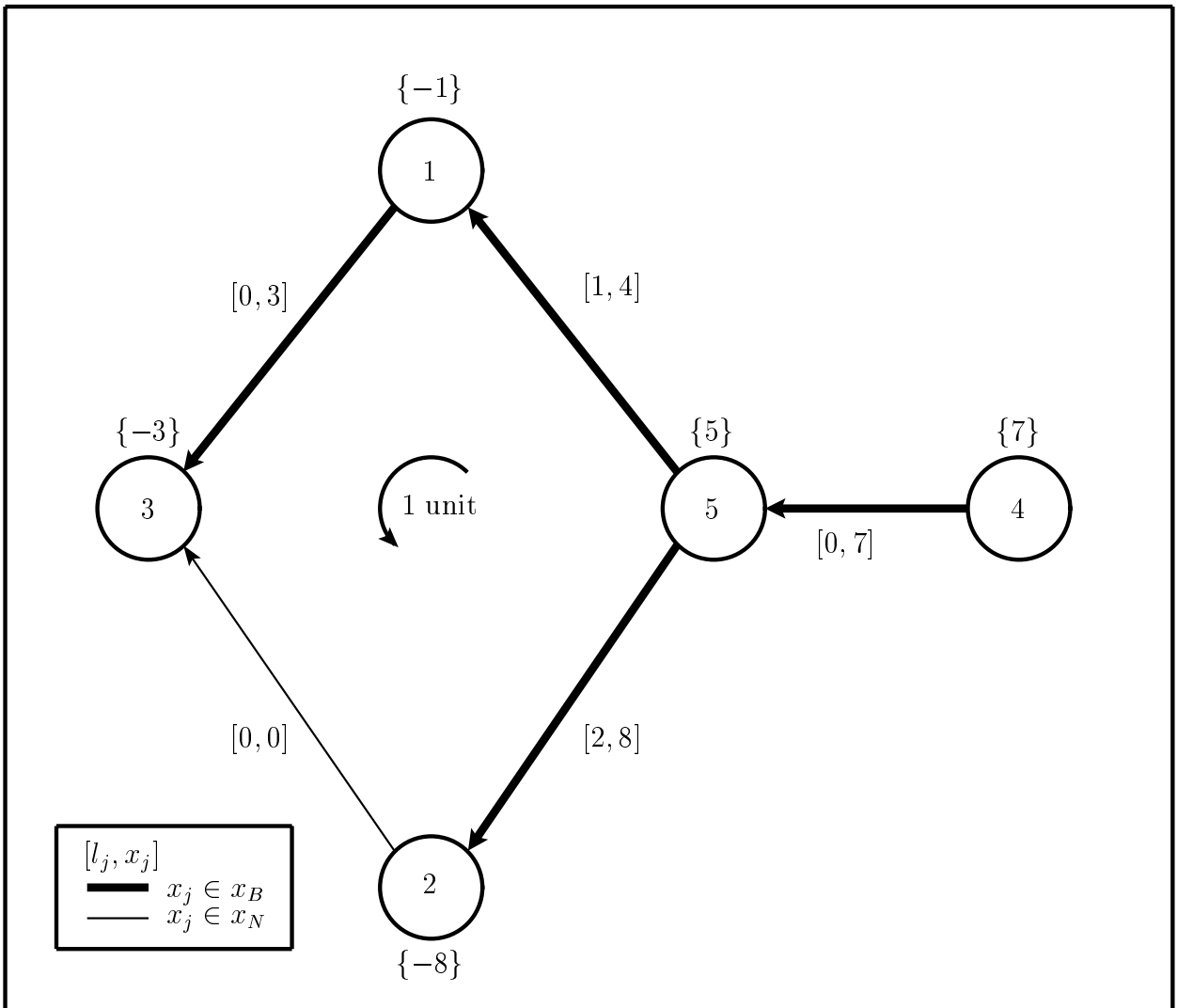


Figure 7: A Basis for  $P$

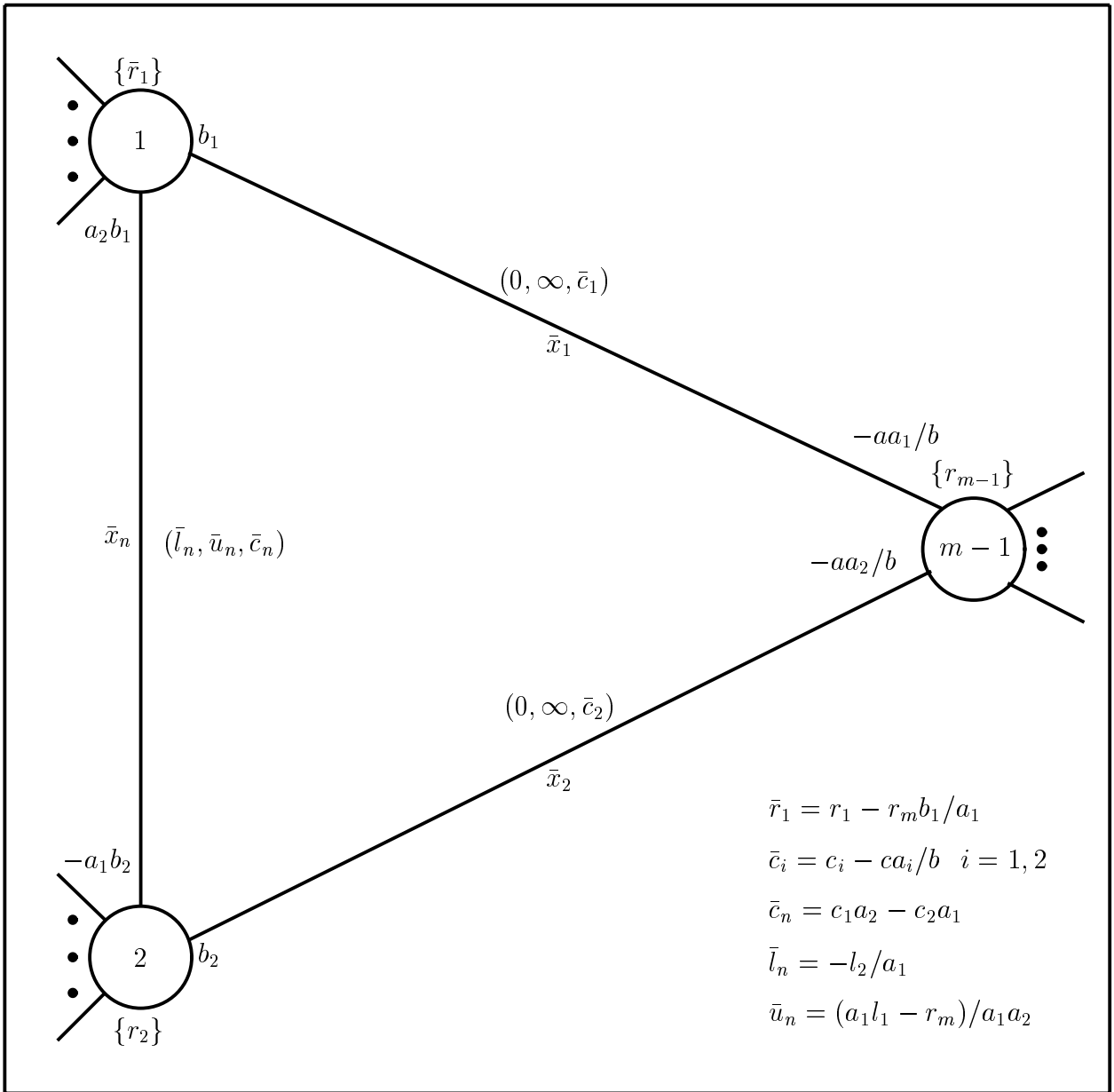


Figure 8: Replacement for Node of Degree 3 with  $r_m < 0$

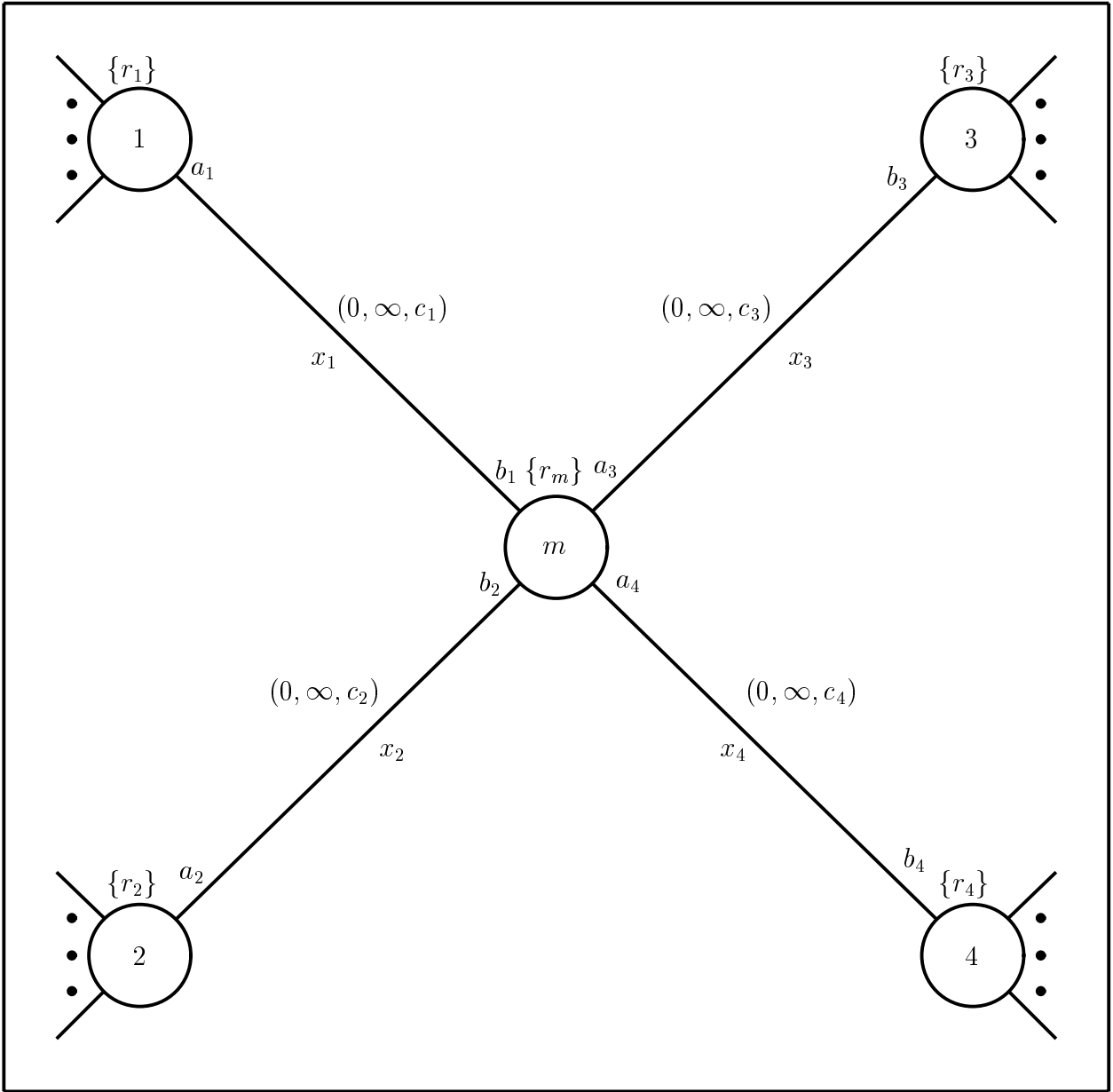


Figure 9: A Network with Node of Degree 4

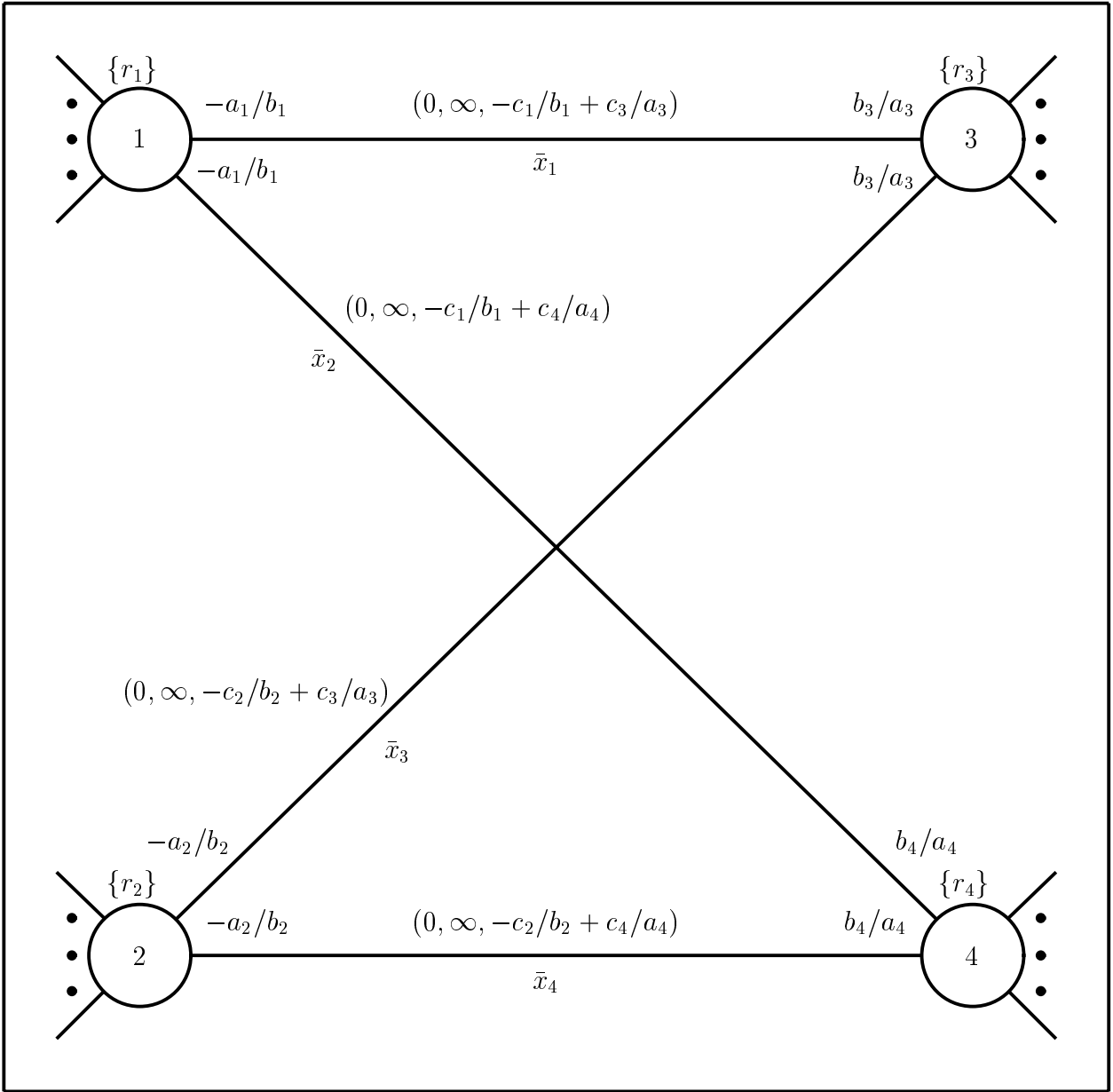


Figure 10: A Network with Node of Degree 4 Eliminated

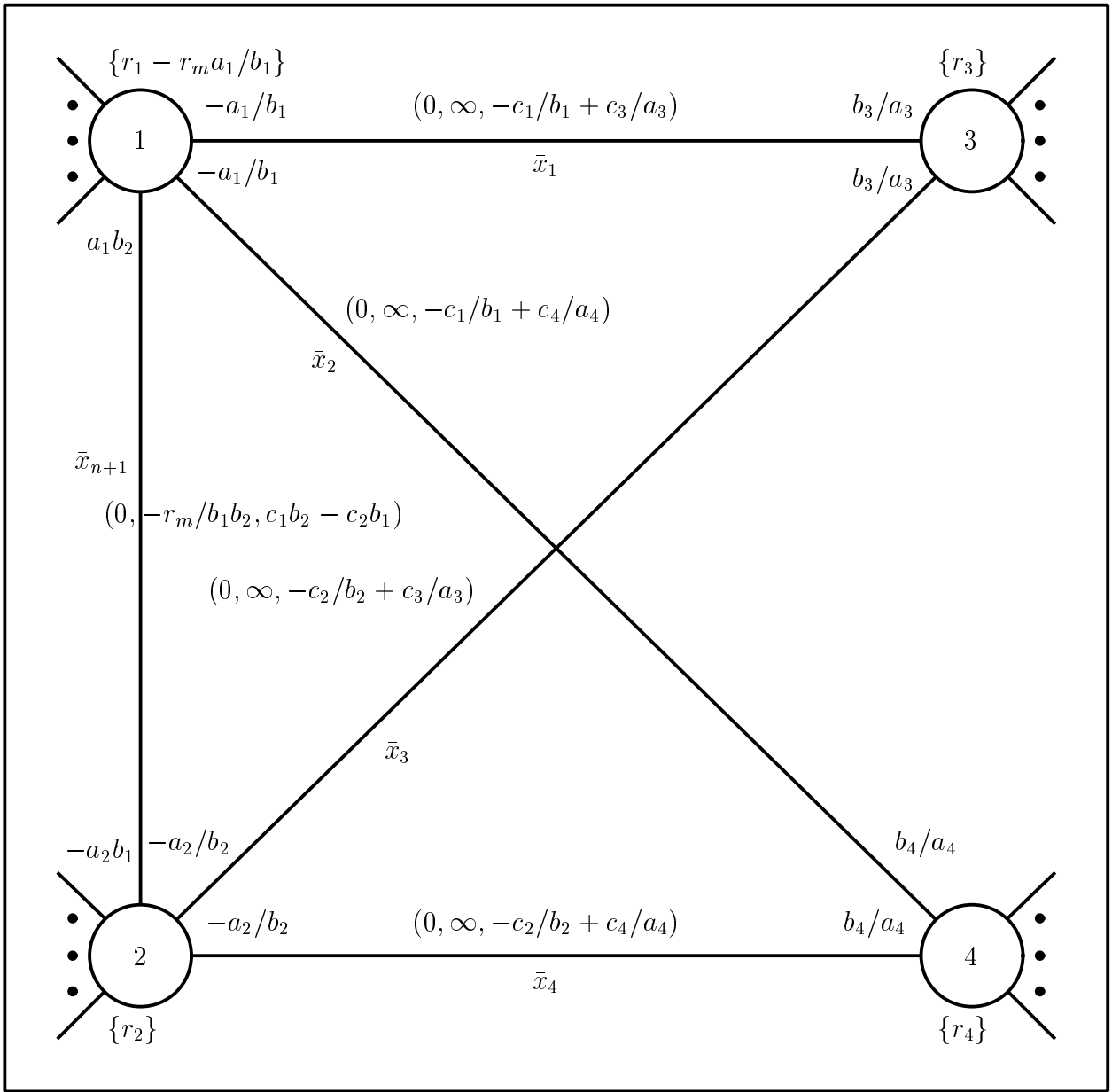


Figure 11: Replacement for Node of Degree 4 if  $r_m < 0$

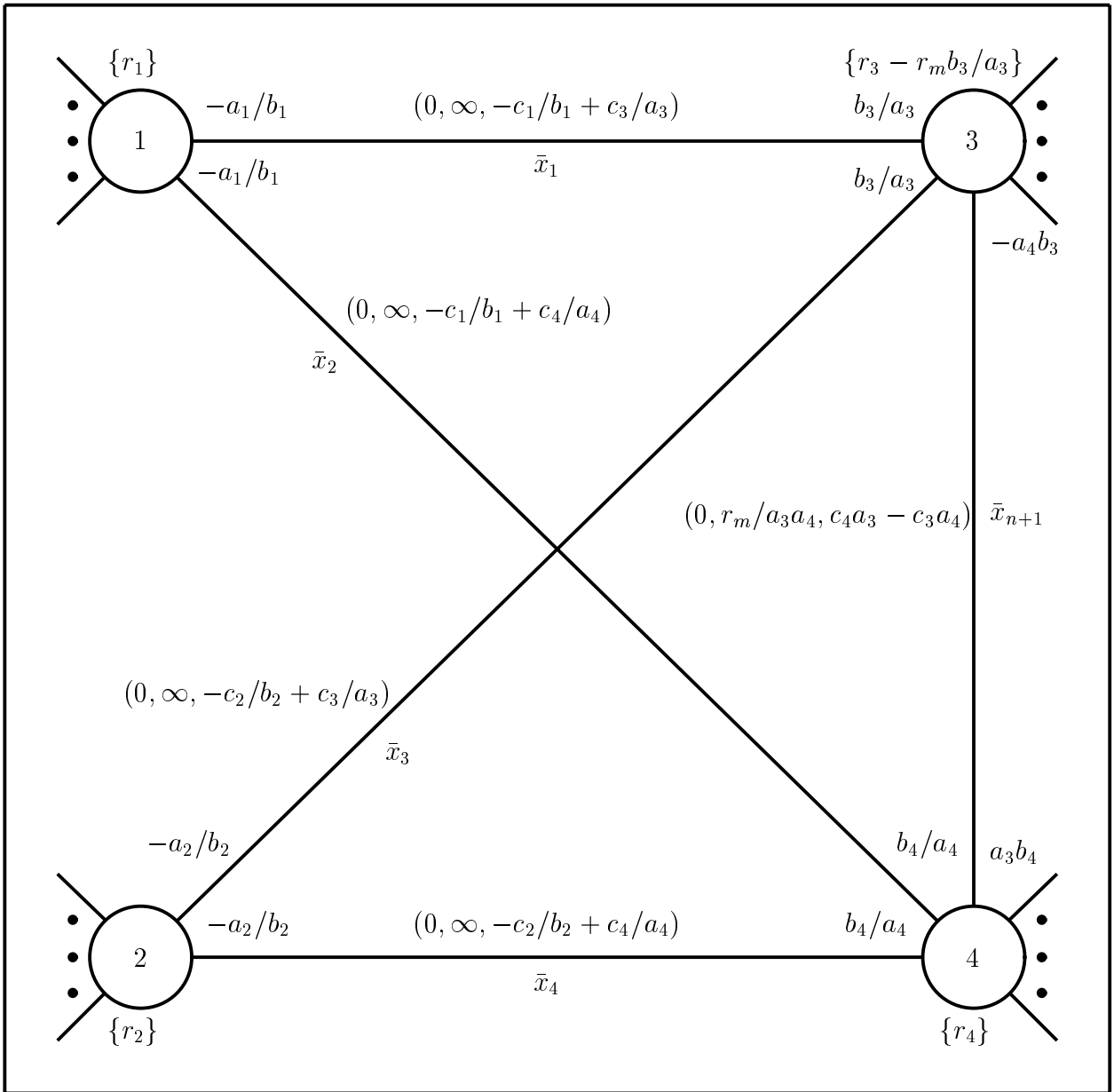


Figure 12: Replacement for Node of Degree 4 if  $r_m > 0$

Table 1: Description of Problems with 100,000 Nodes

Prob Name	Arcs	Supply/Demand Nodes	Transshipment Supply/Demand	Nonzero Lower Bound	Finite Upper Bound
KL01	150,000	2000	0	0	0
KL02	150,000	2000	0	0	75,162
KL03	150,000	2000	0	37,862	0
KL04	150,000	2000	0	37,862	75,162
KL05	150,000	2000	1000	0	0
KL06	150,000	2000	1000	0	75,320
KL07	150,000	2000	1000	37,884	0
KL08	150,000	2000	1000	37,884	75,320
KL09	150,000	50,000	0	0	0
KL10	150,000	50,000	0	0	75,045
KL11	150,000	50,000	0	37,644	0
KL12	150,000	50,000	0	37,644	75,045
KL13	150,000	50,000	25,000	0	0
KL14	150,000	50,000	25,000	0	75,342
KL15	150,000	50,000	25,000	37,651	0
KL16	150,000	50,000	25,000	37,651	75,342
KL17	350,000	2000	0	0	0
KL18	350,000	2000	0	0	174,455
KL19	350,000	2000	0	87,307	0
KL20	350,000	2000	0	87,307	174,455
KL21	350,000	2000	1000	0	0
KL22	350,000	2000	1000	0	174,453
KL23	350,000	2000	1000	87,406	0
KL24	350,000	2000	1000	87,406	174,453
KL25	350,000	50,000	0	0	0
KL26	350,000	50,000	0	0	175,643
KL27	350,000	50,000	0	87,429	0
KL28	350,000	50,000	0	87,429	175,643
KL29	350,000	50,000	25,000	0	0
KL30	350,000	50,000	25,000	0	175,561
KL31	350,000	50,000	25,000	88,818	0
KL32	350,000	50,000	25,000	88,818	175,561

Table 2: Comparison of Reduction between CPLEX Presolver and the GNP

Prob Name	CPLEX 6.5.3 Preprocessor		Generalized Network Preprocessor (GNP)				CPLEX 6.5.3 on Reduced Problem Produced by GNP	
	Nodes Elim	Arcs Elim	Nodes Elim	Arcs			Nodes Elim	Arcs Elim
				Elim	Added	Net		
KL01	75,643	76,420	79,797	76,116	50	76,066	13	363
KL02	60,660	61,234	52,720	52,850	2	52,848	8,120	8,387
KL03	63,947	64,631	69,940	64,329	5,005	59,324	17	296
KL04	53,298	53,858	48,645	48,712	140	48,572	4,895	5,149
KL05	75,612	76,337	79,717	75,924	69	75,855	12	482
KL06	60,745	61,282	52,789	52,751	3	52,748	8,114	8,534
KL07	64,057	64,696	70,034	64,264	5,057	59,207	6	423
KL08	53,457	53,979	48,721	48,627	131	48,496	4,948	5,360
KL09	66,501	73,796	69,202	68,135	396	67,739	836	6,013
KL10	66,178	72,531	60,894	62,386	6	62,380	5,715	10,319
KL11	62,198	69,240	66,053	63,934	1674	62,260	828	5,098
KL12	63,541	69,726	59,881	61,252	14	61,238	4,214	8,599
KL13	68,993	79,925	71,631	71,080	335	70,745	513	9,359
KL14	67,642	77,676	63,028	64,498	4	64,494	5,015	13,390
KL15	64,228	74,918	67,604	66,133	1,397	64,736	504	8,797
KL16	64,936	74,855	61,864	63,215	13	63,202	3,519	11,787
KL17	15,386	18,693	18,953	17,493	93	17,400	24	1,358
KL18	8,513	10,800	8,474	9,695	10	9,685	312	1,089
KL19	8,423	11,504	12,309	10,418	3,699	6,719	14	1,214
KL20	4,898	7,165	5,024	6,214	253	5,961	162	935
KL21	15,186	18,209	18,697	16,962	66	16,896	14	1,389
KL22	8,339	10,391	8,275	9,190	4	9,186	311	1,196
KL23	8,298	11,133	12,150	9,978	3,725	6,253	10	1,294
KL24	4,778	6,803	4,872	5,740	245	5,495	175	1,057
KL25	10,356	35,800	18,418	17,919	2,637	15,282	900	20,267
KL26	12,146	34,274	12,789	16,540	198	16,342	619	17,523
KL27	8,022	33,201	16,232	15,380	3,836	11,544	849	19,913
KL28	11,055	33,083	11,785	15,427	253	15,174	549	17,424
KL29	12,961	41,903	19,626	19,858	1,707	18,151	517	24,039
KL30	11,708	36,665	12,116	15,969	109	15,860	456	20,549
KL31	9,698	38,157	16,563	16,353	3,906	12,447	451	23,516
KL32	10,117	34,936	10,612	14,337	195	14,142	378	20,450

Table 3: Comparison of Solution Times Using the Network Solver (seconds)

Prob Name	No Preprocessor [1]	CPLEX 6.5.3 Preprocessor [2]	% Diff $([1]-[2])/[1]$	Gen. Network Preprocessor [4]	% Diff $([1]-[4])/[1]$
KL01	12.02	6.79	43.5%	3.33	72.3%
KL02	22.40	13.89	38.0%	10.75	52.0%
KL03	51.30	19.12	62.7%	12.29	76.0%
KL04	50.66	28.75	43.3%	26.84	47.0%
KL05	11.92	6.80	43.0%	3.37	71.8%
KL06	22.14	14.03	36.7%	10.70	51.7%
KL07	52.08	19.18	63.2%	12.07	76.8%
KL08	51.72	28.48	44.9%	26.89	48.0%
KL09	34.73	16.67	52.0%	10.23	70.6%
KL10	31.93	16.59	48.0%	12.79	59.9%
KL11	42.10	19.78	53.0%	12.55	70.2%
KL12	38.43	18.71	51.3%	16.04	58.3%
KL13	35.41	16.27	54.1%	9.68	72.7%
KL14	33.18	20.31	38.8%	12.83	61.3%
KL15	42.87	21.15	50.7%	12.27	71.4%
KL16	38.54	22.47	41.7%	16.55	57.1%
KL17	8.98	22.16	-146.9%	11.53	-28.4%
KL18	21.54	36.27	-68.4%	23.23	-7.8%
KL19	84.91	92.02	-8.4%	76.27	10.2%
KL20	89.90	101.21	-12.6%	88.39	1.7%
KL21	8.97	21.66	-141.5%	11.30	-26.0%
KL22	20.25	37.38	-84.6%	21.66	-6.9%
KL23	84.65	89.82	-6.1%	75.56	10.7%
KL24	88.28	101.38	-14.8%	87.46	0.9%
KL25	43.41	58.14	-33.9%	41.16	5.2%
KL26	53.43	64.09	-20.0%	50.57	5.3%
KL27	67.46	75.83	-12.4%	59.67	11.5%
KL28	69.37	79.22	-14.2%	65.82	5.1%
KL29	41.65	57.40	-37.8%	38.66	7.2%
KL30	51.18	63.64	-24.3%	49.13	4.0%
KL31	66.21	78.23	-18.1%	56.81	14.2%
KL32	69.40	80.01	-15.3%	66.90	3.6%
Totals	1440.94	1347.35	6.5%	1033.23	28.3%

Table 4: Detailed Comparison of Times Using the Network Solver (seconds)

Prob Name	CPLEX 6.5.3			Gen. Network Preprocessor			% Diff
	Presolve [1]	Solve + Postsolve [2]	Total [3]	Presolve [4]	Solve + Postsolve [5]	Total [6]	$([3]-[6])/[3]$
KL01	3.97	2.83	6.79	1.10	2.23	3.33	51.0%
KL02	4.83	9.06	13.89	0.82	9.94	10.75	22.6%
KL03	4.19	14.93	19.12	1.10	11.20	12.29	35.7%
KL04	4.92	23.83	28.75	0.80	26.04	26.84	6.6%
KL05	3.99	2.82	6.80	1.09	2.28	3.37	50.5%
KL06	4.65	9.38	14.03	0.81	9.89	10.70	23.7%
KL07	3.97	15.21	19.18	1.10	10.98	12.07	37.1%
KL08	4.81	23.67	28.48	0.78	26.11	26.89	5.6%
KL09	6.38	10.29	16.67	1.06	9.17	10.23	38.6%
KL10	6.53	10.07	16.59	0.94	11.85	12.79	22.9%
KL11	6.17	13.61	19.78	1.04	11.51	12.55	36.5%
KL12	6.01	12.70	18.71	0.93	15.11	16.04	14.2%
KL13	6.02	10.25	16.27	1.06	8.62	9.68	40.5%
KL14	6.90	13.41	20.31	0.93	11.91	12.83	36.8%
KL15	5.94	15.22	21.15	1.01	11.26	12.27	42.0%
KL16	6.55	15.92	22.47	0.90	15.65	16.55	26.3%
KL17	13.08	9.08	22.16	2.23	9.30	11.53	48.0%
KL18	14.57	21.70	36.27	1.89	21.34	23.23	36.0%
KL19	12.93	79.09	92.02	2.05	74.23	76.27	17.1%
KL20	12.80	88.42	101.21	1.81	86.58	88.39	12.7%
KL21	13.05	8.61	21.66	2.18	9.13	11.30	47.8%
KL22	16.87	20.52	37.38	1.91	19.75	21.66	42.1%
KL23	11.46	78.36	89.82	2.10	73.46	75.56	15.9%
KL24	15.02	86.36	101.38	1.86	85.60	87.46	13.7%
KL25	20.70	37.44	58.14	2.53	38.63	41.16	29.2%
KL26	19.40	44.69	64.09	2.24	48.33	50.57	21.1%
KL27	18.99	56.84	75.83	2.55	57.12	59.67	21.3%
KL28	19.57	59.66	79.22	2.22	63.60	65.82	16.9%
KL29	22.86	34.55	57.40	2.45	36.21	38.66	32.6%
KL30	20.28	43.36	63.64	2.18	46.96	49.13	22.8%
KL31	22.65	55.58	78.23	2.43	54.38	56.81	27.4%
KL32	20.76	59.25	80.01	2.20	64.70	66.90	16.4%
Totals	360.73	986.62	1347.35	50.23	983.00	1033.23	23.3%

## References

- [1] I. Adler, N. Karmarkar, M. G. Resende, and G. Veiga. Data structures and programming techniques for the implementation of Karmarkar's algorithm. *ORSA Journal on Computing*, 1(2):84–106, 1989.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1993.
- [3] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71(2):221–245, 1995.
- [4] G. H. Bradley, G. G. Brown, and G. W. Graves. Structural redundancy in large-scale optimization models. In *Redundancy in Mathematical Programming*, chapter 12. Springer-Verlag, Berlin, 1983.
- [5] A. L. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, 8(1):54–83, 1975.
- [6] J. Gondzio. Presolve analysis of linear programs prior to applying an interior point method. *INFORMS Journal on Computing*, 9(1):73–91, 1997.

- [7] R. V. Helgason and J. L. Kennington. Primal simplex algorithms for minimum cost network flows. In *Handbooks in Operations Research and Management Science*, volume 7, chapter 2. Elsevier, Amsterdam, 1995.
- [8] J. L. Kennington and R. V. Helgason. *Algorithms for Network Programming*. John Wiley and Sons, Inc., New York, N. Y., 1980.
- [9] J. L. Kennington and K. R. Lewis. Preprocessing techniques for generalized networks: Theory and an empirical analysis. Technical Report 00-CSE-10, Southern Methodist University, Dallas, TX 75275, 2000. Available from SMU Engineering and Science Library.
- [10] D. Klein and S. J. Holm. Some reduction of linear programs using bounds on problem variables. In *Redundancy in Mathematical Programming*, chapter 8. Springer-Verlag, Berlin, 1983.
- [11] K. R. Lewis. *Preprocessing Algorithms for Generalized Networks: Theory and an Empirical Analysis*. PhD thesis, Southern Methodist University, 2001.
- [12] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–14, 1994.
- [13] K. Murty. *Network Programming*. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1992.

- [14] J. A. Tomlin and J. S. Welch. Finding duplicate rows in a linear programming model. *Operations Research Letters*, 5(1):7–11, 1986.
  
- [15] H. P. Williams. A reduction procedure for linear and integer programming models. In *Redundancy in Mathematical Programming*, chapter 9. Springer-Verlag, Berlin, 1983.