FINDING A MATRIX OF GIVEN SIGN PATTERN AND LINE SUMS

EDWARD C. KUNG
ADVISOR: DONALD Y. YAU

ABSTRACT. We construct an algorithm that solves the problem described as follows. Given an $m \times n$ sign pattern A, and real vectors $\vec{r} \in \mathbb{R}^m$ and $\vec{c} \in \mathbb{R}^n$, find an $m \times n$ matrix B with sign pattern A and row sums $\vec{r}$ and column sums $\vec{c}$. We also answer the question of when such a problem has a unique solution and give insight into the conditions which will allow for an integer solution to the problem when $\vec{r}$ and $\vec{c}$ have integer entries.

1. INTRODUCTION

A *sign pattern* is a $m \times n$ array whose entries are $+$'s, $-$'s, or 0's. Associated with a sign pattern A is the class $Q(A)$ of all $m \times n$ real matrices B such that $b_{ij} > 0$ when $a_{ij} = +$, $b_{ij} < 0$ when $a_{ij} = -$, and $b_{ij} = 0$ when $a_{ij} = 0$. $\hat{Q}(A)$ is the class of all $m \times n$ real matrices B such that $b_{ij} \geq 0$ when $a_{ij} = +$ or 0, $b_{ij} \leq 0$ when $a_{ij} = -$ or 0, and $b_{ij} = 0$ when $a_{ij} = 0$.

The main interest of this article is the problem $P(A, \vec{r}, \vec{c})$, where A is an $m \times n$ sign pattern and $\vec{r}$ and $\vec{c}$ are vectors in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively. Defining e to be a vector of 1's as usual, a solution to $P(A, \vec{r}, \vec{c})$ is a $m \times n$ matrix $B \in Q(A)$ with row sum vector $\vec{r}$ and column sum vector $\vec{c}$; that is, $Be = \vec{r}$ and $e^T B = \vec{c}$. A *weak* solution to $P(A, \vec{r}, \vec{c})$ is a $m \times n$ matrix $B \in \hat{Q}(A)$ with row sum vector $\vec{r}$ and column sum vector $\vec{c}$; that is, $Be = \vec{r}$ and $e^T B = \vec{c}$.

In [1], Johnson et. al. characterized the problems P that admit a solution, so in our article we will assume that we can easily determine whether a problem P has a solution or not. However, Johnson et. al. left open the question of how a solution to P can be constructed when one is known to exist. The focus of our article is to answer this question.

The first step to answering this question is in fact already given in [1] and [3]: Johnson et. al. showed in [1] that the problem P is equivalent to a special case of feasible circulation problems in graph theory, and an algorithm for solving general feasible circulation problems is given in [3].[1] This implies that an equivalent algorithm should exist for constructing a solution to P. The majority of our article will be devoted to describing this algorithm in the language of linear algebra (thus saving one the trouble of converting our problem into its equivalent graph theory problem).

Date: March 30, 2004.
This research was completed in the summer of 2003 as an undergraduate research project at the University of Illinois at Urbana Champaign.

[1]For technical details on how the problems are related, refer to [1] and [3].

From the algorithm we deduce a number of interesting results regarding the existence of integer solutions to a problem P and the uniqueness of a solution to a problem P. For the interest of the reader, we discuss these results before we introduce the algorithm.

2. Main results

It is obvious that not every problem P in which $\vec{r}$ and $\vec{c}$ have only integer elements has an integer solution (a solution with only integer elements). For example, the simple case where A is

$$\begin{pmatrix} + & + \\ + & + \end{pmatrix}$$

and $\vec{r}$ and $\vec{c}$ are both $(1,1)^T$ cannot have an integer solution. However, we can see easily that

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

is a *weak* integer solution to the problem. Whether or not a problem with $\vec{r}$ and $\vec{c}$ having only integer elements will always have a weak integer solution is not so obvious, but our first result answers this question.

Theorem 1. *Assuming P has a solution, a problem $P(A, \vec{r}, \vec{c})$ has a weak solution with integer elements if and only if all the elements of $\vec{r}$ and $\vec{c}$ are integers.*

We have seen that a problem P with $\vec{r}$ and $\vec{c}$ having only integer elements does not necessarily have an integer solution. A much broader question, then, is whether or not such a problem necessarily has a rational solution. Our second result claims that it does.

Theorem 2. *Assuming P has a solution, a problem $P(A, \vec{r}, \vec{c})$ in which all the elements of $\vec{r}$ and $\vec{c}$ are integers has a solution in which all of its elements are rational numbers.*

Since there is a solution with all of its elements rational numbers, it follows directly that the solution can be multiplied by some integer N to give an integer solution to an equivalent problem in which $\vec{r}$ and $\vec{c}$ have been scaled by N.

Corollary 1. *If a solution exists for problem $P(A, \vec{r}, \vec{c})$, in which all the elements of $\vec{r}$ and $\vec{c}$ are integers, then there exists an integer N such that the problem $P(A, N\vec{r}, N\vec{c})$ has a solution in which all of its elements are integers.*

Finally, we ask the question of when a solution to a problem P is unique. In our fourth result we claim that P either has one unique solution or it has an infinite number of solutions. We characterize the problems P that have a unique solution and, conversely, an infinite number of solutions.

Theorem 3. *Assume a solution exists for problem $P(A, \vec{r}, \vec{c})$. There are an infinite number of solutions to P if and only if there exists a set of positions $(ij) = \{i_1j_1, i_2j_1, i_2j_2, i_3j_2, \ldots, i_kj_k, i_1j_k\}$ in A with at least 4 elements such that a_{ij} is non-zero for all positions in the set. The solution is unique if and only if such a set does not exist.*

We postpone the proofs of these theorems until section 6, after we have discussed the algorithm. In the following section we introduce some definitions that will set the stage for our algorithm, and in the two sections after that we describe the algorithm and give a demonstration of how the algorithm is performed.

3. Preliminary definitions

Before we begin our formal description of the algorithm, it is necessary to introduce the concept of a *directed matrix link* (we use "directed" here to differentiate our term "matrix link" from the term "matrix link" defined in [2]), a *link augmentation operation*, a *residual array*, and a *residual array update operation.*

Definition 1. *Let C be a $m \times n$ matrix or array. A* directed matrix link *$L(C)$ is a set of positions of C defined in one of the following five ways:*

(1) $L(C) = \{i_1 1, i_1 j_1, i_2 j_1, i_2 j_2, i_3 j_2, \ldots, i_k j_k, 1 j_k\}$, where $c_{i_1 1}$ and $c_{1 j_k} > 0$, and $c_{i_x j_y} > 0$ if $x = y$ and < 0 otherwise.
(2) $L(C) = \{i_1 1, i_1 j_1, i_2 j_1, i_2 j_2, i_3 j_2, \ldots, i_{k+1} j_k, i_{k+1} 1\}$, where $c_{i_1 1} > 0$ and $c_{i_{k+1} 1} < 0$, and $c_{i_x j_y} > 0$ if $x = y$ and < 0 otherwise.
(3) $L(C) = \{1 j_1, i_1 j_1, i_1 j_2, i_2 j_2, i_2 j_3, \ldots, i_k j_k, i_k 1\}$, where $c_{1 j_1}$ and $c_{i_k 1} < 0$, and $c_{i_x j_y} < 0$ if $x = y$ and > 0 otherwise.
(4) $L(C) = \{1 j_1, i_1 j_1, i_1 j_2, i_2 j_2, i_2 j_3, \ldots, i_k j_{k+1}, 1 j_{k+1}\}$, where $c_{1 j_1} < 0$ and $c_{1 j_{k+1}} > 0$, and $c_{i_x j_y} < 0$ if $x = y$ and > 0 otherwise.
(5) $L(C) = \{i_1 j_1, i_2 j_1, i_2 j_2, i_3 j_2, \ldots, i_k j_1\}$, where $c_{i_1 j_1} > 0$ and $c_{i_k j_1} < 0$, and $c_{i_x j_y} > 0$ if $x = y$ and < 0 otherwise.

We call a link defined the first way a *"type 1 link"*, a link defined the second way a *"type 2 link"*, and so on. Throughout this article, we will refer to a directed matrix link $L(C)$ simply as a *link on C*. For example, if we let C be the 4×4 matrix

$$\begin{pmatrix} 0 & 1 & -1 & 1 \\ 1 & 2 & -1 & -1 \\ -1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{pmatrix}$$

then the set of positions $\{21, 22, 12\}$ is a type 1 link, the set of positions $\{21, 22, 32, 31\}$ is a type 2 link, the set of positions $\{13, 23, 22, 32, 31\}$ is a type 3 link, the set of positions $\{13, 23, 22, 12\}$ is a type 4 link, and the set of positions $\{22, 24, 44, 42\}$ is a type 5 link.

Definition 2. *Let B and C be $m \times n$ matrices, x be a real number, and $L(C)$ be a link on C. A* link augmentation operation *redefines the elements of B in the following way:*

$$b_{ij} = \begin{cases} b_{ij} + x, & \text{if } c_{ij} > 0; \\ b_{ij} - x, & \text{if } c_{ij} < 0, \end{cases} \quad \text{for all positions } ij \text{ in } L(C).$$

If we perform this operation we say that we *augment B by x at the positions of $L(C)$*.

Definition 3. *A* $m \times n$ residual array *is a* $m \times n$ *array with real numbers in its first row and first column, and ordered pairs in positions not in the first row or first column. For each ordered pair, the first element is nonnegative and the second element is nonpositive.*

For example,

$$\begin{pmatrix} 0 & 2 & 4 & -3 \\ -2 & (+1,-1) & (2,-1) & (+1,-4) \\ 3 & (0,0) & (+2.5,0) & (0,-1) \\ \pi & (+3,-1) & (+1,-1) & (0,0) \end{pmatrix}$$

is a 4×4 residual array.

To specify a position of an element in a residual array R we either use the notation ij, which we call the *general position*, or the notation $\pm ij$, which we call the *specific position*. The general position ij gives the numerical value of the element at ij if ij is in the first row or column, and gives the ordered pair at ij if ij is not in the first row or column. The specific position $\pm ij$ gives the numerical value of the element at ij if ij is in the first row or column (in this case we use $+$ to denote a positive value and $-$ to denote a negative value), and gives the first element of the ordered pair at ij if $+$ precedes ij or gives the second element of the ordered pair at ij if $-$ precedes ij.

Recall that a directed matrix link is a set of positions. A link on a residual array R is defined the same way as a link on a regular matrix, except that each element in the link is a *specific* position of R. For example, using the residual array given above,

$$\{+31, +33, +13\}$$

is a type 1 link where $r_{+31} = 3$, $r_{+33} = 2.5$, and $r_{+13} = 4$,

$$\{+22, -23, +43, -42\}$$

is a type 5 link where $r_{+22} = 1$, $r_{-23} = -1$, $r_{+43} = 1$, and $r_{-42} = -1$, and

$$\{+23, -22, +42, -43\}$$

is a type 5 link where $r_{+23} = 2$, $r_{-22} = -1$, $r_{+42} = 3$, and $r_{-43} = -1$.

Definition 4. *Let B and C be $m \times n$ residual arrays, x be a real number, and $L(C)$ be a link on C. A* residual array update operation *redefines the elements of B in the following way:*

$$b_{ij} = \begin{cases} b_{ij} - x, & \text{if } c_{\pm ij} > 0 \text{ and } i \text{ or } j = 1; \\ b_{ij} + x, & \text{if } c_{\pm ij} < 0 \text{ and } i \text{ or } j = 1; \\ b_{ij} - (x,x) & \text{if } c_{\pm ij} > 0 \text{ and } i,j \neq 1; \\ b_{ij} + (x,x) & \text{if } c_{\pm ij} < 0 \text{ and } i,j \neq 1, \end{cases} \quad \text{for all positions } \pm ij \text{ in } L(C).$$

If we perform this operation we say we *update* B with $L(C)$ and x. This concludes our discussion of the preliminary definitions necessary to describe the algorithm.

4. The algorithm

The entire algorithm is performed in three stages, each of which involve several steps. The parameters of the algorithm are simply the parameters of the problem: the $m \times n$ sign pattern A, the $m \times 1$ row sum vector $\vec{r}$ and the $n \times 1$ column sum vector $\vec{c}$. The algorithm is described in full in this section. An example is included in the following section, and the reader may find it helpful to follow the example as he reads through this section.

Initialization stage:

1: Create a $(m+1) \times (n+1)$ zero matrix U, called the *intermediate matrix*. Define S as the submatrix "block" of U, $S = U[\alpha|\beta]$ where $\alpha = \{2, \dots, m+1\}$ and $\beta = \{2, \dots, n+1\}$, that is, S to be the submatrix of U formed by the intersection of rows α and columns β.

2: Create a $(m+1) \times (n+1)$ residual array V with elements defined as follows:

(1) $v_{11} = 0$,
(2) $v_{i+1,1} = r_i$ for all i,
(3) $v_{1,j+1} = c_j$ for all j,
(4) $v_{i+1,j+1} = (+2M, 0)$ for all $a_{ij} = +$, where M is an integer defined as follows:

$$M > \sum_{i=1}^{m} |r_i| + \sum_{j=1}^{n} |c_j|,$$

(Note that M is just an arbitrarily large number that is large enough to accommodate a solution.)
(5) $v_{i+1,j+1} = (0, -2M)$ for all $a_{ij} = -$, where M is as above,
(6) $v_{i+1,j+1} = (0, 0)$ for all $a_{ij} = 0$.

Call this the *residual array.*

Line sum adjustment stage:

3: Find an i such that $v_{i1} \neq 0$. If no such i exists go to step 9, otherwise proceed.
4: Find a type 1–4 link on V that contains the general position $i1$. Call this link L. If no such link can be found, terminate the algorithm; there is no solution.
5: Define x to be the smallest of the absolute values of the elements of V at the specific positions of the link L.
6: Augment x on U at the positions of link L.
7: Update V with x and L.
8: If $v_{i1} = 0$ go to step 3, otherwise go to step 4.
9: Find a j such that $v_{1j} \neq 0$. If no such j exists go to 15, otherwise proceed.

10: Find a type 1–4 link on V that contains the general position $1j$. Call this link L. If no such link can be found, terminate the algorithm; there is no solution.
11: Define x to be the smallest of the absolute values of the elements of V at the specific positions of the link L.
12: Augment x on U at the positions of link L.
13: Update V with x and L.
14: If $v_{1j} = 0$ go to step 9, otherwise go to step 10.

Sign adjustment stage:

15: Find a position ij, where $i, j \neq 1$, such that the sign of u_{ij} does not correspond to the sign of $a_{i-1,j-1}$. If no such ij exists then terminate the algorithm; S is a solution. Otherwise proceed.
16: If $a_{i-1,j-1} = +$, find a link on V with specific position $+ij$. Otherwise if $a_{i-1,j-1} = -$, find a link on V with specific position $-ij$. Call this link L. If no such link exists, terminate the algorithm; there is no solution.
17: Define x to be a positive number less than the smallest of the absolute values of the elements of V at the specific positions of link L.
18: Augment x on U at the positions of link L.
19: Update V with x and L. Go to step 15.

5. A demonstration and an explanation

The algorithm is relatively simple to perform if understood, but its explanation on paper is complicated. Thus, to give the reader a better understanding of how to perform the algorithm, we include in this section a demonstration. The reader will find it helpful to consider the steps of the algorithm as he or she follows our demonstration. It will also be helpful if the reader performs the algorithm along with our demonstration. We will now demonstrate how to use the algorithm to solve $P(A, \vec{r}, \vec{c})$, where A is the 3×3 sign pattern

$$\begin{pmatrix} + & - & + \\ 0 & + & + \\ + & - & - \end{pmatrix}$$

and $\vec{r} = (4, 3, -1)^T$ and $\vec{c} = (4, 0, 2)^T$.

Initialization stage:

First we initialize the matrix U and V as follows:

$$U = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } V = \begin{pmatrix} 0 & 4 & 0 & 2 \\ 4 & (+50, 0) & (0, -50) & (+50, 0) \\ 3 & (0, 0) & (+50, 0) & (+50, 0) \\ -1 & (+50, 0) & (0, -50) & (0, -50) \end{pmatrix}$$

We have chosen M here to be 25, but M can be arbitrarily large as long as it meets the condition described in section 4.

Line sum adjustment stage:

We see that $v_{21} \neq 0$ so we find a type 1—4 link on V that contains the position 21. $L(V) = \{+21, +22, +12\}$ is such a link (type 1). Now we set x to be the smallest of the absolute values of the elements of V at the specific positions of the link $L(V)$. So in this case, $x = 4$. Now we augment x on U at the positions of $L(V)$ and update V with x and $L(V)$. So

$$U = \begin{pmatrix} 0 & 4 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad V = \begin{pmatrix} 0 & 0 & 0 & 2 \\ 0 & (+46, -4) & (0, -50) & (+50, 0) \\ 3 & (0, 0) & (+50, 0) & (+50, 0) \\ -1 & (+50, 0) & (0, -50) & (0, -50) \end{pmatrix}$$

Now we see that $v_{31} \neq 0$ so we find the type 1 link $L(V) = \{+31, +34, +14\}$. In this case we set $x = 2$. Again we augment U and update V.

$$U = \begin{pmatrix} 0 & 4 & 0 & 2 \\ 4 & 4 & 0 & 0 \\ 2 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad V = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & (+46, -4) & (0, -50) & (+50, 0) \\ 1 & (0, 0) & (+50, 0) & (+48, -2) \\ -1 & (+50, 0) & (0, -50) & (0, -50) \end{pmatrix}$$

Still $v_{31} \neq 0$ so we find another link, the type 2 link $L(V) = \{+31, +33, -43, -41\}$. We set $x = 1$ and we augment U and update V as usual.

$$U = \begin{pmatrix} 0 & 4 & 0 & 2 \\ 4 & 4 & 0 & 0 \\ 3 & 0 & 1 & 2 \\ -1 & 0 & -1 & 0 \end{pmatrix} \quad V = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & (+46, -4) & (0, -50) & (+50, 0) \\ 0 & (0, 0) & (+49, -1) & (+48, -2) \\ 0 & (+50, 0) & (+1, -49) & (0, -50) \end{pmatrix}$$

We see that all $v_{ij} = 0$, where i or $j = 1$, so we may move on to the sign adjustment stage. Notice that the submatrix S of U is indeed a weak solution to the problem.

Sign adjustment stage:

We see that $u_{23} = 0$ while $a_{12} = -$. So we find a type 5 link that includes u_{-23}. The link $L(V) = \{+24, -23, +43, -44\}$ is such a link. We define x to be less than the smallest of the absolute values of the elements of V at the positions of $L(V)$. The smallest of the absolute values is 1 ($v_{+43} = 1$) so we can set $x = 0.5$. We augment and update U and V as usual.

$$U = \begin{pmatrix} 0 & 4 & 0 & 2 \\ 4 & 4 & -0.5 & 0.5 \\ 3 & 0 & 1 & 2 \\ -1 & 0 & -0.5 & -0.5 \end{pmatrix}$$

$$V = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & (+46,-4) & (0.5,-49.5) & (+49.5,-0.5) \\ 0 & (0,0) & (+49,-1) & (+48,-2) \\ 0 & (+50,0) & (+0.5,-49.5) & (+0.5,-49.5) \end{pmatrix}$$

Finally we see that $u_{42} = 0$ while $a_{31} = +$. So we find the link $L(V) = \{+42, -44, +24, -22\}$ and set $x = 1$. We augment and update U and V to obtain

$$U = \begin{pmatrix} 0 & 4 & 0 & 2 \\ 4 & 3 & -0.5 & 1.5 \\ 3 & 0 & 1 & 2 \\ -1 & 1 & -0.5 & -1.5 \end{pmatrix}$$

$$V = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & (+47,-3) & (0.5,-49.5) & (+48.5,-1.5) \\ 0 & (0,0) & (+49,-1) & (+48,-2) \\ 0 & (+49,-1) & (+0.5,-49.5) & (+1.5,-48.5) \end{pmatrix}$$

and we see that the submatrix S of U is a solution to the problem. Note that the first column of U represents the row sums and the first row of U represents the column sums, that is, $u_{i+1,1} = r_i$ and $u_{1,j+1} = c_j$. This completes our demonstration of the algorithm. Also note that the algorithm is not deterministic, that is, we would have obtained a different solution had we chosen different links at each of the steps.

To begin our explanation of why this algorithm works, we first remark that the elements in the first row and first column of V represent how much the corresponding elements in the first row and first column of U must be modified such that the first column of U represents $\vec{r}$ and that the first row of U represents $\vec{c}$. The elements not in the first row and first column of V represent how much the corresponding elements in the submatrix block S of U may be modified such that S is "weakly" the same sign pattern as A, that is: $u_{ij} \geq 0$ (respectively ≤ 0, $= 0$) when $a_{i-1,j-1} = +$ (respectively $-$, 0). Because of the way a link is defined, it is easy to verify that at every step of the algorithm the submatrix block S is always "weakly" the same sign pattern as A. It is also easy to see that at any stage of the algorithm, if there is an element in the first row or column of V that is not zero, or an element in the submatrix S of U that does not have the same sign as its corresponding element in A, and a link containing the position of this element cannot be found, then the problem certainly cannot have a solution. We can now prove that the algorithm actually produces a solution.

Lemma 1. *If a weak solution to problem P exists, then at the end of the line sum adjustment stage, S is a weak solution to P.*

Proof. Note that in each loop of the line sum adjustment stage two of the elements in the first row and column of V approach zero, i.e. the line sums of the submatrix S approach $\vec{r}$ and $\vec{c}$. Since a solution exists, it is sufficient to show that the algorithm will reduce all elements in the first row and column of V to zero in a finite number of steps.

Let ij be a position in the first row or column of V such that $v_{ij} \neq 0$. The algorithm finds a link L containing position ij, augments U by a real number x on link L, and then updates V with x and L. Because x is defined as the minimum of the absolute values of the elements at the positions of L, it is easy to verify that after the update either v_{ij} is zero or the positions of L are no longer a link. Since there are a finite number of links containing v_{ij}, there must also be finitely many steps before v_{ij} is reduced to zero. □

Theorem 4. *If a solution to problem P exists, then at the end of the algorithm, S is a solution to problem P.*

Proof. It is easily verified that after each loop in the sign adjustment stage at least one element in S that previously did not have the correct sign is made to have the correct sign, without changing the signs of other elements in S that already have the same sign as their corresponding elements in A, and without changing the row and column sums of S. Since S is already a weak solution at the end of the line sum adjustment stage, it is sufficient to show that the sign adjustment stage corrects all the signs in a finite number of steps. But there are a finite number of elements in S so there are also a finite number of steps to the sign adjustment stage. □

6. Proof of main results

Now that we have discussed our algorithm, our main results are derived naturally from it. We devote this section to proving the theorems in section 2.

Proof of theorem 1. It is obvious that if a problem $P(A, \vec{r}, \vec{c})$ has a weak integer solution then the elements of $\vec{r}$ and $\vec{c}$ are all integers. To prove the converse, we note that in the line sum adjustment stage, if the elements of $\vec{r}$ and $\vec{c}$ are all integers then a non-integer will never be augmented on U. Because of this fact and lemma 1, a problem $P(A, \vec{r}, \vec{c})$ with the elements of $\vec{r}$ and $\vec{c}$ integers must have a weak integer solution. □

Proof of theorem 2. Unless $\vec{r}$ or $\vec{c}$ has elements that are non-rational numbers, we can always define x to be a rational number at any stage of the algorithm, so we are never forced to augment a non-rational number on U. Since at the end of the algorithm, the submatrix S of U is a solution, there must exist a rational solution when the elements of $\vec{r}$ and $\vec{c}$ are all integers. □

Proof of theorem 3. Let B be a solution to P. Let V be the residual array that would be obtained in the final step if B were constructed from the algorithm. It is easy to verify that if a set of positions as described in theorem 3 exists in B, then a type 5 link $L(V)$ exists in V. We can then augment a real number x on B at the positions of $L(V)$ and still retain a solution. Since there are an infinite number of ways to choose x (recall that x is defined as a positive real number less than the smallest of the absolute values of the elements of V at the specific positions of link L) while still retaining a solution, there are an infinite number of solutions.

Now suppose that a set of positions as described in the theorem does not exist in B. It is easy to verify that no type 5 link exists in V. If a type 5 link cannot be found in V then there is no way to change the elements of the solution without changing the row and column sums, so the solution is unique. □

7. Concluding remarks and possible directions for further study

We have seen that our algorithm answers some interesting questions regarding the existence of integer solutions and the uniqueness of solutions in general to a given problem P. We conclude by posing a number of questions that the reader may find interesting and deserving of further research.

Expanding on corollary 2, we ask: what is the smallest integer N such that a problem $P(A, N\vec{r}, N\vec{c})$, where all the elements of $\vec{r}$ and $\vec{c}$ are integers, has an integer solution? We remark that an obvious upper bound exists for N, that is $N \leq 2^{mn}$. This is obvious because we have a weak integer solution at the end of the line sum adjustment stage of the algorithm and there are at most mn elements whose signs must be changed in order to produce a solution. Thus constructing a solution takes at most mn iterations of the sign adjustment stage (since each iteration makes at least one previously incorrect sign correct). Suppose we set x each time to be half the minimum of the absolute values of the elements of V at the positions of the link. It is easy to see that the largest denominator for any element in the solution that can be generated this way is 2^{mn}, so N cannot be greater than 2^{mn}. This is, of course, a very bad upper bound because it does not take into account the nature of A, $\vec{r}$, or $\vec{c}$ at all. It would be interesting to see if better bounds for N can be defined, and even more interesting to determine when $N = 1$, i.e. when the problem has an integer solution.

In theorem 3 we saw that a problem either has a unique solution or an infinite number of solutions and the proof of it was relatively straightforward. A much more difficult question involves, once again, the existence of integer solutions. We ask: when does a problem P have a unique *integer* solution? And more generally we ask: how may we determine the size of the set of integer solutions to a problem P?

Finally, we remark that the worst-case efficiency of our algorithm is nonpolynomial because the maximum number of possible links in an $m \times n$ matrix is $\sum_{i=3}^{mn} \binom{mn}{i}$, so obviously the maximum number of iterations of the algorithm does not grow as a polynomial function of mn. However, because our problem's graph theory equivalent is only a special case of the general feasible circulation problem, we expect our algorithm to have a better efficiency than the algorithm for solving feasible circulation problems given in [3]. As such, our algorithm may find applications in computer science and optimization problems.

References

[1] Charles R. Johnson, Suzanne A. Lewis, Donald Y. Yau, Possible line sums for a qualitative matrix, Linear Alg. Appl. 327 (2001), no. 1–3, 54—60.

[2] M. Menon, Matrix links, an extrimization problem, and the reduction of a nonnegative matrix to one with prescribed row and column sums, Can. J. Math. 20 (1968), 225—232.

[3] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory Algorithms and Applications, Prentice–Hall, Englewood Cliffs, NJ, 1993.

Department of Mathematics, University of Illinois at Urbana Champaign, Urbana, IL 61801, USA

E-mail address: ekung@uiuc.edu