Variations on Matrix Balancing for Telecommunication Demand Forecasting

Abstract

Matrix balancing is a term that describes the process of altering the elements of a matrix to make it conform to known regularity conditions while still remaining 'close' to the original matrix. Often the regularity conditions relate only to the row and column sums, which yields problems having a desirable 'network' structure. We describe an application in telecommunication demand forecasting that additionally requires the matrix to be symmetric. Although this requirement destroys the network constraint structure, we show that for certain objectives row and column sum conditions are sufficient to ensure symmetry. In addition, we describe a rounding procedure for generating heuristic integer solutions.

The problem of adjusting the elements of a matrix so that they satisfy certain consistency requirements but still remain ‘close’ to the original matrix is generically referred to as *matrix balancing*. Matrix balancing problems arise in a wide range of practical contexts that include accounting, transportation, and demographics. These and several other applications are reviewed in an excellent overview by Schneider and Zenios [20].

In a typical matrix balancing problem, we have a matrix that estimates certain quantities of interest, but these estimates do not satisfy consistency requirements that the actual values are known to satisfy. An example might be estimating the elements of a transition probability matrix which we know to be doubly stochastic. Consistency with the doubly stochastic property requires that the rows and columns sum to one. The doubly stochastic matrix is an example of one of two types of matrix balancing problems discussed by Schneider and Zenios [20]. They are

- adjusting the elements of a matrix so that the row and column sums equal certain prescribed values;
- adjusting the elements of a square matrix so that its row and column sums are equal to each other, but not necessarily to any prescribed values.

Both the application we consider, and the doubly stochastic condition mentioned above yield matrix balancing problems of the first type.

The conditions imposed on the row and column sums are called *balance* conditions, and a matrix that satisfies the balance conditions is said to be *balanced*. In the applications considered by Schneider and Zenios [20], the balance conditions relate only to row and column sums. More generally, the balance conditions can be restrictions on the sums of various combinations of matrix elements. (See, for example, Censor and Zenios [6].) The fair representation problem considered by Balinski and Demange [5] is one example. In the most general case, the balance conditions can be any set of linear restrictions on the matrix entries.

For a particular set of balance conditions there may be a large number of balanced matrices, but in matrix balancing, we seek a balanced matrix that is *close* to the original matrix. Schneider and Zenios [20] review several methods for obtaining such matrices. These methods typically fall into one of two categories: 1) scaling methods [2, 5, 6, 20]; and 2) mathematical programming methods [25, 20]. Of the two approaches, the mathematical programming-based methods are more flexible in the sense that they are easily adapted to accomodate changes in the underlying model, such as including bounds or objective weights [20]. The mathematical programs that result from many matrix balancing problems are nonlinear network flow problems that are solved quite efficiently using general purpose nonlinear network solvers such as that of Mulvey and Zenios [17]. However, the applicability of network flow models depends upon the type of balance conditions that are imposed.

In this paper, we consider a problem that arises in telecommunication demand forecasting and use it to motivate discussion of variations on matrix balancing that incorporate symmetry and integrality conditions. The symmetry requirement essentially changes the constraints of our model from those of a network flow problem to those of a matching problem. However, we show that for certain intuitively appealing objective functions, standard matrix balancing formulations do, in fact, yield optimal *continuous* solutions. Ultimately, our goal is to produce symmetric *integer* matrices. This additional requirement transforms our problem to one of finding a perfect b-matching that optimizes a convex separable non-linear objective. While it may be the case that this problem is infeasible, we can always identify an "almost-perfect" b-matching that suits the needs of this particular application.

1 The Telecommunications Demand Forecasting Problem

Our application arises in forecasting demands between wirecenters in a telecommunication network. These forecasts are used in a variety of network planning activities. Placing and sizing links and switches would be among the most important.

The particular demands that we consider are for symmetric broadband services or narrowband services, like ordinary voice telephone. These services involve symmetric two-way communication, so meaningful forecasts for such products should, likewise, be symmetric. A symmetric forecast requires that the forecasted demand from wirecenter A to wirecenter B be the same as that from B to A.

It is often the case that there are no historical measurements of inter-wirecenter traffic, but there are measurements or forecasts of the aggregate amount of demand at each wirecenter. Given the aggregate demands at the wirecenters (either measured or forecast), econometric models are used to disaggregate these values to obtain forecasts for the inter-wirecenter demands. While these econometric forecasts contain valuable information, they are not typically symmetric, so they cannot be used directly for planning.

This is the point at which our application begins. We are given a matrix M that contains inter-wirecenter forecasts derived from econometric models. An element m_{ij} represents the forecast for the demand from wirecenter i to wirecenter j. The elements along the diagonal are forecasts of within-wirecenter demand. The forecasts are all integer and the sum of the forecasts along any row is equal to the forecast of the aggregate demand at the associated wirecenter. Our goal is to find another matrix that is 'close' to M, that preserves desirable properties that M already possesses, but is also symmetric. The particular properties of M that we wish to preserve are: integrality; its row sums; and its ratio between intra- and inter-wirecenter traffic. The reason for imposing the integrality requirement is simply that these demands would necessarily be integer amounts. If our model does not produce integer forecasts, they would likely be obtained by some ad hoc means. We preserve the ratio between intra- and inter-wirecenter traffic only because there may be some inherent difference in the forecasts—for instance, we may have better data on intra-wirecenter traffic. Preserving this ratio is not really a hard constraint but is more a guideline to prevent altering the character of the solution too much. As a result, this particular condition can be—and ultimately is—relaxed in a controlled way to obtain an integer solution.

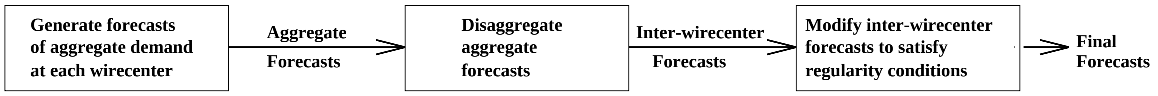
Generate forecasts of aggregate demand at each wirecenter
Aggregate Forecasts
Disaggregate aggregate forecasts
Inter-wirecenter Forecasts
Modify inter-wirecenter forecasts to satisfy regularity conditions
Final Forecasts

Figure 1: Overview of the demand forecasting process.

An overview of the demand forecasting process is provided in Figure 1. The first two steps produce the aggregate and disaggregate forecasts, respectively. The metholodology underlying these processes is described in detail in [23, 24]. This paper addresses only the final step in which the demands are modified to conform to known regularity conditions. This final step seeks a matrix that is close to the matrix, M, computed in the second step and has the following properties:

1. it has all integer entries;
2. its row sums are equal to those of M;
3. it is symmetric; and
4. its diagonal is the same as that of M.

Note that given the second condition, fixing the diagonal is equivalent to fixing the ratio between intra- and inter-wirecenter demand at each wirecenter. The last three conditions could be considered our "balance conditions".

One final point relating to our application is that the techniques we develop will ultimately be imbeded in decision support tools used by network planners. This brings up several issues that may not be of concern in more theoretical treatments of this problem. First, the procedures are likely to be used in tools that are somewhat interactive, so they should be fast on moderate-sized problems of perhaps several hundred nodes. Second, the software employed should be as general as possible (without sacrificing efficieny) so that it is adaptable to variations in the underlying model. Third, the algorithm should always produce a solution to a reasonable instance of the problem. In this case, we consider a problem instance to be "reasonable" whenever a balanced *continuous* solution exists. Our application requires that we provide an integer solution for any reasonable instance.

Of course, it's simple to demonstrate that even when a balanced matrix exists, a balanced integer matrix may not. A simple 3×3 example suffices. Suppose our matrix of forecasts is

$$\begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

We seek a new matrix

$$\begin{bmatrix} 0 & a & b \\ a & 0 & c \\ b & c & 0 \end{bmatrix},$$

whose entries a, b, and c are the unique solution to the system

$$\begin{aligned} a + b &= 3, \\ a + c &= 2, \\ b + c &= 2. \end{aligned} \tag{1}$$

The solution is $a = \frac{3}{2}$, $b = \frac{3}{2}$, and $c = \frac{1}{2}$. Hence, no balanced integer matrix exists.

To provide an integer solution we need to relax some of the constraints. For the current application, we will be allowed alter any diagonal element by one, in order to assure that an integer solution (albeit to a relaxed problem) can be found.

By allowing this flexibility, a solution to the above problem is immediately available:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

We examine these feasibility issues in considerable detail in the next two sections. Our reason for mentioning them here is simply to point out that relaxing the fixed-diagonal requirement is critical to producing integer solutions.

2 Striking a Balance

For the time being, we will omit the integrality requirements and examine the continuous version of the problem. Here, the main difference between our problem and those considered by Schneider and Zenios is the symmetry requirement. Unfortunately, enforcing symmetry in the constraints of the optimization model transforms it from a nonlinear *network* model to a more general linearly constrained nonlinear program. Solving these more general problems typically requires more time and more sophisticated software. Thus, it is certainly advantageous to employ network models and solution techniques wherever possible. One of our goals in this section is to show that it's possible to produce continuous symmetric matrices with the desired properties using network models.

2.1 A brief review of matrix balancing

We begin our discussion of optimization approaches by presenting a standard matrix balancing formulation for producing matrices with prescribed row and column sums. This formulation appears in [20]. Suppose that we are given an $n \times n$ nonnegative matrix M and positive vectors s and d, both in $\Re^n$, that provide target row and column sums. The associated matrix balancing problem can be written

[MB]:

$$\text{minimize} \quad \sum_{i,j} f_{ij}(x_{ij})$$

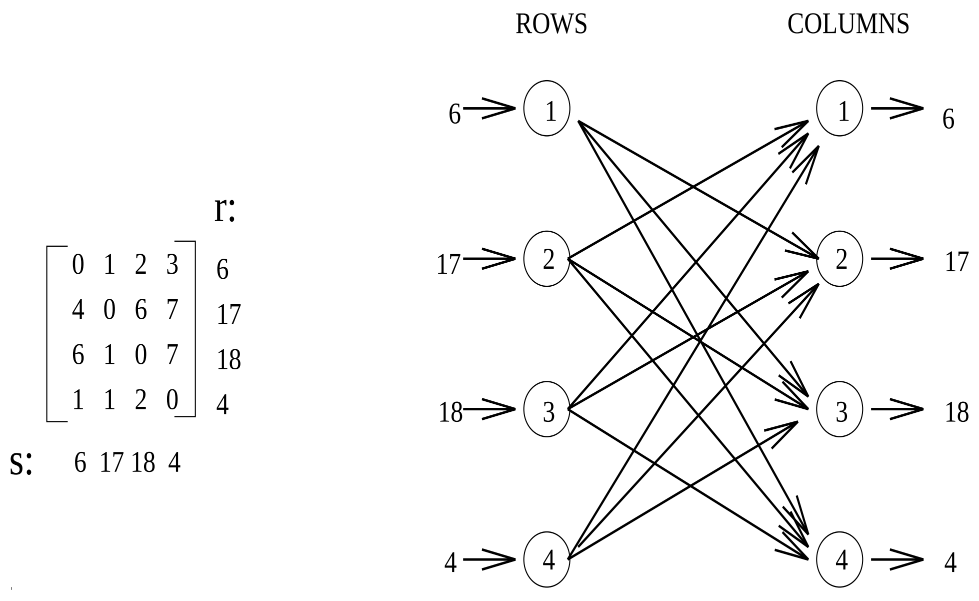

Figure 2: A matrix and it's associated transportation network.

$$\begin{array}{rrcll} \text{subject to:} & \sum_j x_{ij} & = & s_i & \forall\, i = 1, \ldots, n \\ & \sum_i x_{ij} & = & d_j & \forall\, j = 1, \ldots, n \\ & x_{ij} & \geq & 0, & \forall\, i, j = 1, \ldots, n \\ & x_{ij} & > & 0 & \text{only if} \quad m_{ij} > 0. \end{array}$$

The constraints in this model can be viewed as the flow-balance equations in an associated transportation problem. Figure 2 provides an example of a small matrix and its representation as a transportation network. There is one left-hand node associated with each row of M and one right-hand node associated with each column of M. There is a link from left-hand node i to right-hand node j whenever the corresponding matrix element m_{ij} of M is nonzero. Left-hand node i has supply s_i, while right-hand node i has demand d_i. To complete the network flow description of the problem, we associate a cost $f_{ij}(x_{ij})$ of sending x_{ij} units of flow on the link from i to j and try to minimize the total cost of satisfying demands. The optimal flows x_{ij} provide the new balanced matrix.

The objective function employed in matrix balancing is typically separable, nonlinear, and convex. The role of the objective is clearly to penalize deviations from the original matrix. Nonlinear objectives are attractive because they promote balance among the deviations by penalizing large deviations disproportionately more. Schneider and Zenios [20] note that quadratic and entropy penalty functions are the ones that are typically used in practice.

Quadratic objective functions that minimize the (weighted) sum of squared deviations from the target matrix have been more widely studied. For problem [MB], we obtain the quadratic penalty objective by letting

$$f_{ij}(x_{ij}) = w_{ij}(x_{ij} \; - \; m_{ij})^2, \tag{2}$$

where the w_{ij}'s are nonnegative weights. The resulting problem has a separable quadratic objective and transportation constraints. Bachem and Korte [3, 4], Cottle *et al.* [8], and Klincewicz [11] all designed solution algorithms that exploit the inherent structure of such problems. More recently, Cosares and Hochbaum [7] have shown that both the continuous and integer versions of the resulting quadratic transportation problem are solvable in strongly polynomial time for a fixed problem size. *(Note: Check to make sure that this is still true when bipartite graph isn't complete.)* Arbitrary instances of the real-valued problem are solvable in polynomial time by virtue of a result of Minoux [14]. Polynomial solvability is also guaranteed by more general results on the polynomial solvability of convex quadratic programs, such as those of Monteiro and Adler [16]. Minoux [15] and Hochbaum and Shantikumar [10] provide polynomial-time algorithms for the integer-valued problem.

Alternatively, the entropy objective defines

$$f_{ij}(x_{ij}) = w_{ij}x_{ij}\left[\ln(\frac{x_{ij}}{m_{ij}}) - 1\right].$$

Many of the common scaling algorithms for matrix balancing implicitly optimize an entropy-like function. (See, for example, [2, 20, 5].) Zenios, Drud and Mulvey [25] employ entropy functions in the context of network models for matrix balancing. The integer version of the problem is polynomially solvable because convex cost network flow problems are polynomially solvable [15, 10]. (See also [1].)

2.2 Symmetric balancing

With the description of the above problem in mind, we return to the continuous version of the telecommunications balancing problem. Note that the preceding model explicitly fixes some of the matrix elements to zero by omitting the associated link from the model. This presumes that a value of zero in the original matrix reflects an "impossible transaction". Such assumptions are not necessarily called for in our telecommunication model, but one can certainly imagine instances where analogous assumptions are meaningful. Consequently, we formulate subsequent models with similar restrictions, but note that removing them does not impact our analysis in any way.

We now present the following *symmetric matrix balancing* formulation

[SMB]:

$$\text{minimize} \quad \sum_{i,j} f_{ij}(x_{ij})$$

$$\text{subject to:} \quad \sum_j x_{ij} \;=\; s_i, \qquad \forall\, i = \; 1, \;\ldots, \; n \tag{3}$$

$$\begin{aligned} x_{ij} &= x_{ji}, \quad \forall\, i,\ j = 1, \ldots, n && (4) \\ x_{ii} &= m_{ii}, \quad \forall\, i = 1, \ldots, n && (5) \\ x_{ij} &\geq 0, \quad \forall\, i,\ j = 1, \ldots, n \\ x_{ij} &> 0 \quad \text{only if} \quad m_{ij} \text{ or } m_{ji} > 0, \end{aligned}$$

where $s_i = \sum_j m_{ij}$. Note that this model includes each matrix element in the formulation, like the first model in the previous section. The first set of constraints (3) fixes row sums, precisely as in [MB]. The second set of constraints (4) enforces symmetry. The third set of constraints (5) fixes the ratio between intra- and inter-wirecenter traffic, by fixing the diagonal in the new matrix to be equal to that of the original matrix. Since these values are fixed we could, alternatively, remove the associated variables from the model and adjust the row sums to compensate. In either case, [SMB] estimates only the off-diagonal entries. Finally, we fix elements to zero when *both* of the associated elements of the original matrix are zero. This is a slight modification of the usual matrix balancing condition that we make to accomodate symmetric solutions. When seeking a symmetric matrix, we really have two pieces of information associated with each new element. That is, both m_{ij} and m_{ji} can be viewed as observations associated with the single new element $x_{ij} = x_{ji}$. So, to be consistent, we fix the element to zero only when both elements of the original matrix are zero.

The fact that m_{ij} and m_{ji} are two pieces of information associated with a single new value should also play a role in the objective. Rather than set $f_{ij}(\cdot)$ as in (2), it seems natural to have a more 'symmetric' objective. The most obvious choice penalizes deviations from the average of the two observations. Thus, the quadratic penalty becomes

$$f_{ij}(x_{ij}) = w_{ij} \left(x_{ij} - \frac{m_{ij} + m_{ji}}{2} \right)^2, \tag{6}$$

and the entropy penalty becomes

$$f_{ij}(x_{ij}) = w_{ij} x_{ij} \left[\ln \left(\frac{x_{ij}}{\frac{m_{ij}+m_{ji}}{2}} \right) \right]. \tag{7}$$

Clearly, [SMB] does not have the transportation constraints that are so attractive in [MB]. However, the following relaxation of [SMB] does

[RSMB]:

$$\text{minimize} \quad \sum_{i,j} f_{ij}(x_{ij})$$

$$\begin{aligned} \text{subject to:} \quad \sum_j x_{ij} &= s_i, \quad \forall\, i = 1, \ldots, n && (8) \\ \sum_i x_{ij} &= s_j, \quad \forall\, j = 1, \ldots, n && (9) \\ x_{ii} &= m_{ii}, \ \forall\, i = 1, \ldots, n && (10) \\ x_{ij} &\geq 0, \quad \forall\, i,\ j = 1, \ldots, n \\ x_{ij} &> 0 \quad \text{only if} \quad m_{ij} \text{ or } m_{ji} > 0. \end{aligned}$$

This formulation replaces the symmetry constraints (4) with aggregate constraints (9) that require the column sums to be the same as the row sums. It's clear that any symmetric matrix has equal row and column sums, but matrices whose row and column sums are equal are not necessarily symmetric. Thus, the feasible set of solutions of [SMB] is contained in that of [RSMB]. Given a feasible solution to the relaxed problem, the following lemma shows that we can at least construct a feasible solution to [SMB] by averaging pairs of elements across the diagonal.

Lemma 1 *Given any feasible solution $X \in \Re^{n \times n}$ for [RSMB],*

$$Y = \frac{1}{2}(X + X^T),$$

where X^T is the transpose of X, is a feasible solution to both [RSMB] and [SMB].

(Proof.) The constraints of [RSMB] form a convex set that includes both X and X^T. Since Y is a convex combination of two elements in the convex feasible set, it is also feasible in [RSMB]. By construction, Y is symmetric; therefore, it is also feasible in [SMB]. ▮

Thus, a solution to [RSMB] always provides a vehicle for obtaining a feasible solution to [SMB]. However, for certain objective functions, it yields considerably more.

Theorem 1 *Let $X \in \Re^{n \times n}$ be an optimal solution to [RSMB] when the objective is some convex function $F(X)$ such that*

$$F(X) = \sum_{i<j} f_{ij}(x_{ij}) + \sum_{i<j} f_{ij}(x_{ji}).$$

Then $Y = \frac{1}{2}(X + X^T)$ is an optimal solution for both [RSMB] and [SMB].

(Proof.) By the lemma, we know that the new solution is feasible in both [RSMB] and [SMB]. If it is also optimal in [RSMB], then the result is proved. Optimality is an immediate consequence of convexity. The definition of the objective provides that $F(X^T) = F(X)$. Convexity of the objective assures that

$$F(Y) = F(\frac{1}{2}X + \frac{1}{2}X^T) \leq \frac{1}{2}F(X) + \frac{1}{2}F(X^T) = F(X).$$

Thus, the new solution is optimal in both [RSMB] and [SMB]. ▮

When $w_{ij} = w_{ji} \; \forall \; i, \; j \; = \; i, \; \ldots, \; n$, the objective that results from defining f_{ij} by either (6) or (7) has the required form for Theorem 1 to apply. What's more, both of these definitions for the f_{ij}'s yield an objective function that is strictly convex over the feasible region. In this case, the optimal solution for [RSMB] is guaranteed to be symmetric, so it immediately provides the solution of [SMB].

Theorem 2 *When the objective of [RSMB] is defined by either (6) or (7) with $w_{ij} = w_{ji} > 0$, the optimal solution is symmetric (and therefore optimal in [SMB]).*

(Proof.) Note that the resulting objective function is strictly convex. We can see this by just looking at the Hessian of the objective: it is diagonal and strictly positive over the feasible region.

We can prove the claim by contradiction. Assume that we have some optimal solution X that is not symmetric. As before, X^T is also feasible and has the same objective value. Let $F(X) = \sum_{i,j} f_{ij}(x_{ij})$, and let $Y = \frac{1}{2}(X + X^T)$. Convexity once again provides:

$$F(Y) = F(\frac{1}{2}X + \frac{1}{2}X^T) \leq \frac{1}{2}F(X) + \frac{1}{2}F(X^T) = F(X).$$

Strict convexity further provides that the inequality is strict unless $X = X^T$. Since X is not symmetric, this is not the case. Thus, Y is an improved solution that contradicts the optimality of X. ∎

The immediate implication of this result is that finding a symmetric solution is really no more difficult than solving the more typical matrix balancing problems considered in Schneider and Zenios [20]. In fact, our problem is solved as a special case. Theorem 2 clearly remains true in the more general case where there are no special restrictions on the diagonal elements. Consequently, any technique used in solving standard matrix balancing problems can also be applied to solve symmetric matrix balancing problems.

Within the context of producing symmetric matrices, we may prefer objectives that penalize deviations from ranges based on x_{ij} and x_{ji}. For instance, instead of trying to minimize deviations from $\frac{1}{2}(m_{ij} + m_{ji})$, suppose that we minimize deviations from some interval centered at this value. The most obvious choice of such intervals has m_{ij} and m_{ji} as endpoints. The simple intuition is that the two observations m_{ij} and m_{ji} bound a range of acceptable solutions, so any solution within the range is not penalized. The function (6) is just the special case where the target region consists of a single point. When the target region is an interval, the objective is a sum of piecewise quadratic functions like the one in Figure 3. When our objective has this form, it is no longer necessarily true that the optimal solution to [RSMB] is symmetric. However, the objective is convex so Theorem 1 assures that we can construct an optimal symmetric solution by averaging across the diagonal. Solving problems with this type of objective triples the size of the underlying network flow problem but still preserves it's desirable structure. We mention this only as a generalization of the basic quadratic penalty objective (6).

2.3 Feasibility

The network flow nature of the problem allows us to easily assess when the symmetric matrix balancing problem has a feasible solution. Since [SMB] has a feasible solution whenever [RSMB] does, determining feasibility is equivalent to determining the feasibility of the

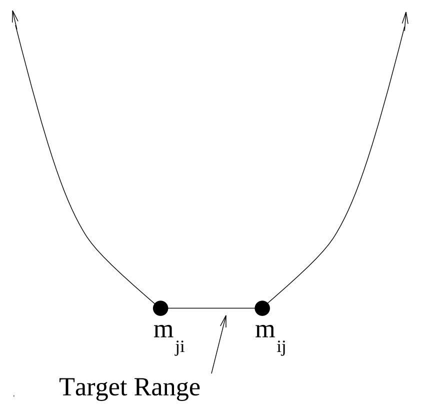

Figure 3: Objective based on penalties outside an interval.

related transportation problem [RSMB]. Thus, it's not difficult to establish just by solving a linear transportation problem.

For completeness, we will give one statement of necessary and sufficient conditions for feasible flows to exist. This statement relys on the following transportation network description of [RSMB]. Let $\mathcal{S}$ denote the left-hand nodes that correspond to the rows of X and let $\mathcal{T}$ denote the right-hand nodes that correspond to columns. In the description that follows, we commit a slight abuse of notation by referring to nodes and their corresponding row or column indices interchangably. Let there be a link from $i \in \mathcal{S}$ to $j \in \mathcal{T}$ whenever at least one of m_{ij} and m_{ji} is nonzero. To complete the description, let there be supply b_i associated with node $i \in \mathcal{S}$ and demand b_j associated with node $j \in \mathcal{T}$, where

$$b_i = s_i - m_{ii}.$$

In subsequent discussions, we refer to b as the *modified row sums*—they are the original row sums with corresponding diagonal values removed to reflect the fact that these values are fixed.

We can now state a simple feasibility theorem that follows as a special case of the circulation feasibility theorem given in Ahuja, Magnanti, and Orlin [1].

Theorem 3 *The matrix balancing problem [SMB] has a feasible solution if and only if for every subset $\bar{\mathcal{S}}$ of $\mathcal{S}$*

$$\sum_{i \in \bar{\mathcal{S}}} b_i \leq \sum_{j \in \delta(\bar{\mathcal{S}})} b_j,$$

where $\delta(\bar{\mathcal{S}})$ is the set of nodes in $\mathcal{T}$ that are reachable from $\bar{\mathcal{S}}$.

Thus, the feasibility of [RSMB] (and, hence, [SMB]) is verified by checking that the aggregate modified row sum of any subset of rows does not exceed the sums of the columns they may meet. In the special case where we are estimating a dense matrix the feasibility conditions are particularly simple: no single modified row sum can exceed the sum of all the others. We say that an input matrix M is *balanceable* if the related problem [SMB] has a feasible solution.

This type of analysis is unnecessary for a symmetric matrix balancing problem in which none of the diagonals are fixed. Here we can always obtain a feasible solution by letting the diagonal equal the associated row sum and setting all other elements to zero.

3 Rounding Things Out

We now consider the issue of obtaining balanced integer matrices. A requirement of our application is that, for any balanceable input matrix, we be able to produce a related symmetric integer matrix that satisfies the row sum conditions (3). This may require adjusting diagonal entries to alter the ratio between intra- and inter-wirecenter demand because it is certainly possible that [SMB] admits no feasible integer solution. In Section 1 we provided a simple example to demonstrate this. In this section, we examine the *integer* feasibility of [SMB], then we show how relaxing the condition on the ratio of intra- and inter-wirecenter forecasts very slightly allows us to construct the desired matrices.

3.1 Integer Feasibility

We begin with a discussion of integer feasibility of [SMB]. The related mathematical program may be stated by simply adding the integrality conditions to [SMB]. Alternatively, we can obtain a more compact formulation by using equations (4) and (5) to substitute for fixed or paired variables. If we do this, the resulting mathematical program is **[SMB2]**:

$$\text{minimize} \quad \sum_{i,j} f_{ij}(x_{ij})$$

$$\begin{aligned} \text{subject to:} \quad \sum_{j>i} x_{ij} + \sum_{j>i} x_{ji} &= \sum_{j \neq i} m_{ij}, \quad \forall\, i = 1, \ldots, n \qquad (11) \\ x_{ij} &\in \mathcal{Z}_+, \quad \forall\, i = 1, \ldots, n;\ j > i \\ x_{ij} &> 0 \quad \text{only if} \quad m_{ij} \text{ or } m_{ji} > 0, \end{aligned}$$

The advantage of this formulation is that it has a natural connection with b-matching that allows us to assess feasibility. Given a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and a vector $b \in \mathcal{Z}_+^{|\mathcal{N}|}$ a *perfect b-matching* is given by a vector $x \in \mathcal{Z}_+^{|\mathcal{A}|}$ such that

$$\sum_{j:(i,j) \in \mathcal{A}} x_{ij} = b_i \quad \forall\, i \in \mathcal{N}.$$

To establish the connection to our problem, we construct an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ that has a node associated with each row of the matrix M, and an arc (i,j) $(i < j)$ whenever

$$\left[\begin{array}{ccccc}3 & 0 & 7 & 8 & 2\\ 0 & 2 & 3 & 5 & 5\\ 1 & 9 & 0 & 0 & 4\\ 3 & 6 & 0 & 1 & 8\\ 4 & 7 & 3 & 5 & 2\end{array}\right]\begin{array}{c}20\\ 15\\ 15\\ 18\\ 21\end{array}$$

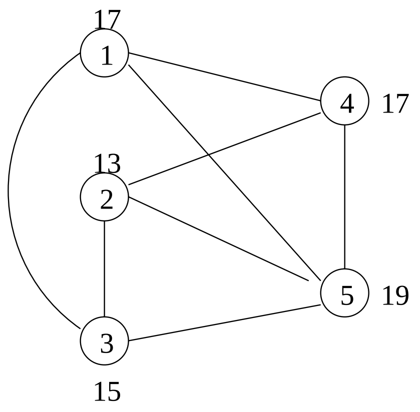

Figure 4: A matrix and an associated b-matching problem.

at least one of m_{ij} and m_{ji} are positive. Thus, there is an arc associated with each (above-diagonal) element that may have a nonzero forecast. The value b_i associated with node $i \in \mathcal{N}$ is the modified row sum defined in the previous section. Figure 4 provides an example of a matrix and its associated graph. (The nodes in Figure 4 are shown with their associated b values.)

Given this description, one can see that the coefficients forming the left-hand side of constraints (11) of [SMB2] are precisely those of the node-arc incidence matrix associated with $\mathcal{G}$. Hence, the constraints themselves are equivalent to the constraints for perfect b-matching in $\mathcal{G}$, where $b_i = \sum_{j \neq i} m_{ij}$. As a consequence of this connection with b-matching, we know that our demand forecasting problem has an integer solution if and only if $\mathcal{G}$ contains a perfect b-matching.

We note that by omitting the integrality requirement, we obtain an alternate representation of the continuous problem as a *fractional perfect b-matching.* One way to interpret the results of the previous section is that the convex separable nonlinear fractional b-matching problem is solvable as a nonlinear network flow problem. Solving linear network flow problems is a well-known means to solve linear fractional b-matching problems, so it is perhaps not surprising that the same device applies more generally. In the remainder of this paper, we will explicitly note when a matching is allowed to be fractional; otherwise, integrality should be assumed.

Matching problems have been widely studied in the literature, and as a result, a great deal is known about when certain types of matchings do and do not exist. Excellent reviews of matching results are provided in Nemhauser and Wolsey [18] and Schrijver [21], while an extensive treatment is available in Lovasz and Plummer [12].

Whether or not a perfect b-matching exists can be determined in polynomial time by virtue of a b-matching algorithm such as that of Marsh [13]. (See also Padberg and Rao [19].) Alternatively, a characterization theorem of Tutte [22] provides necessary and sufficient conditions for a perfect b-matching to exist. Schrijver [21] provides an alternate statement as a max-min result. Those results yield the following theorem.

Theorem 4 *An undirected $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ has a perfect b-matching if and only if for each subset $\mathcal{S}$ of $\mathcal{N}$,*

$$b(\mathcal{S}) \geq \beta(\mathcal{N} \backslash \mathcal{S}),$$

where:

- *$b(\mathcal{S})$ is $\sum_{v \in \mathcal{S}} b_v$; and*
- *$\beta(\mathcal{N} \backslash \mathcal{S})$ is as follows: let $\mathcal{I}$ be the set of isolated nodes in the graph induced by $\mathcal{N} \backslash \mathcal{S}$ and let t be the number of components $\mathcal{C}$ of the graph induced by $\mathcal{N} \backslash (\mathcal{S} \cup \mathcal{I})$ for which $\sum_{v \in \mathcal{C}} b_v$ is odd; then $\beta(\mathcal{N} \backslash \mathcal{S}) = b(\mathcal{I}) + t$.*

We note that for any *balanceable* instance of the problem, Theorem 3 assures that

$$b(\mathcal{S}) \geq b(\mathcal{I})$$

for any $\mathcal{S} \subseteq \mathcal{N}$. Thus, feasibility boils down to assuring that there are not too many odd components in any induced subgraph. These conditions allow us to characterize when perfect b-matchings, and hence balanced integer matrices, are guaranteed to exist. What is perhaps more relevant for our purposes is the fact that there can be many problem instances, corresponding to balanceable matrices, that do not admit integer solutions.

For instance, by letting $\mathcal{S} = \emptyset$, it's clear that $\mathcal{G}$ has no perfect b-matching if the sum of the b_i's is odd. When $\mathcal{G}$ is a complete graph, indicating that all non-diagonal matrix elements may be positive, this is the only condition under which no perfect b-matching exists, given that the corresponding matrix is balanceable. Thus, we can transform this to a feasible perfect b-matching problem by simply decreasing the largest element of b by one. In our matrix problem, this corresponds to decreasing the largest modified row sum by one and then increasing the associated diagonal element to compensate. For sparser graphs, there may be other conditions under which there is no perfect b-matching, so it becomes less obvious how to modify b (in a controlled way) to assure a perfect b-matching. The algorithm we present in the next section provides a partial answer.

Existence of a b-matching only addresses the issue of *feasibility.* To use a b-matching approach to solve our symmetric matrix balancing problem, we need to incorporate a separable nonlinear objective function similar to (6) or (7). Thus, the issue of whether a method for nonlinear b-matching would be computationally viable presents another question that we do not address here.

Some questions:

Is convex separable b-matching polynomially solvable? This is probably(?) the case – both the poly. b-matching algorithm of Marsh and the poly. algs. for convex cost network flows are based on Edmonds Karp kinds of scaling algorithms. It's no worse than pseudo-polynomial.

Can we minimally alter b in such a way that we know a perfect matching exists?

3.2 A heuristic approach

Rather than solve nonlinear b-matching problems, we can take advantage of the fact that preserving the ratio between intra- and inter-wirecenter demands is really a 'soft' constraint in our application. Thus, small violations of this condition are certainly acceptable. Ideally, our goal is to alter no diagonal element by more than one. The remainder of this section is dedicated to showing that we can identify such solutions in a computationally efficient manner that requires nothing more sophisticated than network flow solvers.

Before doing this we note that given a continuous solution it is easy to construct symmetric integer matrices that may fail to preserve the ratio between intra- and inter-wirecenter demands by larger amounts. Generally speaking, we would obtain such a matrix by rounding in some symmetric manner and then altering the diagonal values to preserve the row sums. The simplest construction is to round all elements down and then add the difference in row sums to the diagonal of the rounded matrix. The result is a symmetric, integer matrix with the required row sums. This is certainly a simple post-processing step, but, especially in the case of small wirecenters, may change the character of the demand forecasts by placing too much demand within its local area.

We now describe a method that alters no diagonal element by more than one. The method begins with an integer solution to [RSMB], that may not be symmetric. There are several ways to obtain "good" integer solutions: 1) solve the integer network flow problem to find the optimal solution; 2) round the optimal continuous solution using network flow techniques to preserve the row and column sums (see [1]); and 3) solve a piecewise linearization of the nonlinear problem using the network simplex method. The *optimal* (asymmetric) integer solution may be obtained by polynomial-time algorithms for separable nonlinear network flow problems such as those of Hochbaum and Shantikumar [10] and Minoux [15].

These network flow approaches (can) yield integer matrices in which paired elements x_{ij} and x_{ji} differ by at most one. In this sense, the matrix is almost symmetric, and it has all of the other desirable features. Thus, if we are willing to (slightly) relax the symmetry conditions instead of the fixed-diagonal conditions, network flow methods are all we need.

The fact that paired elements differ by at most one is obvious when we're just rounding the continous solution. That the optimal integer solution necessarily posesses this property is less clear, so we state and prove this in Theorem 5

Theorem 5 *When the objective of [RSMB] is defined by either (6) or (7) with $w_{ij} = w_{ji} > 0$, the optimal integer-valued solution has the property that paired elements x_{ij} and x_{ji} differ by at most one, for all $i,\ j\ =\ 1,\ \ldots,\ n$.*

(Proof.) [[Here's the idea, but it needs to be cleaned up. The simplest way to do this is to piecewise linearize the objective so that each piece is one unit long. An optimal solution to the nonlinear problem is optimal in the piecewise linear one, and vice versa. By convexity, averaging across the diagonal will yield a solution that is equal to or better than

the current one. If two elements differ by more than one unit, then they lie in different pieces of the piecewise linearization, so the solution will be improved by averaging. Probably some augmenting path kind of argument will work also.]]

Given any integer feasible solution of [RSMB], we can obtain another feasible solution that is symmetric and half-integer by averaging paired elements across the diagonal. (A *half-integer* matrix is a matrix whose elements are either an integer or an integer divided by two.) Our algorithm operates on a symmetric half-integer matrix that corresponds to a feasible solution of [RSMB]. Given such a matrix, each iteration of the algorithm selects a group of elements that may be rounded together to preserve both symmetry and the row and column sums. An iteration of the algorithm reduces the number of non-integers and preserves the invariant properties of half-integrality, symmetry, and row and column sums.

The method is stated in Algorithm 1. Given this description, it remains to show that the algorithm produces a solution with the desired properties. We do this in two parts. First we show that, at the end of each iteration, the new matrix still possesses the invariant properties—namely, we show that it is symmetric, half-integer, has a diagonal within one of the original, and has the same row sums. Second, we demonstrate that the algorithm terminates.

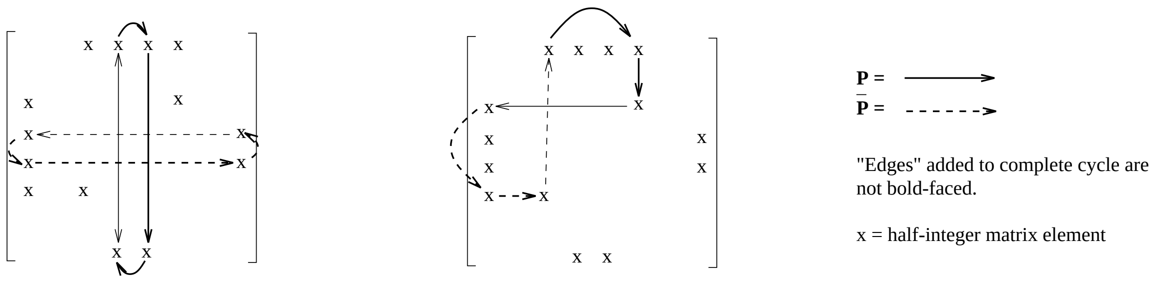
P =
$\bar{P}$ =
"Edges" added to complete cycle are not bold-faced.
x = half-integer matrix element

Figure 5: Possible "cycles" identified by Algorithm 1.

Although the algorithm is stated directly in terms of the matrix, it may be helpful to view it another way. We can construct a graph G whose nodes correspond to non-integer matrix elements and whose edges join two nodes if their associated non-integers are in the same row or column. Thus, a node is associated with a unique matrix element, so we may also use the term "node" to refer to its corresponding matrix element. The algorithm works to construct two paths, P and $\bar{P}$, that are "reflections" or "transposes" of eachother in the sense that when P visits a node, $\bar{P}$ visits the node across the diagonal from it. The algorithm builds these two paths until ultimately they close to form either one big cycle or two disjoint cycles. Figure 5 depicts each of the two cases. An iteration ends in the column scanning step when the two paths can be joined to form a single cycle. An iteration ends in the row scanning step when the paths close on themselves to form two separate cycles. The cycles visit each row and each column an even number (possibly zero) of times, and a row or column that is visited is visited twice consecutively. The properties of the paths and the cycles built from them allow us to use parity arguments to demonstrate the correctness of

```
Input: a symmetric half-integer matrix $X \in \Re^{n\times n}$ with the prescribed (integer) row and column sums.
Set vector $d \in \Re^n$ to zero.
While there are non-integers in $X$ do:
    **Initialization:**
    Let $P$ and $\bar{P}$ be ordered lists of elements of $X$.
    Select some non-integer $x_{i,j}$ to begin.
    Set $P = x_{i,j}$ and $\bar{P} = x_{j,i}$ and mark $x_{i,j}$ and $x_{j,i}$ as visited.
    Set the 'magic column' $m$ to be $j$.
    Set the current row $r = i$.

    **Scan row $r$:**
    If $x_{r,m}$ is an unmarked non-integer then:
        Mark $x_{r,m}$ and $x_{m,r}$.
        Set $P = P, x_{r,m}$ and set $\bar{P} = \bar{P}, x_{m,r}$
        **Do rounding (case 1):**
            Beginning with the first element of $P$,
                round the elements alternately up and then down.
            Beginning with the first element of $\bar{P}$.
                round the elements alternately up and then down.
            Goto **END**.
    Else, select some unmarked non-integer element in row $r$, say $x_{r,k}$
        Mark $x_{r,k}$ and $x_{k,r}$.
        Set $P = P, x_{r,k}$ and set $\bar{P} = \bar{P}, x_{k,r}$
        Let $c = k$.

    **Scan column $c$:**
    If $x_{m,c}$ is an unmarked non-integer then:
        Mark $x_{m,c}$ and $x_{c,m}$.
        Set $P = P, x_{m,c}$ and set $\bar{P} = \bar{P}, x_{c,m}$
        **Do rounding (case 2):**
            If $d_m > 0$,
                Beginning with the first element of $P$,
                    round the elements alternately down and then up.
                Beginning with the first element of $\bar{P}$.
                    round the elements alternately down and then up.
                Set $d_m = 0$, and decrease $x_{m,m}$ by one.
            If $d_m = 0$,
                Beginning with the first element of $P$,
                    round the elements alternately up and then down.
                Beginning with the first element of $\bar{P}$.
                    round the elements alternately up and then down.
                Set $d_m = 1$, and increase $x_{m,m}$ by one.
        Goto **END**.
    Else, select some unmarked non-integer element in column $c$, say $x_{k,c}$
        Mark $x_{k,c}$ and $x_{c,k}$.
        Set $P = P, x_{k,c}$ and set $\bar{P} = \bar{P}, x_{c,k}$
        Let $r = k$.
        Goto **Scan row**.
**END**
```

Algorithm 1: *Symmetric Rounding*

the algorithm. These properties arise because the matrix is half-integer, which means that the non-integers are indistinguishable from eachother, so our algorithm merely has to keep elements are properly "paired" when it rounds.

We now verify that Algorithm 1 produces a solution with the desired properties. We begin by showing that each iteration preserves the invariant properties. We consider that an iteration ends when it reaches the **END** statement. The only two ways to reach the end is via one of the two rounding steps. Thus, to complete an iteration, we must round some element(s) of the matrix. In the next two propositions, we show how the particular choice of elements to round in an iteration and the way they are rounded preserves the invariant properties from one iteration to the next.

Proposition 1 *If we begin an iteration with a symmetric half-integer matrix with integer row sums and the iteration ends after performing a case 1 rounding, then the matrix produced has the same row sums and diagonal as when the iteration began, and it is symmetric and half-integer.*

(Proof.) Since the matrix is half-integral at the beginning of the iteration, and we do nothing more than round some of its elements, the new matrix is half-integral, as well.

The only elements that we round are in P and $\bar{P}$. P and $\bar{P}$ are reflections of eachother in the sense that when P visits x_{ij}, $\bar{P}$ visits x_{ji}. Thus, they visit the same sequence of paired elements, but visit different elements of the pair. Since the sequence of roundings is the same in both P and $\bar{P}$, we are assured that pairs of elements are rounded together and in the same direction. Thus, the new matrix remains symmetric. The diagonal is unchanged because it is necessarily all integer, so there can be no diagonal elements in P or $\bar{P}$.

To see that an iteration preserves row sums, note that P visits each row and each column an even number of times and a row or column that is visited is visited twice consecutively. (Note that for visits to column m to be 'consecutive', we need to view P as a cycle.) By alternating the direction in which we round the elements of P, the number of round-ups in any row or column must equal the number of round-downs. Since the matrix is half-integer, the non-integer part of any non-integer is the same, so the row sum is preserved as long as the number of round-ups is equal to the number of round-downs. Thus, rounding the elements of P preserves row sums. The same type of argument holds for $\bar{P}$. ∎

Proposition 2 *If we begin an iteration with a symmetric half-integer matrix with integer row sums and the iteration ends after performing a case 2 rounding, then the matrix produced has the same row sums as when the iteration began, has a diagonal within one, and is symmetric and half-integer.*

(Proof.) The matrix is symmetric and half-integer for the same reasons as above. Demonstrating the row sum condition is analogous, but slightly more complicated, than for case 1. We once again examine P and $\bar{P}$ and note that both P and $\bar{P}$ visit all rows and columns

except m an even number of times. In addition, except for P's first visit to column m and last visit to row m, any row or column visited is visited twice consecutively. Thus, the alternating rounding of P preserves all row and column sums except m. If the first element of P is rounded up, then the sum across row (and column) m is increased by a half; otherwise, it decreases by a half. Similarly, $\bar{P}$ visits all rows and columns except m an even number of times, and visits are paired except for the first visit to row m and the last visit to column m. When the first element of $\bar{P}$ is rounded up, the sum across row (and column) m increases by a half; otherwise, it decreases by that amount. In total, rounding the elements of P and $\bar{P}$ either increases or decreases the sums across row and column m by one. We compensate by altering the diagonal element x_{mm}. Note that d is an indicator vector whose m^{th} element is one if x_{mm} is already one greater than its original value, and is zero when x_{mm} is at its original value. Thus, if x_{mm} has already been increased, we begin by rounding the elements of P and $\bar{P}$ up and preserve the row sum by restoring x_{mm} to its original value. Otherwise, we begin rounding the elements of P and $\bar{P}$ down and fix the row sum by increasing x_{mm} by one. Thus, the rounding does preserve row sums. It also keeps the diagonal within one of its original value because the indicator d_m prevents us from either increasing or decreasing x_{mm} twice consecutively. Hence, the diagonal iterates between its original value and its original value plus one. ∎

Theorem 6 *If the algorithm begins with a symmetric half-integer matrix with integer row sums, then it terminates with a symmetric integer matrix whose row sums are the same as the original matrix and whose diagonal elements are within one.*

Propositions 1 and 2 guarantee that as long as we begin an iteration with a symmetric, half-integer matrix with integer row sums, and we reach one of the two rounding steps, we end the iteration with a symmetric, half-integer matrix with the same row sums and diagonals within one of the initial matrix. Moreover, each iteration reduces the number of non-integers whenever it reaches a rounding step. Thus, proving the claim reduces to showing that each iteration reaches one of the two rounding steps.

Assume that at some point we cannot reach one of the rounding steps for the first time. This means that we have reached a point where we cannot find another unvisited non-integer in our current row or column. Let's examine both cases. Suppose that we are scanning row r and are unable to find another unmarked non-integer in row r. So far, P has visited row r an odd number of times and $\bar{P}$ has visited row m an odd number of times. They visit all other rows an even number of times. Thus, either r has an odd number of non-integers, which contradicts the fact that the initial matrix is symmetric and half-integer with integer row sums, or $r = m$. However $r \neq m$, because upon reaching row m the iteration would have terminated in rounding case 2. Thus, this situation cannot occur.

Now suppose we are scanning column c and are unable to find another unmarked non-integer. When scanning a column, it is the case that $\bar{P}$ has visited all columns an even number of times, while P has visited columns c and m an odd number of times and all others an even number of times. It cannot be the case that $c = m$ because otherwise the iteration would have terminated in rounding case 1. Thus, the fact that we cannot find another unmarked non-integer in column c contradicts the assumption that we began the iteration with a symmetric half-integer matrix with integer row sums.

Therefore, each iteration is able to reach one of the two rounding steps. Since each iteration is guaranteed to reduce the number of non-integers and produce a matrix with the desired properties, the entire algorithm will ultimately produce an integer matrix with the desired properties. ∎

3.3 Sample Results

Include some sample computational results here. Describe what we did and why: piece-wise linearization, followed by heuristic.

In results note: 1) problem sizes and largest b_i (they're pretty big, so a pseudo-polynomial algorithm probably won't be appropriate); 2) time to solve netflow time to do rounding 3) in how many cases is the forecast outside the range given by x_{ij} and x_{ji} 4) what is the worst violation of this range constraint 5) how many diagonals get changed.

3.4 Matching revisited

Algorithm 1 guarantees that, if a matrix is balanceable, we can find a symmetric integer matrix that changes no diagonal element by more than one. For the related b-matching problem, this means that there exists some $\bar{b}$, such that

$$\bar{b_i} \in \{b_i, b_i - 1\} \quad \forall\, i \;=\; 1,\ \ldots,\ n,$$

for which $\mathcal{G}$ has a perfect $\bar{b}$-matching. Algorithm 1 gives us one such $\bar{b}$.

Ideally, we'd like to select a $\bar{b}$ that yields a matching that is, in some sense, optimal. The ability to change diagonal matrix elements can be incorporated in the associated b-matching model by placing loops at nodes corresponding to rows in which a diagonal element may vary. The variables associated with the loops have upper bounds that effectively limit the change in any associated diagonal element. In the problem that we've presented, all of these upper bounds are equal to one, but more general models could allow diagonal elements to be increased or decreased within individually specified bounds. Changes in the diagonal elements may be penalized in the objective, just as they are for other matrix elements.

It is straightforward to allow diagonal elements to vary by larger amounts in the context of the heuristic approach, as well. If none of the diagonal elements are fixed at a prescribed value, then the heuristic also delivers a feasible solution. To do this, we include variables corresponding to diagonal elements in the network flow model. If the i^{th} diagonal element is allowed to vary between l_i and u_i, the associated variable in the network flow model will have bounds l_i and u_i. Given feasible integer solution for the network flow problem, we average across the diagonal and apply a slight modification of Algorithm 1. The indicator d_i in the algorithm will now be -1, 1, or 0, depending upon whether the diagonal is at its lower bound, its upper bound, or between bounds. Now, if the diagonal must be changed to preserve row and column sums in a rounding step, we round in a direction that would ensure that the diagonal remains within its bounds and update d_i to reflect the new status of the diagonal element.

Loops may also be used to model flexibility in the row sum conditions. If the row sums are allowed to fall within some range, the constraints (11) are replaced by

$$b_i^L \leq \sum_{j>i} x_{ij} + \sum_{j>i} x_{ji} \leq b_i^U, \quad \forall\, i = 1, \ldots, n,$$

where b_i^L and b_i^U denote the allowable range. Censor and Zenios consider allowing range constraints in the context of standard matrix balancing models in [6]. Range constraints in symmetric matrix balancing can be interpreted in the context of the b-matching model as follows: the b-value at node i becomes b_i^U and there is a loop at node i whose corresponding variable has a lower bound of 0 and an upper bound of $b_i^U - b_i^L$. A feasible perfect b_i^U-matching corresponds to a symmetric integer matrix whose row and column sums fall into the prescribed range. Once again, a modification to the network flow model, yields an optimal continuous solution to the ranged problem. Network flow solutions are easily modified to yield feasible symmetric integer solutions when every upper bound is larger than its associated lower bound.

Finally, the matching model can be adapted to find solutions that minimize the *number* of changes to the diagonal. We begin with a graph formed in the usual way, then we add a new node v and a new arc between v and every other node. These new arcs have upper bounds of 1. We also add a loop at node v. If we let b_v be the number of rows in our associated matrix, a maximum weight perfect (capacitated) b-matching, where the weight on the loop at node v is 1 and all other weights are zero, corresponds to a solution that changes the diagonal as little as possible. Once we know how many times the diagonal must change, we can solve a separable nonlinear perfect b-matching in this graph to find a symmetric integer matrix that is close to our original matrix *and* changes as few diagonals as possible. To do this, we replace the objective with one that penalizes deviations from $\frac{1}{2}(x_{ij} + x_{ji})$, and include a lower bound on the variable associated with the loop at node v. This lower bound is equal to the number of rows in the corresponding matrix minus the number of required diagonal changes. For the small example provided in Section 1, we know that one diagonal element must be changed in order to find a balanced, symmetric, integer matrix. Figure 6 shows the associated b-matching model, if our goal is to find a solution that changes only one diagonal element.

4 Conclusions

It's clear that a wide range of symmetric matrix balancing problems can be modeled in the context of b-matching. The problem of forecasting symmetric services that we have presented is one ready application.

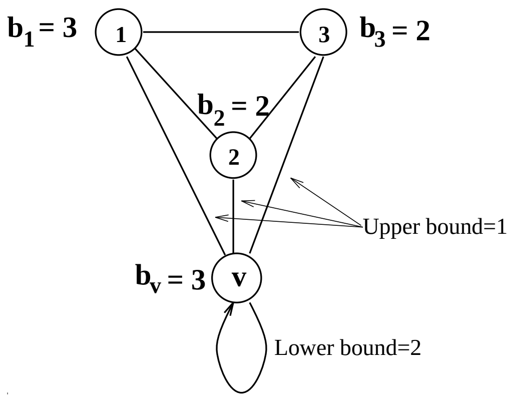

Figure 6: b-matching model that changes only one diagonal element.

We've used this application to motivate the discussion of several simple variations on classical matrix balancing models. The two twists that we've considered are symmetry and integrality. In Table 1, we summarize the effect of adding these constraints individually and in tandem. The conditions that we imposed on allowable changes to diagonal elements don't change the nature of the formulation, so they are not included in Table 1. The constraints on diagonal elements do affect feasibility. However, we've shown that, unless elements are fixed, any balanceable instance of the problem admits an integer symmetric feasible solution.

Additional Constraints	Solution technique
integrality	integer convex separable network flow
symmetry	convex separable network flow
integrality and symmetry	convex separable perfect b-matching

Table 1: Added constraints and the effect on solution technique.

It's clear that fast methods for solving convex separable b-matching problems have a ready application in forecasting demands for symmetric services. The viability of these methods for realistic instances needs to be explored. In the meantime, the heuristic method that we have described for obtaining an "almost-perfect" b-matching satisfies our immediate needs. These heuristics are employed in the forecasting tool described in [24].

References

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

[2] M. Bacharach. *Bi-proportional Matrices and Input-Output Change*, Cambridge University Press, U.K., 1970.

[3] A. Bachem and B. Korte. “An algorithm for quadratic optimization over transportation polytopes”, *Zeitschrift fur Angewante Mathematik und Mechanik*, 58, 549–461.

[4] A. Bachem and B. Korte. “Minimum norm problems over transportation polytopes”, *Linear Algebra and its Applications*, 31, 103–118, 1980.

[5] M. L. Balinski and G. Demange. “Algorithms for proportional matrices in reals and integers”, *Mathematical Programming*, 45, 193–210, 1989.

[6] Y. Censor and S. A. Zenios. “Interval-constrained matrix balancing”, *Linear Algebra and its Applications*, 150, 393–421, 1991.

[7] S. T. Cosares and D. S. Hochbaum. “Strongly polynomial algorithms for the quadratic transportation problem with a fixed number of sources”, *Mathematics of Operations Research*, 19, 94–111, 1994.

[8] R. W. Cottle, S. G. Duvall, and K. Zikan. “Lagrangean relaxation algorithm for the constrained matrix problem”, *Naval Res. Log. Quart.*, 33, 55–76, 1986.

[9] L. H. Cox and L. R. Ernst. “Controlled rounding”, *INFOR*, 20, 423–432, 1982.

[10] D. S. Hochbaum and J. G. Shantikumar. “Convex separable optimization is not much harder than linear optimization”, *Journal of the ACM*, 37, 843–862, 1990.

[11] J. G. Klincewicz. “Implementing an exact Newton method for separable convex transportation problems”, *Networks*, 19, 95-105, 1989.

[12] L. Lovasz and M. D. Plummer. *Matching Theory*, North-Holland, Amsterdam, 1986.

[13] A. B. Marsh, III. *Matching Algorithms*, Johns Hopkins University, Ph.D Thesis, 1979.

[14] M. Minoux. “A polynomial algorithm for minimum quadratic cost flow problems”, *European Journal of Operational Research*, 18, 377-387, 1984.

[15] M. Minoux. “Solving integer minimum cost flows with separable convex cost objective polynomially”, *Mathematical Programming Study*, 26, 237–239, 1986.

[16] R. D. C. Monteiro and I. Adler. “Interior path-following primal-dual algorithms. Part II: Convex quadratic programs”, *Mathematical Programming*, 44, 43–66, 1989.

[17] J. M. Mulvey and S. A. Zenios. “GENOS 1.0: A generalized network optimization system”, Decision Sciences Working Paper 87-12-3, The Wharton School, University of Pennsylvania, Philadelphia, PA.

[18] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.

[19] M. W. Padberg and M. R. Rao. “Odd minimum cut-sets and b-matchings”, *Mathematics of Operations Research*, 7, 67–80, 1982.

[20] M. H. Schneider and S. A. Zenios. “A comparative study of algorithms for matrix balancing”, *Operations Research*, 38: 439–455, 1990.

[21] A. Schrijver. “Max-min results in combinatorial optimization”, in *Mathematical Programming the State of the Art*, A. Bachem, M. Grotschel, and B. Korte, eds., Springer, 1983.

[22] W. T. Tutte. “The factors of graphs.”, *Canadian Journal of Mathematics*, 4, 314–328, 1982.

[23] O. Wasem. “Narrowband interwirecenter forecasting tool: methodology”, Bellcore Technical Memorandum TM-ARH-023543, November, 1993.

[24] O. J. Wasem, A. M. Gross, G. A. Tlapa. “Forecasting broadband demand between geographic areas”, *IEEE Communications Magazine*, 33, Number 2, 50–57, 1995.

[25] S. A. Zenios, A. Drud, J. M. Mulvey. “Balancing large social accounting matrices with nonlinear network programming”, *Networks*, 19, 569–585, 1989.