

**A Splitting Equilibration Algorithm for the  
Computation of Large-Scale Constrained  
Matrix Problems: Theoretical Analysis and  
Applications**

by

*Anna Nagurney and Alexander Eydeland*

OR 223-90

July 1990



**A Splitting Equilibration Algorithm  
for the Computation of  
Large-Scale Constrained Matrix Problems:  
Theoretical Analysis and Applications**

*Anna Nagurney*

Department of General Business and Finance  
School of Management

*Alexander Eydeland*

Department of Mathematics and Statistics  
University of Massachusetts  
Amherst, Massachusetts, 01003

June, 1990



## Abstract

In this paper we introduce a general parallelizable computational method for solving a wide spectrum of constrained matrix problems. The constrained matrix problem is a core problem in numerous applications in economics. These include the estimation of input/output tables, trade tables, and social/national accounts, and the projection of migration flows over space and time. The constrained matrix problem, so named by Bacharach, is to compute the best possible estimate  $X$  of an unknown matrix, given some information to constrain the solution set, and requiring either that the matrix  $X$  be a minimum distance from a given matrix, or that  $X$  be a functional form of another matrix. In real-world applications, the matrix  $X$  is often very large (several hundred to several thousand rows and columns), with the resulting constrained matrix problem larger still (with the number of variables on the order of the square of the number of rows/columns; typically, in the hundreds of thousands to millions). In the classical setting, the row and column totals are known and fixed, and the individual entries nonnegative. However, in certain applications, the row and column totals need not be known a priori, but must be estimated, as well. Furthermore, additional objective and subjective inputs are often incorporated within the model to better represent the application at hand. It is the solution of this broad class of large-scale constrained matrix problems in a timely fashion that we address in this paper.

The constrained matrix problem has become a standard modelling tool among researchers and practitioners in economics. Therefore, the need for a unifying, robust, and efficient computational procedure for solving constrained matrix problems is of importance. Here we introduce an algorithm, the splitting equilibration algorithm, for computing the entire class of constrained matrix problems. This algorithm is not only theoretically justified, but also fully exploits both the underlying structure of these large-scale problems and the advantages offered by state-of-the-art computer architectures, while simultaneously enhancing the modelling flexibility.

In particular, we utilize some recent results from variational inequality theory, to construct a splitting equilibration algorithm which splits the spectrum of constrained matrix problems into series of row/column equilibrium subproblems. Each such constructed subproblem, due to its special structure, can, in turn, be solved simultaneously via exact equilibration in closed form. Thus each subproblem can be allocated to a distinct processor.

We also present numerical results when the splitting equilibration algorithm is implemented in a serial, and then in a parallel environment. The algorithm is tested against another much-cited algorithm and applied to input/output tables, social accounting matrices, and migration tables. The computational results illustrate the efficacy of this approach.

## 1. Introduction

In this paper we introduce a general parallelizable computational method for solving a wide spectrum of *Constrained Matrix* problems. The *Constrained Matrix* problem is a core problem in numerous applications. These include the estimation of input-output tables, trade tables, and social/national accounts, the projection of migration flows over space and time, the treatment of census data, the analysis of political voting patterns, and the estimation of contingency tables in statistics. The constrained matrix problem (see Figure 1), so named by Bacharach (1970), is to compute the best possible estimate of an unknown matrix, given some information to constrain the solution set, and requiring either that the matrix be a minimum distance from the given matrix, or that it be a functional form of another known matrix. In real-world applications, the matrix is often very large (several hundred to several thousand rows and columns), with the resulting constrained matrix problem larger still (with the number of variables on the order of the square of the number of rows/columns; typically, in the hundreds of thousands to millions). In the classical setting, the row totals and the column totals are known and fixed, and the individual matrix elements nonnegative. However, in certain applications, the row and column totals need not be known a priori, but must be estimated, as well. Furthermore, additional objective and subjective inputs are often incorporated within the model to better represent the application being studied. It is the solution of this broad class of large-scale constrained matrix problems in a timely fashion that we address in this paper.

The constrained matrix approach has become a standard modelling tool among researchers and practitioners. Therefore, the need for a unifying, robust, and efficient computational procedure for solving constrained matrix problems is of importance. In this paper we introduce an algorithm, which we call the *Splitting Equilibration Algorithm*, for computing solutions to the entire class of constrained matrix problems. The algorithm splits a large-scale constrained matrix problem into series of row (supply)/column (demand) subproblems. Each such constructed subproblem, due to its special structure, can, in turn, be solved simultaneously via *exact equilibration* in closed form on a distinct processor. This algorithm is not only theoretically justified, but also fully exploits the advantages offered by the state-of-the-art computer architectures, while simultaneously enhancing the modelling flexibility.

Our computational procedure is motivated, in part, by the problem at hand - the “equilibration” of matrices (cf. Van der Sluis (1969)), and by the problem’s connection with spatial price equilibrium problems (Enke (1951), Samuelson (1952), and Takayama and Judge (1971)). Indeed, although as early as 1951, Stone recognized that the same methodology should be applied to the computation of both spatial price equilibrium problems and constrained matrix problems, the computational state of the art at the time precluded such an investigation. Furthermore, although the RAS method, which dates to Deming and Stephan (1940), is currently the most widely applied computational method in practice for the solution of the constrained matrix problem, its limitations include the use of a highly specific set of constraints and objective function and its nonconvergence in certain applications (see, e.g., Mohr, Crown, and Polenske (1987)).

The paper is organized as follows: in Section 2 we present the formulation of the general quadratic constrained matrix problem, specializing it, firstly, to the particular constrained matrix problem arising in the estimation of social accounting matrices in which the row and column totals must “balance”, and, secondly, to the case of known row and column totals. We also relate the general formulation to many currently used models.

In Section 3 we develop the splitting equilibration algorithm and provide a theoretical analysis of the algorithm, including convergence results.

In Sections 4 and 5 we empirically investigate the computational performance of the splitting equilibration algorithm on the largest quadratic constrained matrix problems reported to date using the IBM 3090-600E at the Cornell National Supercomputer Facility. Our goals include: 1). to compare the relative efficiency of the splitting equilibration algorithm to both our earlier equilibration algorithm (RC) (Nagurney, Kim, and Robinson (1990)), and the much-cited Bachem and Korte (1978) algorithm, 2). to investigate the efficiency of the new equilibration approach on the spectrum of very large constrained matrix problems, both diagonal and general, and 3). to investigate the speedups obtained with parallelization of the splitting equilibration algorithm for both diagonal and general large-scale problems.

## 2. Formulation of the General Constrained Matrix Problem

In this Section we present a general quadratic constrained matrix problem, describe several applications, and highlight the special cases. We also identify its relationship to classical spatial price equilibrium problems and asymmetric spatial price equilibrium problems, for which no equivalent optimization formulations exist.

The problem will be formulated as a minimization of the weighted squared sums of the deviations. We denote the given  $m \times n$  matrix by  $X^0 = (x_{ij'}^0)$ , and the matrix estimate by  $X = (x_{ij'})$ . Let  $s_i^0$  denote the row  $i$  total, and  $s_i$  the estimate of the row  $i$  total. Let  $d_{j'}^0$  denote the column  $j'$  total, and  $d_{j'}$  the estimate of the column  $j'$  total. Let the  $mn \times mn$  matrix  $G = (\gamma_{ij'kl'})$  denote the imposed weight matrix for the mixed variable terms  $(x_{ij'} - x_{ij'}^0) \cdot (x_{kl'} - x_{kl'}^0)$  and assume the matrix  $G$  to be strictly positive definite. Let the  $m \times m$  matrix  $A = (\alpha_{ik})$  denote the imposed weight matrix for the mixed variable terms  $(s_i - s_i^0) \cdot (s_k - s_k^0)$  and let the  $n \times n$  matrix  $B = (\beta_{j'l'})$  denote the imposed weight matrix for the mixed variable terms  $(d_{j'} - d_{j'}^0) \cdot (d_{l'} - d_{l'}^0)$ . Assume that the matrices  $A$  and  $B$  are also strictly positive definite.

Then the general constrained matrix problem may be written as follows:

$$\begin{aligned} \text{Minimize } & \left[ \sum_{i=1}^m \sum_{k=1}^m \alpha_{ik} (s_i - s_i^0) \cdot (s_k - s_k^0) \right] \\ & + \left[ \sum_{i=1}^m \sum_{j'=1}^n \sum_{k=1}^m \sum_{l'=1}^n \gamma_{ij'kl'} (x_{ij'} - x_{ij'}^0) \cdot (x_{kl'} - x_{kl'}^0) \right] \\ & + \left[ \sum_{j'=1}^n \sum_{l'=1}^n \beta_{j'l'} (d_{j'} - d_{j'}^0) \cdot (d_{l'} - d_{l'}^0) \right] \end{aligned} \quad (1)$$

subject to:

$$\sum_{j'=1}^n x_{ij'} = s_i, \quad i = 1, \dots, m \quad (2)$$

$$\sum_{i=1}^m x_{ij'} = d_{j'}, \quad j' = 1, \dots, n \quad (3)$$

and the inequality constraints:

$$x_{ij'} \geq 0, \quad \text{for all } i, j' \quad (4)$$



where the objective function represents the weighted squared sums of the deviations.

The objective function (1) permits the utilization of mixed-variable weight terms. An example of possibly fully dense  $A$ ,  $B$ , and  $G$  matrices are the inverses of the respective variance-covariance matrices (see, e.g., Mosteller and Tukey (1977), Judge and Yancey (1986)). Other examples may arise when the matrices  $A$ ,  $B$ , and  $G$  include subjective weights based on the expert knowledge of planners. We note that under the assumption of positive definiteness, the solution to (1), subject to (2) through (4) is unique.

We now describe, as an illustration, the above constrained matrix problem in the context of input/output (I/O) matrices.

An I/O table is constructed from data obtained for a specific economic entity, be it a country, region, state, etc. The economic activity is divided into a number of producing sectors (or industries), with the exchanges of goods between sectors consisting of sales and purchases of goods. Sectors may be general industrial categories (e.g., the aluminum industry), or smaller subdivisions (such as aluminum siding), even larger industrial groups (such as mining ore). The rows of the I/O matrix denote the origin sectors, i.e., the sellers, whereas the columns denote the destination sectors, i.e., the purchasers. The data required to form the I/O table are the flows of the products from each of the producing sectors to the consuming sectors. The data are obtained for a particular time period. The column data represent each sector's inputs while the row data represents each sector's output; thus, the reason for the nomenclature "input/output table". For an overview of input/output tables, including applications to regional economics, see Polenske (1980) and Miller and Blair (1985).

The constrained matrix problem arises in the context of I/O tables, when one has a base I/O table for a given time period, typically, a year, and one wishes to update the table to another time period. Often one is not certain of the row and column totals at the new time period. For example, usually, one is able to estimate row and column totals only under certain simplifying assumptions. Furthermore, in countries or regions with a poor information base, the stated row and column totals may simply reflect informed conjectures (see Harrigan and Buchanan (1984) and the references therein). The above formulation takes into account such knowledge gaps directly, thus permitting modelling flexibility.

In the diagonal case, where  $\alpha_{ik} = 0$ , for  $k \neq i$ ,  $\gamma_{ij'kl'} = 0$ , for  $kl' \neq ij'$ , and  $\beta_{j'l'} = 0$ , for  $l' \neq j'$ , the objective function (1) simplifies to:

$$\text{Minimize} \quad \left[ \sum_{i=1}^m \alpha_i (s_i - s_i^0)^2 + \sum_{i=1}^m \sum_{j'=1}^n \gamma_{ij'} (x_{ij'} - x_{ij'}^0)^2 + \sum_{j'=1}^n \beta_{j'} (d_{j'} - d_{j'}^0)^2 \right] \quad (5)$$

subject to the constraints (2) through (4).

The choice of weights is also flexible in this formulation. When the weights in (5) are all equal to one, the problem becomes a constrained least squares problem, and when  $\alpha_i = \frac{1}{s_i^0}$ ,  $\beta_{j'} = \frac{1}{d_{j'}^0}$ , and  $\gamma_{ij'} = \frac{1}{x_{ij'}^0}$ , for all rows  $i$  and columns  $j'$ , the objective function is the well-known chi-square. Other possible weights, include  $\alpha_i = \frac{1}{s_i^0{}^{1/2}}$ ,  $\beta_{j'} = \frac{1}{d_{j'}^0{}^{1/2}}$ , and  $\gamma_{ij'} = \frac{1}{x_{ij'}^0{}^{1/2}}$ ; or a mixed weighting scheme. Of course, when the inverse of the variance-covariance matrix is diagonal, the objective function (5) can also be used.

This diagonal model has been studied by Nagurney (1989) who identified its isomorphism with classical, separable spatial price equilibrium problems and proposed an equilibration algorithm (albeit serial) for its solution. This model is a special case of a constrained matrix problem applied to I/O estimation by Harrigan and Buchanan (1984), who considered interval constraints, rather than equality constraints (2) and (3).

We now consider a special case of the above general quadratic model which arises in the estimation of social/national accounts.

A social accounting matrix (SAM) is a general equilibrium data system, consisting of a series of accounts in the economy of a nation. A SAM is comprised of  $n$  rows and  $n$  columns, with any particular account,  $i$ , represented by row  $i$  and column  $i$ . The rows represent the receipts of the accounts, the columns the expenditures of the accounts, and the individual matrix entries the transactions in the economy. SAM's have been widely used for policy analyses in developing countries (see, e.g., Pyatt and Round (1985)).

The fact that data used in the construction of SAM's often comes from disparate sources and the need to resolve various inconsistencies motivate the use of the constrained matrix problem in this application. In particular, any specific account represented by the corresponding row and column total must be balanced, that is, the receipts from the accounts must equal the expenditures. This "definitional" constraint that each row and column must balance makes the SAM estimation problem unique within the domain of

constrained matrix problems. In the case when the totals are not known a priori, this gives rise to the following special case of the above general constrained matrix problem:

$$\begin{aligned} \text{Minimize} \quad & \left[ \sum_{i=1}^n \sum_{k=1}^m \alpha_{ik} (s_i - s_i^0) \cdot (s_k - s_k^0) \right] \\ & + \left[ \sum_{i=1}^n \sum_{j'=1}^n \sum_{k=1}^m \sum_{l'=1}^m \gamma_{ij'kl'} (x_{ij'} - x_{ij'}^0) \cdot (x_{kl'} - x_{kl'}^0) \right] \end{aligned} \quad (6)$$

subject to inequality constraints (4) and:

$$\sum_{j'=1}^m x_{ij'} = s_i, \quad i = 1, \dots, n \quad (7)$$

$$\sum_{i=1}^m x_{ij'} = s_{j'}, \quad j' = 1, \dots, n. \quad (8)$$

Stone (1962), Byron (1978), and Van der Ploeg (1982), (1988) considered (6), subject to (7) and (8), but without the inequality constraints (4).

In the special diagonal case, where  $\alpha_{ik} = 0$ , for all  $k \neq i$ , and  $\gamma_{ij'kl'} = 0$ , for all  $kl' \neq ij'$ , the “diagonal” objective function for the SAM estimation problem becomes:

$$\text{Minimize} \quad \left[ \sum_{i=1}^n \alpha_i (s_i - s_i^0)^2 + \sum_{i=1}^n \sum_{j'=1}^n \gamma_{ij'} (x_{ij'} - x_{ij'}^0)^2 \right] \quad (9)$$

subject to the above constraints.

Lastly, in the case where the row and column totals are known with certainty, i.e.,  $s_i = s_i^0$ , for all  $i$ , and  $d_{j'} = d_{j'}^0$ , for all  $j'$ , the above general quadratic model (1), (2), (3), and (4), collapses to the quadratic constrained matrix problem with fixed row and column totals formulated in Nagurney and Robinson (1989) and solved in Nagurney, Kim, and Robinson (1990) via RC equilibration, whose performance will be compared to the new splitting equilibration algorithm in the subsequent computational sections.

In particular, the objective function in this case simplifies to:

$$\text{Minimize} \quad \sum_{i=1}^m \sum_{j'=1}^n \sum_{k=1}^m \sum_{l'=1}^m \gamma_{ij'kl'} (x_{ij'} - x_{ij'}^0) \cdot (x_{kl'} - x_{kl'}^0) \quad (10)$$

with the inequality constraints (4) and:

$$\sum_{j'=1}^n x_{ij'} = s_i^0, \quad i = 1, \dots, m \quad (11)$$

$$\sum_{i=1}^m x_{ij'} = d_{j'}^0, \quad j' = 1, \dots, n. \quad (12)$$

This model may also be applied to the estimation of input/output tables, and social/national accounting matrices, provided that the row and column totals are known with certainty, as well as to the estimation of migration flows. Migration tables are used to study growth patterns in distinct locations and to predict the needs for public services and housing. In migration tables rows represent the origin locations and columns the destination locations. The matrix entries represent the population flows between the origins and destinations. Row and column totals may be available for such applications, for example, when the totals have been obtained for a time period in the past, and the matrix entries which yield those totals are needed, given matrix entries for an earlier time period.

In the much-studied diagonal constrained matrix problem with fixed row and column totals, where the  $\gamma_{ij'kl'} = 0$ , for  $kl' \neq ij'$ , the objective function (10) further simplifies to

$$\text{Minimize} \quad \sum_{i=1}^m \sum_{j'=1}^n \gamma_{ij'} (x_{ij'} - x_{ij'}^0)^2 \quad (13)$$

with constraints (11), (12), and (4).

Deming and Stephan (1940) considered (13) with  $\gamma_{ij'} = \frac{1}{x_{ij'}^0}$ , subject to constraints (11) and (12), whereas Friedlander (1961) considered the case where  $G = I$ . Bachem and Korte (1978) treated (13) with all of the constraints. Cottle, et al. (1986), on the other hand, studied Friedlander's problem with the additional constraints (4). Ohuchi and Kaji (1984) studied the Bachem and Korte problem with upper and lower bounds. Klincewicz (1989) also studied the above diagonal model.

### 3. The Splitting Equilibration Algorithm

In this Section we introduce the splitting equilibration algorithm (SEA) for the computation of the general quadratic constrained matrix problem and its variants.

As mentioned in the Introduction, the algorithm is motivated both by the tremendous need for a unified and robust procedure which can solve the wide class of constrained matrix problems outlined in the preceding section and by the desire to exploit the underlying structure of these problems and the advantages of state-of-the-art computer architectures to allow the solution of problems of a scale larger than heretofore has been practical.

The algorithm constructs a series of diagonal problems of the form (5), subject to the constraints (2), (3), and (4). Each of the constructed diagonal problems, in turn, is then “split” to handle the row constraints (2) and then the column constraints (3). Each such row (supply) or column (demand) subproblem, because of the separability of the diagonal problem and the splitting of the constraints, can be solved simultaneously, i.e., in parallel, on a distinct processor. In a similar manner, the algorithm diagonalizes the SAM objective function (6), and the objective function (10) in the problem with fixed row and column totals, and then splits the row and column constraints (7), (8), and (11), (12), respectively, inducing, again, row and column equilibrium subproblems which can be solved exactly in closed form, and in parallel. Of course, in the diagonal constrained matrix problems with objective functions given, respectively, by (5), (9), and (13), no such diagonalization is required, before the splitting occurs.

We first present the splitting equilibration algorithm for the diagonal problems and then for the general ones. The general problems are solved iteratively using the appropriate diagonal procedure outlined immediately below. A graphical depiction of SEA for the diagonal problems is given in Figure 2. The special structure of the row and column equilibrium subproblems is then highlighted in Figure 3. A graphical depiction of SEA for general problems is given in Figure 4.

#### 3.1 The Splitting Equilibration Algorithm for Diagonal Problems

In this Section we present the splitting equilibration algorithm for the diagonal problems formulated in Section 2. We first focus on the diagonal problem with unknown row and column totals and objective function (1). We then describe SEA for the SAM es-

timization problem with objective function (9) and conclude with the constrained matrix problem with fixed row and column totals and objective function (13). We conclude with a unified interpretation of the algorithm as a dual method and provide theoretical results.

### 3.1.1 SEA for the Constrained Matrix Problem with Unknown Row and Column Totals

This algorithm computes a solution to problem (5), i. e.,

$$\text{Minimize } \Theta_1(x, s, d) = \sum_{i=1}^m \alpha_i (s_i - s_i^0)^2 + \sum_{i=1}^m \sum_{j'=1}^n \gamma_{ij'} (x_{ij'} - x_{ij'}^0)^2 + \sum_{j'=1}^n \beta_{j'} (d_{j'} - d_{j'}^0)^2$$

subject to constraints (2), (3), and (4).

#### Step 0: Initialization Step

Let  $\mu^1 \in R^n = 0$ . Set  $t = 1$ .

#### Step 1: Row Equilibration

Find  $X(\mu^t), S(\mu^t), D(\mu^t)$  such that

$$(X(\mu^t), S(\mu^t), D(\mu^t)) \rightarrow \min_{x,s,d} \Theta_1(x, s, d) - \sum_{j'=1}^n \mu_{j'}^t \left( \sum_{i=1}^m x_{ij'} - d_{j'} \right) \quad (14)$$

subject to:

$$\sum_{j'=1}^n x_{ij'} = s_i, \quad i = 1, \dots, m \quad (15)$$

$$x_{ij'} \geq 0.$$

Compute the corresponding Lagrange multipliers  $\lambda_i^{t+1}$  for this problem, according to  $\lambda_i^{t+1} = 2\alpha_i s_i^0 - 2\alpha_i S_i(\mu^t)$ ,  $i = 1, \dots, m$ , and use them in Step 2.

#### Step 2: Column Equilibration

Find  $X(\lambda^{t+1}), S(\lambda^{t+1}), D(\lambda^{t+1})$  such that

$$(X(\lambda^{t+1}), S(\lambda^{t+1}), D(\lambda^{t+1})) \rightarrow \min_{x,s,d} \Theta_1(x, s, d) - \sum_i \lambda_i^{t+1} \left( \sum_{j'=1}^n x_{ij'} - s_i \right) \quad (16)$$

subject to:

$$\sum_{i=1}^m x_{ij'} = d_{j'}, \quad j' = 1, \dots, n \quad (17)$$

$$x_{ij'} \geq 0.$$

Compute the corresponding Lagrange multipliers  $\mu_j^{t+1}$  for this problem, according to  $\mu_j^{t+1} = 2\beta_{j'} d_{j'}^0 - 2\beta_{j'} D_{j'} (\lambda^{t+1})$ ,  $j' = 1, \dots, n$ , and use them in Step 1.

### Step 3: Convergence Verification

If  $|x_{ij'}^t - x_{ij'}^{t-1}| \leq \epsilon$ ; for all  $i, j'$ , terminate; else, set  $t = t + 1$ , and go to Step 1.

Observe that both the row and column equilibration problems correspond to  $m$  and  $n$  disjoint network subproblems, respectively, each of which can be allocated to a distinct processor and solved, respectively, using the supply market/demand market exact equilibration algorithms discussed in Eydeland and Nagurney (1989).

We now provide a dual interpretation of the above algorithm which is used in establishing convergence and the theoretical analysis.

We introduce the Lagrangian

$$L_1(x, s, d, \lambda, \mu) = \Theta_1(x, s, d) - \sum_{i=1}^m \lambda_i \left( \sum_{j'=1}^n x_{ij'} - s_i \right) - \sum_{j'=1}^n \mu_{j'} \left( \sum_{i=1}^m x_{ij'} - d_{j'} \right) \quad (18)$$

and

$$\zeta_1(\lambda, \mu) = \min_{x \geq 0, s, d} L_1(x, \lambda, \mu). \quad (19)$$

We also let  $\lambda^*, \mu^* \rightarrow \max \zeta_1(\lambda, \mu)$ . In (19), the optimal values  $X = X(\lambda, \mu)$ ,  $S = S(\lambda, \mu)$ ,  $D = D(\lambda, \mu)$  satisfy the following relations:

$$\frac{\partial L_1}{\partial x_{ij'}}(X, S, D, \lambda, \mu) = 2\gamma_{ij'} X_{ij'} - 2\gamma_{ij'} x_{ij'}^0 - \lambda_i - \mu_{j'} \begin{cases} \geq 0, & \text{if } X_{ij'} = 0 \\ = 0, & \text{if } X_{ij'} > 0 \end{cases} \quad (20)$$

$$\frac{\partial L_1}{\partial s_i}(X, S, D, \lambda, \mu) = 2\alpha_i S_i - 2\alpha_i s_i^0 + \lambda_i = 0 \quad (21)$$

$$\frac{\partial L_1}{\partial d_{j'}}(X, S, D, \lambda, \mu) = 2\beta_{j'} D_{j'} - 2\beta_{j'} d_{j'}^0 + \mu_{j'} = 0. \quad (22)$$

Therefore,

$$X_{ij'} = \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ \quad (23a)$$

$$S_i = \frac{1}{2\alpha_i} (2\alpha_i s_i^0 - \lambda_i) \quad (23b)$$

$$D_{j'} = \frac{1}{2\beta_{j'}} (2\beta_{j'} d_{j'}^0 - \mu_{j'}). \quad (23c)$$

Hence,

$$\begin{aligned}
\zeta_1(\lambda, \mu) &\equiv L_1(X, S, D, \lambda, \mu) \\
&= \sum_{ij'} \gamma_{ij'} \left( \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ - x_{ij'}^0 \right)^2 \\
&+ \sum_i \alpha_i \left( \frac{1}{2\alpha_i} (2\alpha_i s_i^0 - \lambda_i) - s_i^0 \right)^2 + \sum_{j'} \beta_{j'} \left( \frac{1}{2\beta_{j'}} (2\beta_{j'} d_{j'}^0 - \mu_{j'}) - d_{j'}^0 \right)^2 \\
&- \sum_i \lambda_i \left( \sum_{j'} \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ - \frac{1}{2\alpha_i} (2\alpha_i s_i^0 - \lambda_i) \right) \\
&- \sum_{j'} \mu_{j'} \left( \sum_i \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ - \frac{1}{2\beta_{j'}} (2\beta_{j'} d_{j'}^0 - \mu_{j'}) \right) \\
&= - \sum_{i,j'} \frac{1}{4\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+^2 \\
&- \sum_i \frac{1}{4\alpha_i} (2\alpha_i s_i^0 - \lambda_i)^2 - \sum_{j'} \frac{1}{4\beta_{j'}} (2\beta_{j'} d_{j'}^0 - \mu_{j'})^2 \\
&+ \sum_{ij'} \gamma_{ij'} x_{ij'}^0{}^2 + \sum_i \alpha_i s_i^0{}^2 + \sum_{j'} \beta_{j'} d_{j'}^0{}^2. \tag{24}
\end{aligned}$$

The last three terms in (24) are constant and, therefore, can be ignored.

Moreover,

$$\begin{aligned}
\frac{\partial \zeta_1(\lambda, \mu)}{\partial \lambda_i} &= - \sum_{j'} \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ + \frac{1}{2\alpha_i} (2\alpha_i s_i^0 - \lambda_i) \\
&\equiv - \sum_{j'} X_{ij'}(\lambda, \mu) + S_i(\lambda, \mu). \tag{25}
\end{aligned}$$

$$\frac{\partial \zeta_1}{\partial \mu_{j'}} = - \sum_i \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ + \frac{1}{2\beta_{j'}} (2\beta_{j'} d_{j'}^0 - \mu_{j'}) \equiv D_{j'}(\lambda, \mu) - \sum_i X_{ij'}(\lambda, \mu). \tag{26}$$

Therefore,

$$\|\nabla \zeta_1(\lambda, \mu)\| \leq \epsilon \sim \|\text{Constraints}\| \leq \epsilon. \tag{27}$$

We define  $\lambda^{t+1}, \mu^{t+1}$  according to

$$\lambda^{t+1} \rightarrow \max_{\lambda} \zeta_1(\lambda, \mu^t) \tag{28a}$$



$$\mu^{t+1} \rightarrow \max_{\mu} \zeta_1(\lambda^{t+1}, \mu). \quad (28b)$$

By the duality principle, step (28a) is equivalent to solving precisely the problem:

$$\text{Minimize } \Theta_1(x, s, d) - \sum_{j'=1}^n \mu_{j'}^t \left( \sum_{i=1}^m x_{ij'} - d_{j'} \right) \quad (29a)$$

subject to

$$\begin{aligned} \sum_{j'=1}^n x_{ij'} &= s_i, \quad i = 1, \dots, m \\ x_{ij'} &\geq 0 \end{aligned}$$

with  $\lambda^{t+1}$  being the Lagrange multipliers for the above row constraints. Note that if  $(X(\mu^t), S(\mu^t), D(\mu^t))$  solves (29a), then we can find  $\lambda^{t+1}$  using the same argument for (23b):

$$\lambda_i^{t+1} = 2\alpha_i s_i^0 - 2\alpha_i S_i(\mu^t). \quad (29b)$$

Observe that (29a-29b) is precisely the Row Equilibration Step 1.

Analogously, the step (28b) is equivalent to:

$$\text{Minimize } \Theta_1(x, s, d) - \sum_{i=1}^m \lambda_i^{t+1} \left( \sum_{j'=1}^n x_{ij'} - s_i \right) \quad (30a)$$

subject to

$$\begin{aligned} \sum_{i=1}^m x_{ij'} &= d_{j'} \\ x_{ij'} &\geq 0 \end{aligned}$$

with  $\mu^{t+1}$  being the corresponding Lagrange multipliers. Again, if  $(X(\lambda^{t+1}), S(\lambda^{t+1}), D(\lambda^{t+1}))$  is the solution of (30a), then

$$\mu_j^{t+1} = 2\beta_{j'} d_{j'}^0 - 2\beta_{j'} D_{j'}(\lambda^{t+1}). \quad (30b)$$

Thus, (30a-30b) is equivalent to the Column Equilibration Step 2.

In the case of the diagonal SAM constrained matrix problem, we note that  $m = n$ , and  $s_i = d_i$ , for all  $i$ . The statement of the splitting equilibration algorithm for this constrained matrix problem is presented in the following section.

### 3.1.2 SEA for the SAM Constrained Matrix Problem

This algorithm computes a solution to problem (9), i.e.,

$$\text{Minimize } \Theta_2(x, s) = \sum_{j'=1}^n \alpha_{j'} (s_j - s_{j'}^0)^2 + \sum_{i=1}^n \sum_{j'=1}^n \gamma_{ij'} (x_{ij'} - x_{ij'}^0)^2 \quad (31)$$

subject to constraints (7), (8), and (4).

#### Step 0: Initialization Step

Let  $\mu^1 \in R^n = 0$ . Set  $t = 1$ .

#### Step 1: Row Equilibration

Find  $X(\mu^t), S(\mu^t)$  such that

$$(X(\mu^t), S(\mu^t)) \rightarrow \min_{x,s} \Theta_2(x, s) - \sum_{j'=1}^n \mu_{j'}^t \left( \sum_{i=1}^n x_{ij'} - s_{j'} \right) \quad (32)$$

subject to:

$$\sum_{j'=1}^n x_{ij'} = s_i, \quad i = 1, \dots, n \quad (33)$$

$$x_{ij'} \geq 0.$$

Compute the corresponding Lagrange multipliers  $\lambda_i^{t+1}$ , by  $\lambda_i^{t+1} = -2\alpha_{i'} S_i(\mu^t) + 2\alpha_{i'} s_i^0 - \mu_{i'}^t, i = 1, \dots, n$ , and use them in Step 2.

#### Step 2: Column Equilibration

Find  $X(\lambda^{t+1}), S(\lambda^{t+1})$  such that

$$(X(\lambda^{t+1}), S(\lambda^{t+1})) \rightarrow \min_{x,s} \Theta_2(x, s) - \sum_{i=1}^n \lambda_i^{t+1} \left( \sum_{j'=1}^n x_{ij'} - s_i \right) \quad (34)$$

subject to:

$$\sum_{i=1}^m x_{ij'} = s_{j'}, \quad j' = 1, \dots, n \quad (35)$$

$$x_{ij'} \geq 0.$$

Compute the corresponding Lagrange multipliers  $\mu^{t+1}$ , by  $\mu_{j'}^{t+1} = -2\alpha_{j'} S_{j'}(\mu^t) + 2\alpha_{j'} s_{j'}^0 - \lambda_{j'}^{t+1}$ , and use them in Step 1.

### Step 3: Convergence Verification

If  $|\sum_{j'} x_{ij'} - s_i|/s_i \leq \epsilon'$ , for all  $i$ , terminate; else, set  $t = t + 1$ , and go to Step 1.

The above row and column equilibration problems have the same special structure as that encountered in the analogous subproblems in Section 3.1.1, and again are amenable to solution via exact equilibration on distinct processors.

We now provide a dual interpretation of the above algorithm.

We introduce the Lagrangian

$$L_2(x, s, \lambda) = \Theta_2(x) - \sum_i \lambda_i (\sum_{j'} x_{ij'} - s_i) - \sum_{j'} \mu_{j'} (\sum_i x_{ij'} - s_{j'}) \quad (36)$$

and

$$\zeta_2(\lambda, \mu) = \min_{x \geq 0, s} L_2(x, s, \lambda, \mu) \quad (37)$$

and we let  $\lambda^*, \mu^* \rightarrow \max \zeta_2(\lambda, \mu)$ .

Since

$$\frac{\partial L_2}{\partial x_{ij'}} = 2\gamma_{ij'} X_{ij'} - 2\gamma_{ij'} x_{ij'}^0 - \lambda_i - \mu_{j'} \begin{cases} \geq 0, & \text{if } X_{ij'} = 0 \\ = 0, & \text{if } X_{ij'} > 0 \end{cases} \quad (38)$$

and

$$\frac{\partial L_2}{\partial s_{j'}} = 2\alpha_{j'} S_{j'} - 2\alpha_{j'} s_{j'}^0 + \lambda_j + \mu_{j'} = 0, \quad (39)$$

implies that:

$$X_{ij'} = \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ \quad (40a)$$

$$S_{j'} = \frac{1}{2\alpha_{j'}} (2\alpha_{j'} s_{j'}^0 - \lambda_j - \mu_{j'}), \quad (40b)$$

with  $\zeta_2(\lambda, \mu)$  taking the form:

$$\begin{aligned} \zeta_2(\lambda, \mu) = & - \sum_{ij'} \frac{1}{4\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+^2 - \sum_j \frac{1}{4\alpha_{j'}} (2\alpha_{j'} s_{j'}^0 - \lambda_j - \mu_{j'})^2 \\ & + \sum_{ij'} \gamma_{ij'} x_{ij'}^0 + \sum_{j'} \alpha_{j'} s_{j'}^0 \end{aligned} \quad (41)$$

with the last two terms in (41) being fixed.

Moreover, in regards to the gradient  $\nabla\zeta_2(\lambda, \mu)$  we observe that

$$\frac{\partial\zeta_2}{\partial\lambda_i} = -\sum_{j'} \frac{1}{2\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+ + \frac{1}{2\alpha_{i'}} (2\alpha_{i'} s_{i'}^0 - \lambda_i - \mu_{i'}) \quad (42)$$

which implies that:

$$\|\text{Constraints}\| \leq \epsilon \sim \|\nabla\zeta_2(\lambda, \mu)\| \leq \epsilon. \quad (43)$$

SEA for the SAM estimation problem, hence, has the following dual interpretation:

$$\lambda^{t+1} \rightarrow \max_{\lambda} \zeta_2(\lambda, \mu^t) \quad (44a)$$

and

$$\mu^{t+1} \rightarrow \max_{\mu} \zeta_2(\lambda^{t+1}, \mu) \quad (44b)$$

where (44a) correspond to the row equilibration step and (44b) correspond to the column equilibration step.

Finally, we present the statement of the splitting equilibration algorithm for the case of fixed row and column totals, which is equivalent to the diagonal RC algorithm described in see Nagurney, Kim, and Robinson (1990), but which will be theoretically analyzed below.

### 3.1.3 SEA for the Constrained Matrix Problem with Fixed Row and Column Totals

This algorithm computes the solution to problem (13), i.e.,

$$\text{Minimize } \Theta_3(x) = \sum_{i=1}^m \sum_{j'=1}^n \gamma_{ij'} (x_{ij'} - x_{ij'}^0)^2$$

subject to (2), (3), and (4).

#### Step 0: Initialization Step

Let  $\mu^1 \in R^n = 0$ . Set  $t = 1$ .

#### Step 1: Row Equilibration

Find  $X(\mu^t)$  such that

$$X(\mu^t) \rightarrow \min_x \Theta_3(x) - \sum_{j'=1}^n \mu_{j'}^t \left( \sum_{i=1}^m x_{ij'} - d_{j'}^0 \right) \quad (45)$$

subject to

$$\sum_{j'=1}^n x_{ij'} = s_i^0, \quad i = 1, \dots, m \quad (46)$$

$$x_{ij'} \geq 0.$$

Compute the corresponding Lagrange multipliers  $\lambda^{t+1}$ .

### Step 2: Column Equilibration

Find  $X(\lambda^{t+1})$  such that

$$(X(\lambda^{t+1})) \rightarrow \min_x \Theta_3(x) - \sum_{i=1}^m \lambda_i^{t+1} \left( \sum_{j'=1}^n x_{ij'} - d_{j'}^0 \right) \quad (47)$$

subject to

$$\sum_{i=1}^m x_{ij'} = d_{j'}^0, \quad j' = 1, \dots, n \quad (48)$$

$$x_{ij'} \geq 0.$$

Obtain the corresponding Lagrange multipliers  $\mu^{t+1}$ .

### Step 3: Convergence Verification

Same as Step 3 above with  $s_i = s_i^0$ , for all  $i$ .

The row and column equilibration problems above again have a characteristic network structure where, however, the respective equilibration problems differ from those encountered in 3.1.1 and 3.1.2 in that they are of the “fixed” type, rather than elastic, in that  $s_i^0$ ,  $i = 1, \dots, m$  and  $d_{j'}^0$ ,  $j' = 1, \dots, n$  are known and fixed.

The dual interpretation now follows.

We introduce the Lagrangian

$$L_3(x, \lambda, \mu) = \Theta_3(x) - \sum_{i=1}^m \lambda_i \left( \sum_{j'=1}^n x_{ij'} - s_i^0 \right) - \sum_{j'=1}^n \mu_{j'} \left( \sum_{i=1}^m x_{ij'} - d_{j'}^0 \right) \quad (49)$$

and

$$\zeta_3(\lambda, \mu) = \min_{x \geq 0} L_3(x, \lambda, \mu). \quad (50)$$

By direct computations we obtain that

$$\zeta_3(\lambda, \mu) = - \sum_{ij'} \frac{1}{4\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+^2 + \sum_i \lambda_i s_i^0 + \sum_{j'} \mu_{j'} d_{j'}^0 + \sum_{ij'} \gamma_{ij'} x_{ij'}^0{}^2. \quad (51)$$

Again,

$$\|\zeta_3(\lambda, \mu)\| \leq \epsilon \sim \|\text{Constraints}\| \leq \epsilon. \quad (52)$$

The algorithm now is the same, as the two previous ones, i.e.,

$$\lambda^{t+1} \rightarrow \max_{\lambda} \zeta_3(\lambda, \mu^t) \quad (53a)$$

$$\mu^{t+1} \rightarrow \max_{\mu} \zeta_3(\lambda^{t+1}, \mu). \quad (53b)$$

Step (53a) is equivalent to solving the problem

$$\text{Minimize}_x \Theta_3(x) - \sum_{j'=1}^n \mu_{j'} \left( \sum_i x_{ij'} - d_{j'}^0 \right) \quad (54a)$$

subject to

$$\begin{aligned} \sum_{j'=1}^n x_{ij'} &= s_i^0 \\ x_{ij'} &\geq 0, \end{aligned}$$

with  $\lambda^t$  being the Lagrange multipliers for this problem. If  $X(\mu^t)$  is a solution of (54a) then

$$\frac{\partial \Theta_3}{\partial x_{ij'}}(X(\mu^t)) - \mu_{j'}^t = \lambda_i^{t+1}, \quad \text{for every } i = 1, \dots, m, \quad j' = 1, \dots, n. \quad (54b)$$

This expression indicates how to update the  $\lambda$ 's.

Step (53b), analogously, is equivalent to

$$\text{Minimize}_x \Theta_3(x) - \sum_{i=1}^m \lambda_i^{t+1} \left( \sum_{j'=1}^n x_{ij'} - d_{j'}^0 \right) \quad (55a)$$

subject to

$$\begin{aligned} \sum_{i=1}^m x_{ij'} &= d_{j'}^0 \\ x_{ij'} &\geq 0, \end{aligned}$$

with  $\mu^{t+1}$  being the corresponding Lagrange multipliers. Also, we have that

$$\frac{\partial \Theta_3}{\partial x_{ij'}}(X(\lambda^{t+1})) - \lambda_i^{t+1} = \mu_{j'}^{t+1}, \quad \text{for every } i = 1, \dots, m, \quad j' = 1, \dots, n, \quad (55b)$$

where  $X(\lambda^{t+1})$  is a solution of (55a).

We now provide a summarization and unification of the above algorithms and refer, henceforth, to the method as SEA.

**The Algorithm:**

$$\begin{aligned}\lambda^{t+1} &\rightarrow \max \zeta_i(\lambda, \mu^t) \\ \mu^{t+1} &\rightarrow \max \zeta_i(\lambda^{t+1}, \mu) \\ &i = 1, 2, 3\end{aligned}$$

where

$$\begin{aligned}\zeta_1(\lambda, \mu) &= - \sum_i \sum_{j'} \frac{1}{4\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+^2 - \sum_i \frac{1}{4\alpha_i} (2\alpha_i s_i^0 - \lambda_i)^2 \\ &\quad - \sum_{j'} \frac{1}{4\beta_{j'}} (2\beta_{j'} d_{j'}^0 - \mu_{j'})^2 + \sum_{ij'} \gamma_{ij'} x_{ij'}^0{}^2 + \sum_i \alpha_i s_i^0{}^2 + \sum_{j'} d_{j'}^0{}^2 \\ \zeta_2(\lambda, \mu) &= - \sum_i \sum_{j'} \frac{1}{4\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+^2 - \sum_{j'} \frac{1}{4\alpha_{j'}} (2\alpha_{j'} s_{j'}^0 - \lambda_j - \mu_{j'})^2 \\ &\quad + \sum_{ij'} \gamma_{ij'} x_{ij'}^0{}^2 + \sum_{j'} \alpha_{j'} s_{j'}^0{}^2 \\ \zeta_3(\lambda, \mu) &= - \sum_{ij'} \frac{1}{4\gamma_{ij'}} (2\gamma_{ij'} x_{ij'}^0 + \lambda_i + \mu_{j'})_+^2 + \sum_i \lambda_i s_i^0 + \sum_{j'} \mu_{j'} d_{j'}^0 + \sum_{ij'} \gamma_{ij'} x_{ij'}^0{}^2.\end{aligned}$$

We now state the proof of convergence using the stopping criterion:  $\|\zeta_i(\lambda, \mu)\| \leq \epsilon$ .

Observe that

$$\zeta_i(\lambda^{t+1}, \mu^t) \geq \max_{\tau} \zeta_i(\lambda^{t+1}, \mu^t + \tau d^t) \quad (56)$$

with

$$d^t = \frac{\nabla_{\mu} \zeta_i(\lambda^{t+1}, \mu^t)}{\|\nabla_{\mu} \zeta_i(\lambda^{t+1}, \mu^t)\|} = \frac{\nabla \zeta_i(\lambda^{t+1}, \mu^t)}{\|\nabla \zeta_i(\lambda^{t+1}, \mu^t)\|}. \quad (57)$$

Let us now estimate  $\tau_{\max}$  where  $\tau$  is the value for which the max in (56) is attained.

Since on the interval  $[\mu^t, \mu^t + \tau_{\max} d^t]$  the function  $\theta(\tau) = \nabla \zeta_i(\lambda^t, \mu^t + \tau d^t) \times d^t$  changes from  $\|\nabla \zeta_i(\lambda^{t+1}, \mu^t)\|$  to 0 and  $M_l \geq |\frac{\partial \theta_l}{\partial \tau}| \geq m_l$ ,  $l = 1, 2, 3$ , where

$$m_1 = \min\left\{\min_{ij'} \frac{1}{2\gamma_{ij'}}, \min_i \frac{1}{2\alpha_i}, \min_{j'} \frac{1}{2\beta_{j'}}\right\} \quad (58a)$$

$$m_2 = \min\left\{\min_{ij'} \frac{1}{2\gamma_{ij'}}, \min_i \frac{1}{2\alpha_i}\right\} \quad (58b)$$

$$m_3 = \min_{ij'} \frac{1}{2\gamma_{ij'}} \quad (58c)$$

and

$$M_1 = \max\left\{\max_{ij'} \frac{1}{2\gamma_{ij'}}, \max_i \frac{1}{2\alpha_i}, \max_{j'} \frac{1}{2\beta_{j'}}\right\} \quad (59a)$$

$$M_2 = \max\left\{\max_{ij'} \frac{1}{2\gamma_{ij'}}, \max_i \frac{1}{2\alpha_i}\right\} \quad (59b)$$

$$M_3 = \max_{ij'} \frac{1}{2\gamma_{ij'}}, \quad (59c)$$

we can conclude that:

$$\tau_{\max} \geq \frac{1}{M_i} \|\nabla \zeta_i(\lambda^{t+1}, \mu^t)\|. \quad (60)$$

Expanding now  $\zeta_l(\lambda^{t+1}, \mu^t + \tau d^t)$  around  $\tau = \tau_{\max}$ , we get

$$\zeta_l(\lambda^{t+1}, \mu^t) \leq \zeta_l(\lambda^{t+1}, \mu^t + \tau_{\max} d^t) - \frac{\tau_{\max}^2}{2} m_l. \quad (61)$$

Hence,

$$\zeta_l(\lambda^{t+1}, \mu^t) \leq \zeta_l(\lambda^{t+1}, \mu^{t+1}) - \frac{m_l}{2M_l^2} \|\nabla \zeta_l(\lambda^{t+1}, \mu^t)\|^2 \quad (62)$$

or

$$\delta^t = \zeta_l(\lambda^{t+1}, \mu^{t+1}) - \zeta_l(\lambda^{t+1}, \mu^t) \geq \frac{m_l}{2M_l^2} \|\nabla \zeta_l(\lambda^{t+1}, \mu^t)\|^2 \geq \frac{m_l}{2M_l^2} \epsilon^2 \quad (63)$$

where  $\epsilon$  is the stopping criterion.

Thus, our algorithm must stop in no more than

$$T = \frac{\zeta_{\max} - \zeta_l(\lambda^0, \mu^0)}{\frac{m_l}{2M_l^2}} \times \frac{1}{\epsilon^2} \text{ steps.} \quad (64)$$

We note that while dual algorithms have been proposed by Cottle, Duvall, and Zikan (1986) and Ohuchi and Kaji (1984) for the fixed model with objective function (13) and constraints (11) and (12), this is the first such unified treatment of both fixed and the more general elastic versions. For other dual methods and associated applications, see Bertsekas and Tsitsiklis (1989). Moreover, our proof of convergence is new and specifically uses the parameters of the problem without any other assumptions or imposed conditions. We also provide further theoretical results, including a rate of convergence, which is also a new contribution.

We now establish additional theoretical results.



Recall that at step  $t + 1$

$$\begin{aligned}\lambda^{t+1} &\rightarrow \max_{\lambda} \zeta_l(\lambda, \mu^t) \\ \mu^{t+1} &\rightarrow \max_{\mu} \zeta_l(\lambda^{t+1}, \mu^t), \quad l = 1, 2, 3.\end{aligned}$$

Let  $\lambda_{t+1}^*, \mu_t^* \in \text{Arg max } \zeta_l(\lambda, \mu)$  and are chosen to be the closest points to the iterates  $\lambda^{t+1}, \mu^t$  in  $\text{Arg max } \zeta_l(\lambda, \mu)$ ,  $l = 1, 2, 3$ . In order to prevent the notation from becoming cumbersome, we do omit the subscript  $l$  of the  $\lambda$ 's and  $\mu$ 's. We assume for the time being that there exists the bounded set  $\Omega_l$  which contains both sequences:  $\{\lambda^{t+1}, \mu^t\}$  and  $\{\lambda_{t+1}^*, \mu_t^*\}$ . It clearly exists for  $l = 1$ , since  $\zeta_1(\lambda, \mu)$  is a strictly concave function as can be seen from its explicit form (24). For  $l = 2, 3$  we later provide a modification of the algorithm that assures that the iterates  $\lambda^{t+1}, \mu^t$  always lie in the bounded set and, therefore, their projections on  $\text{Arg max } \zeta_l$  also belong to a bounded set.

We have the following inequality for  $\tau \in [0, 1]$ .

$$\begin{aligned}\zeta_l(\lambda^{t+1}, \mu^{t+1}) &\geq \zeta_l(\lambda^{t+1}, \mu^t + \tau(\mu_t^* - \mu^t)) \\ &= \zeta_l(\lambda^{t+1}, \mu^t) + \tau \nabla_{\lambda, \mu} \zeta_l(\lambda^{t+1}, \mu^t) \times \begin{pmatrix} (\lambda_{t+1}^* - \lambda^{t+1}) \\ (\mu_t^* - \mu^t) \end{pmatrix} \\ &\quad - \frac{\bar{M}_l}{2} \tau^2 (\|\lambda_{t+1}^* - \lambda^{t+1}\|^2 + \|\mu_t^* - \mu^t\|^2) \\ &\geq \zeta_l(\lambda^{t+1}, \mu^t) + \tau (\zeta_l(\lambda_{t+1}^*, \mu_t^*) - \zeta_l(\lambda^{t+1}, \mu^t)) - \frac{\bar{M}_l}{2} \tau^2 [\|\lambda_{t+1}^* - \lambda^{t+1}\|^2 + \|\mu_t^* - \mu^t\|^2]\end{aligned}\tag{65}$$

where  $\bar{M}_l$  is a bound of the norm of the Hessian of  $\zeta_l$ .

Let us prove now that for every  $(\lambda^*, \mu^*) \in \Omega_l$  and for every direction  $\psi = (\psi_\lambda, \psi_\mu)$  from the normal cone  $N_{\lambda^*, \mu^*}$  to  $\text{Arg max } \zeta_l(\lambda, \mu)$  at  $(\lambda^*, \mu^*)$ ,

$$\zeta_l(\lambda^*, \mu^*) - \zeta_l(\lambda^* + \psi_\lambda, \mu^* + \psi_\mu) \geq \frac{A}{2} \|\psi\|^2\tag{66}$$

where  $A$  is a positive constant and a normal cone  $N_{\lambda^*, \mu^*}$  is defined as a set of directions  $\psi$  which satisfy

$$\psi_\lambda \cdot (\Lambda^* - \lambda^*) + \psi_\mu \cdot (M^* - \mu^*) \leq 0$$

for every  $(\Lambda^*, M^*) \in \text{Arg max } \zeta_l(\lambda, \mu)$ .

From our definition of  $\lambda_{t+1}^*, \mu_t^*$  it is clear that

$$\psi = \begin{pmatrix} \lambda^{t+1} - \lambda_{t+1}^* \\ \mu^t - \mu_t^* \end{pmatrix} \in N_{\lambda_{t+1}^*, \mu_t^*}.$$

Since, for every  $l = 1, 2, 3$  the function

$$Z_l(\tau) = \zeta_l(\lambda^* + \tau\psi_\tau, \mu^* + \tau\psi_\mu), \quad \tau > 0 \quad (67)$$

is a piecewise quadratic concave function of  $\tau$  (as can be seen from the explicit expressions for  $\zeta_l$ ). Hence, for every  $\tau > 0$ ,

$$Z_l(\tau) \leq Z_l(0) + \frac{\partial Z_l}{\partial \tau_+}(0) + \frac{1}{2} \frac{\partial^2 Z_l}{\partial \tau_+^2}(0) \tau^2,$$

and for  $\tau \in [0, \epsilon]$ ,  $\epsilon > 0$ ,

$$Z_l(\tau) = Z_l(0) + \frac{\partial Z_l}{\partial \tau_+}(0) + \frac{1}{2} \frac{\partial^2 Z_l}{\partial \tau_+^2}(0) \tau^2 \quad (68)$$

where  $\frac{\partial}{\partial \tau_+}$  denotes a directional derivative.

Since  $(\lambda^*, \mu^*) \in \text{Arg max } \zeta_l(\lambda, \mu)$ ,  $\frac{\partial Z_l}{\partial \tau}(0) = 0$ .

Moreover, there exists a positive number  $a_{\lambda^*, \mu^*, \psi} > 0$ , such that

$$\frac{\partial^2 Z_l}{\partial \tau_+^2}(0) \leq -a_{\lambda^*, \mu^*, \psi} \|\psi\|^2 < 0 \quad (69)$$

since if  $\frac{\partial^2 Z_l}{\partial \tau_+^2}(0) = 0$ , then for  $\tau \in [0, \epsilon]$ ,  $Z_l(\tau) = Z_l(0) = \zeta_l(\lambda^*, \mu^*)$ , which means that  $(\lambda^* + \tau\psi_\lambda, \mu^* + \tau\psi_\mu) \in \text{Arg max } \zeta_l$ . This contradicts the fact that the direction  $(\psi_\lambda, \psi_\mu)$  belongs to the normal cone.

We now introduce  $\bar{a}_{\lambda^*, \mu^*} = \min_{\psi \in N_{\lambda^*, \mu^*}} a_{\lambda^*, \mu^*, \psi}$ . By (67) and compactness of the normal cone, we have that  $\bar{a}_{\lambda^*, \mu^*} > 0$ .

Finally, we define

$$A = \min_t \{\bar{a}_{\lambda_{t+1}^*, \mu_t^*}\}.$$

We now prove that  $A > 0$ . Indeed, assume that there exists a sequence  $\{\bar{a}_{\lambda_{\nu+1}^*, \mu_\nu^*}\}$  such that  $\bar{a}_{\lambda_{\nu+1}^*, \mu_\nu^*} \rightarrow 0$ . Then by the definition of  $\bar{a}_{\lambda^*, \mu^*}$  and by the boundedness of  $\Omega_l$  there exists a limit point  $(\Lambda^*, M^*)$  of this subsequence and a vector  $\Psi \neq 0$  with the following properties:

$(\Lambda^*, M^*) \in \Omega_l$ ,  $\Psi \in \text{normal cone to Arg max } \zeta_l$  and by our assumption  $a_{\Lambda^*, M^*, \Psi} = 0$ .

But, as we have shown above, the last equality is in contradiction with  $\Psi$  being a vector in a normal cone. Thus,  $A > 0$ . We then obtain (66) by choosing in (67)  $\psi_\lambda = (\lambda^{t+1} - \lambda_{t+1}^*)$  and  $\psi_\mu = (\mu^{t+1} - \mu_{t+1}^*)$  and then using the inequality (69) for  $\tau = 1$ .

We obtain then that

$$\begin{aligned} \zeta_l(\lambda^{t+1}, \mu^{t+1}) &\geq \zeta_l(\lambda^{t+1}, \mu^t) + \tau(\zeta_l(\lambda_{t+1}^*, \mu_t^*) - \zeta_l(\lambda^{t+1}, \mu^t)) \\ &\quad - \tau^2 \frac{\bar{M}_l}{A} (\zeta_l(\lambda_{t+1}^*, \mu_t^*) - \zeta_l(\lambda^{t+1}, \mu^t)). \end{aligned} \quad (70)$$

It follows then that

$$\zeta_l(\lambda^{t+2}, \mu^{t+1}) \geq \zeta_l(\lambda^{t+1}, \mu^{t+1}) \geq \dots \quad (71)$$

Let now

$$\delta^t = \zeta_l(\lambda_{t+1}^*, \mu_{t+1}^*) - \zeta_l(\lambda^{t+1}, \mu). \quad (72)$$

Using then (70), (71), and (72) we obtain

$$\begin{aligned} \zeta_l(\lambda_{t+2}^*, \mu_{t+1}^*) - \zeta_l(\lambda^{t+2}, \mu^{t+1}) &\leq \zeta_l(\lambda^*, \mu^*) - \zeta_l(\lambda^{t+1}, \mu^t) \\ &\quad - \tau(\zeta_l(\lambda^*, \mu^*) - \zeta_l(\lambda^{t+1}, \mu^t)) + \tau^2 \frac{\bar{M}_l}{A} (\zeta_l(\lambda^*, \mu^*) - \zeta_l(\lambda^{t+1}, \mu^t)) \end{aligned} \quad (73)$$

or

$$\delta^{t+1} \leq \delta^t (1 - \tau + \tau^2 \frac{\bar{M}_l}{A}) \quad (74)$$

for  $\tau \in [0, 1]$ .

Minimizing (74) with respect to  $\tau$ , we obtain:

$$\tau_{\min} = \frac{A}{2\bar{M}_l}. \quad (75)$$

Hence,

$$\delta^{t+1} \leq \delta^t \left(1 - \frac{A}{4\bar{M}_l}\right) \quad (76)$$

where  $1 - \frac{A}{4\bar{M}_l} < 1$ .

Therefore, if for some fixed  $\bar{\epsilon} > 0$ , we use the stopping criterion  $\delta^{\bar{T}} \leq \bar{\epsilon}$ , then the number of steps  $\bar{T}$  for convergence, is given by:

$$\bar{T} = \frac{\ln \left[ \frac{\bar{\epsilon}}{\delta^0} \right]}{\ln \left[ \left(1 - \frac{A}{4\bar{M}_l}\right) \right]}. \quad (77)$$

Observe that the # of iterations  $\bar{T}$  is additive with respect to  $\bar{\epsilon}$ . Hence if we decrease  $\bar{\epsilon}$  by a factor of 10, we should expect to see only an additive increase in the number of iterations.

For each iteration, cf. Eydeland and Nagurney (1989), (assuming that  $m = n$  each demand/supply exact equilibration algorithm takes

$$7n + n \ln n + 2n \quad \text{operations.}$$

Hence, it takes

$$n(9n + n \ln n) \quad \text{operations}$$

for all  $n$  rows/columns to be equilibrated. The overall number of operations  $N$  (and correspondingly the overall CPU time) is then proportional to

$$N = \bar{T}(n^2)(9 + n \ln n).$$

Note, that if there are  $p$  processors available, where we assume that  $p \leq n$  then

$$N_p = \frac{\bar{T}(n^2)(9 + n \ln n)}{p}.$$

In particular, for  $p = n$ , we have that

$$N_n = \bar{T}n(9 + n \ln n).$$

We shall now prove how to ensure that  $(\lambda^{t+1}, \mu^t)$  belongs to a bounded set in the case of  $l = 2, 3$ .

It is clear that there exists a  $d_{max}$  such that if  $\lambda_i + \mu_{j'}$  is larger than  $d_{max}$ , then  $\zeta_l$  is  $< \zeta_l^0$ , which cannot be true since at any step  $\zeta_l \geq \zeta_l^0$  because we are maximizing  $\zeta_l$ . Also, it is clear that if  $\lambda_i + \mu_{j'} < -d_{max}$ , then  $x_{ij'} = 0$ .

If at step  $t$ , then, we have that  $x_{ij'}^t > 0$ , then  $\lambda_i + \mu_{j'} > -d_{max}$ . Hence, we can conclude that

$$-d_{max} < \lambda_i + \mu_{j'} < d_{max}. \quad (78)$$

Note that  $d_{max}$  depends only on the given data of the problem and not on  $t$ .

Each iterate  $(x^t, \lambda^{t+1}, \mu^t)$  of our procedure defines a graph  $G^t$  whose nodes  $(ij')$  are connected only if  $x_{ij'} \neq 0$ . In this graph we can introduce the definition of adjacency of two edges: the edge  $(ij')$  is adjacent to  $(kl')$  if either  $i = k$  or  $l' = j'$ . Thus, we have a definition of a new graph  $G^{t*}$  whose nodes, corresponding to edges in  $G^t$ , are connected if

and only if the edges in  $G^t$  have a common endpoint. Having a definition of connectedness of two nodes in  $G^{t^*}$  (edges in  $G^t$ ) we may now define the connected component in  $G^{t^*}$  in a standard way. It is clear from the definition of  $\zeta_l (l = 2, 3)$  that within a connected component one can add a certain constant to  $\lambda_i$ 's and subtract the same constant from  $\mu_{j'}$  without changing the values of  $\zeta_l$ . Moreover, by (78), if edges  $(ij')$  and  $(kl')$  belong to a connected component then

$$|\lambda_i - \lambda_k| < 2nd_{max}$$

$$|\mu_{j'} - \mu_{l'}| < 2nd_{max}.$$

Combining these properties together we define the following modification of our algorithm which would keep the iterates  $(\lambda^{t+1}, \mu^t)$  in the bounded set.

#### Modified Algorithm:

Choose a large  $R > 0$ .

Let  $\lambda^t, \mu^t$  be known.

If all  $\lambda_i$ 's are  $< R$ , continue to the next  $t$ -th step.

If there exists a  $\tilde{\lambda}_i$  such that  $|\tilde{\lambda}_i| > R$ , then subtract  $\tilde{\lambda}_i$  from all  $\lambda_i$ 's in the connected component and add it to all  $\mu_{j'}$ 's in this connected component. This should bring all of the  $\lambda$ 's and  $\mu$ 's in the cube  $[-2nd_{max}, 2nd_{max}]$ . Then check other connected components.

By the properties discussed above this modification does not change the values of  $\zeta_l$ . Moreover, since  $\tilde{\lambda}_i^{modified} = 0$  all other  $\lambda_i$ 's in the connected component will be less than  $2nd_{max}$  in absolute values. Clearly,  $|\mu_{j'}^{modified}|$  are bounded in the same way.

We now present the splitting equilibration algorithm for the general problem.

### 3.2 The Splitting Equilibration Algorithm for General Problems

SEA for general problems solves a series of diagonal problems as outlined in Section 3.1. The diagonal problems, in turn, are constructed via the projection method of Dafermos (1982, 1983) which is based on variational inequality theory. For a brief introduction to variational inequality theory and associated applications, see Nagurney (1987). In particular, the projection method constructs a series of quadratic programming problems which are simpler than the original problem. It uses fixed matrices and modifies only the fixed linear terms in the corresponding objective functions. In particular, we select as the

fixed matrices the diagonals of general matrices  $A$ ,  $B$ , and  $G$ . Hence, only the linear terms are updated from iteration to iteration.

We now present the splitting equilibration algorithm for the general constrained matrix problem, (1) through (4). For a graphical depiction of SEA for general problems, we refer the reader to Figure 4.

### 3.2.1 SEA for General Constrained Matrix Problems with Unknown Row and Column Totals

#### Step 0: Initialization Step

Start with any feasible  $(s, x, d)$ , i.e., one which satisfies constraints (2), (3), and (4). Set  $t = 1$ .

#### Step 1: Projection Step

Given  $(s^{t-1}, x^{t-1}, d^{t-1})$ , find  $(s^t, x^t, d^t)$  by solving the following problem:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}s^T \tilde{A}s + (-\tilde{A}s^{t-1} - As^0 + As^{t-1})^T s + \frac{1}{2}x^T \tilde{G}x + (-\tilde{G}x^{t-1} - Gx^0 + Gx^{t-1})^T x \\ & + \frac{1}{2}d^T \tilde{B}d + (-\tilde{B}d^{t-1} - Bd^0 + Bd^{t-1})^T d \end{aligned} \quad (79)$$

subject to constraints (2), (3), and (4), via SEA for Diagonal Problems (see Section 3.1), where  $\tilde{A}$ ,  $\tilde{G}$ , and  $\tilde{B}$  denote the diagonal matrices  $\text{diag}(A)$ ,  $\text{diag}(G)$ , and  $\text{diag}(B)$ , respectively.

#### Step 2: Convergence Verification

If  $|x_{ij}^t - x_{ij}^{t-1}| \leq \epsilon$ , for all  $i, j$ , then stop; otherwise, set  $t = t + 1$ , and go to Step 1.

The general splitting equilibration algorithm applied to both the SAM problem and the problem with fixed row and column totals can be constructed in an analogous manner.

## 4. Computation of Large-Scale Diagonal Constrained Matrix Problems

In this and the subsequent Sections we describe the computational experiments conducted and the results obtained for the splitting equilibration algorithm (SEA) on large-scale quadratic constrained matrix problems. We begin with computational experience on diagonal problems and then turn to the solution of general problems. For each class of problems we first present the results of serial computations and then those of parallel computations. All of the computational experiments were conducted on the IBM 3090-600E at the Cornell National Supercomputer Facility (CNSF).

### 4.1 Serial Experiments

In this Section we investigated the computational efficiency of the SEA algorithm on the class of diagonal constrained matrix problems outlined in Section 2. All of the programs used throughout the study were coded in FORTRAN and run on the IBM 3090-600E at the CNSF. The serial programs were compiled under VS FORTRAN at optimization level (3) running under VM/XA 5.5. The CPU times are exclusive of input and output, but include initialization times.

#### 4.1.1 Computational Experience with SEA

In this Section we studied the performance of SEA on very large problems with fixed row and column totals. The examples were generated as follows. We generated matrix examples ranging in size from 750 rows  $\times$  750 columns through 3000 rows  $\times$  3000 columns with the percentage of positive  $x_{ij}^0$ 's at 100%. Each non-zero  $x_{ij}^0$  was generated uniformly in the range  $[.1, 10000]$ , to simulate the wide spread of the initial data which are characteristic of both input/output and social accounting matrices. The weighting terms, the  $\gamma_{ij}$ 's were set to  $\frac{1}{x_{ij}^0}$ . In the examples, we set each row total  $s_i^0 = 2 \sum_{j'} x_{ij'}^0$ , and each column total  $d_{j'}^0 = 2 \sum_i x_{ij'}^0$ .

SEA was implemented in accordance with the suggestions and theoretical guidance for equilibration algorithms contained in Eydeland and Nagurney (1989). In particular, each row equilibrium subproblem and each column equilibrium subproblem was solved via exact equilibration. Since exact equilibration requires sorting and since the arrays to be sorted are, typically, in the applications considered here substantially larger than one hundred elements, the sorting procedure used in the implementation of SEA for fixed row

and column totals was HEAPSORT. The  $\epsilon$  in the convergence test was set at .01. The results of the numerical experiments are reported in Table 1.

The examples solved ranged from 562,500 to 9 million non-zero initial matrix elements. As can be seen from Table 1, the smallest example with 562,500 variables required only minutes of CPU time on a serial machine, whereas the largest, with 9,000,000 variables required approximately  $3\frac{1}{2}$  hours of CPU time. On the basis of these runs we now consider applications with fewer than 1 million variables to be solvable feasibly in a reasonable time-frame in a serial manner on a machine such as the IBM 3090-600E when an efficient algorithm such as SEA is used for the computation.

#### 4.1.2 Experiments on Input/Output Matrices, Social Accounting Matrices, Migration Tables, and Spatial Price Equilibrium Problems

In this Section we provide further numerical results with SEA on real-world economic and demographic datasets. The datasets include input/output matrices, social accounting matrices, migration tables, and spatial price equilibrium problems.

We now briefly describe the datasets. The computational results are reported in Tables 2, 3, 4, and 5. In Table 2 we report the results of the performance of SEA on input/output matrices with known row and column totals. In Table 3 we report the results of SEA on social accounting matrices, in which the row and column totals must balance and must be estimated, as well. In Table 4 we report the results of SEA on migration tables, in which the row and column totals are also to be estimated. In Table 5 we report the results of SEA's performance on spatial price equilibrium problems, which, as discussed in Section 2, are equivalent to constrained matrix problems in which both row and column totals need to be estimated.

The first set of three examples was constructed from an aggregated 1972 input/output matrix of construction activity in the United States consisting of 205 rows  $\times$  205 columns. This I/O matrix retained the construction sectors in the United States in detail, and aggregated those sectors in the United States in which the construction inputs were zero or negligible. The first example, IOC72a, was formed by generating a 10% growth factor, while the second, IOC72b, by generating a 100% growth factor. The percentage of non-zero  $x_{ij}^0$ 's was 52%. The third datapoint, termed, IOC72c, consisted of the average of



10 examples, where each example consisted of the 1972 matrix perturbed by a randomly generated additive term in the range  $[1, 10]$ .

The second series of three examples was constructed from an aggregated 1977 input/output matrix of construction activity in the United States consisting of 205 rows  $\times$  205 columns in the same manner as those in the first series, and are called, respectively, IOC77a, IOC77b, and IOC77c. Examples IOC77a and IOC77b consisted of 58% non-zero elements in the  $X^0$  matrix.

The third and final series of I/O examples was constructed from a 1972 input/output matrix for the United States consisting of 485 rows  $\times$  485 columns in a manner similar to the examples in the first two series. These examples are referred to, respectively, as IO72a, IO72b, and IO72c. These matrices were the sparsest, with only 16% non-zero elements in the  $X^0$  matrix.

As can be seen from Table 2, all of the examples, with the exception of the largest set based on the disaggregated 1972 I/O matrix, required only seconds of CPU time for computation of the solution via SEA. The largest examples required less than 8 minutes.

We now describe the SAM estimation problems. The SAM estimation problems selected were of various sizes. The first four examples were selected because they represented real economic datasets. The last three examples were generated to introduce large-scale SAM problems of a size larger than heretofore considered computationally tractable.

The smallest example, STONE, had also been solved in Byron (1978). The USDA82E example, was a perturbed SAM developed at the United States Department of Agriculture for 1982 (For a description of its development, we refer the reader to Hanson and Robinson (1989)). It was perturbed in order to make it fully dense, and a “difficult” problem. The example SRI is a perturbed example of the SAM for Sri Lanka for 1970 contained in King (1985). TURK is a perturbed SAM for the 1973 Turkish economy discussed in Dervis, De Melo, and Robinson (1982). Examples S500, S750, and S1000 are large-scale randomly generated SAM’s. The convergence tolerance was set at  $\epsilon = .001$ .

As can be seen from Table 3, SEA was very efficient, computing the solution for the first five examples in only fractions of a CPU second. The largest economic dataset, USDA82E, required only several CPU seconds for convergence of SEA. The largest three problems demonstrate the scale of SAM estimation problems that are now solvable in a

reasonable time-frame, even in a serial environment, provided that a robust and efficient algorithm such as SEA is utilized.

In Table 4 we report the performance of SEA on United States migration tables for different time periods. The objective function used was again diagonal, of the form (5). The rows of each migration table represent the origin states, and the columns, the destination states. Alaska, Hawaii, and Washington, DC were removed, creating tables with 48 rows and 48 columns.

The first set of three migration table examples in Table 4 was constructed from a 1955 – 1960 U.S. state to state migration table. The first example, MIG5560a, was formed by generating a distinct random growth factor for each row and column total in the range 0 – 10 %, and by then using the resultant as the  $s_i^0$ , or  $d_j^0$ , (cf. (5)), respectively. All of the weights were set equal to one. The second example, MIG5560b, was formed by generating a distinct growth factor again for each original row and column total, but now within the larger range of 0 – 100%. The third example, MIG5560c was formed by keeping the  $s_i^0$ 's and  $d_j^0$ 's equal to the sums of the corresponding matrix entries in the original table; the  $X^0$  matrix was then constructed by perturbing each element randomly by 0 – 10 %.

The second set of three examples was constructed from the 1965 – 1970 U.S. state to state migration table in a manner similar to the first set, and these examples were named: MIG6570a, MIG6570b, and MIG6570c, respectively.

The third and final set of migration tables was constructed from a 1975 – 1980 U.S. state to state migration table in a manner similar to the examples in the first two sets, and were named MIG7580a, MIG7580b, and MIG7580c, respectively.

As can be seen from Table 4, the migration table examples were computed in only seconds of CPU time. The examples with the greater growth factor were more difficult to solve (as expected) than the corresponding ones with the smaller growth factor range. The examples with the perturbed matrix entries were solved most quickly.

Finally, we turned to the computation of classical spatial price equilibrium problems. Spatial price equilibrium models have been widely applied to the study of agricultural and energy markets. Specifically, we consider spatial price equilibrium problems, characterized by linear supply price, demand price, and transportation cost functions which are also separable. We generated large-scale spatial price equilibrium problems ranging in size from

50 supply markets and 50 demand markets, with a total of 250 variables, to 750 supply markets and 750 demand markets, with a total of 562,500 variables. The convergence tolerance was  $\epsilon = .01$ . As can be seen from Table 5, SEA converged for all of the examples. Although serial equilibration algorithms have been proposed for such problems by Dafermos and Nagurney (1989) and Eydeland and Nagurney (1989), the problem with 750 supply markets and 750 demand markets represents the largest of this class solved to date.

## 4.2 Parallel Experiments

In this Section we describe our experiences concerning a parallel implementation of SEA for the computation of large-scale diagonal quadratic constrained matrix problems. The experiments were carried out on the IBM 3090-600E, a shared memory machine, using its full multiprocessor capabilities.

The SEA algorithm (diagonal version) was embedded with the parallel constructs provided by Parallel FORTRAN (PF) for purposes of task allocation (cf. Figure 2). Task allocation was required for the row equilibration phase and the column equilibration phase, with cycling between the two phases until the convergence criterion was satisfied.

For the computational testing, SEA was compiled using the Parallel FORTRAN (PF) compiler, optimization level (3). We selected four previously solved examples for the parallel tests, specifically, IO72b from Table 2, the  $1000 \times 1000$  example from Table 1, and SP500  $\times$  500 and SP750  $\times$  750 from Table 5. Recall that both IO72b and  $1000 \times 1000$  assume fixed row and column totals, whereas SP500  $\times$  500 and SP750  $\times$  750 are spatial price equilibrium problems, isomorphic to constrained matrix problems with unknown row and column totals.

The speedup measure for  $N$  processors was defined as follows:

$$\text{Speedup } S_N = \frac{T_1}{T_N},$$

where  $T_1$  is the elapsed time to solve the problem using the serial implementation of SEA on a single processor, and  $T_N$  is the elapsed time to solve the problem using the parallel implementation of SEA on  $N$  processors.

The efficiency measure for  $N$  processors was defined as:

$$\text{Efficiency } E_N = \frac{T_1}{T_N \times N}.$$

In Table 6 we report the speedup measurements and the corresponding efficiencies obtained. These measurements were obtained in a standalone environment. The speedup measurements are then displayed graphically in Figure 5.

SEA required 2 iterations for convergence for example IO72b and only 1 iteration for the  $1000 \times 1000$  example. As can be seen from Table 6, SEA exhibited identical speedups of 1.93, or, equivalently, efficiencies of 96.5 % on both diagonal examples when 2 CPU's were used. In the case of 4 CPU's SEA applied to IO72b exhibited 93.5 % efficiency, whereas the  $1000 \times 1000$  example induced an efficiency of 89.4 %. For 6 CPU's, SEA again exhibited the higher speedup for IO72b of 5.15, at an efficiency of 85.5 %. This difference in relative speedups can be explained by the portion of time spent in the serial phase which consists solely of the convergence criterion verification stage for diagonal SEA. The larger example,  $1000 \times 1000$  required more serial time spent in this serial phase. Although SEA required only 1 iteration for this example, the serial convergence step is on the order of  $m^2$  operations, where  $m$  in this example is equal to 1000. On the other hand, even though SEA required 2 iterations in the case of IO72b, since  $m$  in this example is equal to 485, the total time spent in the serial phase would be approximately 50 % less in the smaller example. Enhanced speedups may be obtained by verifying convergence not after every iteration, as was done in these tests, but after every other iteration when the number of iterations is small or by implementing the convergence phase in parallel. Nevertheless, practitioners are interested in the solutions themselves, and, therefore, convergence verification is a vital step.

SEA required 84 iterations for convergence of SP500  $\times$  500 and 104 iterations for convergence of SP750  $\times$  750, where the convergence check was done after every other iteration. In these "elastic" examples, convergence verification again comprised the only serial phase, and was of the order  $m^2$ . Again, the larger example required greater time in the serial phase of convergence verification. Here, enhanced speedups may be obtained by verifying convergence, say, after every five iterations and/or by implementing the convergence step in parallel. The explanation of the greater number of iterations required for convergence of SEA in the case of the elastic examples versus the fixed examples may lie in the initialization phase;  $\mu = 0$  may be closer to the optimal for the latter examples, than for the former ones.

## 5. Computation of Large-Scale General Constrained Matrix Problems

In this Section we describe the computation of large-scale general quadratic constrained matrix problems, formulated in Section 2. Recall that the general quadratic constrained matrix problem is computed via the iterative solution of diagonal constrained matrix problems. SEA for the general case needs substantial storage since the  $G$  matrix may, in fact, be fully dense. Hence, for an initial matrix  $X^0$ , consisting of 100 rows  $\times$  100 columns, the corresponding  $G$  matrix would be of dimension  $10,000 \times 10,000$ .

### 5.1 Serial Experiments

In this Section we provide the results of serial experimentation. The SEA, RC (Nagurney, Kim, and Robinson (1990)), and B-K (Bachem and Korte (1978)) programs for the general quadratic problems were coded in FORTRAN, compiled under VS FORTRAN at optimization level (3) running under VM/XA 5.5. Details of the implementation of the B-K algorithm are given in Nagurney, Kim, and Robinson (1990).

#### 5.1.1 Computational Comparisons of SEA, RC, and B-K

Similar to the general SEA, the RC algorithm is also an equilibration algorithm based on the projection method, and involves the iterative solution of diagonal constrained matrix problems. It, however, first considers the general objective function (1) subject to only the row constraints, and then subject to the column constraints. A graphical representation of the RC algorithm for the general problem is presented in Figure 6.

Our computational comparisons of SEA versus RC and B-K are conducted for the general constrained matrix problem with fixed row and column totals, since both RC and B-K were designed for this class of constrained matrix problems.

The matrix  $G$  was generated to be symmetric and strictly diagonally dominant, which ensured positive definiteness, with each diagonal term generated in the range [500, 800], but allowing for negative off-diagonal elements to simulate variance-covariance matrices. Each element of the linear term coefficients in the expansion of (1) was generated uniformly in the range [100, 1000]. The same convergence criterion was used for B-K, RC, and SEA, with  $\epsilon' = .001$ .

The implementation of both SEA and general RC was done in accordance with the guidelines for the implementation of equilibration algorithms contained in Eydeland and

Nagurney (1989) . The general problems computed with both SEA and RC ranged in size of  $X^0$  matrices from  $10 \times 10$  to  $120 \times 120$ , with the corresponding  $G$  matrices ranging in size from  $100 \times 100$  to  $14400 \times 14400$ , respectively. The STRAIGHT INSERTION SORT was used for the implementation of exact equilibration, since the arrays to be sorted ranged in length from 10 elements to 120 elements.

Table 7 presents computational comparisons of SEA versus RC and B-K on general quadratic constrained matrix problems solved in Nagurney, Kim, and Robinson (1990) with 100% dense  $G$  matrices. As can be seen from Table 7, SEA outperformed RC by a factor of 3 to 4, and outperformed B-K by as much as two orders of magnitude. The larger problems were not solved using B-K because it became prohibitively expensive to do so.

### 5.1.2 Computational Experience with SEA on Migration Tables

In this Section we considered United States migration tables for different time periods, for which the constrained matrix formulation with objective function (1) was again used. The weighting matrix  $G$  was generated in the same manner as in Section 4.1.1. These United States migration tables, from which we constructed the examples, consisted of 48 rows and 48 columns. The rows of each migration table represented the origin states and the columns the destination states. Alaska, Hawaii, and Washington, DC, were removed, thus creating tables with 48 rows and columns. The  $G$  matrices were, hence, of dimension  $2304 \times 2304$ . The examples, reported in Table 8, were as follows.

The first set of two examples, GMIG5560a and GMIG5560b, were based on the 1955–1960 U.S. state to state migration table. GMIG5560a consisted of the baseline table with row and column totals being fixed and consisting of a growth factor in the range 0 – 10 %. GMIG5560b, then, in addition, to the row and column total perturbations, had each individual matrix entry perturbed by a distinct growth factor, also in the range 0 – 10 %.

The second and third sets of two examples each were based on the 1965 – 1970 and the 1975 – 1980 U.S. state to state migration tables, respectively, and were constructed in a manner similar to the examples in the first set. As can be seen from Table 8 below, all of the examples were solved via SEA in approximately 25 seconds of CPU time with  $\epsilon'$  set to .001.

## 5.2 Parallel Experiments with SEA

For purposes of parallel experimentation, we selected the  $10000 \times 10000$  example contained in Table 7, which had also been solved using a parallel implementation of RC in Nagurney, Kim, and Robinson (1990). The parallel implementation of the SEA algorithm (cf. Figure 4) used Parallel FORTRAN (PF) as did the RC algorithm. For the computational testing, both SEA and RC were compiled using the PF compiler, optimization level 3.

Recall that SEA resolves general quadratic constrained matrix problems into series of diagonal row and column equilibrium subproblems. In RC, the parallel phases of exact row equilibration and exact column equilibration are separated via the serial phase of projection method convergence verification, which adds a serial phase not encountered in the parallelization of the SEA algorithm (cf. Figures 4 and 6), in which convergence verification of the projection method is only done once.

The example was solved in 2 iterations of general RC and in 1 iteration via SEA. The  $10000 \times 10000$  example required for RC in the first iteration, 4 iterations of the projection method for row equilibration and 3 iterations of the projection method for column equilibration, whereas in the second iteration, 4 iterations for both equilibrations were required. SEA, on the other hand, besides requiring only a single outer iteration, only required two inner iterations.

Table 9 contains the speedups and the efficiencies obtained, whereas Figure 7 graphically depicts the speedups for both SEA and RC. These speedups and efficiencies were obtained in a standalone environment.

As can be seen from Table 9, SEA exhibited higher speedups than RC for the example. In the case of 2 CPU's, SEA exhibited a speedup of 1.82 versus 1.75 obtained with RC; in the case of 4 CPU's, SEA exhibited a speedup of 2.62 versus 2.24 obtained with RC. Hence, SEA registered an improvement in absolute efficiency of 3.03 % in the case of 2 CPU's and 9.59 %, in the case of 4 CPU's. These parallel results, and the serial results contained in Section 5.1.1, strongly suggest that the new Splitting Equilibration Algorithm is better suited for parallelization than RC and more effective for the computation of large-scale constrained matrix problems, in either a serial or a parallel environment.

## Acknowledgements

The first author's work was supported by NSF Grant RII-880361 under the NSF VPW program and by a 1990 Faculty Fellowship Award from the University of Massachusetts at Amherst while she was a visiting faculty member at the Transportation Systems Division, the Operations Research Center, and the Sloan School of Management at the Massachusetts Institute of Technology. The cordiality and hospitality of the host institution are warmly appreciated. The second author's work was supported by NSF Grant: DMS-8602316 and completed while he was on sabbatical leave at the Courant Institute, New York University.

The research was conducted at the Cornell National Supercomputer Facility, a resource of the Center for Theory and Simulation in Science and Engineering at Cornell University, which is funded in part by the National Science Foundation, New York State, and the IBM Corporation.

The authors would like to thank Karen Polenske and Nic Rockler of the Department of Urban Planning at MIT for providing the input/output datasets, Waldo Tobler of the Department of Geography at the University of California at Santa Barbara for providing the migration tables, and Kenneth Hanson of the US Department of Agriculture, Economic Research Service, for providing the USDA social accounting matrix.

The authors would like to thank Dae-Shik Kim for assistance with the numerical runs and Alan G. Robinson of the Department of General Business and Finance in the School of Management at the University of Massachusetts at Amherst for stimulating conversations during early phases of this research.

The authors are indebted to Francesca Verdier of the Cornell Theory Center for assistance in the standalone runs.



## References

Bacharach, M. (1970), **Biproportional scaling and input-output change**, Cambridge University Press, Cambridge, UK.

Bachem, A. and Korte, B., (1978), "Algorithm for quadratic optimization over transportation polytopes," *Zeitschrift fur Angewandte Mathematik und Mechanik*, **58**: T459-T461.

Bertsekas, D. P. and Tsitsiklis, J. N., (1989), **Parallel and Distributed Computation: Numerical Methods**, Prentice-Hall, Englewood Cliffs, New Jersey.

Byron, R. P., (1978), "The estimation of large social accounts matrices," *Journal of the Royal Statistical Society Series A*, **141**: 359-369.

Cottle, R. W., Duvall, S. G., and Zikan, K., (1986), "A lagrangean relaxation algorithm for the constrained matrix problem," *Naval Research Logistics Quarterly*, **33**: 55-76.

Dafermos, S., (1982), "The general multimodal network equilibrium problem with elastic demand," *Networks* **12**: 57-72.

Dafermos, S., (1983), "An iterative scheme for variational inequalities," *Mathematical Programming*, **26**: 40- 47.

Dafermos, S. and Nagurney, A., (1989), "Supply and demand equilibration algorithms for a class of market equilibrium problems," *Transportation Science*, **23**: 118-124.

Deming, W. E. and Stephan, F. F., (1940), "On a least-squares adjustment of a sampled frequency table when the expected marginal totals are known," *Annals of Mathematical Statistics*, **11**: 427-444.

Dervis, K., De Melo, J., and Robinson, S., (1982), **General equilibrium models for development policy**, Cambridge University Press, Cambridge, United Kingdom.

Enke, S., (1951), "Equilibrium among spatially separated markets: solution by electric analogue," *Econometrica*, **19**, 40-48.

Eydeland, A. and Nagurney, A., (1989), "Progressive equilibration algorithms: the case of linear transaction costs," *Computer Science in Economics and Management*, **2**, 197-219.

Friedlander, D., (1961), "A technique for estimating a contingency table given the marginal total and some supplementary data," *Journal of the Royal Statistical Society A*, **124**: 412-420.

Hanson, K. A. and Robinson, S., (1989), "Data, linkages, and models: U.S. national income and product accounts in the framework of a social accounting matrix," Agriculture and Rural Economy Division, Economic Research Service, U. S. Department of Agriculture. Staff Report No. AGES 89-5.

Harrigan, F. and Buchanan, I., (1984), "Quadratic programming approach to input-output estimation and simulation," *Journal of Regional Science*, **24**, no. 3: 339-358.

Judge, G. G. and Yancey, T. A., (1986), **Improved methods of inference in econometrics**, North-Holland.

King, B. B., (1985), "What is a SAM?", in **Social accounting matrices: a basis for planning**, G. Pyatt and J. I. Round (eds.), The World Bank, Washington, DC.

Kliniewicz, J., (1989), "Implementing an exact Newton method for separable convex transportation problems," *Networks*, **19**: 95-105.

Miller, R. E. and Blair, P. D., (1985), **Input/output analysis: foundations and extensions**, Prentice-Hall, Englewood Cliffs, N. J.

Mohr, M., Crown, W. H., and Polenske, K. E., (1987), "A linear programming approach to solving infeasible RAS problems," *Journal of Regional Science*, **27**, no. 4: 587-603.

Mosteller, F. and Tukey, J. W., (1977), **Data analysis and regression**, Addison-Wesley Publishing Co., Inc.

Nagurney, A., (1987), "Competitive equilibrium problems, variational inequalities, and regional science," *Journal of Regional Science*, **27**: 503-514.

Nagurney, A., (1989), "An algorithm for the solution of a quadratic programming problem with application to constrained matrix and spatial price equilibrium problems," *Environment and Planning A*, **21**: 99-114.

- Nagurney, A., Kim, D. S., and Robinson, A. G., (1990), "Serial and parallel equilibration of large-scale constrained matrix problems with application to the social and economic sciences," *The International Journal of Supercomputer Applications*, **4.1**, Spring, 49-71.
- Nagurney, A. and Robinson, A. G., (1989), "Equilibration operators for the solution of constrained matrix problems," Operations Research Center Working Paper, OR 196-89, Operations Research Center, MIT, Cambridge, MA.
- Ohuchi, A. and Kaji, I., (1984), "Lagrangian dual coordinatewise maximization for network transportation problems with quadratic costs," *Networks*, **14**: 525-530.
- Polenske, K. E., (1980), **U. S. multiregional input-output accounts and model**, Lexington Books, Lexington, MA.
- Pyatt, G. and Round, J. I., (1985), (eds.), **Social accounting matrices: a basis for planning**, The World Bank, Washington, DC.
- Samuelson, P. A., (1952), "A spatial price equilibrium and linear programming," *American Economic Review*, **42**: 283-303.
- Stone, R., (1951), "Simple transaction models, information, and computing," *The Review of Economic Studies*, **XIX (2)**, **49**: 67-84.
- Stone, R., (1962), "Multiple classifications of social accounting," *Bulletin de l'Institut International de Statistique*, **39**: 215-233.
- Takayama, T. and Judge, G. G., (1971), **Spatial and temporal price and allocation models**, North-Holland, Amsterdam.
- Van der Ploeg, F., (1982), "Reliability and adjustment of sequences of large economic accounting matrices," *Journal of the Royal Statistical Society A*, **145**: 169-184.
- Van der Ploeg, F., (1988), "Balancing large systems of national accounts," *Computer Science in Economics and Management*, **1**: 31-39.
- Van der Sluis, A., (1969), "Condition numbers and equilibration of matrices," *Numerische Mathematik*, **14**: 14-23.

$$\begin{pmatrix} x_{11}^0 & x_{12}^0 & \dots & x_{1n}^0 \\ x_{21}^0 & x_{22}^0 & \dots & x_{2n}^0 \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1}^0 & x_{m2}^0 & \dots & x_{mn}^0 \end{pmatrix} \implies \begin{matrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{matrix} \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix}$$

$$\begin{matrix} d_1 & d_2 & \dots & d_n \end{matrix}$$

Initial

Matrix  $X^0$

Matrix

Estimate  $X$

Figure 1: The Constrained Matrix Problem

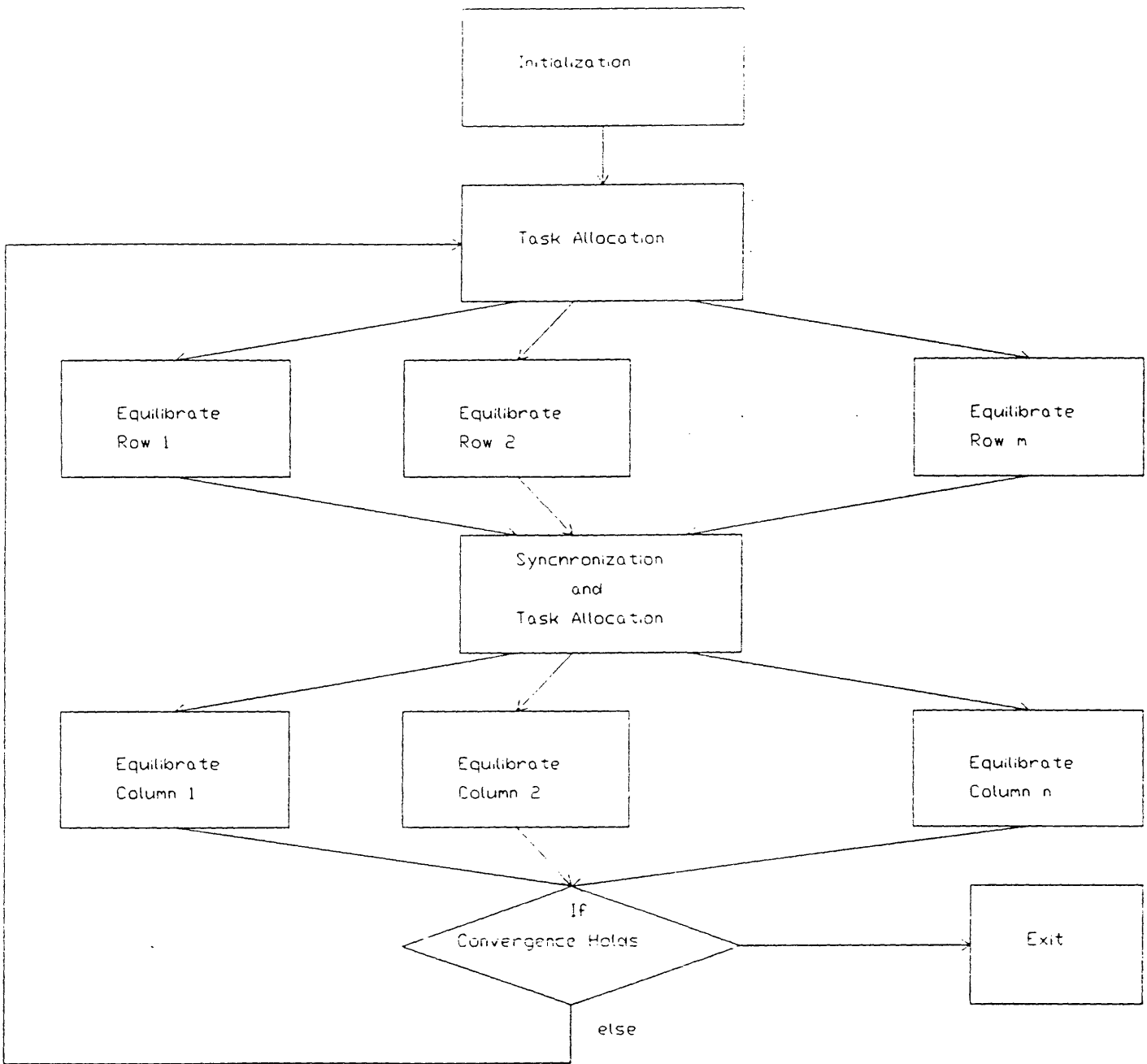
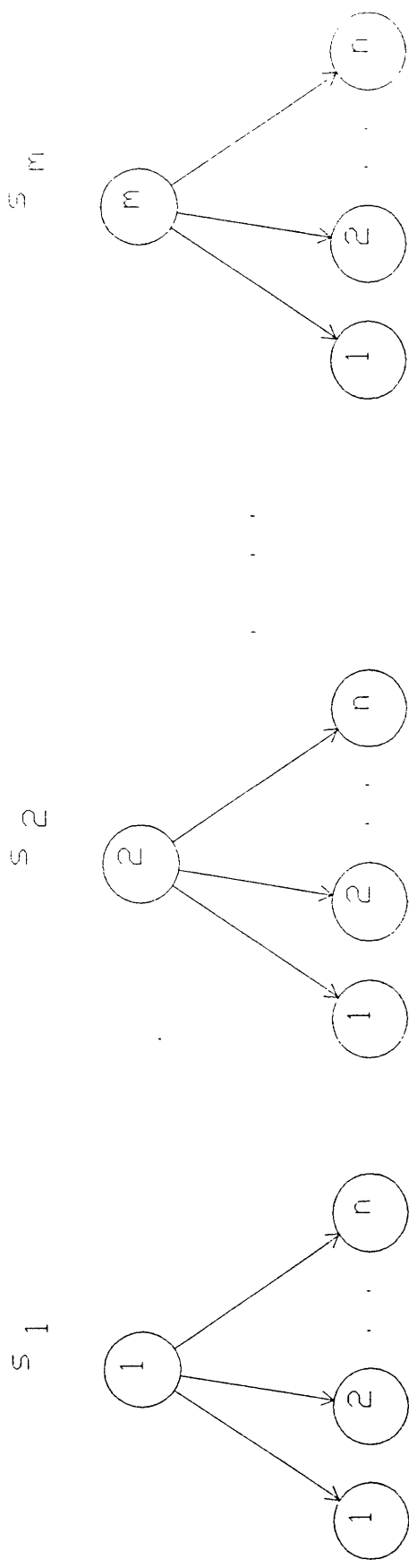
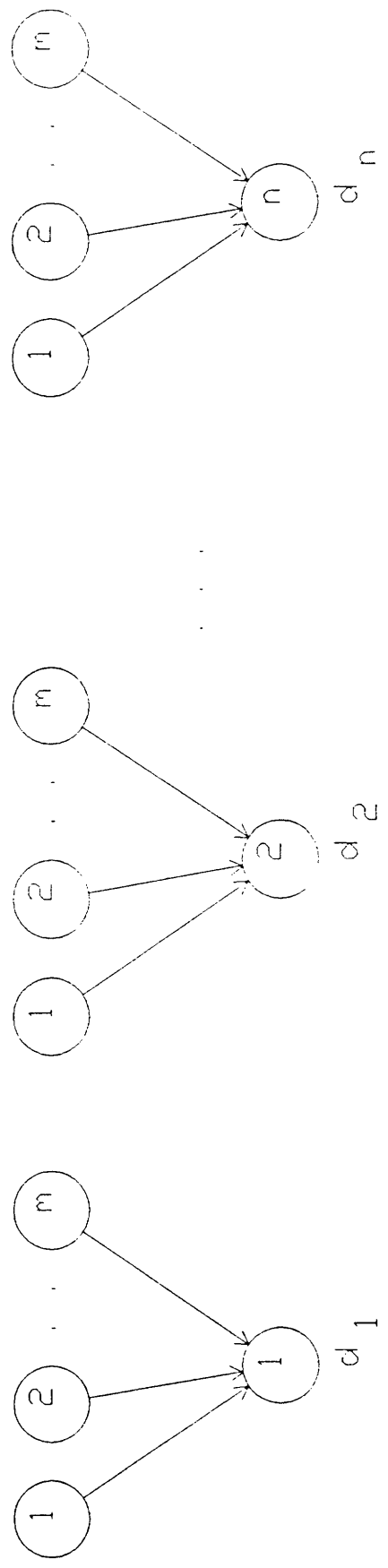


Figure 2: Flowchart of SEA - the Diagonal Case



Row Equilibration Step



Column Equilibration Step

Figure 3: Network Structure of Row and Column Equilibrium Subproblems

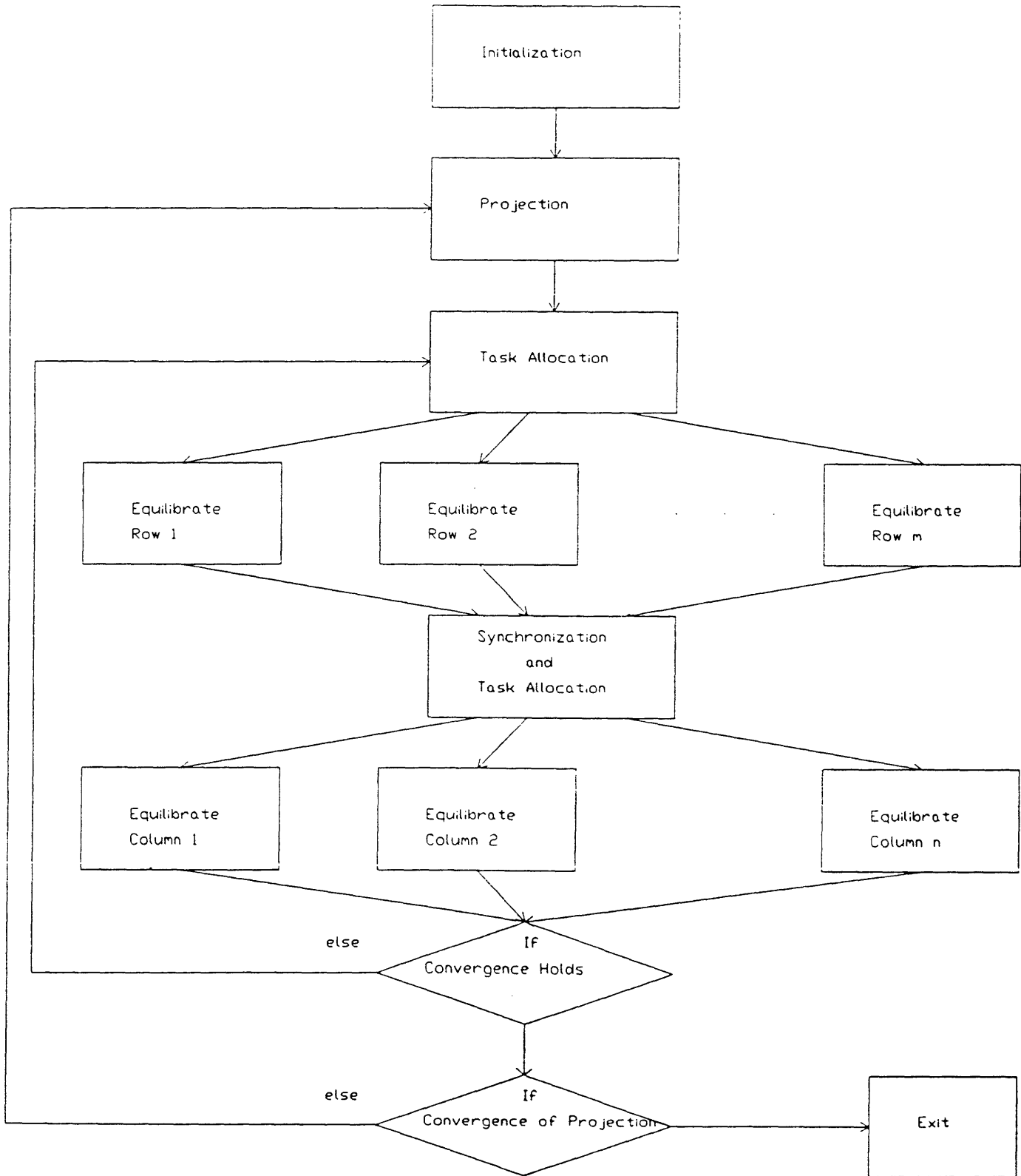
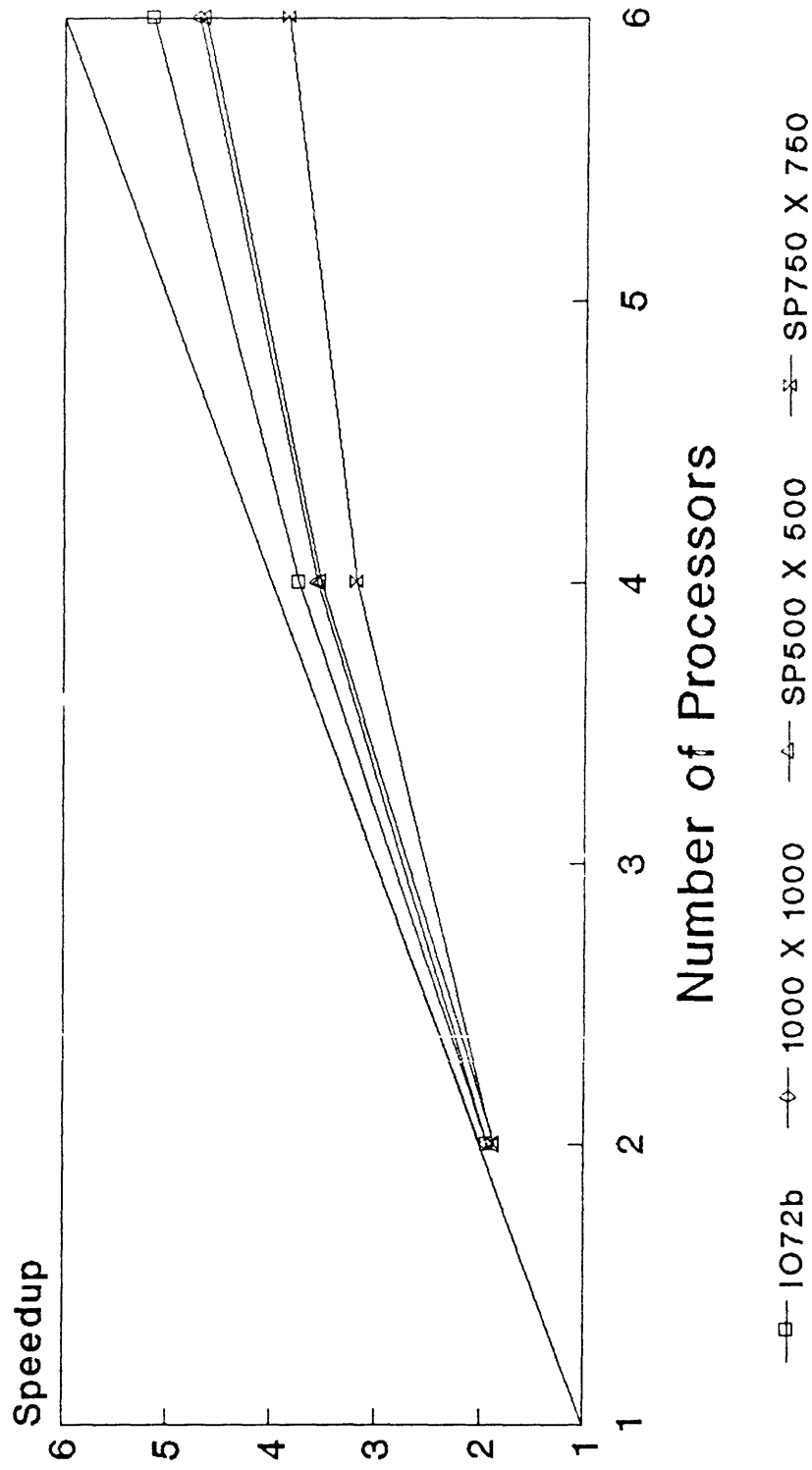


Figure 4: Flowchart of SEA - the General Case

# Figure 5: Speedup for SEA on Diagonal Problems





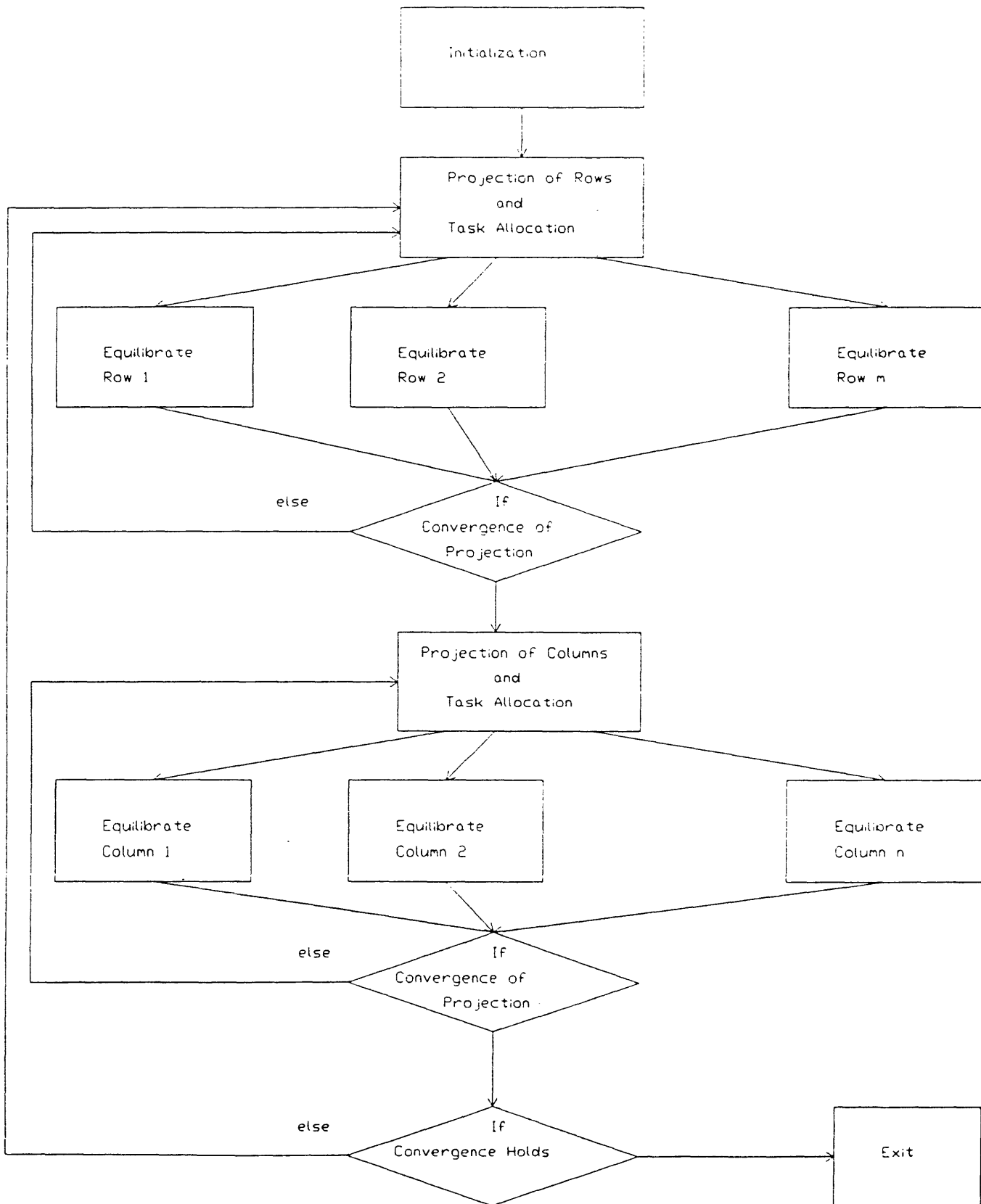


Figure 6: Flowchart of the RC Algorithm - the General Case

# Figure 7: Speedup for SEA vs RC on General Problems

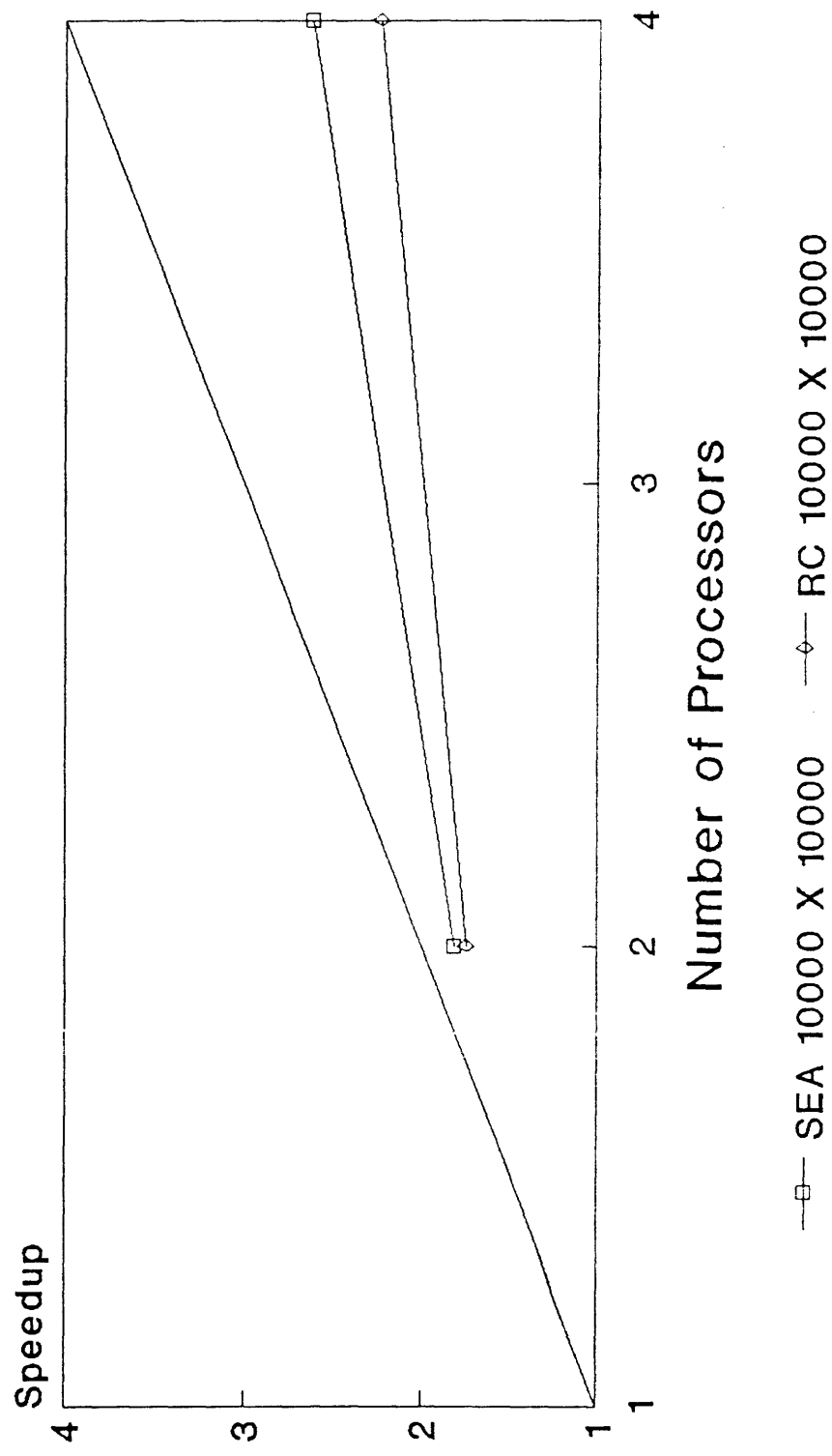


Table 1: Computational Experience with SEA on Large-Scale Diagonal Quadratic Constrained Matrix Problems\*

$m \times n$	# of non-zero $x_{ij}^0$ variables	CPU time (seconds)
$750 \times 750$	562,500	204.7476
$1000 \times 1000$	$1 \times 10^6$	483.2065
$2000 \times 2000$	$4 \times 10^6$	3,823.2139
$3000 \times 3000$	$9 \times 10^6$	13,561.5703

\*CPU time based on a single example

Table 2: Computational Experience with SEA on United States Input/Output Matrix Datasets

Dataset	CPU time (seconds)
IOC72a	18.6697
IOC72b	18.9923
IOC72c	25.6035
IOC77a	13.6168
IOC77b	19.1338
IOC77c	30.2037
IO72a	333.2691
IO72b	438.3519
IO72c	335.6124

Table 3: Computational Experience with SEA on Social Accounting Matrix Datasets

Dataset	# of accounts	# of transactions	CPU time (seconds)
STONE	5	12	.0024
TURK	8	19	.0210
SRI	6	20	.009
USDA82E	133	17,689	5.7598
S500	500	250,000	28.99
S750	750	562,500	52.60
S1000	1000	1,000,000	95.08

Table 4: Computational Experience with SEA on United States Migration Tables

Dataset	CPU time (seconds)
MIG5560a	1.5935
MIG5560b	4.1367
MIG5560c	.8932
MIG6570a	1.2915
MIG6570b	3.9714
MIG6570c	.8203
MIG7580a	3.5168
MIG7580b	9.1067
MIG7580c	.8041

Table 5: Computational Experience with SEA on Spatial Price Equilibrium Problems

$m \times n$	# of Variables	CPU time (seconds)
SP50 $\times$ 50	250	1.3822
SP100 $\times$ 100	10,000	11.2621
SP250 $\times$ 250	62,500	129.4597
SP500 $\times$ 500	250,000	540.7056
SP750 $\times$ 750	562,500	1589.0613

Table 6: Parallel Speedup and Efficiency Measurements for SEA on Diagonal Problems

Example	$N$	$S_N$	$E_N$
IO72b	2	1.93	96.5%
	4	3.74	93.5%
	6	5.15	85.8%
$1000 \times 1000$	2	1.93	96.5%
	4	3.57	89.4%
	6	4.71	78.5%
SP500 $\times$ 500	2	1.86	92.85%
	4	3.52	88.10%
	6	4.66	77.75%
SP750 $\times$ 750	2	1.87	93.79%
	4	3.19	79.80%
	6	3.86	64.34%



Table 7: Computational Comparisons of SEA, RC, and B-K on General Quadratic Constrained Matrix Problems with 100% Dense G Matrix

Dimension of G	# of of runs	CPU time (seconds)		
		SEA	RC	B-K
$100 \times 100$	10	.0194	.1270	.7725
$400 \times 400$	10	.5694	1.8373	78.9557
$900 \times 900$	2	2.9767	9.5129	1458.3820
$2500 \times 2500$	1	21.4607	71.4807	—
$4900 \times 4900$	1	81.2640	428.8780	—
$10000 \times 10000$	1	353.6885	1305.5940	—
$14400 \times 14400$	1	1254.731	3000.5200	—

Table 8: Computational Experience with SEA on General Constrained Matrix Problems Consisting of United States Migration Tables with 100% Dense G Matrices — Dimension of G —  $2304 \times 2304$

Dataset	CPU time (seconds)
GMIG5560a	23.16
GMIG5560b	22.99
GMIG6570a	23.57
GMIG6570b	23.28
GMIG7580a	28.73
GMIG7580b	23.49

Table 9: Parallel Speedup and Efficiency Measurements for SEA and RC on General Problems

Example	$N$	$S_N$	$E_N$
SEA	2	1.82	90.77%
10000 $\times$ 10000	4	2.62	65.49%
RC	2	1.75	87.7%
10000 $\times$ 10000	4	2.24	55.9%