1-1-1992

Parallel computation of large-scale network equilibria and variational inequalities.

Dae-Shik Kim
University of Massachusetts Amherst

PARALLEL COMPUTATION OF LARGE-SCALE NETWORK EQUILIBRIA AND VARIATIONAL INEQUALITIES

A Dissertation Presented

by

Dae-Shik Kim

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 1992

School of Management

PARALLEL COMPUTATION OF LARGE-SCALE NETWORK EQUILIBRIA AND VARIATIONAL INEQUALITIES

A Dissertation Presented

by

DAE-SHIK KIM

Approved as to style and content by:

Anna Nagurney, Chair

Robert Nakosteen, Member

James Smith, Member

Alexander Eydeland, Member

Ronald Karren, Program Director
Ph.D. Program
School of Management

To my wife Young-Soon
and
to my sons Tae-Woo and Jeong-Woo

ACKNOWLEDGMENTS

I would like to thank Prof. Anna Nagurney for her willingness to discuss Management Science at any time. Her illuminating comments have been essential to this work. Many useful discussions I had with Prof. Robert Nakosteen, Prof. James Smith and Prof. Alexander Eydeland are gratefully acknowledged. I am also grateful to all my committee members for patiently going through my thesis, making useful comments on the contents, and helping me to finish all requirements before the last day for the May graduation.

I wish to acknowledge the support, understanding, praying of my parents and family through chanllenging times. Most of all, I would like to thank my Lord, Jesus Christ, who gave me ideas and wisdom throughout this thesis.

This research was supported, in part, by assistantships and summer research grants from the School of Management, University of Massachusetts, Amherst, by NSF Grant, RII-8800361, and by cooperative agreement No. 58-3AEN-0-80066 from the Economic Research Service of the United States Department of Agriculture.

This research was conducted at the Cornell National Supercomputing 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 by the IBM Corporation, and the Northeast Parallel Architecture Center (NPAC) at Syracuse University. This support is greatly acknowledged.

ABSTRACT

PARALLEL COMPUTATION OF LARGE-SCALE NETWORK EQUILIBRIA AND VARIATIONAL INEQUALITIES

May 1992

DAE-SHIK KIM

B.S., SEOUL NATIONAL UNIVERSITY, SEOUL, KOREA

M.S., KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY

M.S., UNIVERSITY OF MASSACHUSETTS

M.B.A., UNIVERSITY OF MASSACHUSETTS

Ph.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor Anna Nagurney

Equilibrium of a network is obtained when each user who competes to optimize his utility can not improve his utility any further. Equilibrium problems governed by distinct equilibrium concepts can be formulated in one general framework - that of variational inequalities. The synthesis of variational inequalities and networks induces the creation of highly efficient algorithms which are especially suited for the large-scale equilibrium problems. Motivated by the recent technological advances in parallel computing architectures, parallel algorithms of large-scale equilibrium problems were developed using the theory of variational inequalities.

In the case where the feasible constraint set of a network equilibrium problem can be expressed as a Cartesian product of subsets, the application of variational

inequality decomposition algorithms for the parallel computation becomes possible. A new spatial price equilibrium model, which is not based on the path flows, but, rather, on the link flows to allow the decomposition by time periods, was developed and used as a prototype of large-scale network equilibrium problems. The variational inequality formulations were decomposed first by commodities, then by time periods, and, subsequently, by markets. The coarse grain parallel architectures used were the IBM 3090-600E and the IBM 3090-600J at the Cornell Theory Center with six processors each. The maximum speed-ups obtained were 1.93 for two processors, 3.74 for four processors, and 5.15 for six processors.

The market subproblems were further decomposed by links, resulting in a fine grain parallel implementation. The Thinking Machine's Connection Machine, CM-2, with 32,768 processors was used for the numerical experimentation. The fine grain parallel algorithm solved input/output matrix problems more than 20 times faster, when compared to the results on the IBM 3090-600J.

It is expected that further enhancements to parallel languages and parallel architectures will make even more efficient implementations realizable, and that parallel computing and the theory of variational inequalities can be successfully applied to solve more efficiently other large-scale problems with an underlying network structure, such as traffic equilibrium problems, general economic equilibrium problems, and financial equilibrium problems.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Network models have been used to formulate many problems in management science as well as in engineering. The advantages of a network formalism are twofold: network models can visually represent a wide variety of applications and can also suggest decomposition algorithms that exploit the underlying problem structure. Historically, network optimization problems have been studied widely. Another large class of network models is that of network equilibrium problems. For example, if more than one user is involved in a system and each user competes to optimize his utility, then at some point he will not be able to improve his utility any further; in other words, the properties of the system will not change with time, that is, the system reaches an equilibrium state.

Equilibrium is, hence, a central concept in diverse problems in management science as well as in engineering. Examples in management science include: traffic network equilibrium problems in which users of the congested network seek routes to minimize travel costs, oligopolistic market equilibrium problems in which profit-maximizing firms are engaged in the production and sale of one or more goods, spatial price equilibrium problems in which optimal commodity supply and consumption levels and interregional shipments are to be determined, and

general economic equilibrium problems. Furthermore, many business application problems such as financial planning can be formulated as network equilibrium models. The equilibrium solutions can assist in governmental policy analyses, where the effects of regulatory instruments such as price controls in the form of price floors and ceilings, trade restrictions, tariffs, and taxes are to be determined. Each user can also obtain useful information for personal decision-making under competitive situations from such models.

Historically, equilibrium models were usually reformulated as optimization problems, provided that a certain symmetry or integrability assumption held for the underlying functions. Convex programming algorithms could then, at least in principle, be used to compute the equilibrium pattern. Utilizing such an approach, Samuelson [77] and Takayama and Judge [82] introduced a variety of spatial price equilibrium models, and Beckmann, McGuire, and Winsten [7] studied traffic network equilibrium models.

Recently, it has been realized that equilibrium problems governed by distinct equilibrium conditions can be formulated in one general framework - that of variational inequalities. Variational inequalities (VI) had originally been introduced by Hartmann and Stampacchia [36] for the study of partial differential equations, in which the applications were derived from mechanics. The discovery by Dafermos [15] that the traffic network equilibrium conditions, as stated by Smith [79], had the structure of a finite-dimensional VI problem opened new vistas not only for the study of qualitative properties of existence, uniqueness, and sensitivity of equilibrium solutions, but also for the development of more general, asym-

metric multicommodity and multimodal models. Further, VI theory permits the design of mathematically correct and convergent algorithms for the computation of solutions to general equilibrium problems for which no equivalent optimization formulation exists.

Indeed, many equilibrium problems have now been formulated as variational inequality problems. For example, Gabay and Moulin (1980) formulated the oligopolistic equilibrium problem governed by Cournot-Nash equilibrium as a VI problem; Florian and Los [30] formulated the spatial price equilibrium problem as a VI problem; Border [10] and Dafermos [18], in turn, formulated exchange price equilibria. More recently, Zhao [88] utilized the VI formulations of the pure exchange problem and the general economic equilibrium problem with production to develop convergent algorithms for these equilibrium problems under a monotonicity assumption on the excess demand functions. For the exposition of the status of the theory, alternative model formulations, and applications, as well as a comprehensive list of references, see the review articles of Magnanti [45], Dafermos [19], and Nagurney [52], and the recent thesis of Zhao [88].

Even though many efficient algorithms have been developed, they may not work as efficiently as the application demands when the problem size increases, due to the limitation of serial computers. The technological advances in computer architecture provide new approaches solving such large-scale problems efficiently. The combination of parallel computing and variational inequality algorithms is an area, hence, that merits further research. In this thesis, parallel algorithms for large-scale network equilibrium and optimization problems were developed

and implemented. To develop the parallel algorithms, the theory of variational inequalities was used.

1.1 Background

In this section, two basic concepts of the thesis, parallel computation and variational inequalities, are briefly introduced. In the first subsection, the need for parallel computers is highlighted and then some important basic terminologies of parallel computation are explained. In the second subsection, the theory of variational inequalities, which is a powerful method to develop parallel algorithms, is briefly discussed.

1.1.1 Parallel Computation

Traditional serial computers are characterized by the presence of a single locus of control that determines the next instruction to be executed. The data to be operated upon, during the execution of each instruction, are fetched from a global memory, one at a time. Thus, only one instruction is executed at a time and the speed of computation is slow. As the problem size increases, serial computers can not solve problems as efficiently as the application may demand. To circumvent this bottleneck, supercomputers with vector facility were developed. More recently, parallel computers with many cheaper processors have been developed.

Parallel computing systems consist of several processors that are located within a small distance of each other to execute jointly a computational task.

Parallel computation is motivated by a variety of factors. There has been a need for the solution of very large-scale computational problems, but it is only recently that technological advances have raised the possibility of massively parallel computation and have made the solution of such problems possible. Furthermore, the availability of powerful parallel computers is generating interest in new types of problems that were not addressed in the past. Accordingly, the development of parallel algorithms is guided by this interplay between old and new computational needs on the one hand, and technological progress on the other.

The original needs for fast computation arose in a number of contexts involving, principally, partial differential equation applications, such as computational fluid dynamics and weather prediction, as well as image processing, etc. In these applications, there is a large number of numerical computations to be performed. The desire to solve more and more complex problems has always been running ahead of the capabilities of the time, and has provided a driving force for the development of faster, and possibly parallel, computing machines. The above mentioned types of problems can be easily decomposed along a spatial dimension, and have, therefore, been prime candidates for parallelization, with a different computational unit (processor) assigned the task of manipulating the variables associated with a small region in space, thus leading to the design of parallel computers consisting of a number of processors with nearest neighbor connections. More recently, there has been increased interest in other types of large-scale computation. Some examples in management science relate to the solutions of mathematical programming, optimization problems, equilibrium

problems, and queuing problems. A common property of such problems, as they arise in practice, is that they can be decomposed. After being properly decomposed, each subproblem is assigned to a separate processor, and each processor runs independently. The details of the parallel algorithms depend upon the types of problems and the parallel architectures.

Parallel computers can be classified in many different ways. A parallel system with thousands of processors is called *massively parallel*, while a system with a small number of processors is called *coarse-grained parallel*. By the communication method of the data, the parallel computers can be grouped into *shared memory* and *message passing* computers. In a shared memory system, all the data are in one big memory that is available to all processors. When in such a machine one processor needs data that must be calculated by another processor, then it must know whether the data in the shared memory are already the required data or whether the other processor is still busy with the calculation. In a message passing system, the memory is distributed among the processors. in other words, each processor has its own local memory. Hence, the data are also distributed and when a processor needs data from another processor it must send a message asking for those data. Processors communicate through specially designed interconnection networks and this data communication becomes more expensive when two processors are spatially very separated.

The most popular classification of parallel architectures is based upon the level of detail at which the operation of parallel processors is controlled. In a SIMD (Single Instruction, Multiple Data) machine, each instruction can act on

more data streams. For example, in a multi-processor, a number of processors all execute the same instruction but on different data. SIMD system comprises a master control unit and a number of identical processing elements. The control unit transmits the same instruction stream to each of the processing elements. In a MIMD (Multiple Instruction, Multiple Data) machine, a number of independent processors execute different instructions on different data concurrently.

The innovation of parallel computing has added a new dimension to the design of algorithms and programs. Parallel programming is not a simple extension of serial programming. In order to exploit the possibilities offered by parallelism, programmers need to think in parallel and reconsider the solution process. For example, the matrix addition can be easily parallelized by assigning each element addition to each processor. but for matrix multiplication, it is not that easy to parallelize. Programmers should think in parallel in order to find the most efficient way in which to both construct and implement an algorithm. Experience has shown that the judgments of the efficiency based on serial techniques can be easily overturned in a parallel environment. Algorithms that are obsolete so far as implementation on serial computers is concerned often show a high degree of parallelism - i.e., they may contain numerous subcalculations that are independent of one another and may, therefore, be executed simultaneously - and can, hence, outperform the best serial techniques. In addition to demanding new techniques of algorithm design, the introduction of extended parallelism also requires us to rethink what is known about systems, parallel languages, non-numerical problems, and numerical methods in general.

In the applications of parallel programming, the main concerns are cost and speed: the hardware should not be prohibitively expensive, and the computation should terminate within an amount of time that is acceptable for the particular application. There are several issues related to parallelization that do not arise in a serial context. A first issue is task allocation, that is, the breakdown of the total workload in smaller tasks assigned to different processors, and the proper sequencing of the tasks when some of them are interdependent and cannot be executed simultaneously. A second issue is the communication of data between the processors. A third issue is the synchronization of the computations of different processors. In synchronous methods, processors must wait at predetermined points for the completion of certain computations or for the arrival of certain data. In asynchronous methods, there is no requirement for waiting at predetermined points.

Two theoretical indices have been used for measuring the performance of a developed parallel algorithm; speed-up and efficiency. Speed-up is defined as the ratio of the time required for the parallel calculation to that of a serial computation. The theoretical maximum value of the speed-up is equal to the number of processors, and it is obtained when the algorithm is fully parallel and the calculation is distributed equally among the processors. This would require all the processors to be operable simultaneously, each processor being capable of performing all the arithmetic operations, and incurring no memory data-movement penalties. The efficiency is the ratio of obtained speed-up to the theoretical maximum value of speed-up. When actual performance on particular machines is compared, the

number of MFLOPS (Mega Floating Point Operations per Second) and GFLOPS (Giga Flops) is used.

The speed-up of parallel algorithms will generally increase with the number of processors. But the efficiency will decrease after a certain number of processors. This phenomena is explained by Amdahl's Law (see Riele, Dekker, and Vorst [76]), where

$$\text{Speed-up} = \frac{T_s}{(1-f)\cdot T_s + f \cdot T_s/p}, \tag{1.1}$$

f the parallelized fraction, T_s is the serial computing time, and p is the number of processors of the parallel implementation. When f approaches one, the speed-up would be close to the maximum value and it increases linearly with the increase of processors. When f is not close enough to one, speed-up does not increase linearly. As p increases, the second term of the denominator becomes smaller and the first term dominates the speed-up. Then the limit of the speed-up will be $\frac{1}{1-f}$. Therefore, there is no speed-up if f is zero. To maximize the speed-up, the parallelized fraction, f, should be close enough to one.

1.1.2 *Variational Inequalities*

In this section, the basic theory of variational inequalities is briefly reviewed. For amplified discussions, see the book by Kinderlehrer and Stampacchia [41], the surveys of Magnanti [45], Dafermos [19], Nagurney [52], and the thesis of Zhao [88].

The finite-dimensional variational inequality problem, heretofore denoted as VI(K, f), is to determine the vector $x \in R^n$ such that

$$f(x) \cdot (x' - x) \geq 0, \quad \text{for all } x' \in K \tag{1.2}$$

where K is a closed convex subset of R^n and $f(\cdot)$ is a known function from K to R^n. The qualitative theory of variational inequalities in terms of existence and uniqueness has reached an advanced state. For example, in the case where $f(\cdot)$ is continuous and the feasible set K is bounded, there exists a solution of inequality (1.2). If K is bounded and $f(\cdot)$ is strictly monotone, then there exists a unique solution of inequality (1.2). If K is unbounded, then strong monotonicity of $f(\cdot)$, that is,

$$(f(x^1) - f(x^2)) \cdot (x^1 - x^2) \geq \alpha \|x^1 - x^2\|^2, \quad \text{for all } x^1, x^2 \in K \tag{1.3}$$

where α is a positive constant, and $\|\cdot\|$ denotes the Euclidean norm, guarantees both existence and uniqueness of the solution x to VI(K, f). In terms of applications, this condition implies, for example, that f_i depends primarily on x_i and less so on the $x_j's$, for $j \neq i$. A necessary and sufficient condition for expression (1.3) to hold is that the Jacobian of f, $[\frac{\partial f}{\partial x}]$, is positive definite over $x' \in K$.

On the other hand, if $f(\cdot)$ is only strictly monotone, that is, the right-hand side of equation (1.3) is now 0 for all $x^1, x^2 \in K, x^1 \neq x^2$, and $>$ replaces $\geq$, then the solution is unique, if one exists. Under such monotonicity assumptions, the theory of variational inequalities becomes particularly powerful.

It should be noted that even in applications where equation (1.3) fails to hold - in which case there may exist multiple equilibria - or, similarly, the condition cannot be verified a priori, such VI algorithms can, nevertheless, be applied for practical purposes. If convergence in such cases is observed, then the solution obtained will be guaranteed to be a solution of the variational inequality problem (1.2). Indeed, such computational experience has been recently reported by Mahmassani and Mouskos [46], who successfully applied the diagonalization method for large-scale traffic network equilibrium problems in which sufficient conditions for convergence were known to be violated. Nevertheless, in order to prove the global convergence of variational inequality algorithms, thus far, equation (1.3) has often been utilized.

The following relationship holds between variational inequality problems and often-studied minimization problems: In the special case where the *symmetry* condition $\frac{\partial f_i}{\partial x_j} = \frac{\partial f_j}{\partial x_i}$ holds, VI(K, f) is equivalent to the solution of the convex minimization problem with objective function $\int f(x)dx$ over the feasible set K. In other words, let $F(\cdot)$ be a continuously differentiable scalar-valued function defined over K and denote its gradient by $\nabla F(\cdot)$. If there exists an $x \in K$ such that

$$F(x) = \text{Min } F(x'), \quad \text{for all } x' \in K \tag{1.4}$$

then x is a solution to the variational inequality

$$\nabla F(x) \cdot (x' - x) \geq 0, \quad \text{for all } x' \in K. \tag{1.5}$$

On the other hand, if $f(\cdot)$, again on an open neighborhood of K, is the gradient of a convex continuously differentiable function $F(\cdot)$, then VI (1.2) and the minimization problem (1.4) are equivalent; in other words, x solves (1.2) when x minimizes $F(\cdot)$ over K. Note that $f(\cdot)$ is a gradient mapping if and only if its Jacobian matrix $\frac{\partial f}{\partial x}$ is symmetric, in which case the objective function is equal to $\int f(y)dy$.

Moreover, if $F(\cdot)$ is convex, strictly convex, or uniformly convex, then its gradient mapping is, respectively, monotone, strictly monotone, or strongly monotone. Therefore, we can see that monotonicity in variational inequality problems plays a role analogous to that of convexity in minimization problems. However, there does not exist an equivalent optimization problem under an *asymmetric* assumption for the underlying function, $f(\cdot)$. Note that variational inequality (1.2) can also contain, as special cases, all the classical problems of mathematical programming, such as linear and nonlinear complementarity problems, fixed point problems, and minimax problems (see Cottle [12]; Kinderlehrer and Stampacchia [41]; and Lemke [44]). Therefore, variational inequality theory can be said to be a powerful unifying framework.

The following is a brief review on how to solve variational inequalities. Variational inequality problems are usually solved iteratively as mathematical pro-

gramming problems. When an initial solution is given, a new solution is found at the end of the first iteration. The new solution is then used to find the next solution until the algorithm converges. Each iteration can be described using the following notation. Let

$$x(t+1) = T(x(t)), \quad t = 0, 1, \ldots, \tag{1.6}$$

where each $x(t)$ is an n-dimensional vector, t is the iteration number, and T is some mapping function from a set K of R^n into itself. Alternative notation that is sometimes used in place of (1.6) is $x := T(x)$. Notice that if the mapping function T is continuous and the sequence $\{x(t)\}$ generated by the above iteration follows a contraction

$$\|x_{t+1} - x_t\| \leq \alpha \|x_t - x_{t-1}\|, \quad 0 \leq \alpha < 1, \tag{1.7}$$

then the iteration converges to a limit x, which is a fixed point of T, that is, it satisfies $x = T(x)$. Such a mapping is called a contraction mapping, or simply a contraction, and the iteration (1.6) is called a contracting iteration and the scalar α is called the modulus of the mapping.

If the strong monotonicity condition (1.3) holds, then the solution of equation (1.2) can be computed via a general iterative scheme devised by Dafermos [17]. This scheme contains, as special cases, the projection method (Dafermos [15, 16]; Bertsekas and Gafni [8]), which resolves the solution of the variational inequality into a series of, in general, quadratic programming problems, and the relaxation/diagonalization method (Florian and Spiess [30]), which resolves the

problem into a series of, in general, nonlinear programming problems. For computational experience with these methods, see Nagurney [50] [51] [53]. The overall scheme of parallel computation to solve variational inequalities will be introduced in Section 1.3 and the detailed parallel algorithms will be explained in Chapters 2 and 3.

1.2 Large-Scale Network Problems

This thesis focuses on large-scale network problems to which both variational inequality theory and network theory are applied with a goal of creating efficient parallel algorithms. In particular, both network equilibrium problems and network optimization problems are considered.

Spatial price equilibrium problems (Samuelson [77, 78], Takayama and Judge [81, 82]) were considered in this thesis as prototypes of network equilibrium problems. Spatial price equilibrium models have been widely applied in agricultural commodity and energy markets (see,e.g., Judge and Takayama [39]). In spatial price equilibrium problems, one seeks to compute the commodity production, consumption, and trade patterns, satisfying the equilibrium property that trade takes place between two spatially separated markets if the demand price at the demand market is equal to the supply price of the commodity at the supply market plus the cost of transportation between the pair of supply and demand markets. In real-world applications, the numbers of supply and demand markets can be very large, and the resulting network will be in large scale.

When multiple commodities are considered, the computation of the equilibrium solution becomes even more difficult due to the even larger scale. One approach to the solution of such problems is to reformulate the problems (either linear or nonlinear) as single commodity, asymmetric VI problems via the construction of appropriate replications (for example, see Dafermos [14] [18]; Aashtiani and Magnanti [1] [2]; Nagurney [50] [53]; Nagurney and Aronson [61]). Other approaches to the solution of linear problems have included Bender's decomposition (Polito, McCarl, and Morin [74]), complementarity theory (Pang [71]; Takayama and Uri [83]), and quadratic programming algorithms for simpler models (Takayama and Judge [82]). Published computational experience with nonlinear multicommodity problems, nonetheless, has been limited to the solution of a small cadre of problems with only several markets and commodities (for example, see Frietz et al. [33]; Pang [72]). In the case of nonlinear multimodal problems in transportation, only several modes have been treated (Nagurney [50] [51]; Mahmassani and Mouskos [46]).

Other examples of large-scale equilibrium problems are dynamic market equilibrium problems, where the optimal commodity production, consumption, trade, and inventory patterns over space and time are to be computed. Such models are inherently large-scale and, hence, the development of efficient computational procedures is essential for the operationalism of such models. Nagurney and Aronson [62] developed new models and computational procedures for dynamic spatial price equilibrium problems with gains and losses, that were implemented on serial computers.

In regards to optimization problems, in this thesis, the constrained matrix problem is used as a prototype. Note that network optimization problems are special cases of network equilibrium problems. The constrained matrix problem is a core problem in numerous applications in the social and economic sciences, as well as engineering. These include: the estimation of input-output tables, trade tables, and social/national accounts in economics and regional science, the projection of migration flows over space and time in demography and geography, the treatment of census data, the analysis of political voting patterns, the estimation of contingency tables in statistics, the projection of transportation and telecommunication origin/destination flows , and diagnostic imaging in radiology. The constrained matrix problem, so named by Bacharah [4], 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 known matrix. The constrained matrix problem can be formulated as mathematical programming problems, with an objective function which forces "conservatism" on the process of rationalizing X from the initial estimate X^0. Although the RAS method, which dates to Deming and Stephan [24], 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 applications, such as interregional trade (see,e.g., Mohr, Crown, and Polenske [49]). In real-world applications, the matrix X^0 is 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 square of the number of rows/columns). The relationship of the constrained matrix problems with spatial price equilibrium problems is identified in Section 3.2.

1.3 Parallel Variational Inequality Decomposition

This section is a brief overview of how variational inequalities can be applied to create parallel decomposition algorithms.

The large-scale nature of constrained matrix problems and multicommodity and dynamic equilibrium problems makes the application of decomposition schemes especially appealing - even more so if the resulting subproblems have special structure that can be further exploited computationally. In the case where the feasible set of a network equilibrium problem can be expressed as a Cartesian product of subsets, i.e.,

$$K = \prod_{i=1}^{n} K_i \tag{1.8}$$

where each K_i is a subset of R^{n_i}, which is natural for multicommodity/multimodal equilibrium problems, variational decomposition algorithms can then be applied under suitable conditions for convergence to compute equilibrium problems in parallel. Many network equilibrium problems in management science can be defined over a feasible set as a Cartesian product of subsets. For example, in multicommodity trade problems, each subset could correspond to the constraints

of a particular commodity; in dynamic spatial price equilibrium problems, each subset could correspond to the constraints of each time period and inventories; in multimodal traffic networks, each subset could correspond to the constraints of each transportation mode; in constrained matrix problems, each subset could correspond to each row and/or each column. Hence, if one can derive alternative variational inequality reformulations of a given problem over a Cartesian product of subsets, one can then avail oneself with a plethora of alternative decomposition algorithms. These, however, must be evaluated both theoretically and empirically.

A key observation for the Cartesian product case is that variational inequality (1.2) decomposes into n coupled variational inequalities of smaller dimensions. Here we have a basic theorem for the decomposition of variational inequalities. For the proof, see page 275 in Bertsekas and Tsitsiklis [9].

Theorem 1:

A vector $x \in K$ solves the variational inequality $VI(K, f)$, where K is defined by (1.8) and f is the vector in R^n with subvectors ($f_i \in R^{n_i}$), if and only if

$$f_i(x) \cdot (x_i' - x_i) \geq 0, \quad \text{for all } x_i' \in K_i, i = 1, \ldots, n. \tag{1.9}$$

Hence, in the case where the set K is expressed as a Cartesian product of subsets as (1.8), the iteration procedure to solve VI(K, f), which was briefly explained in Section 1.1, can be decomposed in the same manner. For example, let $x_i(t)$ denote the ith component of $x(t)$ and let T_i denote the ith component of the

mapping function T, to be discussed later. Then, we can write $x(t+1) = T(x(t))$ as

$$x_i(t+1) = T_i(x_1(t), \ldots, x_n(t)), \quad i = 1, \ldots, n. \tag{1.10}$$

The iterative algorithm $x := T(x)$ can be parallelized by letting each one of n processors update a different component of x according to (1.10). At each stage, the ith processor knows the value of all components of $x(t)$ on which T_i depends, computes the new value $x_i(t+1)$, and communicates it directly to other processors or writes it on the shared memory in order to start the next iteration. This updating process of $x_i(t+1)$ depends upon the architecture of the parallel computers. After all $x_i(t+1), i = 1, \ldots, n$ are calculated and communicated, then the next iteration can start. Such a parallel algorithm is said to be synchronous, because the start of each iteration is simultaneous for all processors and the end of the message receptions is simultaneous for all messages. In order to implement a synchronous algorithm in an inherently asynchronous parallel architecture, we need a synchronization mechanism, i.e., an algorithm that is superimposed on the original and by which every processor can detect the end of each iteration.

To solve the variational inequality problem (1.2) using the parallel iterative algorithm (1.10), one needs to have n distinct processors. Sometimes, for example, when there are fewer processors available, one may wish to employ a coarse-grained parallelization of the iteration $x := T(x)$. In particular, one decomposes the vector space R^n as a Cartesian product of lower dimensional subspaces $R^{n_j}, j = 1, \ldots, p$, where $\sum_{j=1}^{p} n_j = n$. Accordingly, any vector $x \in R^n$ is

decomposed as $x = (x_1, \ldots, x_j, \ldots, x_p)$, where each x_j is n_j-dimensional vector, called a block-component of x. or simply a component where no confusion can arise. Similarly, the iteration $x(t+1) = T(x(t))$ can be written as

$$x_j(t+1) = T_j(x(t)), \quad j = 1, \ldots, p, \tag{1.11}$$

where each T_j is a vector function mapping R^n into R^{n_j}. Each one of the p processors is assigned to update a different block-component according to (1.11), and the resulting parallel algorithm is said to be block-parallelized.

The vector $T_i(x)$ of equation (1.10) can be selected to be either linear or nonlinear. The subproblems induced by a linear function, however, are often easier to solve (see Nagurney [54]), as will be the situation in the constrained matrix problems and market equilibrium problems that will be considered in this thesis, due to the special structure. In a parallel linearization algorithm, hence, we compute $x(t+1)$ by solving variational inequalities $VI(K_i, \tilde{f}_{i,t}(x))$ in parallel instead of $VI(K, f)$, where the linearized function $\tilde{f}_{i,t}$ has the form

$$\tilde{f}_{i,t}(x) = f_i(x(t)) + A_i(x(t))(x_i(t+1) - x_i(t)) \tag{1.12}$$

where $A_i(\cdot)$ is block-diagonal for each x. Equivalently, one constructs a vector $x(t+1) = (x_1(t+1), \ldots, x_n(t+1))$ satisfying

$$(f_i(x(t)) + A_i(x(t))(x_i(t+1) - x_i(t)))(y_i - x_i(t+1)) \geq 0,$$

$$\text{for all } y_i \in K_i,\ i = 1, \ldots, n. \tag{1.13}$$

This shows that each linearized variational inequality subproblem with smaller dimension can be solved independently by a distinct processor.

Since variational inequality problems are usually solved iteratively, the overall efficiency of a variational inequality algorithm depends on the embedded mathematical programming algorithm. In case where $A_i(x)$ is a symmetric positive definite matrix, the iteration method will be the well-known projection method. In the case where $A_i(x) = D_i(x)$, where $D_i(x)$ is the diagonal of the Jacobian of $f_i(\cdot)$ with respect to x_i, the resulting function becomes a disjoint separable function. In this thesis, $D_i(\cdot)$ is used to get the linearized function $\tilde{f}_{i,t}(\cdot)$ and it is applied to develop parallel algorithms for solving variational inequality problems.

When a parallel decomposition algorithm is developed, one of the most important issues will be its convergence. The convergence of linearized algorithms in the Cartesian product case is given in Bertsekas and Tsitsiklis [9], pages 276-277.

Theorem 2:

Suppose that the problem VI(K, f) has a solution x and that there exist symmetric positive definite matrices G_i and some $\delta > 0$ such that $D_i(x') - \delta G_i$ is nonnegative definite for every i and $x' \in K$, and that there exists some $\beta \in [0, 1)$ such that

$$\|G_i^{-1}(f_i(x') - f_i(y) - D_i(y)(x_i' - y_i))\|_i \leq \delta\beta \max_j \|x_j' - y_j\|_j, \quad .$$

$$\text{for all } x', y \in K, \tag{1.14}$$

where $\|x_i\|_i = (x^T{}_i G_i x_i)^{1/2}$. Let $\tilde{f}_{i,t}(x)$ be defined now as the linearized function as following:

$$\tilde{f}_{i,t}(x) = f_i(x(t)) + D_i(x(t))(x_i(t+1) - x_i(t)). \tag{1.15}$$

Define $x_i(t+1)$ as a solution of $VI(K_i, \tilde{f}_{i,t})$ and define the mapping function T_i by $x_i(t+1) = T_i(x_i(t))$, for every i. Then, the iteration mapping T_i of the linearized algorithm has the contracting property

$$\|T_i(x) - x_i\| \leq \beta \max_j \|x_j' - x_j\|_j, \quad \text{for all } x_i' \in K_i, i = 1, \ldots, n. \tag{1.16}$$

In particular, when x is the unique solution of VI(K, f), the parallel decomposition algorithm $x(t+1) = T(x(t))$ converges to x geometrically.

Although VI formulations of equilibrium problems over Cartesian products of sets have been derived (for example, see Dafermos [16, 18]; Pang [73]; Nagurney [53]; Nagurney and Aronson [61]) and serial decomposition algorithms implemented (Pang [72]; Nagurney [53, 54]; Nagurney and Aronson [61]), parallel VI decomposition algorithms have, heretofore, not been implemented on parallel architectures, nor their relative efficiencies vis á vis serial decomposition algorithms evaluated.

The recent development of parallel computing systems offers the promise of a quantum leap in the computing power that can be brought to bear on many important problems. When each suitably decomposed subproblem over a subset K_i is allocated to a distinct physical processor and is calculated independently, the time to solve large-scale network equilibrium problems can drop dramatically. Therefore, efficient parallel algorithms should be developed to solve large-scale network equilibrium problems. Coarse grain parallel algorithms, which have several multitasks, can be implemented on a supercomputer such as the IBM 3090-600 or the Cray Y-MP. Fine grain parallel algorithms, which have a very large

number of tasks, can be appropriately implemented on a massively parallel architecture such as the Connection Machine with as many as 64K processors in its full configuration.

Motivated by the recent development of parallel architectures, efficient parallel algorithms are developed in this thesis for large-scale network equilibrium and optimization problems using variational inequality theory and are implemented on the IBM 3090-600 and the Thinking Machine's CM-2 (Connection Machine) as mandated by the particular application.

This thesis is organized as follows. In Chapter 2, a new spatial price equilibrium model as a prototype of network equilibrium problems is developed. Its variational inequality formulation and accompanying parallel decomposition algorithms are also introduced. In Chapter 3, a constrained matrix problem is outlined as a prototype of (structured) network optimization problems. In Chapter 4, numerical implementations of the parallel algorithms which are introduced in Chapters 2 and 3 are explained. In Chapter 5, the numerical results are reported and discussed and, in Chapter 6, the thesis is summarized and directions for future research are given.

CHAPTER 2

NETWORK EQUILIBRIUM PROBLEMS

In this chapter, static spatial price equilibrium problems (SPEP) - single commodity and multicommodity - and dynamic SPEP's are considered as prototypes of large-scale network equilibrium problems. General dynamic multicommodity SPEP's are considered first, and then static SPEP's, both single and multicommodity, which are special cases of the general model.

2.1 Dynamic Multicommodity Spatial Price Equilibrium Problems

In this section, a new spatial price equilibrium model is developed. The model allows inventories at supply markets and contains, as special cases, static spatial price equilibrium models when the number of the finite time periods is equal to one. This model differs from the ones developed in Nagurney and Aronson [61] [62] in a significant way. In particular, the formulation is no longer based on the *path* flows, which is memory expensive, but, rather, on the *link* flows. Furthermore, this model allows the decomposition by *time periods*, which is a natural one based on the structure, as shall be demonstrated.

The model is now presented. Consider a finite time horizon and partition the horizon into discrete time periods $t; t = 1, \ldots, T$. It is assumed that L different commodities, typically denoted by k, are produced at m supply markets and are consumed at n demand markets. Denote a typical supply market by i and a typical demand market by j. Number the supply markets from 1 through m and the demand markets from $m + 1$ through $m + n$.

The state of the system will be described by a number of vectors as follows.

A supply column vector $s = \{s_{it}^k : i = 1, \ldots, m; k = 1, \ldots, L; t = 1, \ldots, T\}$ with nonnegative supply quantity s_{it}^k associated with supply market i and commodity k at time period t.

A demand column vector $d = \{d_{jt}^k : j = m + 1, \ldots, m + n; k = 1, \ldots, L; t = 1, \ldots, T\}$ with nonnegative demand d_{jt}^k associated with demand market j and commodity k at time period t.

A shipment column vector $X = \{X_{itjt}^k : i = 1, \ldots, m; j = m+1, \ldots, m+n; k = 1, \ldots, L; t = 1, \ldots, T\}$ with nonnegative commodity shipment X_{itjt}^k associated with commodity k between supply market i and demand market j in time period t.

An inventory column vector $I = \{I_{iti(t+1)}^k : i = 1, \ldots, m; k = 1, \ldots, L; t = 1, \ldots, T - 1\}$ with nonnegative total carryover quantity $I_{iti(t+1)}^k$ associated with supply market i and commodity k between time periods t and $t + 1$.

A supply price row vector $\pi = \{\pi_{it}^k : i = 1, \ldots, m; k = 1, \ldots, L; t = 1, \ldots, T\}$ with π_{it}^k denoting the supply price of commodity k at supply market i at time period t.

A demand price row vector $\rho = \{\rho_{jt}^k : j = m+1, \ldots, m+n; k = 1, \ldots, L; t = 1, \ldots, T\}$ with ρ_{jt}^k denoting the demand price of commodity k at demand market j at time period t.

A transaction cost (which includes the transportation cost) row vector $c = \{c_{itjt} : i = 1, \ldots, m; j = m+1, \ldots, m+n; k = 1, \ldots, L; t = 1, \ldots, T\}$ with c_{itjt}^k denoting the nonnegative transaction cost associated with shipping commodity k from supply market i to demand market j in time period t.

An inventorying cost row vector $H = \{H_{iti(t+1)}^k : i = 1, \ldots, m; k = 1, \ldots, L; t = 1, \ldots, T-1\}$ with $H_{iti(t+1)}^k$ denoting the nonnegative inventory cost associated with carrying over commodity k from time period t to $t+1$ at supply market i.

Now the dynamic network equilibrium model is constructed. In particular, L copies of a single commodity dynamic network for commodity k are considered as follows (see Figure 1). For each period $t; t = 1, \ldots, T$, m supply market nodes, denoted by the tuples $(1t)^k, \ldots, (mt)^k$, represent the supply markets of commodity k at time period t. Similarly, n demand market nodes, denoted by the tuples $(m+1, t)^k, \ldots, (m+n, t)^k$, represent the demand markets of commodity k at time period t. In the model, mn transaction/transportation links with a typical one originating at node $(it)^k$ and terminating at node $(jt)^k$ are denoted by $(itjt)^k$. Hence, the total number of transaction links is $LmnT$. From each supply

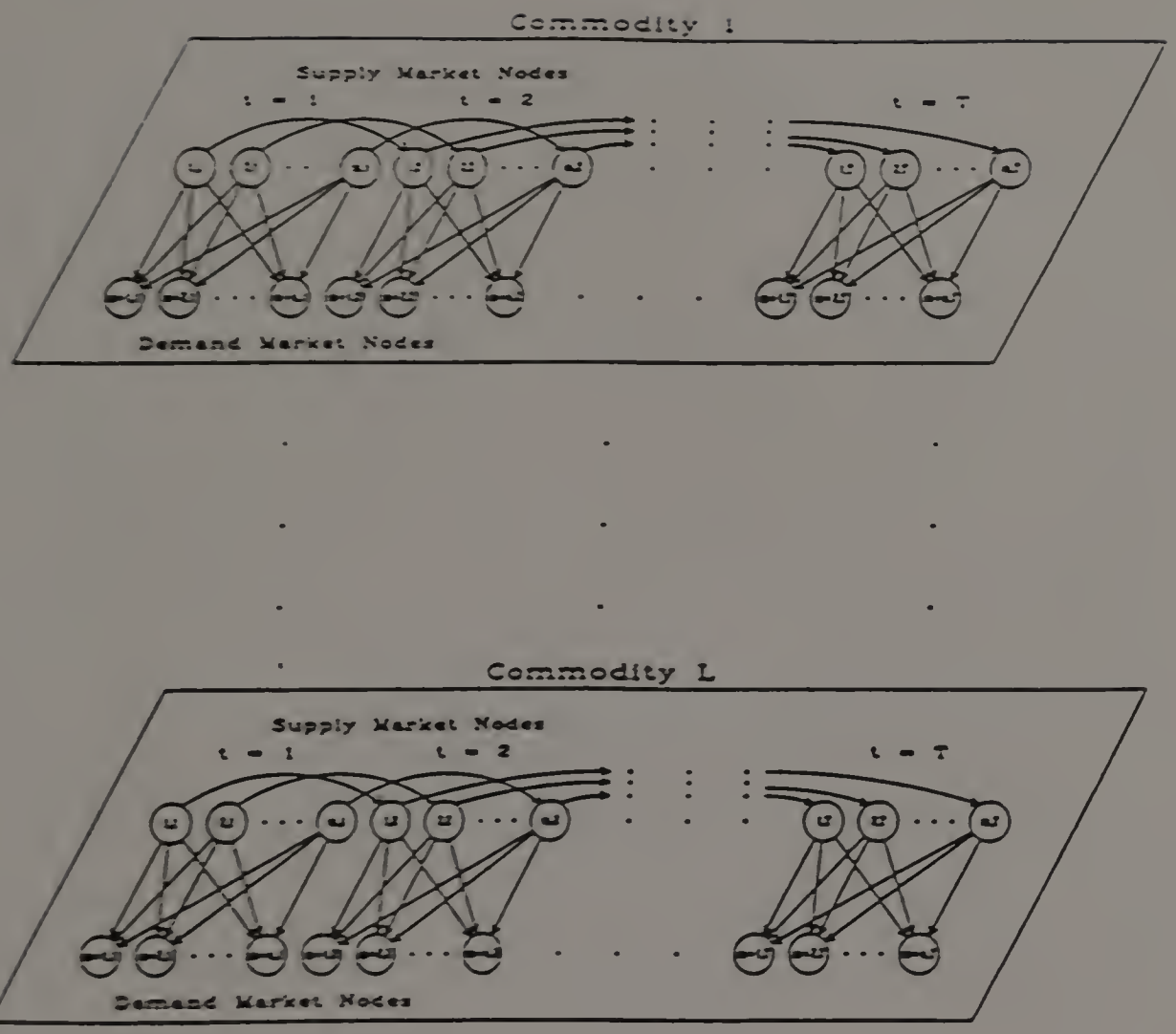


Figure 1 A Network Representation of the Dynamic Multicommodity Market Equilibrium Problem

supply market node $(it)^k$, an inventorying link $(it, i(t+1))^k$ is constructed and, then, the total number of inventory links is $Lm(T-1)$. With each of the links $((itjt')^k; t' = t)$ the corresponding transaction cost $c^k_{itjt'}$ is associated and with each of the links $((itit')^k; t' = t+1)$ the inventory cost $H^k_{itit'}$. The flows on these links correspond, respectively, to $X^k_{itjt'}$ and $I^k_{itit'}$.

The supply and demand of each commodity must satisfy the following flow conservation constraints:

$$s^k_{it} = \sum_{j=1}^{n} X^k_{itjt} + I^k_{iti(t+1)} - I^k_{i(t-1)it}, \quad \text{for all } i, k, t \tag{2.1}$$

and

$$d^k_{jt} = \sum_{i=1}^{m} X^k_{itjt}, \quad \text{for all } j, k, t \tag{2.2}$$

where

$$X^k_{itjt} \geq 0 \text{ and } I^k_{itit'} \geq 0, \quad \text{for all } i, j, k, t. \tag{2.3}$$

Denote the set of all feasible (s, X, I, d) satisfying constraints (2.1), (2.2), and (2.3) by K.

Assuming that the underlying mechanism is that of perfect competition, a dynamic spatial market equilibrium consisting of commodity prices, shipments, and quantities inventoried, is obtained if the following interregional/intertemporal conditions are satisfied: a commodity will be produced, traded, and consumed,

between a pair of markets if the supply price at the supply market plus the transaction cost is equal to the demand price at the demand market. No commodity will be produced, traded, and consumed, if the supply price plus the transaction cost is greater than the demand price. Similarly, the commodity will be inventoried between two time periods if the supply price at the supply market plus the inventory cost is equal to the supply price at the next time period. The inventorying quantity will be zero if the supply price plus the inventory cost is greater than the supply price at the next time period.

Mathematically, the dynamic multicommodity market equilibrium conditions take the form, following Samuelson [78] and Takayama and Judge [82] (see also Nagurney and Aronson [61]): for all $i = 1, \ldots, m; j = m+1, \ldots, m+n; k = 1, \ldots, L; t = 1, \ldots, T$:

$$\pi^k_{it}(s) + c^k_{itjt}(X) \begin{cases} = & \rho^k_{jt}(d), \quad \text{if } X^k_{itjt} > 0 \\ \geq & \rho^k_{jt}(d), \quad \text{if } X^k_{itjt} = 0 \end{cases} \tag{2.4}$$

and for all $i = 1, \ldots, m; k = 1, \ldots, L; t = 1, \ldots, T-1$:

$$\pi^k_{it}(s) + H^k_{iti(t+1)}(I) \begin{cases} = & \pi^k_{i(t+1)}(s), \quad \text{if } I^k_{iti(t+1)} > 0 \\ \geq & \pi^k_{i(t+1)}(s), \quad \text{if } I^k_{iti(t+1)} = 0. \end{cases} \tag{2.5}$$

Here the general situation is considered where the supply price function $\pi^k_{it}(s)$ associated with a supply market i, commodity k, and time period t may, in general, depend upon the supply quantity of every commodity at every supply market in every time period. Similarly, the demand price function ρ^k_{jt} associated with demand market j, commodity k, and time period t may depend upon, in

general, the demand quantity of every commodity at every demand market in every time period. The transaction and inventory costs c^k_{itjt}, $H^k_{iti(t+1)}$, in turn, may depend, respectively, upon the shipments of every commodity between every pair of markets within every time period, and upon the quantities inventoried of every commodity at every supply market between every pair of time periods.

The variational inequality formulation of the above equilibrium conditions (2.4) and (2.5) is presented as follows.

Theorem 3:

A commodity pattern (s, X, I, d) is in equilibrium, if and only if, it satisfies the variational inequality problem:

$$\pi(s) \cdot (s' - s) + c(X) \cdot (X' - X) + H(I) \cdot (I' - I) - \rho(d) \cdot (d' - d) \geq 0 \quad (2.6)$$

$$\text{for all } (s', X', I', d') \in K.$$

Proof:

First variational inequality (2.6) is derived from equilibrium conditions (2.4) and (2.5).

Observe that condition (2.4) implies that for fixed market pair (i, j), commodity k, and time period t:

$$(\pi^k_{it}(s) + c^k_{itjt}(X) - \rho^k_{jt}(d)) \cdot (X^{k'}_{itjt} - X^k_{itjt}) \geq 0, \quad \text{for all } X^{k'}_{itjt} \geq 0. \quad (2.7)$$

But inequality (2.7) holds for all pairs (i, j), k, and t; hence,

$$\sum_{t=1}^{T} \sum_{k=1}^{L} \sum_{j=m+1}^{m+n} \sum_{i=1}^{m} (\pi^k_{it} + c^k_{itjt}(X) - \rho^k_{jt}(d)) \cdot (X^{k'}_{itjt} - X^k_{itjt}) \geq 0. \quad (2.8)$$

Also, observe that condition (2.5) implies that

$$(\pi^k_{it}(s) + H^k_{itj(t-1)}(I) - \pi^k_{i(t-1)}(s)) \cdot (I^{k'}_{itj(t+1)} - I^k_{iti(t+1)}) \geq 0, \text{ for all } I^{k'}_{iti(t+1)} \geq 0, \quad (2.9)$$

and, therefore,

$$\sum_{t=1}^{T-1} \sum_{k=1}^{L} \sum_{i=1}^{m} (\pi^k_{it}(s) + H^k_{itit+1}(I) - \pi^k_{it+1}(s)) \cdot (I^{k'}_{itit+1} - I^k_{itit+1}) \geq 0. \quad (2.10)$$

Combining now inequalities (2.8) and (2.10) and simplifying the resulting expression by using (2.1) and (2.2), we obtain:

$$\sum_{t=1}^{T} \sum_{k=1}^{L} \sum_{i=1}^{m} \pi^k_{it}(s) \cdot (s^{k'}_{it} - s^k_{it}) + \sum_{t=1}^{T} \sum_{k=1}^{L} \sum_{j=m+1}^{m+n} \sum_{i=1}^{m} c_{itjt}(X) \cdot (X^{k'}_{itjt} - X^k_{itjt})$$

$$\sum_{t=1}^{T-1} \sum_{k=1}^{L} \sum_{i=1}^{m} H^k_{iti(t+1)}(I) \cdot (I^{k'}_{iti(t+1)} - I^k_{iti(t+1)}) - \sum_{t=1}^{T} \sum_{k=1}^{L} \sum_{j=m+1}^{m+n} \rho^k_{jt}(d) \cdot (d^{k'}_{jt} - d^k_{jt}) \geq 0, \quad (2.11)$$

$$\text{for all } (s', X', I', d') \in K,$$

or, equivalently, (2.6).

Now equilibrium conditions (2.4) and (2.5) are derived from variational inequality (2.6). For convenience, the expanded form of the variational inequality (2.11) is used. At first, fix $I^{k'}_{itit+1} = I^k_{itit+1}$, for all i and t. Then (2.11) reduces to:

$$\sum_{t=1}^{T} \sum_{k=1}^{L} \sum_{j=m+1}^{m+n} \sum_{i=1}^{m} (\pi^k_{it}(s) + c^k_{itjt}(X) - \rho^k_{jt}(d)) \cdot (X^{k'}_{itjt} - X^k_{itjt}) \geq 0. \quad (2.12)$$

Note, by setting $X^{k'}_{it't''t''} = X^{k}_{it't''t''}$ for all $it't''t'' \neq itjt$, inequality (2.12) reduces to

$$(\pi^k_{it}(s) + c^k_{itjt}(X) - \rho^k_{jt}(d)) \cdot (X^{k'}_{itjt} - X^k_{itjt}) \geq 0, \tag{2.13}$$

and, hence, equilibrium condition (2.4) must hold.

Equilibrium condition (2.5) can be shown to hold using similar arguments, but by, first, setting $X^{k'}_{itjt} = X^k_{itjt}$ for all i, j, k, and t.

Existence of a unique equilibrium pattern (s, X, I, d) can be guaranteed from the theory of variational inequalities under the assumption of strong monotonicity, that is,

$$\begin{aligned}
&(\pi(s^1) - \pi(s^2)) \cdot (s^1 - s^2) + (c(X^1) - c(X^2)) \cdot (X^1 - X^2) \\
&+(H(I^1) - H(I^2)) \cdot (I^1 - I^2) - (\rho(d^1) - \rho(d^2)) \cdot (d^1 - d^2) \\
&\geq \alpha \left(\|(s^1 - s^2\|^2 + \|X^1 - X^2\|^2 + \|I^1 - I^2\|^2 + \|d^1 - d^2\|^2 \right) \qquad (2.14) \\
&\text{for all } (s^1, X^1, I^1, d^1) \text{ and } (s^2, X^2, I^2, d^2) \in K,
\end{aligned}$$

where α is a positive constant. Condition (2.14) will hold when the respective Jacobian matrices $\frac{\partial \pi}{\partial s}$, $\frac{\partial c}{\partial X}$, $\frac{\partial H}{\partial I}$, and $-\frac{\partial \rho}{\partial d}$ are positive definite over the feasible set K. This condition is commonly imposed and means that we can expect that the supply price of a commodity at a supply market and time period will depend primarily upon the supply of the commodity at that supply market in that time period. Similarly, we can expect the demand price of a commodity at a demand market and time period to depend primarily upon the demand of the commodity at that demand market in that time period. The analogous dependencies between

the transaction and inventorying cost functions and the respective commodity shipments and inventory quantities can also be expected to hold.

In the special case when the Jacobian matrices are symmetric, as assumed in the classical models of Samuelson and Takayama and Judge, then it is easy to see that (s, X, I, d) satisfies (2.6), if and only if, it minimizes the functional

$$O(s, X, I, d) = \int \pi(s)ds + \int c(X)dX - \int H(I)dI - \int \rho(d)dd \qquad (2.15)$$

over K. In the symmetric case, then, the equilibrium can be constructed by standard convex programming algorithms.

Dynamic single commodity SPEP's can be easily obtained from equilibrium conditions (2.4) and (2.5) by letting the number of commodities, L, be equal to one. The graphical description (see Figure 2) of this problem is simply obtained by taking one layer from Figure 1. The equivalent variational inequality formulation of the above equilibrium conditions (2.4) and (2.5) and the qualitative properties of equilibria such as existence and uniqueness are then easily obtained. It is emphasized, that although the model is referred to as a single commodity one, it is, nevertheless, quite general, in that the supply price of the commodity at a supply market at a time period may depend upon the supply of the commodity at every supply market and time period. A similar generality holds for the demand price functions and the transaction and inventory cost functions.

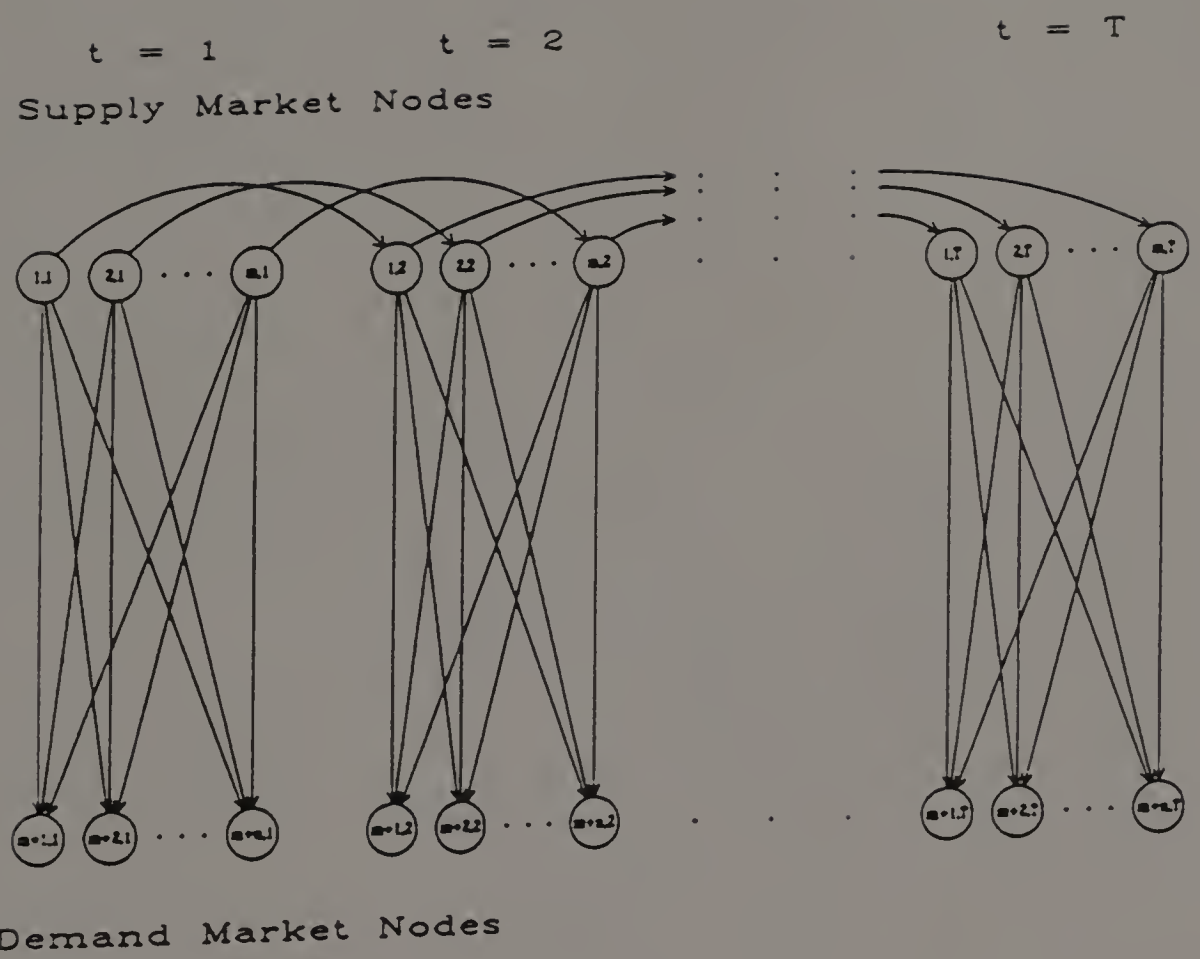


Figure 2 A Network Representation of the Dynamic Single Commodity Market Equilibrium Problem

2.2 Static Multicommodity Spatial Price Equilibrium Problems

In static SPEP's, the products produced by spatially separated supply markets at different prices are transshipped at different transportation costs to the spatially separated demand markets which have different demand prices in one time period. This model is readily obtained from the dynamic multicommodity model described in the preceding section by letting the number of discrete time periods, T, be equal to one. The resulting static multicommodity spatial price equilibrium model is then identical to the multicommodity equilibrium model introduced by Dafermos [18]. Here the model is briefly outlined.

Assume that L different commodities, typically denoted by k, are produced at m supply markets and are consumed at n demand markets. The typical supply and demand markets will be denoted, respectively, by i and j. The state of the system will be described by a number of vectors as follows:

A *supply column vector* $s = \{s_k : k = 1, \ldots, L\} \in R^{Lm}$ where $s_k = \{s_i^k; i = 1, \ldots, m\}$ is a vector $\in R^m$ with nonnegative supply quantity s_i^k associated with commodity k and market i.

A *demand column vector* $d = \{d_k : k = 1, \ldots, L\} \in R^{Ln}$ where $d_k = \{d_j^k : j = 1, \ldots, n\}$ is a vector $\in R^n$ with nonnegative demand quantity d_j^k associated with commodity k and market j.

A *shipment column vector* $X = \{X_k : k = 1, \ldots, L\} \in R^{Lmn}$, where $X_k = \{X_{ij}^k : i = 1, \ldots, m; j = 1, \ldots, n\}$ is a vector $\in R^{mn}$ with nonnegative shipment

X_{ij}^k associated with commodity k between supply market i and demand market j.

A *supply price row vector* $\pi = \{\pi_k : k = 1, \ldots, L\} \in R^{Lm}$ where $\pi_k = \{\pi_i^k : i = 1, \ldots, m\}$ is a vector $\in R^m$ with component π_i^k denoting the supply price of commodity k at supply market i.

A *demand price row vector* $\rho = \{\rho_k : k = 1, \ldots, L\} \in R^{Ln}$ where $\rho_k = \{\rho_j^k : j = 1, \ldots, n\}$ is a vector $\in R^{Ln}$ with component ρ_j^k denoting the demand price of commodity k at demand market j.

A *transaction cost row vector* $c = \{c_k : k = 1, \ldots, L\} \in R^{Lmn}$ where $c_k = \{c_{ij}^k, i = 1, \ldots, m; j = 1, \ldots, n\}$ is a vector $\in R^{mn}$ with component c_{ij}^k denoting the unit transaction cost associated with producing commodity k at supply market i and consuming it at demand market j.

The above fields are related by the following conservation equations

$$s_i^k = \sum_j X_{ij}^k \quad i = 1, \ldots, m; \quad k = 1, \ldots, L \tag{2.16}$$

$$d_j^k = \sum_i X_{ij}^k \quad j = 1, \ldots, n; \quad k = 1, \ldots, L, \tag{2.17}$$

where $X_{ij}^k \geq 0$, for all i, j, k. Define $K_k \equiv (s_k, X_k, d_k)$, such that (2.16) and (2.17) hold for fixed k, $X_k \geq 0$, and $K = \prod_{k=1}^L K_k$.

Again the general situation is considered where the supply price of any commodity at any supply market may, in general, depend upon the supplies of every commodity associated with every supply market. The demand price functions and transaction cost functions are also as general as described earlier.

Assuming, as before, that the underlying mechanism is that of perfect competition, the equilibrium is obtained when the supply price of the commodity at the supply market plus the transaction cost of this commodity associated with this pair of supply and demand markets is equal to the demand price of this commodity at the demand market. Mathematically, this state is characterized by the following equilibrium conditions which must hold for every commodity k and for all pairs of supply and demand markets (i, j):

$$\pi_i^k(s) + c_{ij}^k(X) \begin{cases} = & \rho_j^k(d), \quad \text{if } X_{ij}^k > 0 \\ \geq & \rho_j^k(d), \quad \text{if } X_{ij}^k = 0. \end{cases} \tag{2.18}$$

Equilibrium conditions (2.18) can be obtained from equations (2.4) and (2.5) by letting T be equal to one. The equivalent variational inequality formulation of equilibrium condition (2.18), following Dafermos [20], is given by

$$\pi(s) \cdot (s' - s) + c(X) \cdot (X' - X) - \rho(d) \cdot (d' - d) \geq 0 \tag{2.19}$$

$$\text{for all } (s', X', d') \in K.$$

Static single commodity equilibrium conditions are easily obtained from the conditions (2.18) by letting the number of commodities, L, be equal to one. This simple SPEP is depicted as a bipartite network in Figure 3. The equivalent variational inequality formulation of the above equilibrium conditions and the qualitative properties of equilibria such as existence and uniqueness are then easily obtained.

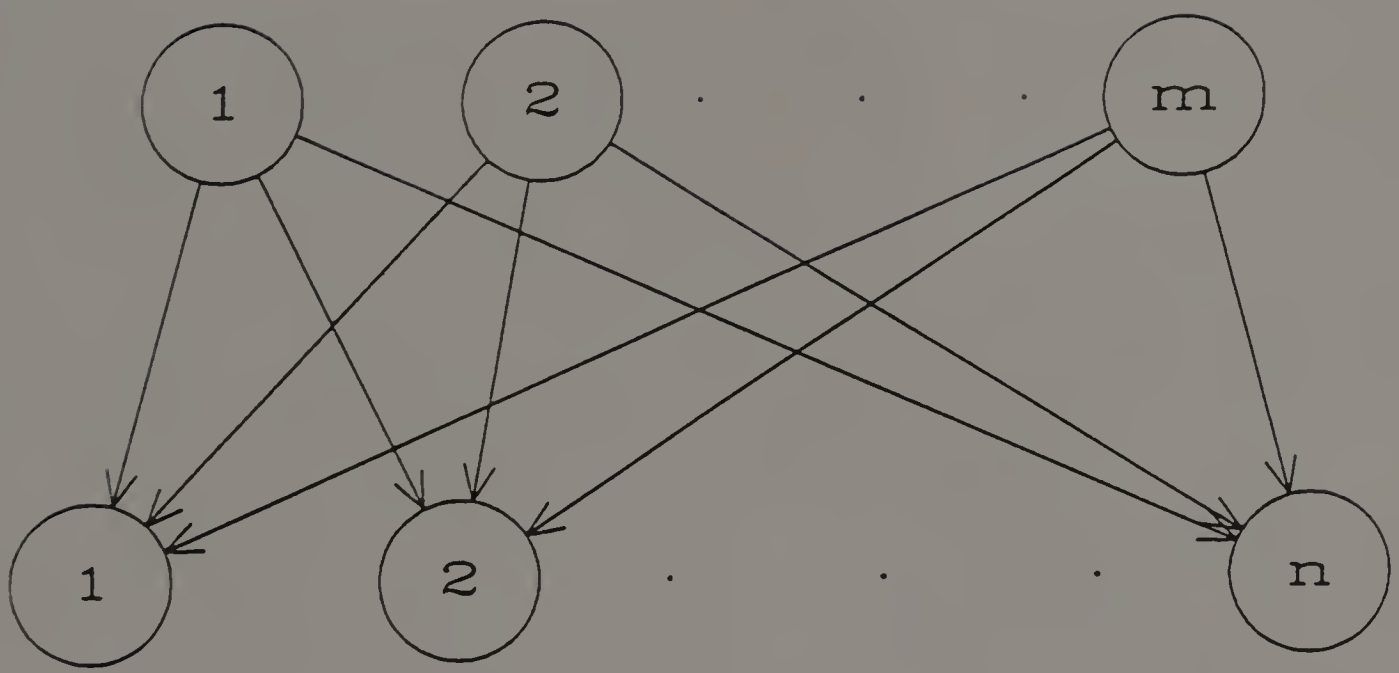


Figure 3 Bipartite Network Representation of Spatial Price Equilibrium Problems

2.3 The Decomposition Procedure

As mentioned in the Introduction in this thesis, parallel computation of the equilibrium solutions will focus on the construction of parallel decomposition schemes for which convergence conditions will also be given. As shall be shown in this section, dynamic multicommodity spatial price equilibrium problems can be decomposed by commodities, by time periods, and/or by demand and supply markets. Figure 4 shows the overview of the decomposition scheme and the detailed decomposition procedures are explained in the following subsections.

2.3.1 Decomposition by Commodities

Multicommodity spatial price equilibrium problem can be decomposed by commodities. In the case of the multicommodity market equilibrium model outlined in the Section 2.1, the feasible set K can be expressed as a Cartesian products of subsets, i.e., $K = \prod_{k=1}^{L} K_k$, where k is a commodity, and the selection of the mapping function $A_k(x) = D_k(x)$, where $D_k(x)$ is the diagonal part of $\nabla_k f_k(x)$, yields a decomposition of the multicommodity problem into L single commodity problems.

Now the parallel decomposition algorithm by commodities to solve the variational inequality problem (2.6) is presented.

Step 0: Initialization Step

Start with an initial feasible solution $(s^0, X^0, I^0, d^0) \in K$ and set the iteration count $\tau = 1$.

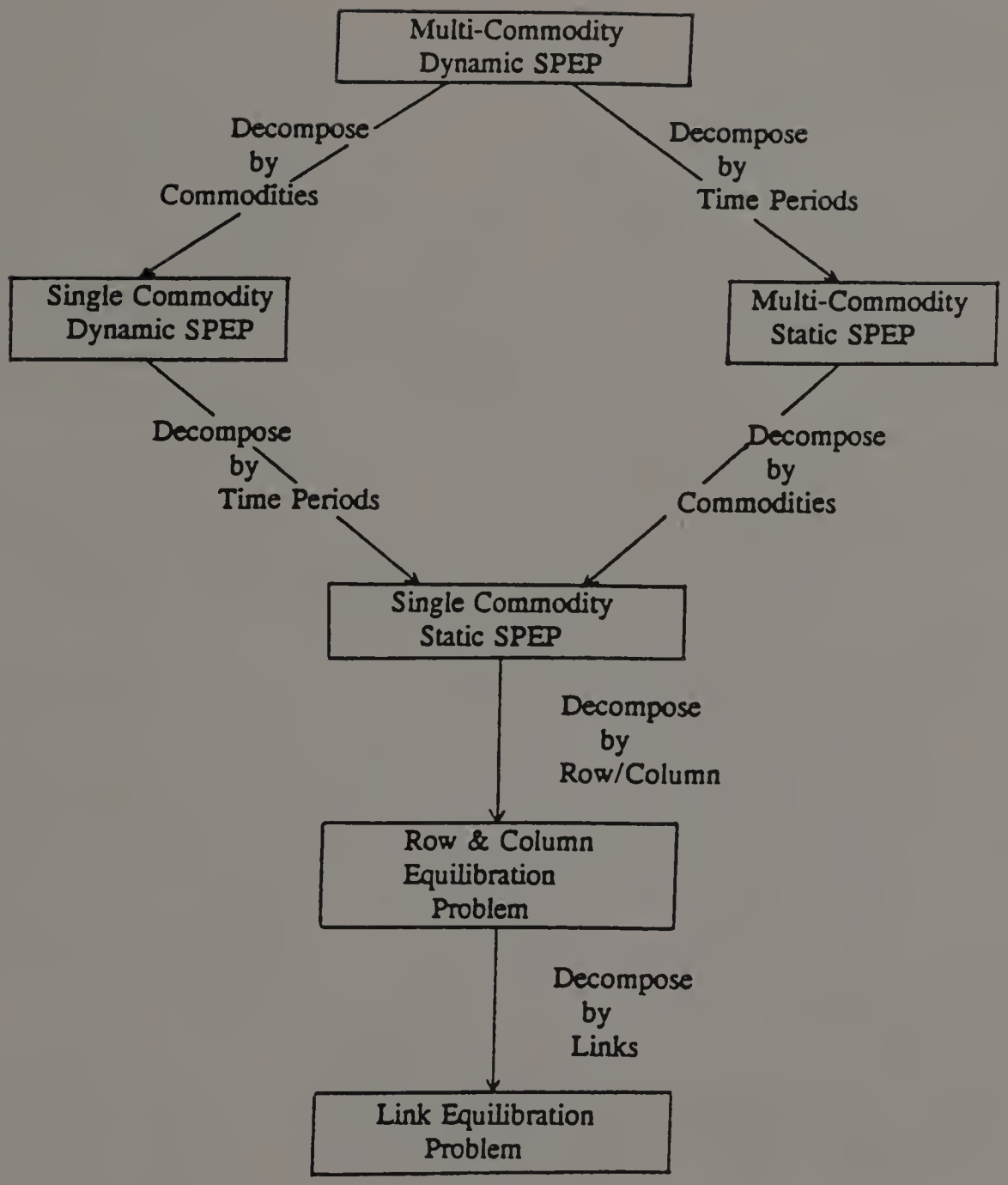


Figure 4 Overview of the Decomposition Schemes of General Spatial Price Equilibrium Problems

Step 1: Linearization Step

At iteration τ, construct new supply price, demand price, transaction cost, and inventorying cost functions which are linear and given for each commodity k by

$$\pi_{it}^{k\tau}(s_{it}^k) = \frac{\partial \pi_{it}^k}{\partial s_{it}^k}(s^\tau) \cdot s_{it}^k + \left\{ \pi_{it}^k(s^\tau) - \frac{\partial \pi_{it}^k}{\partial s_{it}^k}(s^\tau) \cdot s_{it}^{k\tau} \right\} \tag{2.20}$$

for supply markets $i = 1, \ldots, m$, commodities $k = 1, \ldots, L$, and time periods $t = 1, \ldots, T$,

$$\rho_{jt}^{k\tau}(d_{jt}^k) = \frac{\partial \rho_{jt}^k}{\partial d_{jt}^k}(d^\tau) \cdot d_{jt}^k + \left\{ \rho_{jt}^k(d^\tau) - \frac{\partial \rho_{jt}^k}{\partial d_{jt}^k}(d^\tau) \cdot d_{jt}^{k\tau} \right\} \tag{2.21}$$

for demand markets $j = 1, \ldots, n$, commodities $k = 1, \ldots, L$, and time periods $t = 1, \ldots, T$,

$$c_{ijt}^{k\tau}(X_{ijt}^k) = \frac{\partial c_{ijt}^k}{\partial X_{ijt}^k}(X^\tau) \cdot X_{ijt}^k + \left\{ c_{ijt}^k(X^\tau) - \frac{\partial c_{ijt}^k}{\partial X_{ijt}^k}(X^\tau) \cdot X_{ijt}^{k,\tau} \right\} \tag{2.22}$$

for supply and demand market pairs $(i, j) : i = 1, \ldots, m; j = 1, \ldots, n$, commodities $k = 1, \ldots, L$, and time periods $t = 1, \ldots, T$,

$$H_{iti(t+1)}^{k\tau}(I_{iti(t+1)}^k) = \frac{\partial H_{iti(t+1)}^k}{\partial I_{iti(t+1)}^k}(I^\tau) \cdot I_{iti(t+1)}^k$$

$$+ \left\{ H_{iti(t+1)}^k(I^\tau) - \frac{\partial H_{iti(t+1)}^k}{\partial I_{iti(t+1)}^k}(I^\tau) \cdot I_{iti(t+1)}^{k\tau} \right\} \tag{2.23}$$

for supply markets $i = 1, \ldots, m,$, commodities $k = 1, \ldots, L$, and time periods $t = 1, \ldots, T-1$.

Step 2: Equilibration Step

The VI subproblem at iteration τ, for commodity k, is, hence,

$$\sum_{t=1}^{T}\sum_{i=1}^{m} \pi_{it}^{k\tau}(s_{it}^{k}) \cdot (s_{it}^{k\tau} - s_{it}^{k}) + \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ijt}^{k\tau}(X_{ijt}^{k}) \cdot (X_{ijt}^{k\tau} - X_{ijt}^{k})$$

$$\sum_{t=1}^{T-1}\sum_{i=1}^{m} H_{iti(t+1)}^{k\tau}(I_{iti(t+1)}^{k}) \cdot (I_{iti(t+1)}^{k\tau} - I_{iti(t+1)}^{k}) - \sum_{t=1}^{T}\sum_{j=1}^{n} \rho_{jt}^{k\tau}(d_{jt}^{k}) \cdot (d_{jt}^{k\tau} - d_{jt}^{k}) \geq 0, \quad (2.24)$$

$$\text{for all } (s^{k\tau}, X^{k\tau}, I^{k\tau}, d^{k\tau}) \in K_k.$$

The solution to all L subproblems is $(s^{\tau+1}, X^{\tau+1}, I^{\tau+1}, d^{\tau+1})$. Note that L decomposed VI subproblems of (2.24) are solved independently and in parallel, where each subproblem (2.24) is equivalent to an optimization problem.

Step 3: Convergence Verification Step

If the equilibrium conditions (2.4) and (2.5) hold within a prescribed tolerance ϵ, then terminate. Otherwise, update the functions according to (2.20), (2.21), (2.22), and (2.23) for all the commodities and go to Step 1.

The convergence of the above decomposition algorithm by commodities is obtained by applying Theorem 2 in Chapter 1.

In the case of static models, this decomposition algorithm becomes simpler in that equation (2.23) is then removed. As a result, the resulting L single commodity static problems have mn shipment variables and a special bipartite network structure, which can be solved via the supply and demand market equilibration

algorithms introduced by Dafermos and Nagurney [22], (see also Nagurney [55, 56]) that have the remarkable property that, at each step, a relaxed market equilibrium problem is solved exactly. For a theoretical analysis of such equilibration algorithms, see Eydeland and Nagurney [28]. Specifically, at each iteration, the VI decomposition algorithm would yield the functions (2.20), (2.21), and (2.22). In this case, the ensuing subproblems are also equivalent to the solution of a quadratic programming problem of the form:

$$\underset{\text{over } K_k}{\text{Minimize}} \sum_i \int_0^{s_i^k} \pi_i^k(x)dx + \sum_{ij} \int_0^{X_{ij}^k} c_{ij}^k(y)dy - \sum_j \int_0^{d_j^k} \rho_j(z)dz, \qquad (2.25)$$

for each k, $k = 1, \ldots, L$.

2.3.2 Decomposition by Time Periods

The dynamic spatial price equilibrium problems can also be decomposed by time periods. For simplicity, a single commodity dynamic problem is considered rather than a multicommodity one. This novel approach - decomposition by time periods - resolves the dynamic network problem into T static network equilibrium problems, each with mn shipment variables and with a special bipartite network structure, for which numerous efficient algorithms exist. The $T+1$-st subproblem, the inventory problem, is a simple problem in $m(T-1)$ variables. Figure 5 describes graphically the dynamic market equilibrium problem when decomposed by time periods. It is emphasized that although our focus is on the parallel nature of this decomposition scheme, the algorithm can also be implemented in a serial

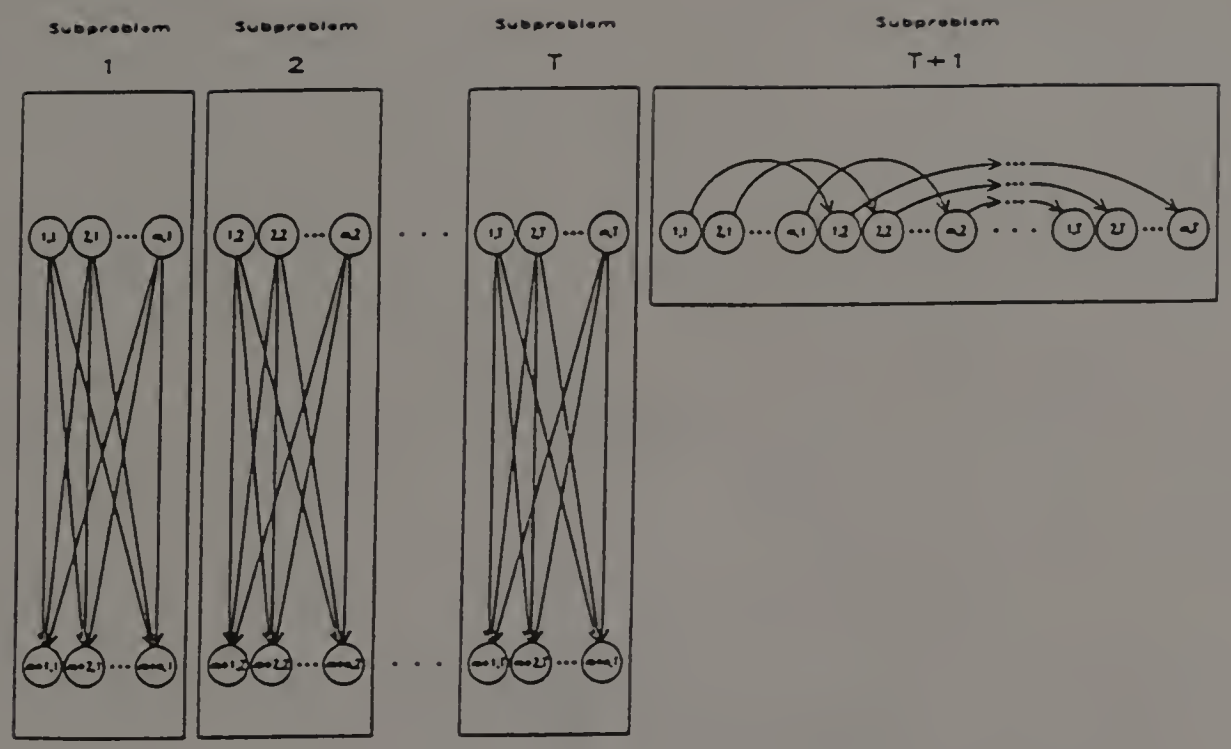


Figure 5 Parallel Decomposition of the Dynamic Market Equilibrium by Time Periods

environment using the appropriate adaptations/extensions to any existing code for static spatial market equilibrium problems. Indeed, the computational results in the next section are for precisely such an implementation.

First some preliminaries are presented. Note that in view of (2.1) and (2.2), we may define the functions $\hat{\pi}_{it}(X, I) \equiv \pi_{it}(s)$ for all i and t and $\hat{\rho_{jt}}(X) \equiv \rho_{jt}(d)$ for all j and t. Hence, the variational inequality (2.11) is equivalent to a variational inequality in only two vectors of variables X and I, that is,

$$\sum_{t=1}^{T} \sum_{j=m+1}^{m+n} \sum_{i=1}^{m} (\hat{\pi}_{it}(X, I) + c_{itjt}(X) - \hat{\rho}_{jt}(X)) \cdot (X'_{itjt} - X_{itjt})$$

$$+ \sum_{t=1}^{T-1} \sum_{i=1}^{m} (\hat{\pi}_{it}(X, I) + H_{iti(t+1)}(I) - \hat{\pi}_{i(t+1)}(X, I)) \cdot (I'_{iti(t+1)} - I_{iti(t+1)}) \geq 0 \quad (2.26)$$

$$\text{for all } (X', I') \in K^2,$$

where $K^2 \equiv K_1 \times K_2$, where $K_1 \equiv \{X' | X' \geq 0\}$ and $K_2 \equiv \{I' | I' \geq 0\}$.

The algorithm is a linearization scheme which resolves (2.26) into two simpler subproblems, each of which is a quadratic programming problem; the first subproblem is in variables X only, and the second, in variables I, only. Each of these subproblems can be solved simultaneously, and in parallel.

Let $f_1(X, I)$ denote the mnT dimensional vector with components $\{\hat{\pi}_{it}(X, I) + c_{itjt}(X) - \hat{\rho}_{jt}(X), i = 1, \ldots, m; j = m + 1, \ldots, m + n; t = 1, \ldots, T\}$ and $f_2(X, I)$ the $m(T - 1)$ dimensional vector with components $\{\hat{\pi}_{it}(X, I) + H_{iti(t+1)}(I) - \hat{\pi}_{i(t+1)}(X, I), i = 1, \ldots, m; t = 1, \ldots, T - 1\}$. Then variational inequality (2.26)

can be written succinctly as:

$$f_1(X,I)\cdot(X'-X)+f_2(X,I)\cdot(I'-I)\geq 0 \quad \text{for all } (X,I)\in K^2. \tag{2.27}$$

Now the algorithm is presented as follows.

Step 0: Initialization Step

Set $X^0=0, I^0=0, \tau=1$.

Step 1: Linearization Step

(1) Construct the function $f_1^\tau(X)\in R^{mnT}$ which is linear and separable according to:

$$f_1^\tau(X)=D_1(X^{\tau-1},I^{\tau-1})\cdot(X)+(f_1(X^{\tau-1},I^{\tau-1})-D_1(X^{\tau-1},I^{\tau-1})\cdot X^{\tau-1}) \tag{2.28}$$

where $D_1(\cdot)$ is the diagonal part of $\nabla_1 f_1(\cdot)$.

(2) Construct the function $f_2^\tau(I)$ which is linear and separable according to:

$$f_2^\tau(I)=D_2(X^{\tau-1},I^{\tau-1})\cdot(I)+(f_2(X^{\tau-1},I^{\tau-1})-D_2(X^{\tau-1},I^{\tau-1})\cdot I^{\tau-1}) \tag{2.29}$$

where $D_2(\cdot)$ is the diagonal part of $\nabla_2 f_2(\cdot)$.

Step 2: Equilibration Step

(1) Solve the variational inequality subproblem,

$$f_1^\tau(X)\cdot(X'-X)\geq 0, \quad \text{for all } X'\in K_1. \tag{2.30}$$

Let the solution to (2.30) be X^τ.

(2) Solve the variational inequality subproblem,

$$f_2^\tau(I) \cdot (I' - I) \geq 0, \quad \text{for all } I' \in K_2. \tag{2.31}$$

Let the solution to (2.31) be I^τ.

Step 3: Convergence Verification Step

If $|\hat{\pi}_{it}^\tau(X^\tau, I^\tau) + c_{itjt}^\tau(X^\tau) - \hat{\rho}_{jt}^\tau(X^\tau)| \leq \epsilon$, for all $X_{itjt}^\tau > 0$;

$\hat{\pi}_{it}^\tau(X^\tau, I^\tau) + c_{itjt}^\tau(X^\tau) - \hat{\rho}_{jt}^\tau \geq -\epsilon$, for all $X_{itjt}^\tau = 0$,

and $|\hat{\pi}_{it}^\tau(X^\tau, I^\tau) + H_{iti(t+1)}^\tau(I^\tau) - \hat{\pi}_{i(t+1)}^\tau(X^\tau, I^\tau)| \leq \epsilon$, for all $I_{iti(t+1)}^\tau > 0$;

$\hat{\pi}_{it}^\tau(X^\tau, I^\tau) + H_{iti(t+1)}^\tau(I^\tau) - \hat{\pi}_{i(t+1)}^\tau(X^\tau, I^\tau) \geq -\epsilon$, for all $I_{iti(t+1)}^\tau = 0$ then stop;

else, set $\tau = \tau + 1$, and go to Step 1.

It is emphasized that subproblem (2.30) decomposes into T subproblems, $t = 1, \ldots, T$, each of which is a static single commodity spatial price equilibrium problem with a special bipartite network structure and because of the construction of the new functions, which are linear, and separable, each subproblem is equivalent to a quadratic programming problem. Special-purpose algorithms for the subproblems, called market equilibration algorithms, have been developed in Dafermos and Nagurney [23] and theoretically analyzed in Eydeland and Nagurney [28]. Subproblem (2.31) is also a quadratic programming problem to which, for example, a Gauss-Seidel method can be applied. Parts (1) and (2) of Steps 1 and 2 can be solved simultaneously. For a graphical depiction of the parallelism, see Figure 5.

The strong monotonicity condition (2.14) is assumed to hold, thus guaranteeing existence and uniqueness of the equilibrium pattern. Convergence of the algorithm then holds under the following condition (Bertsekas and Tsitsiklis [9], Proposition 5.8, Section 3.5).

Theorem 4:

Let $K_1 \equiv \{X' | X' \geq 0\}, K_2 \equiv \{I' | I' \geq 0\}$, and $K^2 = K_1 \times K_2$. Suppose that there exist symmetric positive definite matrices G_i and some $\delta > 0$ such that $D_i(\cdot) - \delta G_i$ is nonnegative definite for $i = 1, 2$ and $y' = (X', I') \in K^2$, and there exists some $\beta \in [0, 1)$ such that

$$\|G_i^{-1}(f_i(y') - f_i(z) - D_i(z) \cdot (y'_i - z_i))\|_i \leq \delta\beta \max_j \|y'_j - z_j\|_j, \qquad (2.32)$$

$$\text{for all } y', z \in K^2$$

where $\|y_i\|_i = (y_i^T G_i y_i)^{1/2}$. Then the above parallel linearization decomposition algorithm converges to the equilibrium solution.

2.3.3 Decomposition by Markets

Static and single commodity spatial price equilibrium can be further decomposed by supply and demand markets. Dafermos and Nagurney [22] and Nagurney [55] [56] introduced supply and demand market equilibration algorithms, which are serial algorithms.

In this subsection, a parallel decomposition algorithm by markets to solve the variational inequality (2.19) is briefly described. In the first, the cost functions $\pi(s), \rho(d)$, and $c(X)$ are projected to get the linear diagonal cost functions

$\tilde{\pi}(s_i), \tilde{\rho}(d_j)$, and $\tilde{c}(X_{ij})$. Then, the resulting variational inequality is equal to a quadratic problem. Since the supply and demand quantities are unknown and the cost functions are linear and separable, the linearized variational inequalities can be solved via the elastic exact algorithm given in Dafermos and Nagurney [23] and Nagurney [59]. Because this problem is isomorphic to the constrained matrix problem with unknown row/column totals, the detailed algorithm will be given in Section 3.3.

CHAPTER 3

NETWORK OPTIMIZATION PROBLEMS

We now turn to the study of a special case of network equilibrium problems, that is, network optimization problems. (Recall that a network equilibrium with the *symmetry* assumption for the underlying functions can be reformulated as a network optimization problem.) As an example, the general constrained matrix problem is used. Note that the constrained matrix problem is characterized by a special bipartite network structure.

3.1 Constrained Matrix Problems

The general constrained matrix problem is to compute the best possible estimate X of an unknown matrix, requiring either that the matrix X be a minimum distance from a given matrix X^0, or that X be a functional form of another known matrix (see Figure 6). The constrained matrix problem can be considered as a network optimization problem having a special bipartite structure as depicted in Figure 3, and this reformulation as a network problem will be used to obtain efficient algorithms which exploit this special structure. Recently, researchers have formulated constrained matrix problems as mathematical programming problems,

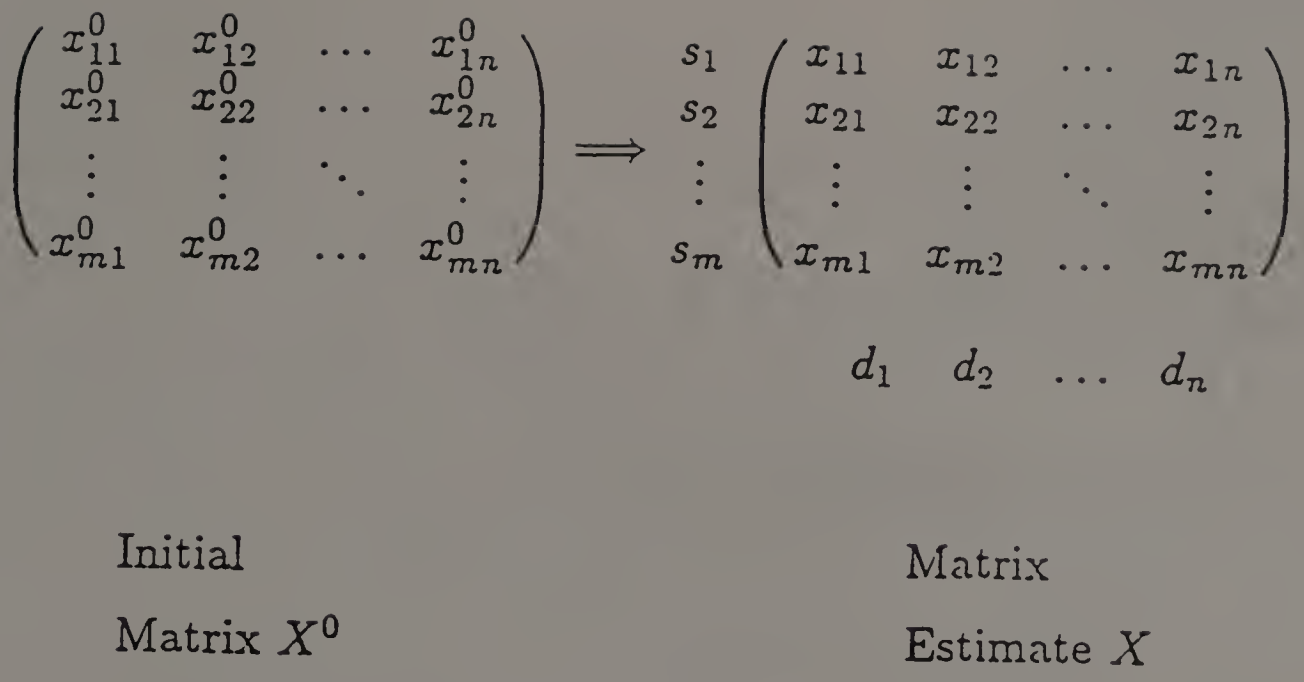


Figure 6 The Constrained Matrix Problem

with an objective function that forces conservatism on the process of rationaliz-ing X from the initial estimate X^0. In the classical setting, the row and column totals are known and fixed. However, in certain applications, the row and column totals are not known a priori, but must be estimated. The relationship of the constrained matrix problem of unknown row and column totals with the spatial price equilibrium problem is explained in Section 3.2.

Now the model is described. Consider the given $m \times n$ matrix by $X^0 = (x^0_{ij})$, and denote the matrix estimate by $X = (x_{ij})$. Let s^0_i denote the row i total, and s_i the estimate of the row i total. Let d^0_j denote the column j total, and d_j the estimate of column j total. Let the $mn \times mn$ matrix $G = (w_{ijkl})$ denote the imposed weight matrix for the mixed variable terms $(x_{ij} - x^0_{ij}) \cdot (x_{kl} - x^0_{kl})$ and assume the matrix G to be strictly positive definite. Let $m \times m$ 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 $B = (\beta_{jl})$ 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 can be written as follows:

$$\text{Minimize } \frac{1}{2}\left\{\sum_{i=1}^{m}\sum_{k=1}^{m}\alpha_{ik}(s_i - s_i^0)\cdot(s_k - s_k^0) + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} w_{ijkl}\right.$$

$$\left.(x_{ij} - x_{ij}^0)\cdot(x_{kl} - x_{kl}^0) + \sum_{j=1}^{n}\sum_{l=1}^{n}\beta_{jl}(d_j - d_j^0)\cdot(d_l - d_l^0)\right\} \tag{3.1}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = s_i, \qquad i = 1, \ldots, m \tag{3.2}$$

$$\sum_{i=1}^{m} x_{ij} = d_j, \qquad j = 1, \ldots, n \tag{3.3}$$

and the nonnegativity conditions. Note that the objective function represents the weighted squared sums of the deviations.

The objective function (3.1) permits the utilization of mixed-variable weight terms. This extends the modeling capabilities of the constrained matrix problem. An example of possibly fully dense A, B, and G matrices are the inverse of the respective variance-covariance matrices (see Judge and Yancey [40]). Other examples may arise when the matrices A, B, and G include subjective weights based on the expert knowledge of planners.

Now, as an illustration, the above constrained matrix problem is described in the context of input/output (I/O) matrices. An I/O table is constructed from data obtained for a specific entity, be it a country, region, state, etc. The economic activity is divided into a number of producing sectors (or industries), with the exchange 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 Miller and Blair [48].

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 the 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. The above formula-

tion takes into account such knowledge gaps directly, thus permitting modeling flexibility.

In the diagonal case, where $\alpha_{ik} = 0$, for $k \neq i$, $w_{ijkl} = 0$, for $kl \neq ij$, and $\beta_{jl} = 0$, for $j \neq l$, the objective function (3.1) simplifies to:

$$\text{Minimize } \frac{1}{2}\left\{\sum_{i=1}^{m}\alpha_{ii}(s_i - s_i^0)^2 + \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ijij}(x_{ij} - x_{ij}^0)^2 + \sum_{j=1}^{n}\beta_{jj}(d_j - d_j^0)^2\right\} \quad (3.4)$$

subject to the constraints (3.2), (3.3), and the nonnegativity assumption.

The choice of weights is also flexible in this formulation. When the weights in (3.4) are all equal to one, the problem becomes a constrained least squares problem, and when $\alpha_{ii} = \frac{1}{s_i^0}$, $\beta_{jj} = \frac{1}{d_j^0}$, and $\gamma_{ijij} = \frac{1}{x_{ij}^0}$, for all rows i and columns j, the objective function is the well-known chi-square. Other possible weights, include $\alpha_{ii} = \frac{1}{s_i^{0\frac{1}{2}}}$, $\beta_{jj} = \frac{1}{d_j^{0\frac{1}{2}}}$, and $\gamma_{ijij} = \frac{1}{x_{ij}^{0\frac{1}{2}}}$; or a mixed weighting scheme. Of course, when the inverse of the variance-covariance matrix is diagonal, the objective function (3.4) can also be used.

This diagonal model has been studied by Nagurney [59] who identified its connection 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 [35], who considered interval constraints, rather than equality constraints (3.2) and (3.3).

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

(3.1), (3.2), and (3.3), collapses to the quadratic constrained matrix problem with fixed row and column totals formulated in Nagurney and Robinson [67] via RC equilibration.

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

$$\text{Minimize } \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} w_{ijkl}(x_{ij}-x_{ij}^0)\cdot(x_{kl}-x_{kl}^0) \tag{3.5}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = s_i^0, \quad i = 1,\ldots,m \tag{3.6}$$

$$\sum_{i=1}^{m} x_{ij} = d_j^0, \quad j = 1,\ldots,n. \tag{3.7}$$

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. Row and column totals may be available 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 $\gamma_{ijkl} = 0$, for $kl \neq ij$, the objective function (3.5) further simplifies to

$$\text{Minimize } \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ijij}(x_{ij}-x_{ij}^0)^2 \tag{3.8}$$

subject to (3.6) and (3.7).

Deming and Stephan [24] considered (3.8) with $\gamma_{ijij} = \frac{1}{x^0_{ij}}$, subject to constraints (3.6) and (3.7), whereas Friedlander [32] considered the case where $G = I$. Bachem and Korte [5] treated (3.8) with all of the constraints. Cottle, et.al. [13], on the other hand, studied Friedlander's problem with upper and lower bounds. Ohuchi and Kaji [69] [70] studied the Bachem and Korte problem with upper and lower bounds. Klincewicz [42] also studied the above diagonal model.

3.2 Relationship to the Spatial Price Equilibrium Problems

As mentioned in the Introduction, the conjecture that a relationship exists between spatial price equilibrium problems and constrained matrix problems dates to Stone [80]. Here the connection between the general constrained matrix problem and spatial price equilibrium problems is further explained.

The Kuhn-Tucker optimality conditions for (3.1), (3.2) and (3.3), can be stated thus: For all rows i, $i = 1, \ldots, m$ and for all columns j, $j = 1, \ldots, n$, the solution (s, x, d) where $s = \{s_i\} \in R^m$, $x \in R^{mn}$ is the vectorization of the matrix X, and $d = \{d_j\} \in R^n$, satisfies constraints (3.2) and (3.3), and:

$$\sum_j \alpha_{ij} s_j - \sum_j \alpha_{ij} s^0_j + \sum_{kl} w_{ijkl} x_{kl} - \sum_{kl} w_{ijkl} x^0_{ijkl}$$

$$\begin{cases} = & -\sum_l \beta_{jl} d_l + \sum_l \beta_{jl} d^0_l, \quad \text{if } x_{ij} > 0 \\ \geq & -\sum_l \beta_{jl} d_l + \sum_l \beta_{jl} d^0_l, \quad \text{if } x_{ij} = 0. \end{cases} \tag{3.9}$$

Note that the Kuhn-Tucker conditions above are equivalent to the well-known Samuelson [77], and Takayama and Judge [82] spatial price equilibrium conditions. In this context, the supply price function for the commodity at the supply market i, π_i, is linear and is given by:

$$\pi_i = \pi_i(s) = \sum_j \alpha_{ij} s_j - \sum_j \alpha_{ij} s_j^0, \tag{3.10}$$

where s_j denotes the supply of the commodity at the supply market j, the demand price function for the commodity at the demand market j, ρ_j is also linear and given by:

$$\rho_j = \rho_j(d) = -\sum_l \beta_{jl} d_l + \sum_l \beta_{jl} d_l^0, \tag{3.11}$$

where d_l denotes the demand for the commodity at the demand market l and, finally, the transportation cost function, c_{ij}, associated with shipping the commodity between the supply market i and the demand market j, is also linear and given by:

$$c_{ij} = c_{ij}(x) = \sum_{kl} w_{ijkl} x_{ijkl} - \sum_{kl} w_{ijkl} x_{ijkl}^0. \tag{3.12}$$

The matrices A, B, and G are assumed to be symmetric and positive definite.

Equations (3.9) through (3.12) state that the commodity shipment x_{ij} is greater than zero, if the supply price at the supply market i plus the cost c_{ij} is equal to the demand price ρ_j, and zero if the supply price plus transportation cost is greater than or equal to the demand price. The above general constrained

matrix problem is, hence, isomorphic to symmetric spatial price equilibrium problems in the case of linear supply price. demand price, and transportation cost functions. This isomorphism is used to develop a fine grain parallel algorithm for such problems which will be described in the next section. Note that, in real-world applications, the large-scale nature of these problems makes the application of parallel decomposition schemes appealing.

It is emphasized, however, that the Splitting Equilibration Algorithm which will be described in the subsequent section, and also be applied to the asymmetric spatial price equilibrium problem, where A, B, and G are no longer symmetric. This enables, for example, the modeling and solution of multicommodity spatial price equilibrium problems. In the asymmetric case, as discussed in Chapter 2, the spatial price equilibrium conditions (2.18) can no longer be formulated as an equivalent quadratic programming problem, but, rather, as a variational inequality problem (2.19). Note, however, that the optimal solution to problem (3.1), nevertheless, always satisfies the variational inequality (2.19), with π_i, ρ_j and c_{ij} defined by (3.10), (3.11), and (3.12). The algorithm that will be described decomposes the problem by markets, rather than commodities, and represents, hence, a new parallelization scheme for asymmetric spatial price equilibrium problems, and, moreover, as we shall be shown, each subproblem can be further decomposed by markets yielding a massively parallel decomposition.

3.3 The Parallel Splitting Equilibration Algorithm

In this section, first the Splitting Equilibration Algorithm (SEA)is reviewed, and then the massively parallel SEA algorithm for the computation of the general quadratic constrained matrix problem is developed. For theoretical results, see Nagurney and Eydeland [63] and for a graphical depiction of SEA, see Figure 7. The special network structure of the row and column equilibration step can be depicted in Figure 8. In the case where the matrices A, B, and G are diagonal, SEA is a Lagrangean dual method.

3.3.1 SEA for the General Constrained Matrix Problem

The SEA algorithm for the general constrained matrix problem (3.1) through (3.3) with unknown row/column totals is now reviewed.

When the row and column totals are unknown, the SEA algorithm constructs a series of diagonal problems of the form (3.4), subject to the constraints (3.2), (3.3), and the nonnegativity assumption, via a projection step. This splitting procedure results in elastic supply and demand subproblems. These elastic subproblems, in turn, are then solved via the elastic exact procedure given in Dafermos and Nagurney [23] and Nagurney [59]. They developed serial equilibration algorithms and applied them to compute the solutions to classical spatial price equilibrium problems with elastic supplies and demands rather than fixed ones. For a theoretical analysis including complexity, see Eydeland and Nagurney [28] and Nagurney and Eydeland [63].

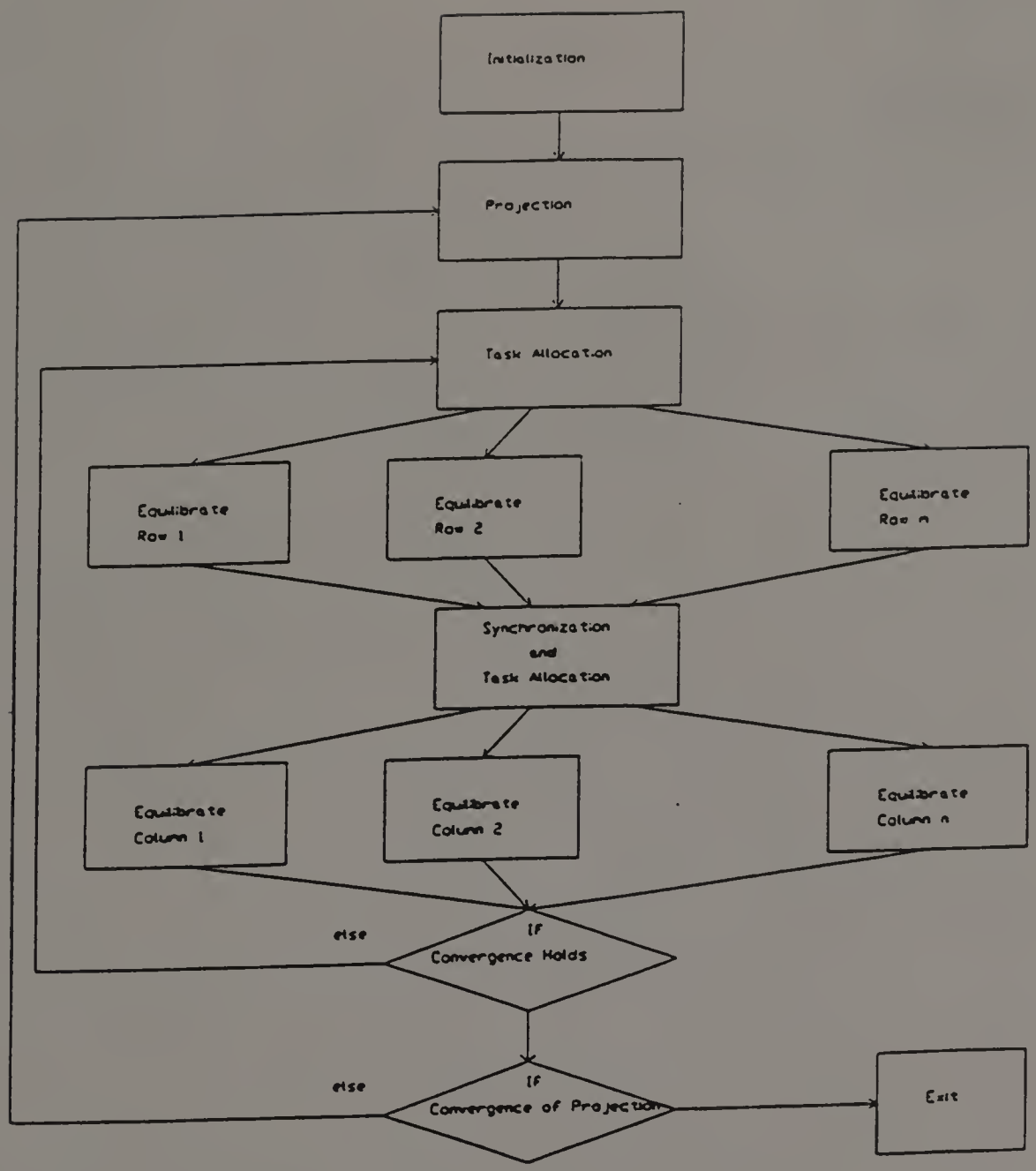
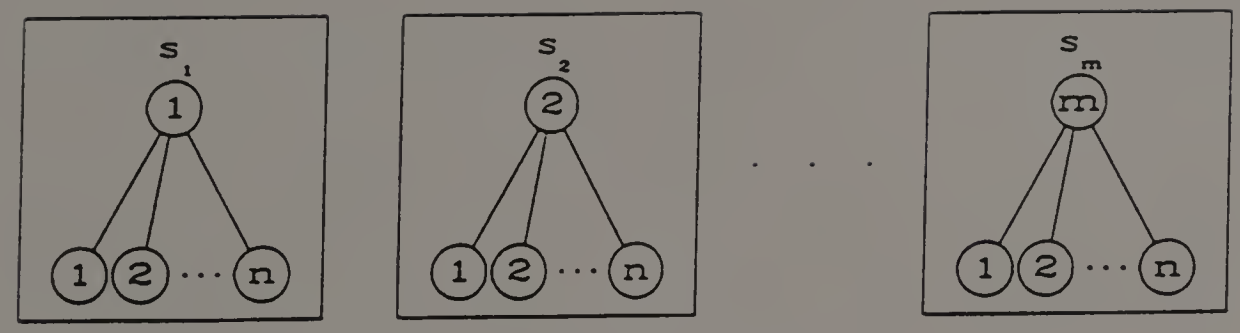
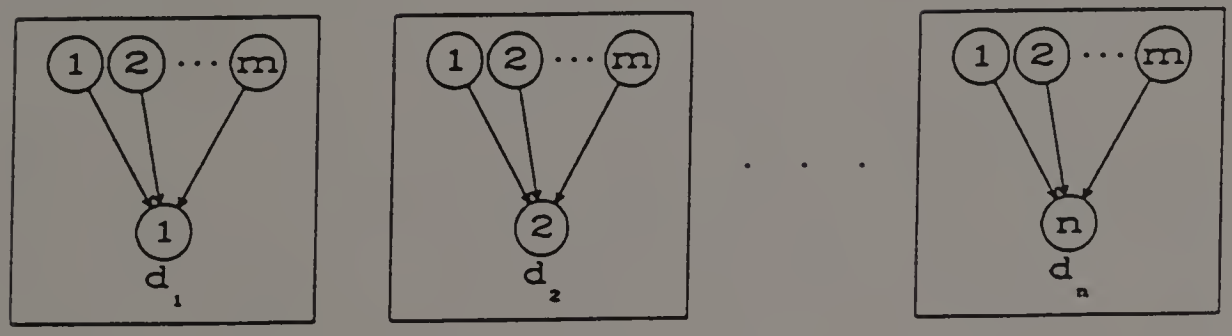


Figure 7 Flowchart of Splitting Equilibration Algorithm - the General Case



Row Equilibration Step



Column Equilibration Step

Figure 8 Decomposition by Supply and Demand Markets

Step 0: Initialization Step

Start with any feasible vector (s^1, x^1, d^1), i.e., one which satisfies constraints (3.2) and (3.3). Set $\xi = 2$.

Step 1: Projection Step

Given $(s^{\xi-1}, x^{\xi-1}, d^{\xi-1})$, the objective function (3.1) can be written as following:

$$\text{Minimize } \frac{1}{2}s^T\tilde{A}s + (\tilde{A}s^{\xi-1} - s^{0T}A - As^{\xi-1})s + \frac{1}{2}x^T\tilde{G}x$$

$$+ (\tilde{G}x^{\xi-1} - x^{0T}G - Gx^{\xi-1})x + \frac{1}{2}d^T\tilde{B}d + (\tilde{B}d^{\xi-1} - d^{0T}B - Bd^{\xi-1})d \qquad (3.13)$$

subject to constraints (3.2) and (3.3). Here $\tilde{A}, \tilde{B}$, and $\tilde{G}$ denote the diagonal matrices of A, B, and G, respectively.

Step 2: Initialization Step for Row/Column Equilibration

Let $\lambda^0 \in R^n = 0$. Set $\tau = 1$.

Step 3: Row Equilibration

For each row i, $i = 1, \ldots, m$, compute the fixed cost terms $\tilde{c}_{ij}^{\tau}$, $j = 1, \ldots, n$ where

$$\tilde{c}_{ij}^{\tau} = -w_{ijij}x_{ij}^0 - \lambda_j^{\tau-1} \qquad (3.14)$$

and solve the i-th row equilibrium subproblem:

$$\text{Minimize } \frac{1}{2}s^T\tilde{A}s - s^{0T}\tilde{A}s + \frac{1}{2}x^T\tilde{G}x + \tilde{c}^Tx \qquad (3.15)$$

subject to (3.2) via elastic exact equilibration (cf. Dafermos and Nagurney [23], Eydeland and Nagurney [28].)

Obtain the Lagrange multiplier μ_i^τ corresponding to the linear constraint in (3.2).

Step 4: Column Equilibration

For each column j, $j = 1, \ldots, n$, compute the fixed cost terms $\bar{\bar{c}}_{ij}^\tau, i = 1, \ldots,$ where

$$\bar{\bar{c}}_{ij}^\tau = -w_{ijij} x_{ij}^0 - \mu_i^\tau \tag{3.16}$$

and solve the j-th column equilibration subproblem:

$$\text{Minimize } \frac{1}{2} x^T \tilde{G} x + \bar{\bar{c}}^T x + \frac{1}{2} d^T \tilde{B} d - d^{0T} \tilde{B} d \tag{3.17}$$

subject to (3.3) via elastic exact equilibration. Obtain the Lagrange multiplier λ_j^τ corresponding to the linear constraint (3.3).

Step 5: Convergence Test of Equilibration Steps

If $|x_{ij}^\tau - x_{ij}^{\tau-1}| \leq \epsilon$, for all i, j, go to Step 6;

else set $\tau = \tau + 1$, and go to Step 3.

Step 6: Convergence Test of Projection

If $|x_{ij}^\xi - x_{ij}^{\xi-1}| \leq \epsilon'$, for all i, j, then stop;

otherwise, set $\xi = \xi + 1$, and go to Step 1.

When the cost functions given are linear, then Steps 1 and 6 are deleted.

When the row/column totals are known with certainty, the general constrained matrix problem simplifies to (3.5) subject to (3.6) -(3.7). In this case, the SEA

algorithm again constructs a series of diagonal problems, but of the form (3.8), subject to the constraint (3.6), (3.7) and nonnegativity, via a projection step. The resulting quadratic programming problems are, hence, simpler than the original problem with objective function given by (3.5). Each of the constructed diagonal problems, in turn, is then "split" to handle the row constraints (3.6) and then the column constraints (3.7). Each such row (supply) or column (demand) equilibrium 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 and exactly via exact equilibration as outlined in [67]. These problems correspond then to fixed, rather than elastic, supply and demand subproblems. Of course, in the diagonal constrained matrix problem with objective function given by (3.8), rather than (3.5), no such diagonalization is required, before the splitting occurs.

Convergence conditions for the outer projection step can be found in Dafermos [17]. An interpretation of row and column equilibration as a dual method and proof of convergence is given in Nagurney and Eydeland [63].

3.3.2 The Massively Parallel Splitting Equilibration Algorithm

As described in Sections 3.3.1 and 3.3.2, SEA decomposes the constrained matrix problem into m row subproblems, each of which can be solved independently and simultaneously on a distinct processor using exact equilibration, and into n column subproblems, each of which can be solved independently and simultaneously. In this context, hence, if $m = n$ then at most m processors would

be used for the parallel implementation; this is, indeed, the case with a coarse-grain architecture. However, in this section, I will construct a massively parallel implementation of SEA in which the exact equilibration algorithm could exploit all $n \times n$ processors. In a sense such a fine-grain implementation would be on the "link" level, whereas the coarse-grain implementation would be on the node level.

The serial exact equilibration algorithm is reviewed first, and then the parallel exact algorithmis described. Since row equilibration is the mirror image of column equilibration, only the column equilibration step is explained here.

Serial Exact Equilibration

Suppose that the cost functions are linear, separable, for example, $c_{ij}(x_{ij}) = g_{ij}x_{ij} + h_{ij}$ and $\rho_j(d_j) = -p_j d_j + q_j$, where g_{ij}, h_{ij}, p_j and q_j are positive for all i and j. We seek to compute the solution x_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, n$, satisfying $d_j = \sum_{i=1}^{n} x_{ij}$, where:

$$g_{ij}x_{ij} + h_{ij} \begin{cases} = & -p_j d_j + q_j, \quad \text{if } x_{ij} \geq 0 \\ \geq & -p_j d_j + q_j, \quad \text{if } x_{ij} = 0, \end{cases}$$

is satisfied. The procedure below accomplishes this.

Step 0: Set the demand market index $j = 1$.

Step 1: Sort the h_{ij}'s, $i = 1, \ldots, n$ in nondescending order and relabel the h_{ij}'s accordingly. Define $h_{n+1,j} = \infty$ and set $v = 1$.

Step 2: Compute

$$\mu_j^v = \frac{\sum_{i=1}^{v} \frac{h_{ij}}{g_{ij}} + \frac{q_j}{p_j}}{\sum_{i=1}^{v} \frac{1}{g_{ij}} + \frac{1}{p_j}}. \tag{3.18}$$

Step 3: If $h_{vj} < \mu_j^v \leq h_{v+1,j}$, then stop; let the critical $s = v$, and go to Step 4. Otherwise, set $v = v + 1$, and go to Step 2.

Step 4: Set

$$x_{ij} = \begin{cases} \frac{\mu_j^s - h_{ij}}{g_{ij}}, & i = 1, \ldots, s \\ \\ 0, & i = s+1, \ldots, n. \end{cases}$$

If $j < n$, let $j = j + 1$ and go to Step 1. Otherwise, terminate.

Row subproblems can be solved in the same manner. When the demand is known with certainty as d_j^0, the equation (3.18) changes to

$$\mu_j^v = \frac{\sum_{i=1}^{v} \frac{h_{ij}}{g_{ij}} + d_j^0}{\sum_{i=1}^{v} \frac{1}{g_{ij}}}. \tag{3.19}$$

Now the parallel exact equilibration algorithm for column equilibration is described. In the serial exact algorithm, the critical s was calculated in a serial way (see Steps 2 and 3 above) and all steps are done again for the next demand market in a serial way until all demand market subproblems are solved. Instead, in the parallel exact algorithm, all demand subproblems will be solved at the same time and each demand subproblem is further parallelized. At the beginning, all h_{ij}'s are sorted columnwise in parallel. Then, assume that all links can be the critical s's one calculates the μ_j^v's in parallel. After that, find the critical s for each

demand market through the parallel comparison of Step 3 below. The equilibrium x_{ij}'s are, finally, calculated in parallel. The detailed algorithm is as follows.

Parallel Exact Equilibration

Step 1: Sort the h_{ij}'s, $i = 1, \ldots, n, j = 1, \ldots, n$, columnwise in nondescending order and relabel the h_{ij}'s accordingly. This columnwise sorting is done in parallel. Define $h_{n+1,j} = \infty$, for all $j = 1, \ldots, n$.

Step 2: Assume that all $v_j = 1, \ldots, m$, for all $j = 1, \ldots, n$, can be the critical s and compute in parallel

$$\mu_j^{v_j} = \frac{\sum_{i=1}^{v_j} \frac{h_{ij}}{g_{ij}} + \frac{q_j}{p_j}}{\sum_{i=1}^{v_j} \frac{1}{g_{ij}} + \frac{1}{p_j}}, \quad \text{for all } v_j = 1, \ldots, n;\ j = 1, \ldots, n. \tag{3.20}$$

Step 3: Find the critical s_j's for all $j = 1, \ldots, n$ in parallel by the parallel comparison, that is, if $h_{vj} < \mu_j^{v_j} \leq h_{v+1,j}$, then set $s_j = v_j$. This parallel comparison will be done using $n \times n$ processors.

Step 4: Calculate all x_{ij}'s

$$x_{ij} = \begin{cases} \frac{\mu_j^{s_j} - h_{ij}}{g_{ij}}, & i = 1, \ldots, s_j;\ j = 1, \ldots, n \\ 0, & i = s_j + 1, \ldots, n;\ j = 1, \ldots, n. \end{cases}$$

This parallel exact equilibration algorithm can solve $n \times n$ subproblems at the same time. The parallel exact equilibration of each row can be described in the same manner. When the demand is fixed, $\mu_j^{v_j}$ of equation (3.20) is given as in equation (3.19).

The parallel exact algorithm is then embedded into the SEA algorithm to obtain the massively parallel SEA algorithm. For the implementation of this algorithm, we need a massively parallel architecture such as Thinking Machine's Connection Machine, CM-2. The detailed implementation will be described in Section 4.2.

CHAPTER 4

NUMERICAL EXPERIMENTATION

In this chapter, the approach to the parallel numerical experimentation is discussed. This chapter first explains how the data-sets were collected and generated, and then how the parallel computations were implemented. Specifically, multicommodity static and dynamic models were considered for the decomposition by commodities, single commodity dynamic models were considered for the parallel decomposition by time periods, and single commodity static models were considered for the decomposition by markets. In this way, we can increase the size of problems to be tested and obtain the correct efficiencies and speed-ups realized by each parallel decomposition scheme by eliminating the possible joint effects from complex parallel decomposition schemes.

4.1 Data Generation and Collection

4.1.1 Decomposition by Commodities

For the performance evaluation of the decomposition scheme by commodities, problems with linear cost functions were considered first, and then those with nonlinear cost functions.

The linear asymmetric supply price, demand price, transaction cost functions, and inventorying cost functions were given, respectively, by

$$\pi_{it}^{k}(s) = \sum_{j,l,t'} r_{itjt'}^{kl} s_{jt'}^{l} + t_{it}^{k} \tag{4.1}$$

$$\rho_{jt}^{k}(d) = -\sum_{i,l,t'} p_{itjt'}^{kl} d_{i}^{l} + q_{jt}^{k} \tag{4.2}$$

$$c_{ij}^{k}(X) = \sum_{l,u,v,t'} g_{ijtuvt'}^{kl} X_{uvt'}^{l} + h_{ijt}^{k} \tag{4.3}$$

$$H_{iti(t+1)}^{k}(I) = \sum_{j,l,t'} v_{iti(t+1),jt'j(t'+1)}^{kl} I_{jt'j(t'+1)}^{l} + w_{iti(t+1)}^{k}. \tag{4.4}$$

It was assumed that the functions (4.1)-(4.4) satisfy the strong monotonicity condition (1.3) to ensure uniqueness of the equilibrium pattern. In terms of applications, this condition means that the supply price of commodity k at supply market i during time period t depends primarily upon the supply of commodity k at supply market i during time period t; the demand price of commodity k at demand market j during time period t depends primarily upon the demand for the commodity k at demand market j during time period t; the transaction cost for commodity k between supply market i and demand market j during time period t depends primarily upon the quantity of commodity k shipped between the pair of supply and demand markets during time period t; and the inventorying cost for commodity k at supply market i during time periods t and $t+1$ depends primarily upon the inventory quantity at supply market i during time periods t and $t+1$. Example data were generated randomly for 20 supply markets and 20

demand markets and 50 supply markets and 50 demand markets, with the number of commodities being set at 9 and 12 and with the number of time periods being set at 2 and 5.

Linear, asymmetric, single commodity problems of similar market number dimensions had been solved by Nagurney [53] using serial decomposition by demand markets and supply markets and the projection method. Pang [71] presented numerical results with a complementarity algorithm for substantially simpler single-price spatial price equilibrium problems, in which the transaction cost consists of only a fixed transaction cost terms h_{ij}^k, and these terms must satisfy a restrictive triangle inequality (see also Theise and Jones [84]).

The coefficients of the supply price, demand price, transaction cost, and inventorying cost functions were generated randomly as follows, so that a strict diagonal dominance condition held in order to guarantee uniqueness of the solution. The diagonal term, $r_{itit'}^{kk}$, in each supply price function was generated uniformly in the range [100, 30,000]. The remaining cross-terms, $r_{itjt'}^{kl}, l \neq k, j \neq i, t' \neq t$, were generated so that the dependence on the commodities and the supply markets was uniform, and the sum of these terms was less than the diagonal term. The fixed supply price term, t_{it}^k, was generated in the range [100, 1,000]. The diagonal term, $-p_{jtjt}^{kk}$, in each demand price function, was generated uniformly in the range [-50, -5,500]. The remaining cross-terms, $-p_{itjt'}^{kl}, l \neq k, i \neq j, t' \neq t$, were generated so that the dependence on the commodities and the demand markets was uniform and the sum of the absolute values of these terms was less than the absolute value of the diagonal term. The fixed demand price term, q_{jt}^k, was gener-

ated in the range [50, 500.000]. The diagonal term, g^{kk}_{ijtijt}, in each transaction cost function, was generated uniformly in the range [20, 6,000]. The remaining cross-terms, $g^{kl}_{ijtuvt'}, l = k, uv = ij, t' = t$, were generated so that their dependence on the commodities and pairs of markets was uniform and the sum of cross-terms was less than the diagonal term. The fixed transaction cost term, h^k_{ijt}, was generated in the range [50, 500]. The diagonal term, $v^{kk}_{iti(t-1),iti(t-1)}$, in each inventorying cost function was generated uniformly in the range defined by 0.075 times the lower and upper limits of the slope of the supply price function. The remaining cross-terms, $v^{kl}_{iti(t-1),jt'j(t'-1)}, l = k, j = i, t' = t$, were generated so that the dependence on the commodities, the supply markets, and the time periods was uniform, and the sum of these terms was less than the diagonal term. The fixed inventorying cost term, $w^k_{iti(t-1)}$, was also generated uniformly in the range defined by 0.075 times the lower and upper limits of the intercept of the supply price function.

Nonlinear supply price, demand price, and transaction cost functions were considered in the form

$$\pi^k_{it}(s) = r^k_{itit}(s^k_{it})^2 + \sum_{j,l,t'} r^{kl}_{ijt'} s^l_j + t^k_{it} \tag{4.5}$$

$$\rho^k_{jt}(d) = -p^k_{jtjt}(d^k_{jt})^2 + \sum_{i,j,t'} p^{kl}_{jt'it} d^l_{it} + q^k_{jt} \tag{4.6}$$

$$c^k_{ijt}(X) = g^k_{ijtijt}(X^k_{ijt})^4 + \sum_{l,u,v,t'} g^{kl}_{ijtuvt'} X^l_{uvt'} + h^k_{ijt} \tag{4.7}$$

$$H^k_{iti(t-1)}(I) = v^k_{iti(t-1),iti(t-1)}(I^k_{iti(t-1)})^2 + \sum_{j,l,t'} v^{kl}_{iti(t-1),jt'j(t'-1)} I^l_{jt'j(t'-1)} + h^k_{iti(t-1)}. \tag{4.8}$$

These functions were also generated to ensure that the strong monotonicity condition held. In terms of applications, the polynomially nonlinear terms give more dependency to each cost function upon its main variable. In other words, each cost function has less effect from the cross-terms. The linear terms in the function (4.5), (4.6), (4.7), and (4.8), were generated in the same manner as their counterparts in (4.1), (4.2), (4.3), and (4.4). The nonlinear term, r^k_{itit} was generated uniformly in the range [100, 30,000] $\times 10^{-6}$; the nonlinear term, $-p^k_{jtjt}$, uniformly in the range [-50, -5,500] $\times 10^{-6}$; and nonlinear term, g^k_{ijtijt}, uniformly in the range [20, 60,000] $\times 10^{-6}$; and nonlinear term, $v^k_{iti(t+1),iti(t+1)}$, uniformly in the range defined by 0.075 times the lower and upper limits of r^k_{itit}. The termination criterion utilized was identical to the one used in the linear tests.

4.1.2 Decomposition by Time Periods

To study the performance of decomposition scheme by time periods, single commodity dynamic problems were considered rather than multicommodity problems due to the memory limitation. The linear diagonal supply price, demand price, transaction cost, and inventorying cost functions are of the form:

$$\pi_{it}(s_{it}) = r_{it}s_{it} + u_{it} \tag{4.9}$$

$$\rho_{jt}(d_{jt}) = -p_{jt}d_{jt} + q_{jt} \tag{4.10}$$

$$c_{itjt}(X_{itjt}) = g_{itjt}X_{itjt} + h_{itjt} \tag{4.11}$$

$$H_{iti(t+1)}(I_{iti(t+1)}) = v_{iti(t+1)}I_{iti(t+1)} + w_{iti(t+1)} \tag{4.12}$$

where $r_{it}, u_{it}, p_{jt}, q_{jt}, g_{itjt}, h_{itjt}, v_{iti(t+1)}, w_{iti(t+1)} > 0$. Under this assumption, the equilibrium must be unique, since this is a strictly convex quadratic programming problem. In terms of application, these linear diagonal cost functions mean that the supply price at supply market i during the time period t depends only upon the supply quantity at supply market i during the same time period; the demand price at demand market j during the time period t depends only upon the demand quantity at demand market j during the same time period; the transaction cost between supply market i and demand market j depends only upon the quantity shipped between the pair of markets (i, j); and the inventory carrying cost at supply market i between the time periods depends only upon the inventory quantity at the same market between the same time periods.

The supply price function coefficients r_{it} and u_{it} were generated randomly and uniformly in the ranges [3, 10] and [10, 25], respectively; the demand price coefficients p_{jt} and q_{jt} were generated in the ranges [1, 5] and [150, 650], respectively; the transaction cost function coefficients g_{itjt} and h_{itjt} were generated in the ranges [1, 16] and [10, 25] respectively; and the inventorying cost coefficients $v_{iti(t+1)}$ and $w_{iti(t+1)}$ in the ranges $.075 \times [3, 10]$ and $.075 \times [10, 25]$, respectively.

Problems ranging in size from 5 supply markets and 5 demand markets to 50 supply markets and 50 demand markets with the number of time periods ranging from 5 to 50 time periods were considered. The convergence tolerance ϵ was set to 1 and convergence was checked after every other iteration.

For the general nonlinear, asymmetric dynamic single commodity market problems, the functions were now of the form:

$$\pi_{it}(s) = r_{it}s_{it}^2 + \sum_{jt'} r_{itit'}s_{jt'} + u_{it} \tag{4.13}$$

$$\rho_{jt}(d) = -p_{jt}d_{jt}^2 + \sum_{lt'} p_{jtlt'}d_{lt'} + q_{jt} \tag{4.14}$$

$$c_{itjt}(X) = g_{itjt}X_{itjt}^4 + \sum_{kt'kt''} g_{itjt,kt'lt''}X_{kt'lt''} + h_{itjt} \tag{4.15}$$

$$H_{iti(t+1)}(I) = \sum_{kt'lt''} u_{iti(t+1),kt'lt''}I_{kt'lt''} + w_{iti(t+1)}. \tag{4.16}$$

The function coefficients were generated as follows. The fixed coefficients in functions (4.13)-(4.16) and the diagonal linear terms were generated in the same manner as their counterparts in (4.9)-(4.12). The off-diagonal terms were then generated to ensure strict diagonal dominance. The term r_{it} was generated in the range [3,10] $\times 10^{-5}$; the term p_{jt} was generated in the range [1,5] $\times 10^{-5}$; and the term g_{itjt} was generated in the range [1,16] $\times 10^{-5}$. The same convergence tolerance as before was used.

The ranges of the market sizes and number of time periods was as before, with the number of cross-terms in each of the functions being fixed at five.

4.1.3 Decomposition by Markets or by Row/Column

In the performance evaluation of the decomposition scheme by markets, two kinds of problems are considered; static single commodity spatial price equilibrium problems and constrained matrix problems with fixed demands. The latter being special cases of the general elastic constrained problems, which are isomor-

phic to the static single commodity spatial price equilibrium problems (see the Section 3.2).

To increase the data size, the matrix G of the equation (3.1) was generated to be diagonal, with each diagonal term generated in the range [.1,10000], to simulate the wide spread of the initial of the initial data. The weighting terms, the w_{ijij}'s were set to $\frac{1}{x_{ij}^0}$ for $x_{ij}^0 \neq 0$, and to one, otherwise. The selected problem for computation had a dimension of the X^0 matrix of 1000×1000, with the corresponding G matrix of dimension $10^6 \times 10^6$. In addition, since the efficiency of algorithms may depend not only upon the dimensionality of the problem, but, also upon the difficulty of the problems themselves, each row total s_i^0 was generated as $2\sum_i x_{ij}^0$, and each column total d_j^0 as $2\sum_i x_{ij}^0$, to represent 100% growth from the initial time period.

For the numerical results of SEA applied to real-world economic data, three data sets were used. The first example IO72 was constructed, assuming a 100% growth factor, from an aggregated 1972 input/output matrix of construction activity in the United States consisting of 485 rows and 485 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 second example, IO77, was constructed from an aggregated 1977 input/output matrix in the same manner as IO72. The third data-set, USDA133, was obtained from a social accounting matrix for the United States. A social accounting matrix(SAM) is a general equilibrium data matrix, consisting of a series of accounts in the economy of a nation. A SAM is comprised of m rows

and m columns, with any particular account, i, represented by row i and column i. The rows represent the receipts 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 Pyatt and Round [75]). USDA133, which is comprised of 133 rows and 133 columns, was a perturbed SAM developed at the United States Department of Agriculture for 1982 (For a description of its development, see Hanson and Robinson [34]).

For the unknown row/column totals, two data-sets of spatial price equilibrium problems with linear and separable cost functions were generated. The equivalence of such spatial price equilibrium problems with diagonal quadratic constrained matrix problems was described in Section 3.2. SPEP500 $\times$ 500 was comprised of 500 supply markets and 500 demand markets. SPEP750 $\times$ 750 was comprised of 750 supply markets and 750 demand markets.

4.2 Parallel Implementation Method

4.2.1 Coarse Grain Parallel Implementation

The decomposition by time periods, commodities and markets (row/columns) generate, respectively, parallel subproblems in the number of time periods, commodities and markets. This coarse grain parallel implementation was tested on the IBM 3090 which has six processors.

Data-sets described in the preceding section were tested using algorithms developed in Chapters 2 and 3. In particular, the IBM 3090-600E and the IBM

3090-600J were utilized with six processors in a stand-alone environment, with an exception made for the decomposition by commodities. since it was implemented in the early stage of this research. Instead, a high Strategic User Priority was used, in which up to three processors were allocated to this experiment. Within the Strategic Users' higher priority, the elapsed wall-clock time was not significantly affected by the multiuser environment. Parallel Fortran was used to execute the algorithms and the elapsed wall clock time was measured using the intrinsic function of Parallel Fortran.

The speed-up measures were defined as follows:

$$\text{Speedup} \quad S = \frac{T_1}{T_N}, \tag{4.17}$$

where T_1 is the elapsed wall clock time to solve the problem using the best serial algorithm and T_N is the elapsed time to solve the problem using the parallel algorithm on N processors. Note that it is very difficult to find the best serial algorithm for given problems. The alternative measurement would be defined as

$$\text{Speedup} \quad S' = \frac{T_1'}{T_N}, \tag{4.18}$$

where T_1' is the elapsed time to solve the problem using the parallel algorithm.

The efficiencies of the parallel implementation using N processors were then defined as:

$$E_N = \frac{S_N}{N} = \frac{T_1}{T_N \times N} \tag{4.19}$$

$$E'_N = \frac{S'_N}{N} = \frac{T'_1}{N \times T_N}. \tag{4.20}$$

In the ideal parallel implementation, the efficiency will be 100%, so the maximum speed-up ratio will be equal to the number of processors. But, in the practical implementation, we cannot obtain this speedup due to the hardware limitation.

4.2.2 Fine Grain Parallel Implementation

The decomposition by demand and supply market pairs introduced in Section 3.2 was implemented on a massively parallel architecture, the CM-2. Thinking Machine's Connection Machine Model CM-2 has, in its full configuration, 64K processors which have distributed memories and obtain information through communication between processors. Since massively parallel implementations are in the beginning stages, I briefly describe the architecture of the CM-2.

The processors are interconnected as a 12-dimensional hypercube, with each processor containing a local memory of 8K bytes. Each processing element is under the control of a microcontroller that sends instructions from a front-end computer to all of the elements for execution. The mode of computation is data level parallelism, that is, all processors execute identical operations. Therefore a CM-2 is called a SIMD machine.

The language used for the implementation was CM FORTRAN version 1.0. It is a high-level language that compiles into PARIS, the assembly level language of the machine. It is a very compact language with, for example, the addition of

two matrices being expressed in a single step. Since matrix operations must be conformable, i.e., the operated on matrices must be of the same dimensions, one may need to change a matrix into a vector, or vice versa; for such transformations the intrinsic functions "pack" and "unpack" are very useful. Also one may use the "spread" command to replicate a vector into a matrix.

Some of the intrinsic functions of CM FORTRAN are briefly described here. These intrinsic functions make CM FORTRAN very suited for implementing the parallel exact algorithm outlined in Section 3. For example, the intrinsic function "cmforder" sorts elements of a matrix either row-wise or column-wise and returns the indices. Since a matrix can not be used as indices of another matrix, but a vector can only when there is not collision among the values, the index matrix is packed first using "pack" command. To avoid collisions among values of indices, the index matrix was modified by adding some numbers, and then packed to get the index vectors without collision. Then the sorted matrix h_{ij} was obtained.

The "minval" and "maxval" functions, in turn, return the smallest, respectively, largest element of a row or a column in an array. These are used to find critical S's in Step 3 for the parallel exact algorithm in Chapter 3. The "cshift" and "eoshift" functions are useful in minimizing the cost of communication between processors in which the data elements are located. These "shift" commands bring the neighboring h_{ij}, $h_{i+1,j}$ values to the same virtual processor before the comparison, in order to minimize the communication.

Finally, logic statements, such as the "where, else, end" statement, are available to check conditions on vector/matrix elements in.parallel.

Since the "pack" command packs a matrix columnwise, the row equilibration step requires the "transpose" command before the "pack" command is used.

In order to take advantage of the data level parallelism a large number of processors are needed to operate on multiple copies of the data simultaneously. Note that in an input/output matrix consisting of 500 rows and 500 columns we would need 250,000 processors which is greater than the number of physical processors available to us even in a fully configured CM-2. The CM-2, however, has the notable feature known as *virtual processor* that permits a processor to operate on multiple copies of data. This feature is identical to having multiple physical processors operating on their own copy of the data. To measure and analyze the performance of the massively parallel implementation, CM time is measured with the number of processors.

CHAPTER 5

RESULTS AND DISCUSSION

In this chapter, all numerical results of the coarse-grain and fine-grain parallel implementations of the parallel algorithms are reported and discussed. The detailed algorithms were described in Chapters 2 and 3 and the numerical experimentation methods were explained in Chapter 4.

5.1 Decomposition by Commodities

In this section, the results of computational experience with the parallel algorithms introduced in Section 2.3 and Chapter 3 on the IBM 3090-600E in a production environment with a high Strategic User priority (to eliminate contention) are presented. In this decomposition, each feasible set K_k corresponds to a particular commodity k. The six numerical data-sets were generated in the manner explained in Section 4.1. The first example, MSP1, was a static problem consisting of 12 commodities, 50 supply markets and 50 demand markets and with quadratic supply price and demand price functions and quartic transportation cost functions. The second example, MSP2, was a static problem consisting of 9 commodities, 50 supply markets and 50 demand markets and with quadratic

supply and demand price functions and quartic transportation cost functions. Note that the decomposition of the multicommodity static problem results in the solution of as many static problems of the form of the static single commodity problems in Figure 2 as there are commodities. For each of the classical bipartite spatial price equilibrium problems, the equilibration algorithms introduced in Dafermos and Nagurney [23] were applied. The next four examples, from DMSP1 to DMSP4, were dynamic spatial price equilibrium problems. The third example, DMSP1, consisted of 12 commodities, 20 supply markets, 20 demand markets, and 2 time periods, whereas the fourth example, DMSP2, consisted of the number of commodities and markets as DMSP1, but had 5 time periods. The form of the functions was as in the static example, with the cost functions on the horizontal links now being quadratic. The fifth and sixth examples, DMSP3 and DMSP4, had the same dimension as DMSP1 and DMSP2, respectively, but their cost functions were linear. The equilibration algorithm embedded in the dynamic problems was the one in Nagurney and Aronson [61].

These algorithms were also embedded with Parallel Fortran constructs and compiled using the Parallel Fortran compiler, optimization level 3. The parallel runs were conducted under the Strategic Users' Program on the IBM 3090-600E and are reported in Table 1 and Figure 9. Additional serial results on the IBM 3090/600E for the dynamic market equilibrium problems can be found in Nagurney [58] and additional parallel results for the static problems can be found in Nagurney and Kim [65]. The elapsed wall-clock times were measured, rather than CPU times since these are the appropriate measurements in parallel processing.

Indeed, it is well known that the CPU time will increase in any parallel processing environment.

The construction of new cost functions for each commodity (cf (2.20), (2.21), and (2.22)), as well as the subsequent solution of L commodity problems was parallelized. In this series of experiments, the number of commodities was greater than, or equal to, the number of processors. In the case where the number of commodities exceeds the number of processors, the commodity subproblems are assigned asynchronously to a processor, once it becomes available; otherwise, each processor is allocated a single commodity subproblem. Table 1 and Figure 9 show that the speed-up was improved as the number of commodities and the number of markets increase.

5.2 Decomposition by Time Periods

In this section, the results of computational experience with the parallel algorithm introduced in the Section 2.2.2 on the IBM 3090-600J in a stand-alone environment will be given. These are finite time horizon problems with T time periods in which a commodity is produced at m supply markets, inventoried at the supply markets, and shipped to the consumers at the n demand markets. These problems were generated in the manner explained in Section 4.2.

The motivation for the parallel decomposition algorithm by time periods lies in the special network structure. In particular, recall that the Cartesian product K was defined as the product of two subsets K_1 and K_2, where K_1 contains all

Table 1 Parallel Implementation of the Decomposition Algorithm by Commodities with Linear and Nonlinear Cost Functions

Data Name	Number of Processors	Wall-Clock Time (sec)	Speedup Ratio	Eff. (%)
MSP1	1	128.92	-	-
	2	73.14	1.76	88 %
	3	55.73	2.31	77 %
MSP2	1	124.88	-	-
	2	73.46	1.70	85 %
	3	57.28	2.18	73 %
DMSP1	1	34.73	-	-
	2	19.91	1.74	87 %
	3	16.75	2.07	69 %
DMSP2	1	392.22	-	-
	2	203.42	1.93	96 %
	3	149.41	2.63	88 %
DMSP3	1	33.38	-	-
	2	19.99	1.67	83 %
	3	14.76	2.26	75 %
DMSP4	1	358.11	-	-
	2	194.59	1.84	92 %
	3	134.04	2.67	89 %

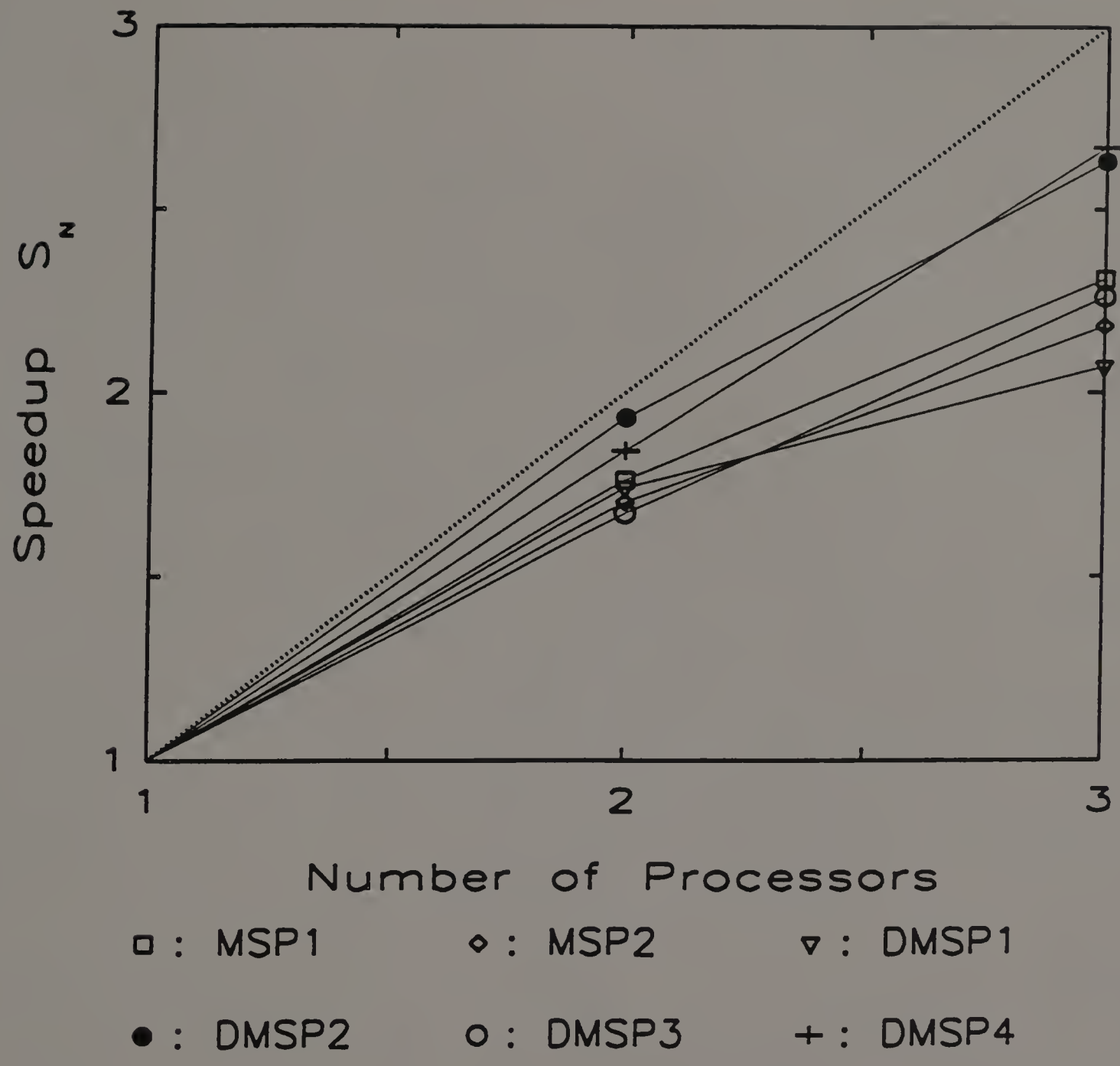


Figure 9 Parallel Speedup of the Decomposition Algorithm by Commodities

the commodity shipment variables (the vertical arcs in Figure 2), whereas K_2 contains all the inventory variables (the horizontal arcs). The constraints require only that these variables be nonnegative. The algorithm then decomposes the problem into $T+1$ subproblems, of the form depicted in Figure 3; the first T problems are static spatial equilibrium problems with a special bipartite network structure in $m \times n$ variables each, for which numerous efficient algorithms exists, whereas the $T+1$-st subproblem, is a very simple inventory problem in $m(T-1)$ variables, which can be solved using a Gauss-Seidel algorithm.

Large-scale dynamic equilibrium problems with linear separable functions and with nonlinear asymmetric functions with five cross-terms were considered. Recall that, in the case of the nonlinear examples, the supply and demand price functions associated with the nodes were quadratic functions, the transportation cost functions associated with the vertical arcs were quartic, and the inventory cost functions associated with the horizontal arcs were linear. Four series of problems were solved, two of which were linear and two of which were nonlinear, with 25 supply markets and 25 demand markets, and 50 supply markets and 50 demand markets, ranging from 5 time periods to 50 time periods. The computations were conducted on the IBM 3090-600J at the Cornell National Supercomputer facility using a FORTRAN code of the algorithm, which was compiled using FORTVS compiler, optimization level 3. Tables 2 - 3 and Figures 10 - 13 depict the CPU behavior of the algorithm when the algorithm is implemented in serial as the number of time periods is increased. The *linear* behavior of the algorithm is to

be contrasted with the behavior of earlier algorithms, which is at least *quadratic* (see Nagurney and Aronson [61]).

The algorithm was embedded with Parallel constructs provided by Parallel FORTRAN and compiled using the Parallel Fortran compiler, optimization level 3. Four example data-sets were proceeded to run in a stand-alone environment on the IBM 3090-600J. The speed-up measures were reported in Table 4 and Figure 14, when the convergence verification was done in serial and after every other iteration. As can be seen from Figure 14, the linear separable problems exhibited substantial speedups, whereas the nonlinear, asymmetric problems, lower speed-ups. This is due, in part, to the serial bottleneck of the convergence verification, which is more time-consuming for the general problems. However, practitioners are concerned with obtaining solutions and, hence, convergence verification is essential. Moreover, these results illustrate a substantial overall savings in elapsed time on extremely large problems. Indeed, prior to this research, the largest problems of this form that had been solved consisted of only 20 supply markets, 20 demand markets, and 10 time periods. (see Nagurney and Aronson [61]).

5.3 Decomposition by Rows/Columns

In this section, the results of computational experience with the Splitting Equilibration Algorithm, which was described in the Section 2.2.3 and Chapter 3 on the IBM 3090-600J, are reported. The algorithm is a general parallelizable procedure which decomposes a wide spectrum of constrained matrix problems into

Table 2 Serial Implementation of the Decomposition Algorithm by Time Periods on Large-Scale Dynamic Market Equilibrium Problems with Linear Separable Cost Functions

Number of Markets	Number of Time Periods	CPU Time (seconds)	Number of Iterations	Percentage of Positive Variables at Solution Inventory	Transshipment
5 x 5	5	0.1005	36	60.0	73.6
	10	0.2114	38	68.9	66.4
	25	0.5797	46	72.5	71.7
	50	1.2371	54	72.2	74.2
10 x 10	5	0.7583	62	75.0	63.6
	10	1.4543	62	66.7	67.2
	25	4.2041	76	65.0	72.3
	50	7.7656	70	69.2	67.6
25 x 25	5	9.0560	62	40.0	55.6
	10	20.8145	70	32.0	60.4
	25	80.3637	116	62.2	60.9
	50	161.2377	110	59.0	62.7
50 x 50	5	88.2942	104	56.5	55.6
	10	188.6683	122	56.9	52.7
	25	470.3525	118	59.0	53.9
	50	984.2568	126	58.9	53.9

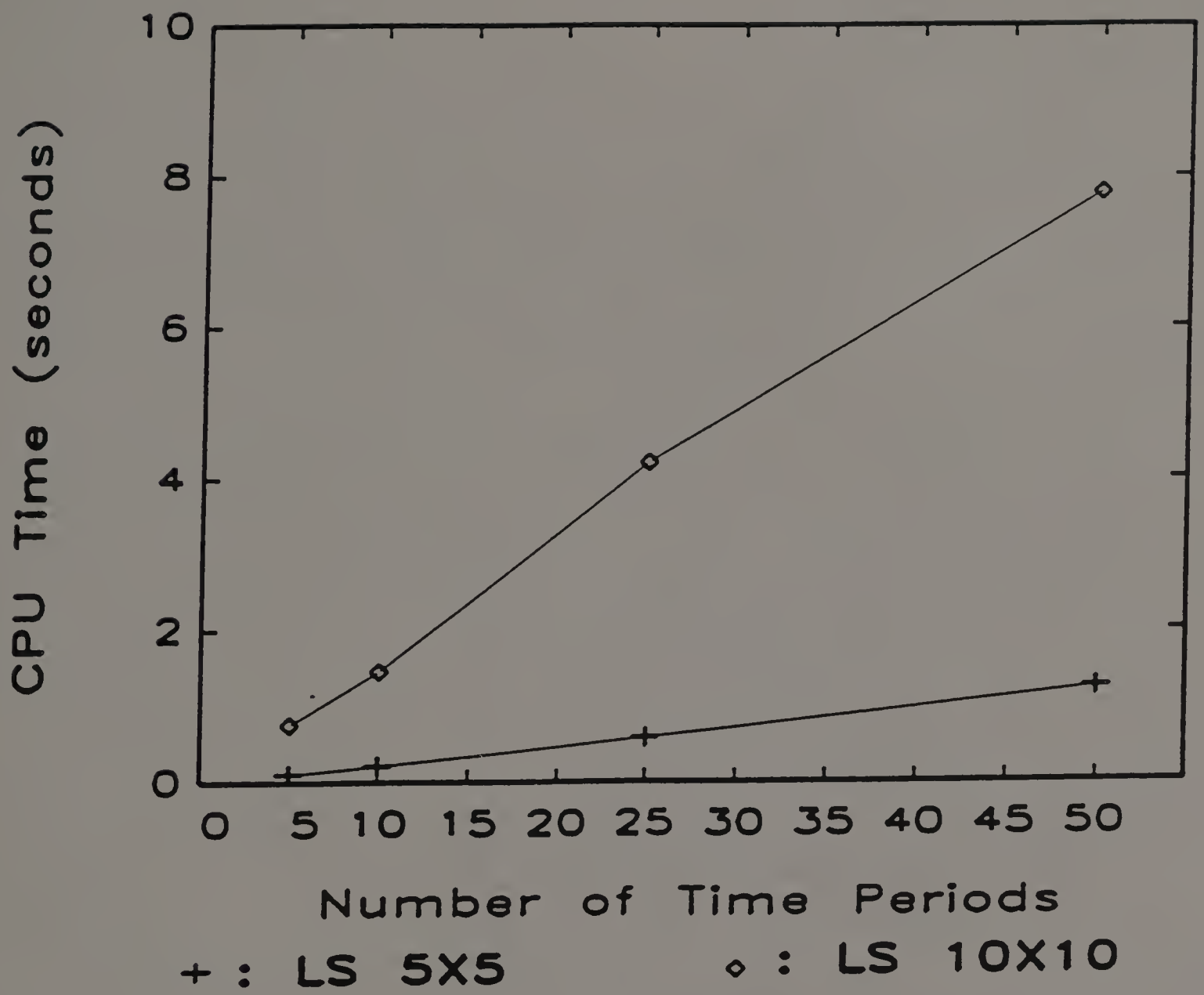


Figure 10 CPU Behavior of the Decomposition Algorithm by Time Periods for Smaller Linear Separable Examples

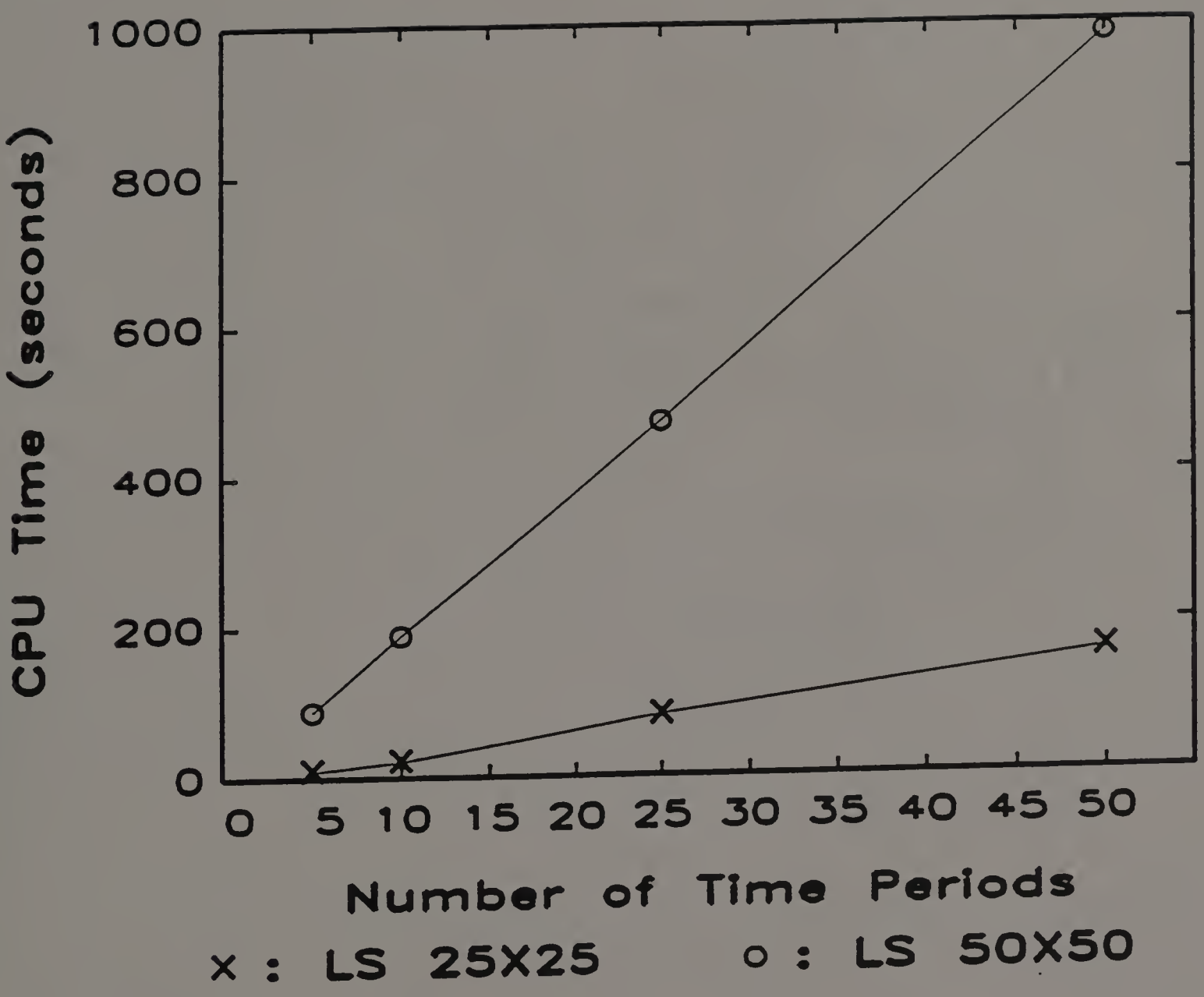


Figure 11 CPU Behavior of the Decomposition Algorithm by Time Periods for Larger Linear Separable Examples

Table 3 Serial Implementation of the Decomposition Algorithm by Time Periods on Large-Scale Dynamic Market Equilibrium Problems with Nonlinear Asymmetric Cost Functions

Number of Markets	Number of Time Periods	CPU Time (seconds)	Number of Iterations	Percentage of Positive Variables at Solution Inventory	Transshipment
5 x 5	5	0.1385	28	65.0	76.8
	10	0.2665	26	55.6	70.8
	25	0.9402	38	60.8	78.2
	50	1.9524	38	71.0	76.2
10 x 10	5	1.3423	50	60.0	62.8
	10	2.4266	46	65.6	66.9
	25	7.9535	60	68.8	72.0
	50	15.3436	60	64.7	68.2
25 x 25	5	19.3264	76	72.0	56.0
	10	50.7079	98	57.3	64.8
	25	119.2013	98	56.0	62.0
	50	271.3679	114	62.4	60.7
50 x 50	5	150.8648	114	74.5	57.6
	10	282.5073	114	47.3	53.8
	25	754.5261	118	57.0	53.1
	50	1427.2461	122	58.2	53.7

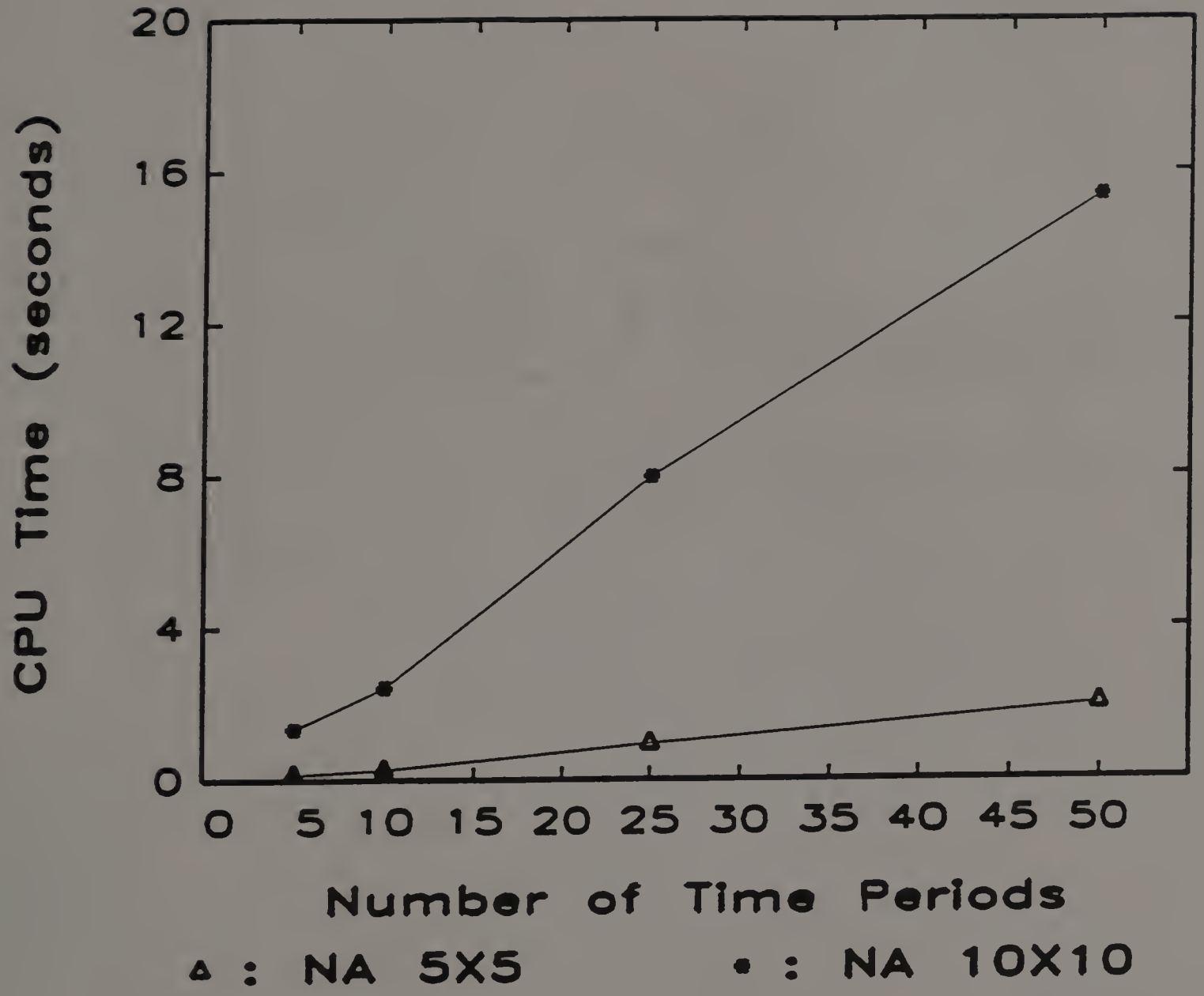


Figure 12 CPU Behavior of the Decomposition Algorithm by Time Periods for Smaller Nonlinear Asymmetric Examples

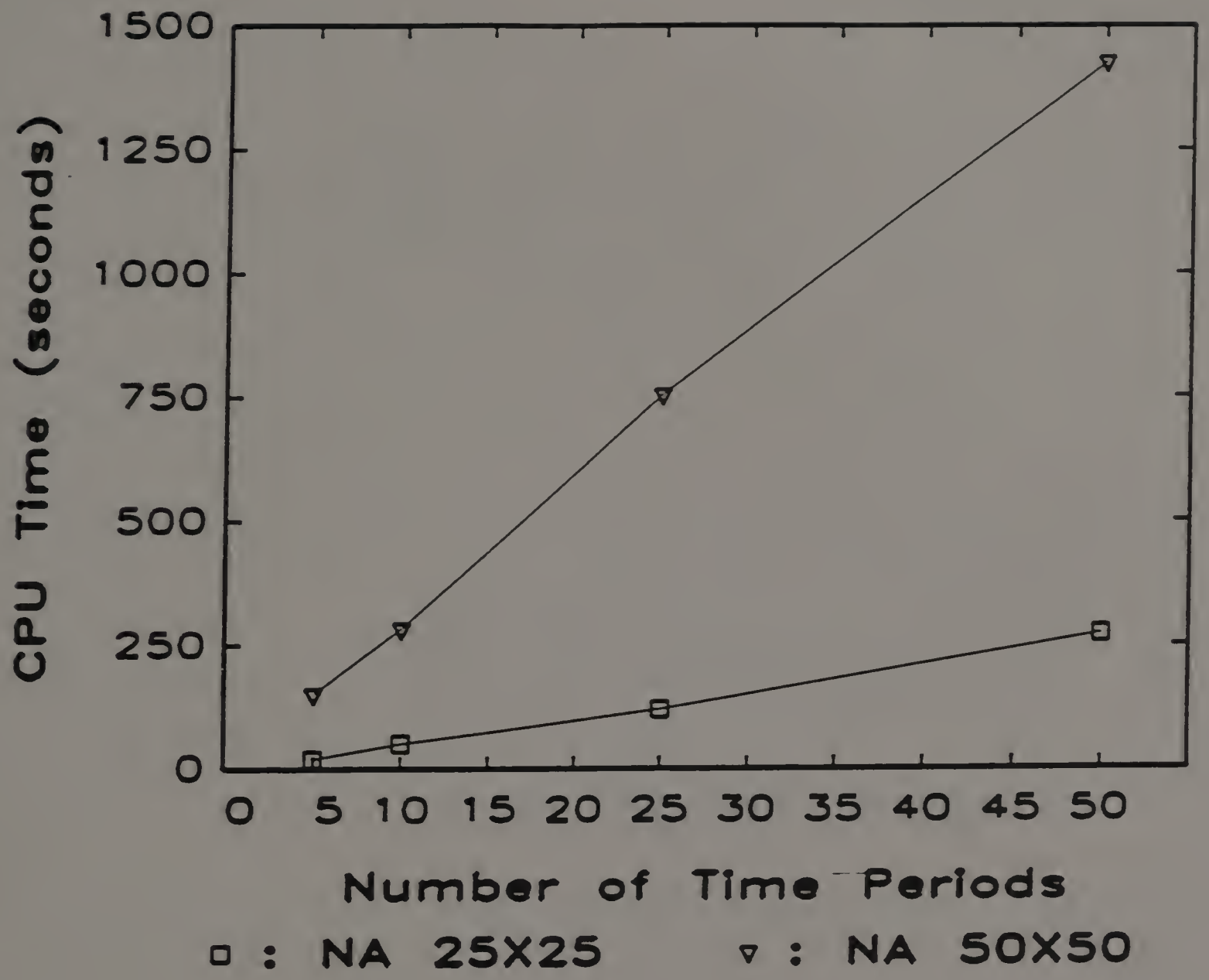


Figure 13 CPU Behavior of the Decomposition Algorithm by Time Periods for Larger Nonlinear Asymmetric Examples

Table 4 Parallel Implementation of the Decomposition Algorithm by Time Periods

(a) Linear Separable Case

Number of Markets	Number of Time Periods	Number of Processors	Speedup S_N	Efficiency E_N (percentage)
50 x 50	25	2	1.58	79.12
		4	2.59	64.67
		6	3.31	55.21
50 x 50	50	2	1.60	80.11
		4	2.68	67.02
		6	3.51	58.46

(b) Nonlinear Asymmetric Case

Number of Markets	Number of Time Periods	Number of Processors	Speedup S_N	Efficiency E_N (percentage)
50 x 50	25	2	1.498	74.90
		4	2.271	56.77
		6	2.717	45.28
50 x 50	50	2	1.498	74.91
		4	2.322	58.05
		6	2.823	47.06

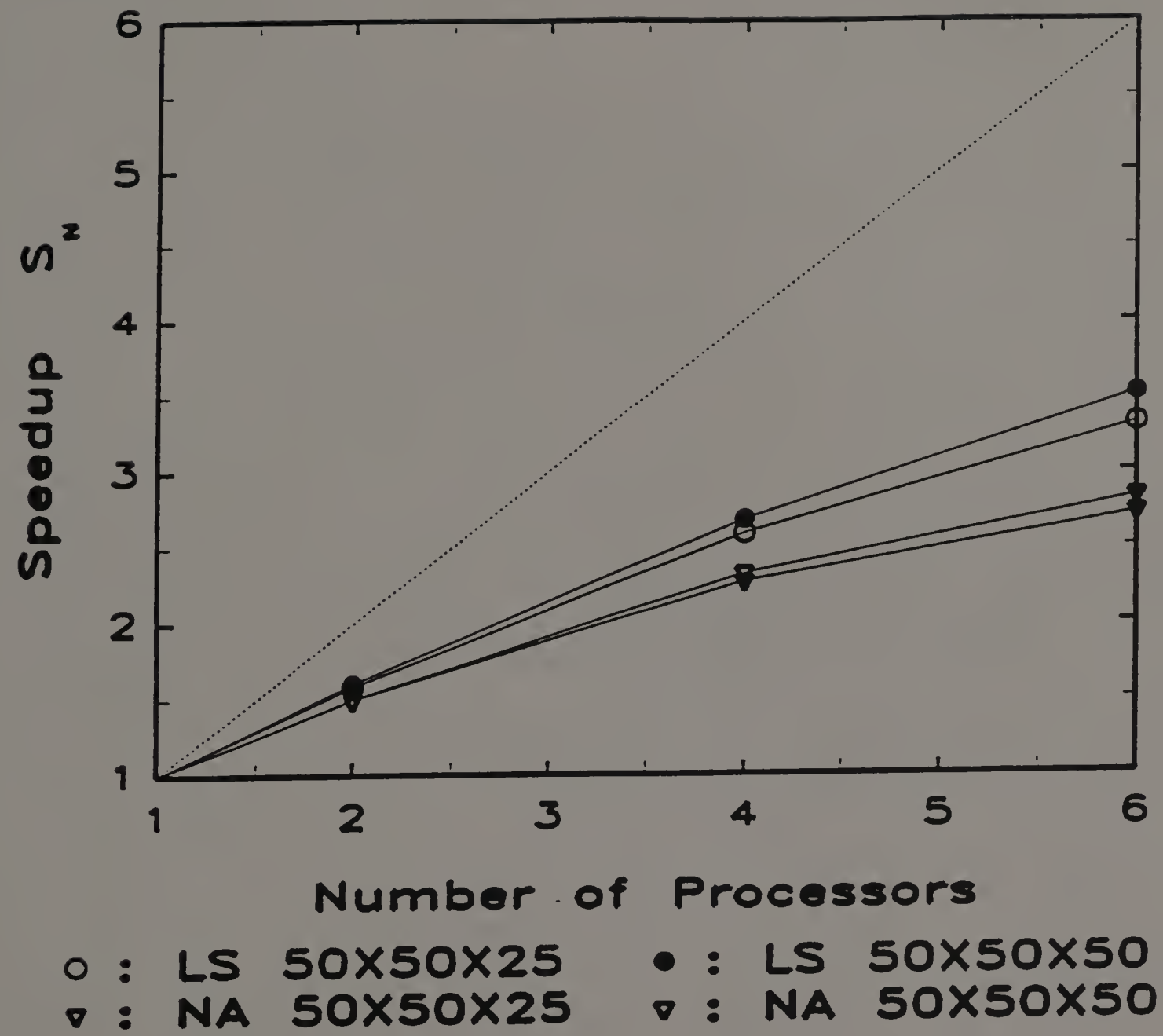


Figure 14 Parallel Speedup of the Decomposition Algorithm by Time Periods

a series of row/column equilibration problems, each of which can be solved exactly in closed form and allocated to a distinct processor. SEA splits the constraints which are of transportation type so that the objective function is considered subject to either the row constraints, or the column constraints. Its theoretical analysis is based on its interpretation as a dual method. The selected problem, FT1000 $\times$ 1000 for the fixed row/column totals had a size of the X^0 matrix of 1000 $\times$ 1000, with the corresponding G matrix of dimension $10^6 \times 10^6$. SEA was also applied to compute the solution to input/output matrix, IO72. Since these two data-sets were for the fixed row and column totals, the convergence criterion was based on the relative residuals $R(s_i) = (\sum_j x_{ij} - s_i)/s_i$ and $R(d_j) = (\sum_i x_{ij} - d_j)/d_j$, with each residual required to be less than .01.

For the elastic row/column totals, two data-sets, SPEP500 $\times$ 500 and SPEP750 $\times$ 750, were generated in the same manner as described in Chapter 4. The convergence tolerance was $\epsilon = .01$.

Figure 15 and Table 5 show the speed-ups obtained for SEA on four examples; first two examples, IO72 and FT1000 $\times$ 1000, were for the fixed row and column totals, and next two examples, SPEP500 $\times$ 500 and SPEP750 $\times$ 750, were for the unknown row and column totals. These data-sets were tested on the IBM 3090-600E in a standalone environment. Maximum speed-ups obtained were 1.93 for two processors, 3.74 for four processors, and 5.15 for six processors when IO72 was used. The minimum speed-ups were 1.87 for two processors, 3.19 for four processors, and 3.86 for six processors, when SPEP750 $\times$ 750 was used.

Table 5 Coarse Grain Parallel Implementation of SEA

Data Name	Number of Processors	Wall-Clock Time(sec.)	Speedup Ratio	Eff. (%)
IO72	1	438.52	-	-
	2	227.12	1.93	96 %
	4	117.21	3.74	93 %
	6	85.12	5.15	86 %
FT 1000X1000	1	483.20	-	-
	2	250.37	1.93	96 %
	4	135.35	3.57	89 %
	6	102.59	4.71	78 %
SPEP 500X500	1	540.71	-	-
	2	290.70	1.86	93 %
	4	153.61	3.52	88 %
	6	116.03	4.66	78 %
SPEP 750X750	1	1589.06	-	-
	2	849.76	1.87	94 %
	4	498.14	3.19	80 %
	6	411.67	3.86	64 %

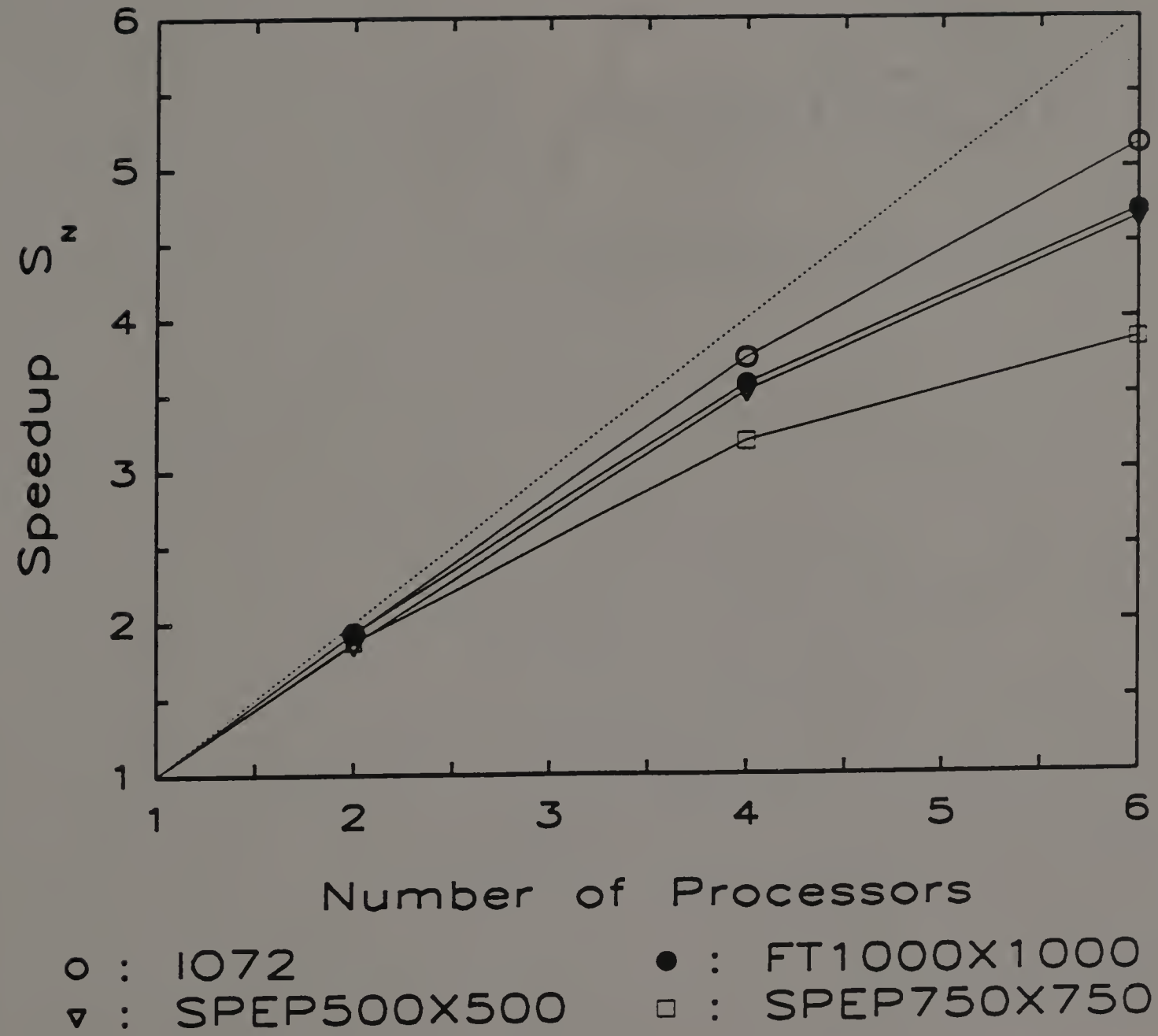


Figure 15 Parallel Speedup of SEA Decomposition Algorithm by Row/Column

5.4 Fine Grain Parallel Implementation of the Massively Parallel SEA Algorithm

In this section, the results of the computations on the CM-2 system for a data-set based on an input/output matrix, IO72, consisting of 485 rows and 485 columns and representing a data-set of a 1972 input/output matrix for the United States are presented. This problem consisted of 23,225 variables. The problem was solved using 8K (8,192) processors, 16K (16,384) processors, and finally, 32K (32,768) processors. Table 6 and Figure 16 show the relationship between the CM CPU time and the number of processors. Observe that the CM CPU time decreases approximately linearly as the number of processors is increased. I note that the same problem was solved on an IBM 3090-600E and required 438.35 CPU seconds for the serial FORTRAN code, compiled using the FORTVS compiler, optimization level 3, and 291.54 seconds on an IBM 3090-600J. The number of iterations required for the convergence was 4 for SEA both on the CM-2 and on the IBM 3090-600. In terms of the parallel runs on the IBM 3090-600E, the wall clock time required for the convergence of the parallel implementation of the Splitting Equilibration Algorithm, compiled using the PF compiler, was 444.18 seconds for one processor, 229.85 seconds for two processors, 118.76 seconds for four processors, and 86.32 seconds for six processors.

The second data-set, D512, was a randomly generated data-set and consisted of 512 rows and columns, and had 262,144 variables. The results of its solution on the CM-2 are reported in Table 10. The problem was solved with 8K and 16K

Table 6 Fine-Grain Parallel Implementation of Massively Parallel SEA

Data Name	Number of Processors	Real Time (sec.)	CM Time (sec.)
IOT2	8K	52.05	51.74
	16K	29.86	29.58
	32K	16.76	16.34
D512	8K	10.92	10.90
	16K	6.27	6.26
IOT7	8K	5.38	5.36
	16K	3.25	2.82
USDA	8K	19.49	19.43
	16K	9.80	9.73

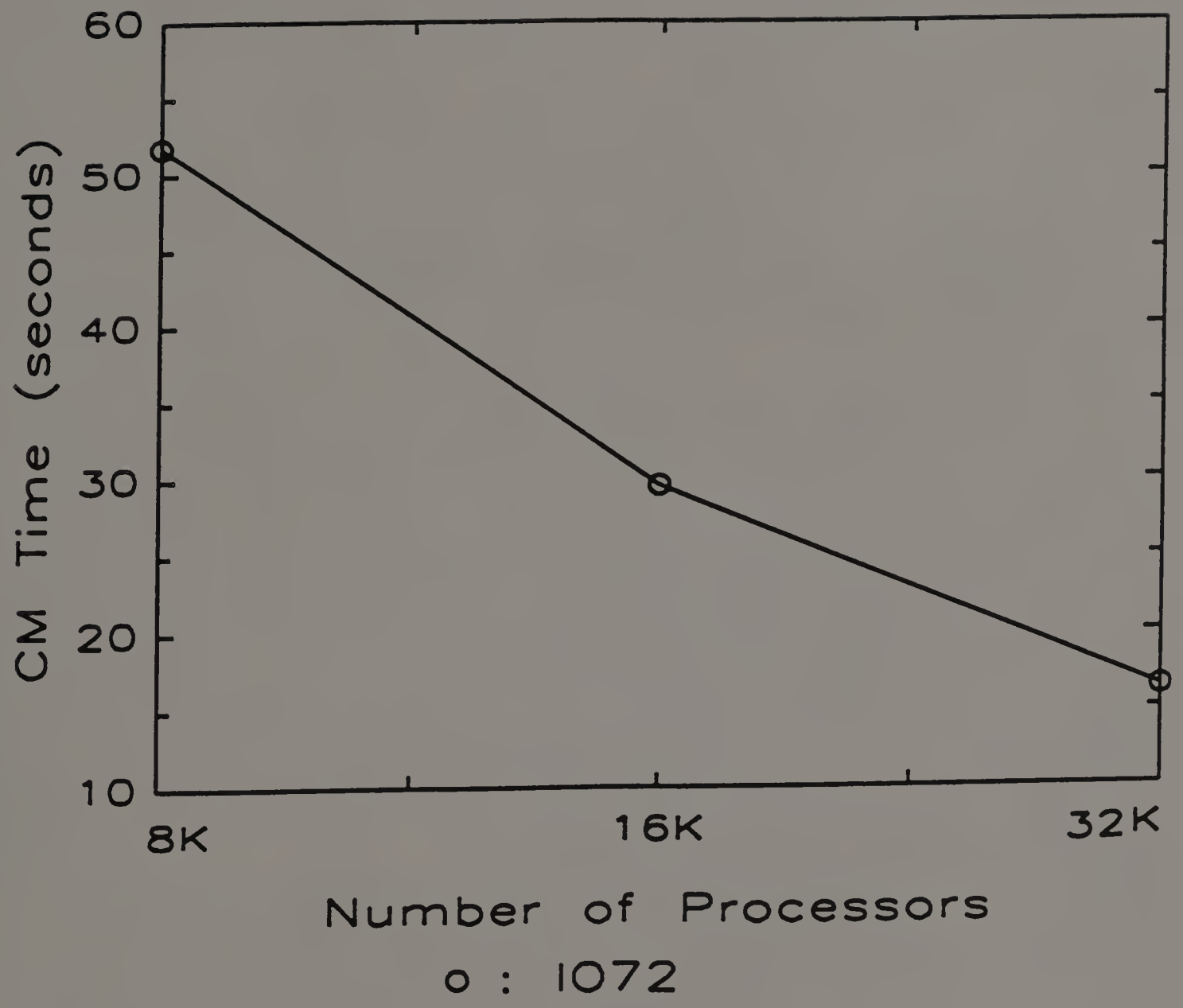


Figure 16 CM Time vs Number of Processors

processors. The same problem required 150.99 CPU seconds on an IBM 3090-600J, compiled using the FORTVS compiler, optimization level 3. The number of iterations for convergence was one for SEA on each of the two architectures.

The third example was an input/output matrix, IO77, consisting of 205 rows and 205 columns, and was based on a 1977 input/output matrix for the US. This problem had 42,025 variables. The results for this problem are reported in Table 9. The same problem required 19.37 CPU seconds for the convergence on the IBM 3090-600J. The number of iterations was two on both architectures.

The final problem, USDA133, was based on a social accounting matrix for the US, consisting of 133 rows and 133 columns, (see Hanson and Robinson [34]). This example had 17,689 variables. The results on the CM-2 machine for this problem are reported in Table 6. The same problem required 17.67 CPU seconds for the convergence on the IBM 3090-600J. The number of iterations required for the convergence was eight on the CM-2 and also on the IBM 3090-600J.

Note that SEA exhibits approximately linear speed-ups, as predicted by the theory in Nagurney and Eydeland [63]. The numerical results strongly suggest that the implementation on the CM-2 using CM FORTRAN is very promising. I believe that further enhancements to the language will make even more efficient implementations realizable.

CHAPTER 6

SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH

In this chapter, the thesis is summarized and directions for future research in the parallel computation of network problems are given.

A new spatial price equilibrium model, which is not based on path flows, but, rather, on link flows, was developed for the full range of parallel decomposition. Variational inequality formulations of this model were derived and the qualitative properties were discussed in Chapter 2. Existence of a unique equilibrium solution is guaranteed under the assumption of strong monotonicity of supply price, demand price, tranaction cost, and inventorying cost functions. The data-sets tested were generated under the strong monotonicity assumption using random number generation, and three data-sets were collected from input/output matrices and social accounting matrices for the United States.

For the parallel implementation, the variational inequalities were decomposed first by commodities, then by time periods, then by markets. The coarse grain parallel architectures used were the IBM 3090-600E and the IBM 3090-600J at the Cornell Theory Center. The maximum speed-ups of the decomposition algorithm by commodities were 1.93 for two processors, and 2.67 for three processors on the

IBM 3090-600E under the Strategic Users' Program. The maximum speed-ups of the decomposition algorithm by time periods were 1.60 for two processors, 2.68 for four processors, and 3.51 for six processors on IBM 3090-600J under a stand-alone environment. The maximum speed-ups of the decomposition algorithm by markets or row/columns were 1.93 for two processors, 3.74 for four processors, and 5.15 for six processors.

The row/column equilibration problems were further decomposed by links, resulting in a fine grain parallel implementation. The Thinking Machine's Connection Machine, CM-2 was used for the numerical experimentation. Input/output matrix problems with 485 rows and 485 columns were solved in 16.76 seconds with 32 K processors, and a spatial price equilibrium problem with 512 supply markets and 512 demand markets with linear separable cost functions was solved in 6.27 seconds with 16 K processors. Time required to solve the same problems on the IBM 3090-600J using one processor was 444.18 CPU seconds and 150.99 CPU seconds, respectively.

Several interesting research directions can be suggested based on the results of this thesis. The algorithms developed in this thesis were *synchronous*, so all processors must wait for all other processors to finish the current calculations to share new information at the end of each iteration. *Asynchronous* parallel algorithms, which do not need to wait at the end of each iteration, can be developed and tested to solve network problems. Before the parallel implementation, however, convergence of the asynchronous parallel algorithms needs to be established mathematically.

In the fine grain parallel implementation, linear separable cost functions were tested. Since all coefficients of the cost functions, which may be nonlinear and *asymmetric* in practice, are stored at each processor, each processor needs to communicate with other processors which are located far away. In future research, to solve this problem efficiently, the programmer should minimize the communication cost in the linearization step.

Similar approaches can be successfully applied to solve other large-scale problems with underlying network structure, such as traffic equilibrium problems, general economic equilibrium problems, and financial equilibrium problems. It is expected that in the future the theory of variational inequalities along with parallel computation will continue to yield very effective algorithms for solving large-scale equilibrium and network problems.

BIBLIOGRAPHY

[1] Aashtiani,M.Z. and Magnanti,T.L., Equilibria on a congested transportation network, *SIAM Journal of Algebraic Discrete Methods* **2**, pp.213-226, 1981.

[2] Aashtiani,M.Z. and Magnanti,T.L., A linearization and decomposition algorithm for computing urban traffic equilibria, *Proceedings of the 1982 IEEE International Large Scale Systems Symposium*, pp.8-19, 1982.

[3] Baase,S., *Computer Algorithms - Introduction to Design and Analysis*, 2nd ed., Addison-Wesley Publishing Company, pp.362-386, 1988.

[4] Bacharach,M., *Biproportional Scaling and Input-Output Change*, Cambridge University, UK, 1970.

[5] Bachem,A. and Korte,B., Algorithm for quadratic optimization over transportation polytopes, *Zeitschrift fur Angewandte Mathematik und Mechanik* **58**, pp.459-461, 1978.

[6] Bachem,A. and Korte,B., Minimum norm problems over transportation polytopes, *Linear Algebra Application* **31**, pp.102-118, 1980.

[7] Beckmann,M.J., McGuire,C.B., and Winsten,C.B. *Studies in the Economics of Transportation*, New Haven, Conn., Yale University Press, 1956.

[8] Bertsekas,D.P. and Gafni,E., Projection methods for variational inequalities and the application to the traffic assignment problem, *Mathematical Programming Studies* **17**, pp.139-159, 1982.

[9] Bertsekas,D.P. and Tsitsiklis,J.N., *Parallel and Distributed Computation, Numerical Methods*, Prentice Hall, pp.264-292, 1989.

[10] Border,K.C., *Fixed Point Theorems with Applications to Economics and Game Theory*, Cambridge, U.K.: Cambridge University Press, 1985.

[11] Byron,R.P., The estimation of large social accounts matrices, *Journal of the Royal Statistical Society Series A* **141**, pp.359-369, 1978.

[12] Cottle,R.W., Complementarity and variational inequality problems, *Symposia Mathematica* **19**, pp.59-70, 1974

[13] Cottle,R.W., Duvall,S.G., and Zikan,K., A lagrangian relaxation algorithm for the constrained matrix problem, *Naval Research Logistics Quarterly* **33**, pp.55-76, 1986.

[14] Dafermos,S., The traffic assignment model with applications to two-way traffic. *Transportation Science* 6 pp.73-87, 1972.

[15] Dafermos,S., Traffic equilibrium and variational inequalities, *Transportation Science* **14**, pp.42-54, 1980.

[16] Dafermos,S., The general multimodal network equilibrium problem with elastic demand, *Networks* **12**, pp.57-72, 1982.

[17] Dafermos,S., An iterative scheme for variational inequalities, *Mathematical Programming* **28**, pp.57-72, 1983.

[18] Dafermos,S., Isomorphic multiclass spatial price and multimodal traffic network equilibrium models, *Regional Science and Urban Economics* **16**, pp. 197-209, 1986.

[19] Dafermos,S., Congested transportation networks and variational inequalities, In *Flow Control of Congested Networks (NATO ASI series), series F: Computer and Systems Sciences* **38**, edited by Odoni, Bianco, and Szego, New York: Springer-Verlag, 1987.

[20] Dafermos,S., Exchange price equilibria and variational inequalities, *Mathematical Programming* **46**, pp.391-402, 1990.

[21] Dafermos,S. and Nagurney,A., Sensitivity analysis for the general spatial economic equilibrium problem, *Operations Research* **32**, pp.1069-1086, 1984.

[22] Dafermos,S. and Nagurney,A., Oligopolistic and competitive behavior of spatially separated markets, *Regional Science and Urban Economics* **17**, pp.245-254, 1987.

[23] Dafermos,S. and Nagurney,A., Supply and demand equilibration algorithms for a class of market equilibrium problems, *Transportation Science* **23**, pp.118-124, 1989.

[24] Deming,W.E. and Stephen,F.F., On a least-squares adjustment of a sampled frequency table when the expected marginal totals are known, *Annals of Mathematical Statistics* **11**, pp.427-444, 1940.

[25] Dervis,K., De Melo,J. and Robinson,S., *General Equilibrium Models for Development Policy*, Cambridge University Press, Cambridge, United Kingdom, 1982.

[26] Erickson,J., On solving linearly constrained maximum entropy problems, *Mathematical Programming* **18**, pp.146-154, 1980.

[27] Erlander,S., Jornsten,K.O. and Lundgren,J.T., On the estimation of trip matrices in the case of missing and uncertain data, *Transportation Research* **19b(2)**, pp.123-141, 1985.

[28] Eydeland,A. and Nagurney,A., Progressive equilibration algorithms: the case of linear transaction costs, *Computer Science in Economics and Management* 2, pp.197-219, 1989.

[29] Florian,M., Nonlinear cost network models in transportation analysis, *Mathematical Programming Study* **26**, pp.167-196, 1986.

[30] Florian,M. and Los,M., A new look at static spatial price equilibrium models, *Regional Science and Urban Economics* **12**, pp.579-597, 1982.

[31] Florian,M. and Spiess,H., The convergence of diagonalization algorithms for asymmetric network equilibrium problems, *Transportation Science*, **16**, pp.477-483, 1982.

[32] Friedlander,D., A technique for estimating a contingency table given the marginal total and some supplementary data, *Journal of the Royal Statistical Society A* **124**, pp.412-420, 1961.

[33] Friesz,T.L., Tobin,R.L. and Harker,P.T., A nonlinear complementarity formulation and solution procedure for the general derived demand network equilibrium, *Journal of Regional Science* **23**, pp.337-359, 1983.

[34] Hanson,K. and Robinson,S., Data, linkages, and models: U.S. national income and product accounts in the framework of a social accounting matrix, Agriculture Service, U.S.Department of Agriculture, Staff Report No.AGES 89-5, 1989.

[35] Harrigan,F. and Buchanan,I., Quadratic programming approach to input-output estimation and simulation, *Journal of Regional Science* **24(3)**, pp.339-358, 1984.

[36] Hartmann,P. and Stampacchia,G., On some nonlinear elliptic differential functional equations, *Acta Math.*, pp.271-310, 1966.

[37] Hildreth,C., A quadratic programming procedure, *Naval Research Logistics Quarterly* 4, pp.79-85, 1957.

[38] Johnston,R.J., Hay,A.M. and Taylor,P.J., Estimating the sources of spatial change in election results: a multiproportional approach, *Environment and Planning A* **14**, pp.951-961, 1982.

[39] Judge,G.G. and Takayama,T., *Studies in Economic Planning over Space and Time*, North-Holland, Amsterdam, 1973.

[40] Judge,G.G. and Yancey,T.A., *Improved Methods of Inference and Extensions*, Prentice-Hall, Englewood Cliffs, N.J., 1985.

[41] Kinderlehrer,D. and Stampacchia,G., *An Introduction to Variational Inequalities*, Academic Press,New York, 1980.

[42] Klincewicz,J., Implementing an exact Newton method for separable convex transportation problems, *Networks* **19**, pp.95-105, 1989.

[43] Labys,W.C., Takayama,T. and Uri,N.D., *Quantitative Methods for Market-Oriented Economic Analysis over Space and Time*, Evebury, Gower Publishing Co., England, 1989.

[44] Lemke,C.E., A survey of complementarity theory, in *Variational Inequalities and Complementarity Problems*, chapter 15, edited by Cottle, Giannessi, and Lions, New York, Wiley, 1980.

[45] Magnanti,T., Models and algorithms for predicting urban traffic equilibrium, In *Transportation Planning Models*, edited by M. Florian, Amsterdam, North-Holland, 1984.

[46] Mahmassani,H.S. and Mouskos,K.C., Some numerical results on the diagonalization algorithm for network assignment with asymmetric interactions between cars and trucks, *Transportation Research* **22**, pp.275-290, 1988.

[47] Mathiesen,L., Computational experience in solving equilibrium models by a sequence of linear complementarity problems, *Operations Research* **33**, pp.1225-1250, 1985.

[48] Miller,R.E. and Blair,P.D., *Input/Output Analysis: Foundations and Extensions*, Englewood Cliffs, N.J., Prentice-Hall, 1985.

[49] Mohr,M. Crown,W.H. and Polenske,K.E., A linear programming approach to solving infeasible RAS problems, *Journal of Regional Science* **27**, No. 4, pp.587-603, 1987.

[50] Nagurney,A., Comparative tests of multimodal traffic equilibrium methods, *Transportation Research* **18B**, pp.469-485, 1984.

[51] Nagurney,A., Computational comparisons of algorithms for general asymmetric traffic equilibrium problems with fixed and elastic demands, *Transportation Research* **20B**, pp.78-84, 1986.

[52] Nagurney,A., Computational comparisons of spatial price equilibrium methods, *Journal of Regional Science* **27**, pp.55-76, 1987.

[53] Nagurney,A., Competitive equilibrium problems, variational inequalities, and regional science, *Journal of Regional Science* **27**, pp.503-517, 1987.

[54] Nagurney,A., Algorithms for oligopolistic market equilibrium problems, *Regional Science and Urban Economics* **18**, pp.425-445, 1988.

[55] Nagurney,A., Import and export equilibration algorithms for the net import spatial price equilibrium problem, *Journal of Cost Analysis* **7**, pp.73-88, 1989.

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

[57] Nagurney,A., Migration equilibrium and variational inequalities, *Economic Letters*, **31**, pp.109-112, 1989.

[58] Nagurney,A., The formulation and solution of large-scale multicommodity equilibrium problems over space and time, *European Journal of Operational Research* **42**, pp.166-177, 1989.

[59] Nagurney,A., 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**, pp.99-104, 1989.

[60] Nagurney,A., Nonlinear networks in the social and economic sciences. *Forefronts* **2**, pp.3-7, Cornell National Supercomputer Facility, Cornell University, Ithaca, NY, 1990.

[61] Nagurney,A. and Aronson,J., A general dynamic spatial price equilibrium model: Formulation, solution, and computational results. *Journal of Computational and Applied Mathematics* **22**, pp.359-377, 1988.

[62] Nagurney,A. and Aronson,J., A general dynamic spatial price network equilibrium model with gains and losses. *Networks* **19**, pp.751-769, 1989.

[63] Nagurney,A. and Eydeland,A., A Splitting Equilibration Algorithm for the computation of large-scale constrained matrix problems: Theoretical analysis and applications, *Advanced Studies in Theoretical and Applied Econometrics* **22**, Kluwer Publishing Co., pp. 65-105, 1992.

[64] Nagurney,A., Eydeland,A. and Kim,D. Computation of large-scale constrained matrix problems: the Splitting Equilibration Algorithm, *Proceedings of Supercomputing* '90, 1990.

[65] Nagurney,A. and Kim,D., Parallel and serial variational inequality decomposition algorithms for multicommodity market equilibrium problems, *The International Journal of Supercomputer Applications* **3**, pp.34-59, 1989.

[66] Nagurney,A., Kim,D. and Robinson,A., 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**, pp.49-71, 1990.

[67] Nagurney,A. and Robinson,A., Algorithms for quadratic constrained matrix problems, *Mathematical and Computer Modeling*, in press.

[68] Nagurney,A. and Zhao,L., A network equilibrium formulation of market disequilibrium and variational inequalities, *Networks* **21**, pp.109-132, 1991.

[69] Ohuchi,A. and Kaji,I., An algorithm for the Hitchcock transportation problem with quadratic cost functions, *Journal of Operational Research Society Japan* **24**, pp.170-181, 1981.

[70] Ohuchi,A. and Kaji,I., Lagrangian dual coordinatewise maximization for network transportation problems with quadratic costs, *Networks* **14**, pp.525-530, 1984.

[71] Pang,J.S., A hybrid method for the solution of some multicommodity spatial equilibrium problems, *Management Science* **27**, pp.1142-1157, 1981.

[72] Pang,J.S., Solution of the general multicommodity spatial equilibrium problem by variational and complementarity methods, *Journal of Regional Science* **24**, pp.403-414, 1984.

[73] Pang,J.S., Asymmetric variational inequality problems over product sets: applications and iterative methods, *Mathematical Programming* **31**, pp.206-219, 1985.

[74] Polito,J., McCarl,B.A. and Morin,T.L., Solution of spatial equilibrium problems with Bender's decomposition, *Management Science* **26**, pp.509-605, 1980.

[75] Pyatt,G. and Round,J.I., *Social Accounting Matrices: a Basis for Planning*, The World Bank, Washington, D.C., 1985.

[76] Riele,H.J,. Dekker,Th.J. and Van Der Vorst,H.A., *Special Topic in Supercomputing* vol. 3, "Algorithms and Applications on Vector and Parallel Computers", Elsevier Science Publishers B.V., North-Holland, 1987.

[77] Samuelson,P.A., A spatial price equilibrium and linear programming, *American Economic Review* **42**, pp.283-303, 1952.

[78] Samuelson,P.A., Intertemporal price equilibrium: A prologue to the theory of speculation. *Weltwirschaftliches Archiv* **79**, pp.181-219, 1957.

[79] Smith,M., An algorithm for solving asymmetric equilibrium problems with a continuous cost-flow function, *Transportation Research* **17B**, pp.365-371, 1979.

[80] Stone,J., Formulation and solution of economic equilibrium problems, Systems Optimization Laboratory Technical Report 88-7, Stanford University, Department of Operations Research, 1988.

[81] Takayama,T., Hashimoto,H. and Uri,N.P., Spatial and temporal price and allocation modeling: Some extensions, *Socio-Economic Planning Science* **18**, pp.227-234, 1984.

[82] Takayama,T. and Judge,G.G., *Spatial and Temporal Price and Allocation Models*, North-Holland, Amsterdam, 1971.

[83] Takayama,T. and Uri,N.P., A note on spatial and temporal price and allocation modeling. *Regional Science and Urban Economics* **13**, pp.455-470, 1983.

[84] Theise,E. and Jones,P.C., Alternative implementations of a diagonalization algorithm for multicommodity spatial price equilibria, Technical Report 88-06, Northwestern University, Department of Industrial Engineering and Management Sciences, 1988.

[85] Thore,S., Spatial disequilibrium, *Journal of Regional Science*, **26**, pp.660-675, 1986.

[86] Van der Ploeg,F., Balancing large systems of national accounts, *Computer Science in Economics and Management* **1**, pp.31-39, 1988.

[87] Van der Sluis,A., Condition numbers and equilibration of matrices, *Numerische Mathematik* **14**, pp.14-23, 1969.

[88] Zhao,L., Variational inequalities in general equilibrium: analysis and computation, Ph.D. dissertation, Brown University, Division of Applied Mathematics, 1989.