March 1987 Report LIDS - P - 1653

Revised September 1987

THE AUCTION ALGORITHM: A DISTRIBUTED RELAXATION METHOD FOR THE ASSIGNMENT PROBLEM *

by

Dimitri P. Bertsekas

Laboratory for Information and Decision Systems

Massachusetts Institute of Technology

Cambridge, Mass. 02139

Abstract

We propose a massively parallelizable algorithm for the classical assignment problem. The algorithm operates like an auction whereby unassigned persons bid simultaneously for objects thereby raising their prices. Once all bids are in, objects are awarded to the highest bidder. The algorithm can also be interpreted as a Jacobi - like relaxation method for solving a dual problem. Its (sequential) worst - case complexity, for a particular implementation that uses scaling, is O(NAlog(NC)) where N is the number of persons, A is the number of pairs of persons and objects that can be assigned to each other, and C is the maximum absolute object value. Computational results show that, for large problems, the algorithm is competitive with existing methods even without the benefit of parallelism. When executed on a parallel machine, the algorithm exhibits substantial speedup.

* Work supported by Grant NSF-ECS-8217668. Thanks are due to J. Kennington and L. Hatay of Southern Methodist Univ. for contributing some of their computational experience.

1. Introduction

Relaxation methods for optimal network flow problems resemble classical coordinate descent, Jacobi, and Gauss-Seidel methods for solving unconstrained nonlinear optimization problems or systems of nonlinear equations. They operate on a dual problem which is unconstrained and involves a dual variable for every node (also called a node price). In their pure form they modify the node prices one at a time using only local node information while aiming to improve the dual cost. They are well suited for distributed implementation on massively parallel machines.

For problems with strictly convex arc costs relaxation methods can be shown to converge to the correct solution under mild additional assumptions. An important property of strictly convex cost problems is that the dual cost function is differentiable. This facilitates the use of the relaxation idea because at any nonoptimal dual vector one can find a single node price coordinate along which the dual cost can be improved. However, the convergence properties of the method do not follow from classical results of unconstrained optimization because the dual cost function is not strictly convex and does not have bounded level sets. Nonetheless there is sufficient structure to guarantee convergence, which can also be shown for a totally asynchronous implementation where price updates at each node are carried out asynchronously with out-of-date price information from neighboring nodes [1], [2]. Computational experiments with parallel relaxation methods applied to strictly convex arc cost problems have been very encouraging [3], [4].

By contrast, for problems where the arc costs are linear the dual cost function is nondifferentiable (piecewise linear). As a result the relaxation idea may encounter difficulty at points where the dual cost cannot be improved in any coordinate direction as seen in Fig. 1.

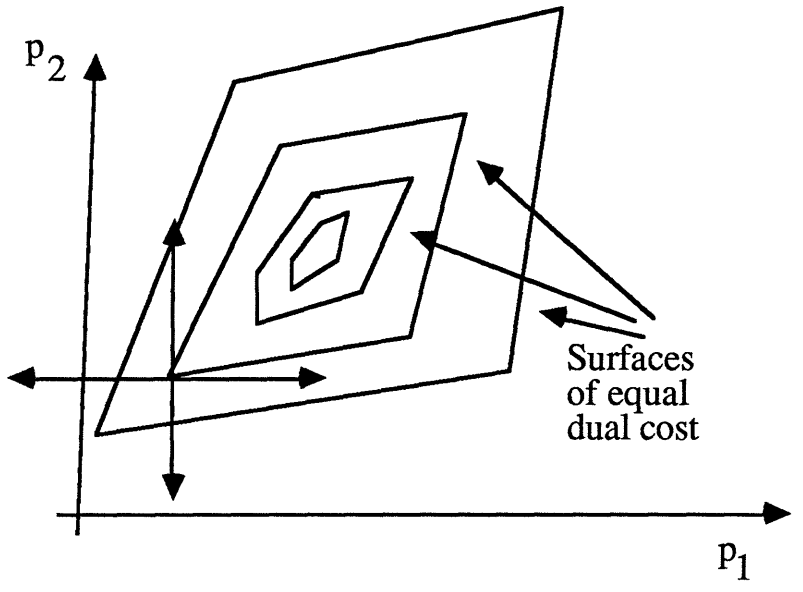

Figure 1: Illustration of the difficulty of the relaxation approach for linear arc costs. At the indicated point it is impossible to improve the dual cost by changing any single price.

One approach to overcome the difficulty has been pursued by the author and his coworkers for the past several years [5]-[8]. In this approach, in addition to the single coordinate relaxation steps, one occassionally changes the prices of several nodes as a group as illustrated in Figure 2. Unfortunately, however, these multiple node price changes need *global* node price information, thereby substantially complicating the distributed implementation of the algorithm. Nonetheless, public domain sequential codes based on this approach have proved highly successful, and have outperformed the classical primal simplex and primal-dual methods by a substantial margin on standard benchmark problems [6]-[8], and by an overwhelming margin on large problems [7], [8].

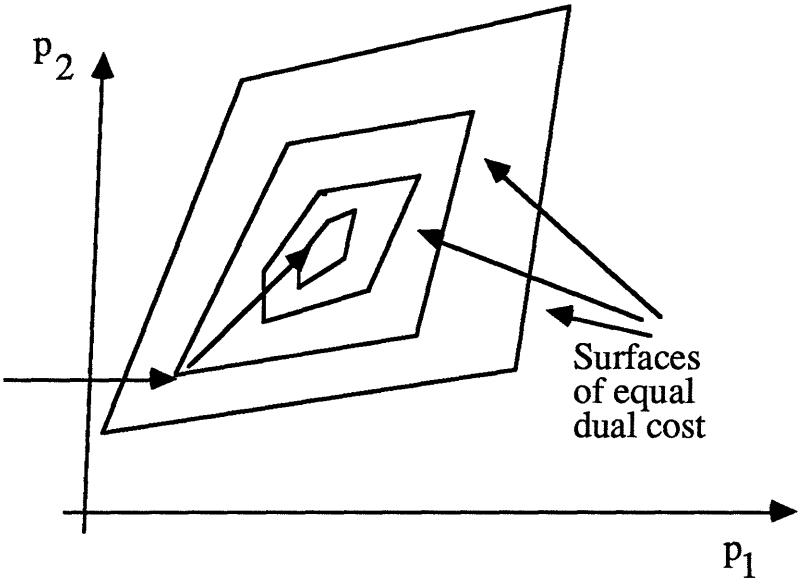

Figure 2: Illustration of the first approach for overcoming the difficulty of Fig. 1. The idea is to reduce the dual cost by changing simultaneously the prices of several nodes.

A second approach for overcoming the difficulty shown in Fig. 1, that is more amenable to parallelization, is based on performing single node price changes exclusively. Some of these price changes may worsen the dual cost by small amounts, but Fig. 3 illustrates that if the price changes are properly regulated, the algorithm can evenually approach the optimal solution.

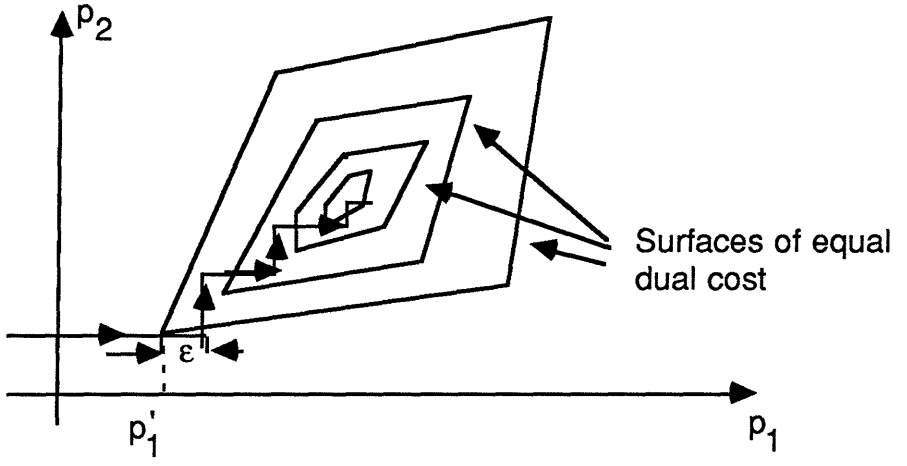

Figure 3: Relaxation approach whereby single node price changes are performed exclusively even if this leads to deterioration of the dual cost. If the size of the price changes is properly regulated, the method can approach the optimal solution.

This second approach was first proposed for the classical assignment problem in 1979 by the author in an unpublished report [9], and was also briefly described in an extended abstract [10]. The corresponding method, called the *auction algorithm*, is the subject of the present paper. Among the many methods for the assignment problem [11] - [25], the auction algorithm seems to be the only one that has a naturally parallel character and is well suited for implementation on a massively parallel machine. It also admits an intuitively appealing economic interpretation, although we make no claim about its utility as a model for market price adjustment. (See [26]-[28] for work on the assignment problem with an economic modeling orientation. In particular, [27] describes an algorithm which involves ideas of competitive bidding, but differs substantially from ours; for example it does not use the idea of ε-complementary slackness described later in this paper.)

In addition to the idea of independent single node price changes which is responsible for the parallel character of the method, the auction algorithm embodies several ideas that were new at the time of its original proposal and have recently found considerable extension and use in network algorithms and computational complexity analysis. These include the notion of ε-

complementary slackness (ε-CS) and the idea of *ε-scaling*.

In Section 2, we define ε-CS as a condition whereby the usual complementary slackness relation for assignment of a person to an object can be violated by a small (ε) amount. A key fact is that a feasible assignment satisfying ε-CS is optimal if ε is sufficiently small. This idea was generalized and was used algorithmically for linear network flow problems in [7], and for nonlinear network flow problems in [2]. A related idea was also used in a different context in the analysis of [29]. ε-CS is central in the extension of the auction algorithm to general linear network flow problem, first given in [30], [31], and further analyzed later in [32]-[34]. ε-CS is also likely to prove useful in other contexts. For example it plays an important role in the recent assignment algorithm of [35] which also uses ε-scaling. This algorithm has complexity $O(N^{1/2}Alog(NC))$ where N is the number of persons to be assigned, A is the number of person - object pairs than can be assigned to each other, and C is the maximum absolute object value. For reasonably small values of C, this is the best complexity bound for assignment problems. However, the algorithm of [35] is not naturally parallelizable.

The idea of ε-scaling is similar to the idea of gradually reducing the penalty parameter in penalty function methods in order to alleviate ill-conditioning, and is essential for obtaining computational efficiency and polynomial complexity. In the pure form of the auction algorithm, ε is kept constant, and the complexity can be shown to be $O(NAC/\varepsilon)$. It is necessary to take ε sufficiently small (less than 1/N assuming object values are integer), in order that the assignment obtained by the algorithm be optimal (see Proposition 1 in Section 3). As a result, the pure form of the algorithm has pseudopolynomial complexity, and its computational performance can be poor as shown by an example given in Section 3. ε-scaling corrects this by applying the auction algorithm with successively smaller values of ε starting with a large value and ending with a value less than 1/N. The final prices and assignment corresponding to each value of ε are used to obtain "good" starting prices and assignment for the next smaller value. ε-scaling was first implemented by the author and tested computationally in 1979 and again in 1985 for the auction algorithm with encouraging results (unpublished work). It was first analyzed in the context of the

more general min-cost flow problem in [36] where polynomial complexity results were given that were more fully established in [33], [34]. Another scaling procedure that is closer in spirit to traditional scaling methods in network flow problems [37]-[40], and can serve as an alternative to ϵ-scaling was later proposed and analyzed in [32], using some of the ideas of [36] and also of [40]. The method of analysis of [32], [33], [34] can be used to show that the auction algorithm with ϵ-scaling as well as the more traditional cost scaling method has complexity O(NAlog(NC)). This is relatively straightforward, but requires replication of large portions of the analysis in these references, and so will not be given. The computational results of the present paper suggest that the auction algorithm with ϵ-scaling is competitive with the best known implementations of assignment methods even without the benefit of parallelism. When implemented in a parallel machine, the auction algorithm should converge much faster.

The purpose of the present paper is to describe the auction algorithm, to interpret it as a dual Jacobi-like relaxation method, and to provide computational results. While a version of the auction algorithm can be derived from its min-cost flow generalization of [30], [31], this derivation is neither obvious nor straightforward; indeed one must introduce modifications to the basic form of the ϵ-relaxation method of [30], [31] in order to be able to derive the auction algorithm as a special case. (The reverse is also true. The min-cost flow problem can be convertied into a transportation problem ([19], p. 149), which can be converted to an assignment problem by replacing single sources and single sinks of the transportation problem with multiple persons and objects respectively in the assignment problem. By applying the auction algorithm to this assignment problem, one can derive a generic form of the ϵ-relaxation method of [30], [31].) The conceptually useful interpretation as a Jacobi-like relaxation method is also nontrivial to obtain starting from the general network flow framework. It is therefore worthwhile to derive the auction algorithm from basic principles, and Section 2 is devoted to this purpose. The validity of the algorithm is established in Section 3, and the computational results are given in Section 5. The auction algorithm can also be implemented in a totally asynchronous (chaotic) distributed environment; this appears to be a generic characteristic of relaxation methods

for both linear and nonlinear network flow problems. We describe briefly and somewhat informally in Section 4 how this can be done, and we refer to [30], [31] for discussion and analysis of totally asynchronous implementations in the context of the general min-cost flow problem.

2. Assignment by Means of an Auction

Consider N persons wishing to divide among themselves N objects. We number persons and objects as 1, 2, ..., N. For each person i there is a nonempty subset A(i) of objects that can be assigned to i. An *assignment* S is a (possibly empty) set of person-object pairs (i,j) such that $j \in A(i)$ for all $(i,j) \in S$, for each person i there is at most one pair $(i,j) \in S$, and for each object j there is at most one pair $(i,j) \in S$. In the context of a given assignment S, we say that person i is *assigned* if there exists an object j such that $(i,j) \in S$; otherwise we say that i is *unassigned.* We use similar terminology for objects. A *complete assignment* is an assignment containing N pairs (*i.e.* every person is assigned to a distinct object). There is a given integer value a_{ij} that a person i associates with an object $j \in A(i)$. We want to find a complete assignment that maximizes

$$\sum_{(i,j)\in S} a_{ij}$$

over all complete assignments S. We call this the *primal assignment problem* and note its well-known equivalence to a linear programming (linear network flow) problem as shown in Fig. 4.

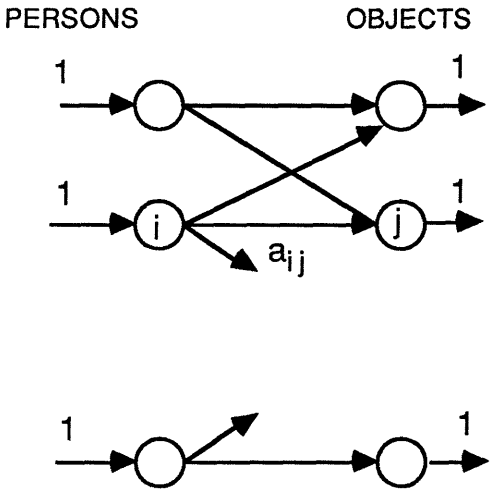

Figure 4: Formulation of the primal assignment problem as a linear programming/network flow problem. The nodes are the persons and the objects, and an arc (i,j) connects every person i and object j such that j belongs to A(i). Denoting the flow of arc (i,j) by f_{ij} the corresponding linear programming problem is:

$$\text{minimize} \sum_{i=1}^{N} \sum_{j \in A(i)} a_{ij} f_{ij}$$

subject to

$$\sum_{j \in A(i)} f_{ij} = 1, \quad i=1, ..., N$$

$$\sum_{\{i \mid j \in A(i)\}} f_{ij} = 1, \quad j=1, ..., N$$

$$0 \leq f_{ij}$$

For each object j, it is useful to introduce a dual variable p_j called the *price* of j. We call the vector with coordinates p_j, j=1, ..., N a *price vector*. For a given price vector p the scalar

$$\pi_i = \max_{j \in A(i)} \{a_{ij} - p_j\} \qquad (1)$$

is called the *profit margin* of person i corresponding to p. It is helpful to think of p_j as the amount of money that a person must pay when assigned to j. Therefore, for a given price vector p, a_{ij} - p_j may be thought of as the benefit person i associates with being assigned to object j. In this

context the name "profit margin" for π_i as given by (1) becomes meaningful.

A dual problem to the assignment problem is

$$\text{minimize} \sum_{i=1}^{N} \pi_i + \sum_{j=1}^{N} p_j$$

$$\text{subject to } \pi_i + p_j \geq a_{ij}, \quad \forall\, i, \text{ and } j \in A(i)$$

For a given price vector p, the cost of this problem is minimized when π_i equals the maximum value of $a_{ij} - p_j$ over $j \in A(i)$. Therefore an equivalent form of the dual problem is the *unconstrained* minimization problem

$$\text{minimize} \quad q(p) \tag{2}$$

subject to no constraints on p

where p is the vector of object prices p_j, and

$$q(p) = \Sigma_i \max_{j \in A(i)} \{a_{ij} - p_j\} + \Sigma_j p_j \tag{3}$$

From linear programming theory it is known that a complete assignment $S=\{(i,j_i) \mid i=1, ..., N\}$ and a price vector p are simultaneously primal and dual optimal respectively if and only if

$$a_{ij_i} - p_{j_i} = \max_{j \in A(i)} \{a_{ij} - p_j\} \qquad \text{for all } i = 1, ..., N.$$

This is known as the *complementary slackness condition* and states that at an optimum each person is assigned to a "best" object, *i.e.* to one attaining the maximum in the definition of profit margin (1).

A relaxation of the complementary slackness condition is to allow persons to be assigned to objects that come within ε of attaining the maximum in (1). Formally we say that an assignment S (not necessarily complete) and a price vector p satisfy *ε-complementary slackness* (ε-CS) if

$$\pi_i - \varepsilon \leq a_{ij} - p_j \leq \pi_i, \qquad \text{for each } (i,j) \in S, \tag{4}$$

where π_i is given by (1), and ε is a nonnegative constant.

We now describe formally the auction algorithm. We fix $\varepsilon>0$, and we start with some

assignement (possibly empty) and price vector satisfying ε-CS. The algorithm proceeds iteratively and terminates when a complete assignement is obtained. At the start of the generic iteration we have an assignment S and a price vector p satisfying ε-CS. At the end of the iteration, S and p are updated while maintaining the ε-CS condition. There are two phases in each iteration, the *bidding phase*, and the *assignment phase* described below:

Bidding Phase: For each person i that is unassigned under the assignment S:

Compute the "current value" of each object $j \in A(i)$ given by

$$v_{ij} = a_{ij} - p_j \tag{5}$$

Find a "best" object j* having maximum value

$$v_{ij^*} = \max_{j \in A(i)} v_{ij},$$

and find the best value offered by objects other than j*

$$w_{ij^*} = \max_{j \in A(i),\, j \neq j^*} \{a_{ij} - p_j\}. \tag{6}$$

(If j* is the only object in A(i) we define w_{ij^*} to be $-\infty$, or, for computational purposes, a number that is much smaller than v_{ij^*}.)

Compute the "bid" of person i for object j* given by

$$b_{ij^*} = p_{j^*} + v_{ij^*} - w_{ij^*} + \varepsilon = a_{ij^*} - w_{ij^*} + \varepsilon \tag{7}$$

[We characterize this situation by saying that person i bid for object j*, and that object j^* received a bid from person i. The algorithm works if the bid has any value between $p_{j^*} + \varepsilon$ and $p_{j^*} + v_{ij^*} - w_{ij^*} + \varepsilon$, but it tends to work fastest for the maximal choice (7).]

Assignment Phase: For each object j:

Let P(j) be the set of persons from which j received a bid in the bidding phase of the iteration.

If P(j) is nonempty increase p_j to the highest bid

$$p_j := \max_{i \in P(j)} b_{ij}, \tag{8}$$

remove from the assignment S any pair (i,j) (if one exists), and add to S the pair (i^*,j) where i^* is some person in P(j) attaining the maximum above.

It is seen that at the end of the iteration we have a new price vector that differs from the preceding vector in the prices of the objects that received a bid during the iteration. We also have a new assignment that differs from the preceding one in that each object that received a bid is now assigned to some person that was unassigned at the start of the iteration. However, the assignment at the end of the iteration need not have more pairs than the one at the start of the iteration because it is possible that all objects that received a bid were assigned at the start of the iteration.

A first important fact is that *the algorithm preserves ε-CS throughout its execution, i.e.* if the assignment and price vector available at the start of an iteration satisfy ε-CS, the same is true for the assignment and price vector obtained at the end of the iteration. To see this suppose that object j* received a bid from person i and was assigned to i during the iteration. Then if p_j and p'_j are the object prices before and after the assignment phases we have [cf. (7), (8)]

$$p'_{j^*} = b_{ij^*} = a_{ij^*} - w_{ij^*} + \varepsilon \tag{9}$$

Using this equation and the fact $p'_j \geq p_j$ for all j, it follows that, if π'_i is the profit margin of i after the assignment phase [cf. (1)], we have

$$\pi'_i = \max_{j \in A(i)} \{a_{ij} - p'_j\} = \max\{a_{ij^*} - p'_{j^*}, \max_{j \in A(i),\, j \neq j^*} \{a_{ij} - p'_j\}\}$$

$$\leq \max\{a_{ij^*} - p'_{j^*}, \max_{j \in A(i),\, j \neq j^*} \{a_{ij} - p_j\}\} = \max\{w_{ij^*} - \varepsilon, w_{ij^*}\} = w_{ij^*}$$

Combining this equation with (9) we obtain

$$a_{ij^*} - p'_{j^*} = w_{ij^*} - \varepsilon \geq \pi'_i - \varepsilon, \tag{10}$$

which shows that (4) continues to hold after the assignment phase.

Note that both the bidding and the assignment phases are highly parallelizable. In the extreme case of a fine grain parallel computing environment where there is a processor associated with each person and a processor associated with each object, all unassigned persons/processors can compute their bids simultaneously and communicate them to the relevant objects/processors. Those object/processors that received at least one bid can determine the highest bidder simultaneously and communicate the changes in the current assignment and price vector to the

relevant persons/processors.

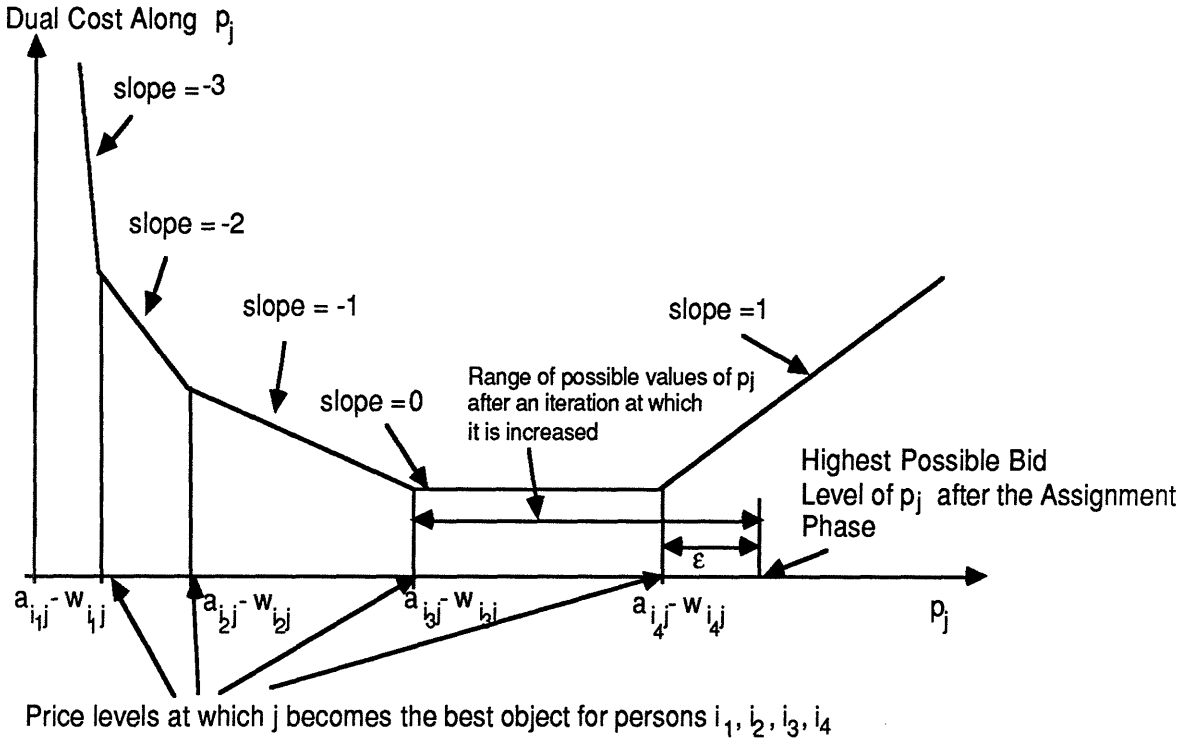

Figure 1: Form of the dual cost along the price coordinate p_j. From (3) the right directional derivative of q along p_j is

$$d_j^+ = 1 - (\text{number of persons i with } j \in A(i) \text{ and } p_j < a_{ij} - w_{ij})$$

where w_{ij} is given by (6). The break points are $a_{ij} - w_{ij}$ for all i such that $j \in A(i)$. If $p_j < a_{ij} - w_{ij}$, then an unassigned person i bids for object j the amount $a_{ij} - w_{ij} + \varepsilon$. The price p_j after the assignment phase is increased to ε plus the highest level $a_{ij} - w_{ij}$ over all unassigned persons i with $j \in A(i)$. .

Figure 5 indicates how each bidding and subsequent assignment phase can be interpreted as a Jacobi - like relaxation step for minimizing the dual function q(p) of (3). In particular, *the price p_j of each object j that received a bid during the assignment phase is increased to either a*

value that minimizes q(p) when all other prices are kept constant, or else exceeds the largest such value by no more than ε. To see this suppose that at some iteration object j received a bid and its price was raised from p_j to p'_j. Then

$$p'_j = \max\{a_{ij} - w_{ij} \mid \text{i was unassigned and j received a bid from i}\} + \varepsilon \quad (11)$$

$$p_j \geq \max\{a_{ij} - w_{ij} \mid \text{i was unassigned and j did not receive a bid from i}\} \quad (12)$$

Since whenever an object receives a bid, its price increases by at least ε, we have

$$p'_j \geq p_j + \varepsilon,$$

so from (11) and (12) we obtain

$$p'_j \geq \max\{a_{ij} - w_{ij} \mid \text{i was unassigned}\} + \varepsilon \quad (13)$$

Since the algorithm maintains property (4) throughout, we have that if at the start of the iteration person i was assigned to some $k \neq j$ and $j \in A(i)$ then

$$a_{ij} - p_j \leq \pi_i \leq a_{ik} - p_k + \varepsilon \leq w_{ij} + \varepsilon$$

Using this relation and the fact $p'_j \geq p_j + \varepsilon$ we obtain

$$p'_j \geq p_j + \varepsilon \geq a_{ij} - w_{ij}$$

and

$$p'_j \geq \max\{a_{ij} - w_{ij} \mid \text{i was assigned to some } k \neq j\} \quad (14)$$

Combining (13) and (14) we obtain

$$p'_j \geq \max\{a_{ij} - w_{ij} \mid \text{i was not assigned to j}\}$$

Since there can be at most one person assigned to j, it follows from the form of the dual cost shown in Figure 5, that p'_j is no less than the smallest value of p_j that minimizes q(p). Combining this fact with (11), we see that p'_j has the property stated in the beginning of this paragraph.

Note that our method is not quite equivalent to a coordinate descent method which reduces the dual cost along each price coordinate, since the dual cost (3) may deteriorate strictly after a price increase. However, the cost deterioration is at most ε. It will be seen shortly that, for ε small enough, an optimal solution can still be obtained thanks to the integer nature of the problem data, and the fact that ε-CS holds at termination [cf. (4)].

Figure 5 also suggests a variation of the algorithm whereby, in addition to all unassigned persons, each *assigned* person i bids for its *own* assigned object j the amount $a_{ij} - w_{ij} + \varepsilon$. This variation has not been tested, but may accelerate convergence in some parallel computing environments.

The above algorithm may be viewed as a *Jacobi version* of the relaxation idea since the bids of all unassigned persons are calculated simultaneously, and the prices of objects that receive a bid are raised simultaneously. An alternative is a *Gauss-Seidel version* whereby a single unassigned person bids for an object, and the price rise of the object is taken into account when the next bid by an unassigned person takes place. This version is just as valid as the Jacobi version and in fact tends to converge a little faster, but is generally much less parallelizable.

3. Properties of the Algorithm

Suppose that the algorithm terminates with the final (complete) assignment $\{j_i \mid i=1, \ldots, N\}$, the object prices p_j, and the profit margins π_i given by (1). Then, by adding (4) over i,

$$\Sigma_i a_{ij_i} \geq \Sigma_i (\pi_i + p_{j_i}) - N\varepsilon$$

If A^* is the optimal primal value and the (equal) optimal dual value we have using the relation above

$$A^* \geq \Sigma_i a_{ij_i} \geq \Sigma_i (\pi_i + p_{j_i}) - N\varepsilon = q(p) - N\varepsilon \geq A^* - N\varepsilon$$

Therefore the assignment $\{j_i \mid i=1, \ldots, N\}$ is within $N\varepsilon$ of being optimal. Since a_{ij} are integer, an optimal assignment is obtained when $\varepsilon < 1/N$. Thus we have shown:

Proposition 1: A complete assignment obtained upon termination of the algorithm is within $N\varepsilon$ of being optimal, and is optimal if $\varepsilon < 1/N$.

The next result asserts that the algorithm terminates assuming existence of at least one complete assignment. The proof relies on the following facts:

a) Once an object is assigned, it remains assigned throughout the remainder of the algorithm's duration. Furthermore, except at termination, there will always exist at least one object that has never been assigned, and has a price equal to its initial price. This is due to the fact that a bidding and assignment phase can result in a reassignment of an already assigned object to a different person, but cannot result in the object becoming unassigned.

b) Each time an object receives a bid its price increases by at least ε [cf. (7), (8)]. Therefore if the object receives a bid an infinite number of times, its price increases to ∞.

c) Each time a person i bids for an object a number of times at most equal to the cardinality of A(i), his profit margin π_i as defined by (1) must decrease by at least ε. This is due to the fact that a bid by person i either decreases π_i by at least ε, or else leaves π_i unchanged because there is more than one object j attaining the maximum in (1). However, in the latter case the price of the object j* receiving the bid will increase by at least ε, and object j* will not receive a bid again from person i until π_i decreases by at least ε. The conclusion is that if a person bids an infinite number of times his profit margin must decrease to $-\infty$.

Proposition 2: If at least one complete assignment exists, the algorithm terminates in a finite number of iterations.

Proof: If the algorithm continues indefinitely, the prices of a proper [cf. a) above] subset J^∞ of objects increases to ∞, while the profit margins π_i of a subset I^∞ of persons decrease to $-\infty$, [cf. c) above]. Furthermore, eventually, in view of (4), at any given time each object in J^∞ can only be assigned to a person from I^∞, and a person from I^∞ will either be assigned to an object in J^∞ or be unassigned. Also, in view of c) above, eventually only persons from I^∞ will be unassigned. Therefore the cardinality of I^∞ is greater than the cardinality of J^∞ while, in view of (1), we have $J^\infty \supset A(i)$ for all i in I^∞. This contradicts the existence of a complete assignment. **Q.E.D.**

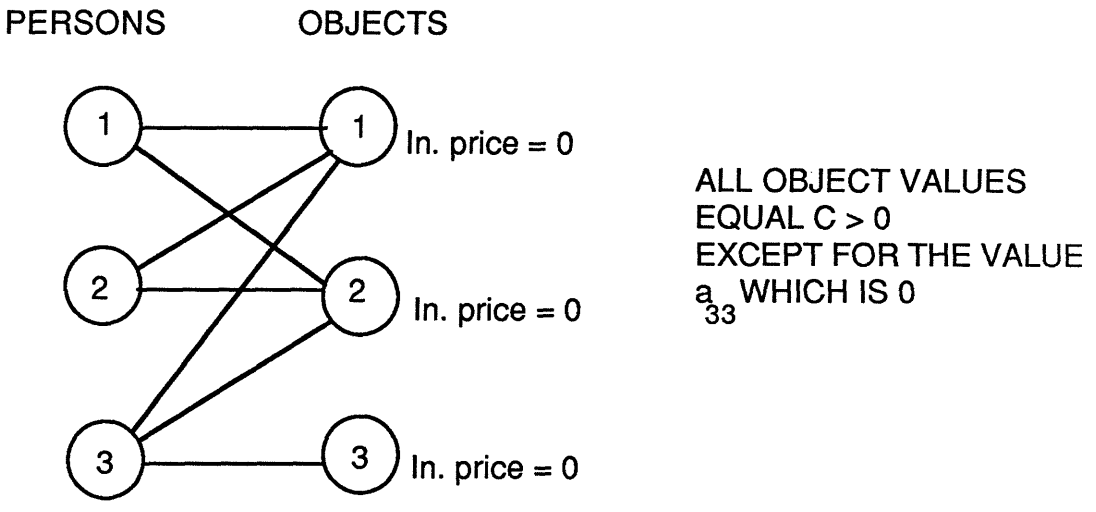

Figure 6: Example where the number of bidding phases is proportional to C/ε. Here at each bidding phase one of the persons 1, 2, or 3 bids the price of object 1 or 2 up by an increment ε until the time that these prices reach the level C.

It can be shown (see Fig. 6) that the computational complexity of the algorithm is sensitive to the maximal absolute object value

$$C = \max\{|a_{ij}| \mid i=1, ..., N, j \in A(i)\}$$

It was found experimentally that this sensitivity manifests itself often in the practical performance of the algorithm. To improve this performance, scaling the value of ε suggests itself. Here we multiply all costs a_{ij} with (N+1) and apply the algorithm with progressively lower value of ε up to the point where ε becomes 1 or smaller. The sequence of ε values used is

$$\varepsilon(k) = \max\{1, \Delta/\theta^k\}, \qquad k = 0, 1, \ldots$$

where θ and Δ are parameters with $\Delta > 0$ and $\theta > 1$. Scaling procedures are discussed in more detail in [32] -[40]. The scaling procedure given above, called *ε-scaling*, was implemented and tested by the author in 1979 and 1985. The computational results of Section 5 show that, with proper choice of the parameters Δ and θ, the algorithm has excellent performance even without

the benefit of parallelism.

4. Asynchronous Implementation

The auction algorithm given in Section 2 may be described as *synchronous* since the bidding and assignment are carried out simultaneously for all persons and objects respectively. A totally asynchronous (chaotic) algorithm (see [41]-[45] for computation models and convergence analysis) results if each unassigned person makes a bid at arbitrary times on the basis of object price information that may be outdated (because of additional bidding of which the person is not informed). Furthermore assignment of objects may be decided even if some potential bidders have not been heard from. We can similarly show the same termination properties as for the synchronous version of the algorithm subject to two conditions.

a) An unassigned person will bid for some object within finite time, and cannot bid twice (*i.e.* cannot bid for a second object while waiting for a reply regarding the disposition of an earlier bid for another object).

b) Whenever one or more bids are received that raise the price of an object then, within finite time, that price must be updated, and its value must be communicated (not necessarily simultaneously) to all persons. Furthermore the new bidder assigned to the object must be informed of this fact simultaneously with receiving the new price.

A more careful formulation of the asynchronous model, and a proof of validity for the more general case of the minimum cost flow problem is given in [30]-[32].

5. Computational Results

In this section we provide the results of computational experimentation on both a serial machine (VAX 11-750), and a parallel machine (Sequent Balance 21000). A more extensive computational study is currently under way.

A code called AUCTION was used to test the performance of the algorithm on a serial machine. The code implements ε–scaling as described at the end of Section 3 for various values of the parameters θ and Δ. A standard initialization procedure was used that sets the initial object prices to $p_j = \min_i a_{ij}$ for all j. At the end of the kth subproblem (*i.e.* the problem corresponding to $\varepsilon(k-1)$) the assignment obtained is checked to see if condition (4) is satisfied for $\varepsilon = \varepsilon(k)$. If (4) is not satisfied for a person i and the corresponding object j_i, the pair (i,j_i) is removed from the assignment, so that condition (4) is enforced at the beginning of each subproblem.

The test problems were generated using the widely used public domain generator NETGEN [46] with the random seed number set at the value used in the original paper [46]. The AUCTION code was compared with a version of the relaxation code RELAX-II [7]-[8] specially adapted to solve assignment problems. This code, called RELAX-IIA, outperformed a variety of specialized relaxation and primal dual codes for the assignment problem based on existing methods. On the basis of this fact and in view of the excellent performance of the RELAX-II code for assignment problems, as documented in [7]-[8], we believe that AUCTION has been given state-of-the-art competition.

The codes compared were written in standard FORTRAN on a VAX 11-750, and were compiled under VMS version 4.1. A rough indication of the potential speedup of AUCTION in a parallel machine is provided by the *degree of parallelism* defined as the number of unassigned persons in a bidding phase averaged over all bidding phases.

Our results are summarized in Figures 7-10. Figure 7 gives the solution times for five problems with equal density (1.5%). The parameters used were $\theta = 8$ and $\Delta = NC/2$. Figure 8 gives the times for five problems where the set A(i) has average cardinality equal to 5. The parameters used were $\theta = 6$ and $\Delta = NC/2$. Figure 9 gives the times for the five standard assignment problems (problems 11-15 in [46]). The parameters used were $\theta = 4$ and $\Delta = NC/2$. The parallelism factor for these problems is roughly constant near 10 as Figure 10 shows. This indicates that the efficiency of the algorithm may be quite low when implemented on a parallel machine with a large number of processors.

The auction algorithm was also independently implemented and tested on dense assignment problems by Prof. J. Kennington and Mr. L. Hatay at Southern Methodist University using a Sequent Balance 21000 computer - a shared memory parallel machine. Their results, briefly described here with their kind permission, show that the auction algorithm implemented using a single processor outperformed substantially and consistently the NETFLO code of [17] and a Hungarian code described in [47]. Figure 11 shows their results with a multiple processor implementation of the auction algorithm and, in particular, the speedup as a function of the number of CPUs employed. It is seen that the efficiency of the algorithm for a small number of processors is quite satisfactory. Note that the speedup with one processor is less than one because we are comparing a serial with a parallel code, and, even on a single processor, the parallel code is not as efficient as the parallel code.

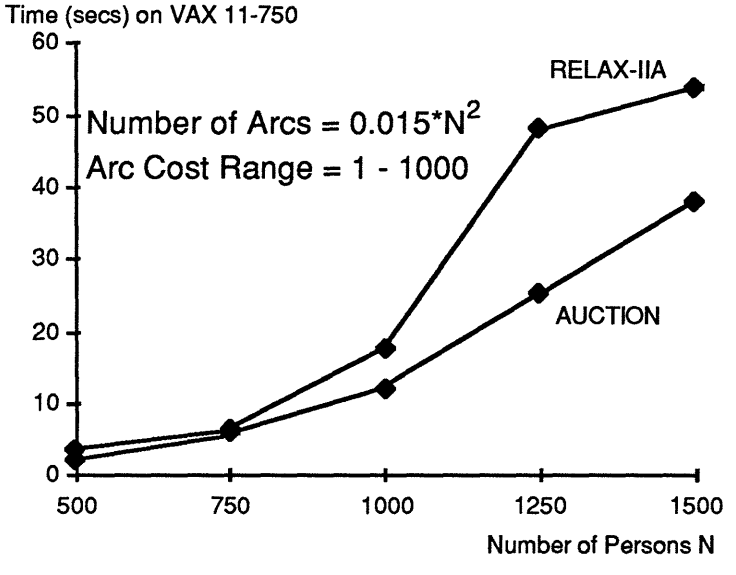

Figure 7: Solution times for AUCTION and RELAX-IIA on VAX 11-750. All problems generated by NETGEN.

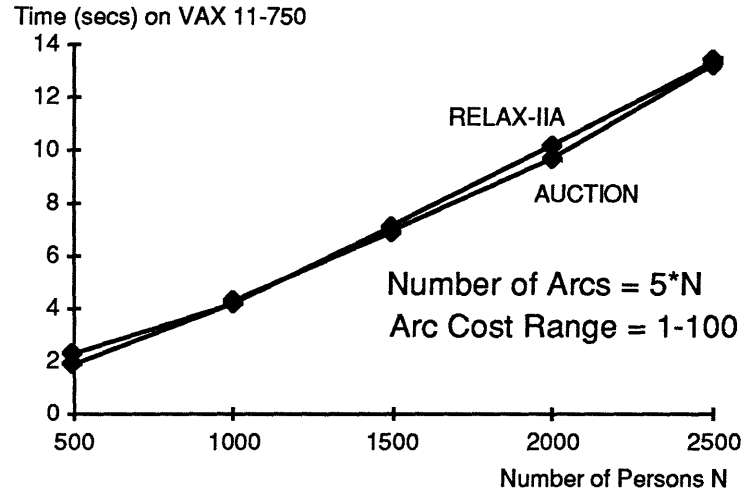

Figure 8: Solution Times for AUCTION and RELAX-IIA on VAX 11-750. All problems generated by NETGEN.

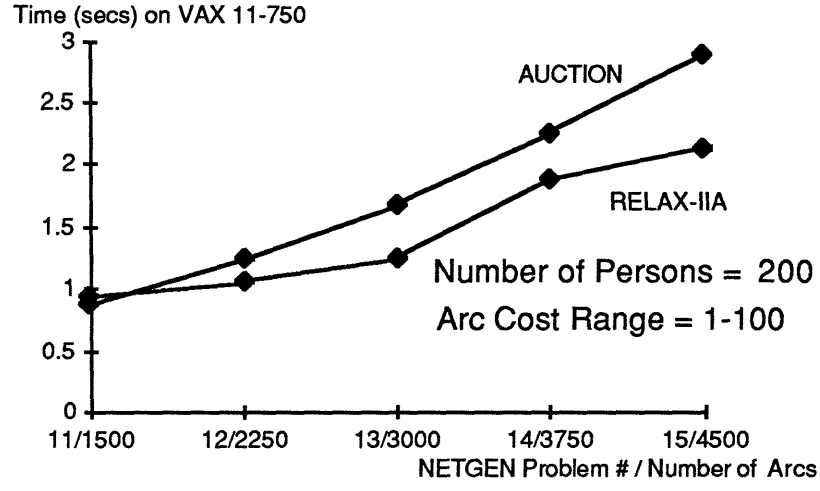

Figure 9: Solution times in secs for AUCTION and RELAX-IIA on VAX 11-750 for standard NETGEN assignment problems.

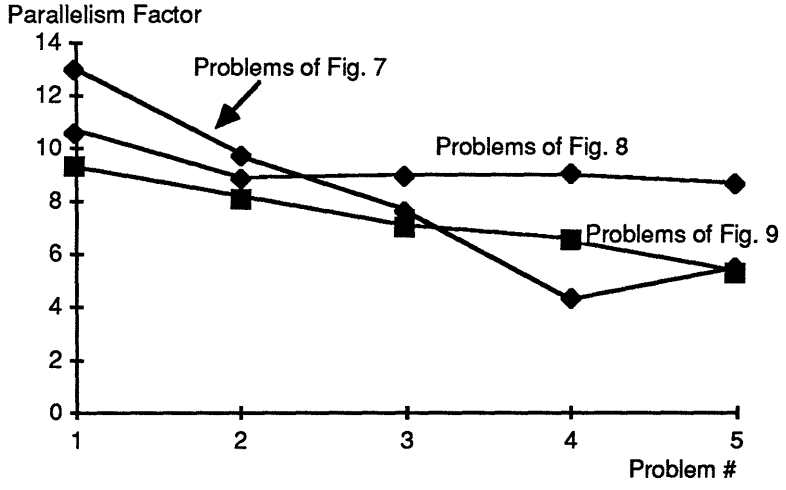

Figure 10: Parallelism factor for the problems of Figs. 7, 8, and 9.

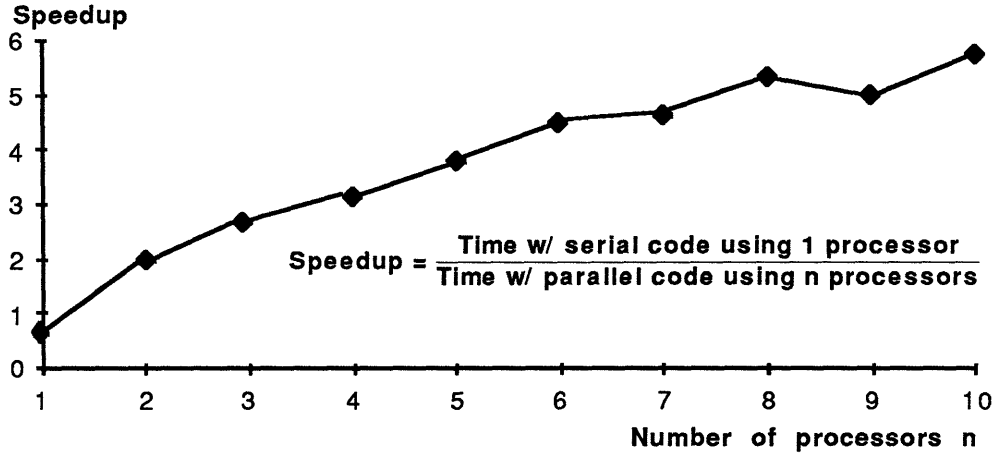

Figure 11: Speedup of a parallel implementation of the auction algorithm as a function of the number of processors used in a Sequent Balance 21000 computer. The problem solved is a randomly generated 800x800 fully dense problem with arc cost range 1 - 10000. The time required by the serial auction code using a single processor is 336.13 secs.

References

[1] Bertsekas, D. P., and El Baz, D., "Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems", *SIAM J. on Control and Optimization*, Vol. 25, 1987, pp. 74-85.

[2] Bertsekas, D. P., Hosein, P. A., and Tseng, P., "Relaxation Methods for Network Flow Problems", *SIAM J. on Control and Optimization*, Vol. 25, 1987, to appear.

[3] Zenios, S. A. and Mulvey, J. M., "Relaxation Techniques for Strictly Convex Network Problems", *Annals of Operations Research*, Vol. 5, 1985/6, pp. 517-538.

[4] Zenios, S. A. and Lasken, R. A., "Nonlinear Network Optimization on a Massively Parallel Connection Machine", Preprint, 1987.

[5] Bertsekas, D. P., "A New Algorithm for the Assignment Problem", *Math Programming*, Vol. 21, 1981, pp. 152-171.

[6] Bertsekas, D. P., "A Unified Framework for Primal-Dual Methods in Minimum Cost Network Flow Problems", *Math. Progr.*, Vol. 32, 1985, pp. 125-145.

[7] Bertsekas, D. P., and Tseng, P., "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems", LIDS Report P-1462, M.I.T., May 1985, to appear in *Operations Research*.

[8] Bertsekas, D. P., and Tseng, P., "RELAX: A Code for Linear Network Flow Problems", in *FORTRAN Codes for N etwork Optmization*, (B. Simeone, ed.), Vol. to be published by the *Annals of Operations Research.*

[9] Bertsekas, D. P., "A Distributed Algorithm for the Assignment Problem", unpublished LIDS Report, M.I.T., March 1979.

[10] Bertsekas, D. P., "A Distributed Asynchronous Relaxation Algorithm for the Assignment Problem", *Proc. 24th IEEE Conference on Decision and Control,* Ft Lauderdale, Fla., Dec. 1985, pp. 1703-1704.

[11] Kuhn, H. W., "The Hungarian Method for the Assignment Problem", *Naval Research Logistics Quarterly,* Vol. 2, 1955, pp. 83-97

[12] Dinic, E., and Kronrod, M., "An Algorithm for the Solution of the Assignment Problem", *Soniet Math Doklady*, Vol. 10, 1969.

[13] Rockafellar, R. T., Network Flows and Monotropic Programming, J. Wiley, N. Y., 1984.

[14] Barr, R., Glover, F., and Klingman, D., "The Alternating Basis Algorithm for Assignment Problems", *Math Programming*, Vol. 13, 1977, pp. 1-13.

[15] Weintraub, A., and Barahona, F., "A Dual Algorithm for the Assignment Problem", Pub. No. 79/02/C, Departmento de Industrias, Universidad de Chile-Sede Occidente, Santiago, Chile, 1979.

[16] Hung, M., and Rom, W., "Solving the Assignment Problem by Relaxation", *Operations Research*, Vol. 28, 1980, pp. 969-982.

[17] Kennington, J., and Helgason, R., *Algorithms for Network Programming*, Wiley, NY., 1980.

[18] Engquist, M., "A Successive Shortest Path algorithm for the Assignment Problem", *INFOR*, Vol. 20, 1982, pp. 370-384.

[19] Papadimitriou, C. H., and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J. 1982.

[20] Glover, F., Glover, R., and Klingman, D., "Threshold Assignment Algorithm", Center for Business Decision Analysis Report CBDA 107, Graduate School of Business, Univ. of Texas at Austin, Sept. 1982.

[21] McGinnis, L. F., "Implementation and Testing of a Primal-Dual Algorithm for the Assignemnt Problem", *Operations Research* , Vol. 31, 1983, pp. 277-291.

[22] Hung, M., "A Polynomial Simplex Method for the Assignment Problem", *Operations Research*, Vol. 31, 1983, pp. 595-600.

[23] Balinski, M. L., "Signature Methods for the Assignment Problem", *Operations Research*, Vol. 33, 1985, pp. 527-537.

[24] Goldfarb, D., "Efficient Dual Simplex Methods for the Assignment Problem", *Math Programming*, Vol. 33, 1985, pp. 187-203.

[25] Balinski, M. L., "A Competitive (Dual) Simplex Method for the Assignment Problem", *Math Programming*, Vol. 34, 1986, pp. 125-141.

[26] Shapley, L. S., and Shubik, M., "The Assignment Game I: The Core", *Intern. J. of*

Game Theory, Vol. 1, 1972, pp. 117-130.

[27] Crawford, V. P., and Knoer, E. M., "Job Matching with Heterogeneous Firms and Workers", *Econometrica*, Vol. 49, 1981, pp. 437-450.

[28] Roth, A. E., "The Economics of Matching: Stability and Incentives", *Math. of Operations Research*, Vol. 7, 1982, pp. 617-628.

[29] Tardos, E., "A Strongly Polynomial Minimum Cost Circulation Algorithm", Combinatorica, Vol. 5, 1985, pp. 247-255

[30] Bertsekas, D. P., "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems", LIDS Report P-1606, M.I.T., Sept. 1986, revised Nov. 1986.

[31] Bertsekas, D. P., "Distributed Relaxation Methods for Linear Network Flow Problems", *Proceedings of 25th IEEE Conference on Decision and Control, Athens, Greece*, 1986, pp. 2101-2106.

[32] Bertsekas, D. P., and Eckstein, J., "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems", Report LIDS-P-1606, M. I. T., Sept. 1986 (rev. Jan. '87), *Proc. of IFAC '87*, Munich, Germany, July 1987.

[33] Goldberg, A. V., "Efficient Graph Algorithms for Sequential and Parallel Computers", Tech. Report TR-374, Laboratory for Computer Science, M. I. T., Feb. 1987.

[34] Goldberg, A. V., and Tarjan, R. E., "Solving Minimum Cost Flow Problems by Successive Approximation", *Proc. 19th ACM STOC*, May 1987.

[35] Gabow, H. N., and Tarjan, R. E., "Faster Scaling Algorithms for Graph Matching", Revised Abstract, 1987.

[36] Goldberg, A. V., "Solving Minimum-Cost Flow Problems by Successive Approximations", extended abstract, submitted to *STOC 87*, Nov 1986.

[37] Edmonds, J. and Karp, R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", *Journal of the ACM*, Vol. 19, 1972, pp. 248-264

[38] Rock, H., "Scaling Techniques for Minimal Cost Network Flows", in *Discrete Structures and Algorithms*, by V. Page (ed.), Carl Hansen, Munich, 1980

[39] Orlin, J. B., "Genuinely Polynomial Simplex and Non-Simplex Algorithms for the Minimum Cost Flow Problem", Working Paper No. 1615-84, Sloan School of Management, M. I. T., Dec. 1984

[40] Bland, R. G., and Jensen, D. L., "On the Computational Behavior of a Polynomial-Time Network Flow Algorithm", Tech. Report 661, School of Operations Research and Industrial Engineering, Cornell University, June 1985

[41] Chazan, D., and Miranker, W., "Chaotic Relaxation", *Linear Algebra and its Applications*, Vol. 2, 1969, pp. 199-222

[42] Baudet, G. M., "Asynchronous Iterative Methods for Multiprocessors", *Journal of the ACM*, Vol. 15, 1978, pp. 226-244

[43] Bertsekas, D. P., Tsitsiklis, J. N., and Athans, M., "Convergence Theories of Distributed Asynchronous Computation: A Survey", LIDS Report P-1412, M. I. T., Oct. 1984; also in *Stochastic Programming*, by F. Archetti, G. Di Pillo, and M. Lucertini (eds.), Springer-Verlag, N.Y., 1986, pp. 107-120

[44] Bertsekas, D. P., "Distributed Asynchronous Computation of Fixed Points", *Math Programming*, Vol. 27, 1983, pp. 107-120.

[45] Bertsekas, D. P., Tsitsiklis, J. N., *Parallel and Distributed Algorithms*, Prentice-Hall, Englewood Cliffs, N.J., 1988 (to appear).

[46] Klingman, D., Napier, A., and Stutz, J., "NETGEN -- A Program for Generating Large Scale (Un)Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems", *Management Science*, Vol. 20, 1974, pp. 814-822.

[47] Parker, R. G., and Rardin, R. L., *Discrete Optimization*, to be published by Academic Press, N.Y.