The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial

DIMITRI P. BERTSEKAS

Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

The auction algorithm is an intuitive method for solving the classical assignment problem. It outperforms substantially its main competitors for important types of problems, both in theory and in practice and is also naturally well suited for parallel computation. I derive the algorithm from first principles, explain its computational properties, and discuss its extensions to transportation and transshipment problems.

In the classical assignment problem there are n persons and n objects that we have to match on a one-to-one basis. There is a benefit a_{ij} for matching person i with object j, and we want to assign persons to objects so as to maximize the total benefit. Mathematically, we want to find a one-to-one assignment [a set of person-object pairs $(1, j_1), \ldots, (n, j_n)$, such that the objects $j_1, \ldots, j_n$ are all distinct] with maximum total benefit $\sum_{i=1}^{n} a_{ij_i}$.

The assignment problem is important in many practical contexts. The most obvious ones are resource allocation problems, such as assigning personnel to jobs, machines to tasks, and the like. There are also situations where the assignment problem appears as a subproblem in various methods for solving more complex problems; for example, in an important method for solving traveling salesman problems [Held and Karp 1970, 1971].

The assignment problem is also of great theoretical importance because, despite its simplicity, it embodies a fundamental linear programming structure. The most important type of linear programming problems, the linear network flow problem, can be reduced to the assignment problem by means of a simple reformulation [Bertsekas

0091-2102/90/2004/0133$01.25
This paper was refereed.

PROGRAMMING—MATHEMATICAL
NETWORKS/GRAPHS

INTERFACES 20: 4 July–August 1990 (pp. 133–149)

and Tsitsiklis 1989, p. 335; Papadimitriou and Steiglitz 1982, p. 149]. Thus, any method for solving the assignment problem can be generalized to solve the linear network flow problem. For this reason, the assignment problem has served as a convenient starting point for important algorithmic ideas in linear programming. For example, the primal-dual method [Ford and Fulkerson 1962; Minty 1960] was motivated and developed through Kuhn's Hungarian method [Kuhn 1955], the first specialized algorithm for the assignment problem. (The name of the algorithm honors its connection with the work of the Hungarian mathematician Egervary, dating to 1931.)

In the 35 years since Kuhn's original proposal a plethora of algorithms for the assignment problem have been proposed; for a representative but incomplete sample, see Balinski [1985, 1986], Barr, Glover, and Klingman [1977], Balas et al. [1989], Bertsekas [1981], Carpaneto, Martello, and Toth [1988], Derigs [1985], Engquist [1982], Glover, Glover, and Klingman [1982], Goldfarb [1985], Hall [1956], Hung [1983], Jonker and Volegnant [1987], McGinnis [1983], and Thompson [1981]. All of these methods are based on iterative improvement of some cost function; for example, a primal cost (as in primal simplex methods), or a dual cost (as in Hungarian-like methods, dual simplex methods, and relaxation methods).

The auction algorithm, which I first proposed in 1979 and discussed further in Bertsekas [1985, 1988], departs significantly from the cost improvement idea; at any one iteration, it may deteriorate both the primal and the dual cost, although in the end it finds an optimal assignment. It is based on a notion of approximate optimality, called *ϵ-complementary slackness,* and while it implicitly tries to solve a dual problem, it actually attains a dual solution that is not quite optimal. I originally conceived the auction algorithm as a method for massively parallel solution of the assignment problem, but it has also turned out to be very effective for serial computation.

Prices and Equilibria

To develop an intuitive understanding of the auction algorithm, it is helpful to introduce an economic equilibrium problem that turns out to be equivalent to the assignment problem.

Let us consider the possibility of matching the n objects with the n persons through a market mechanism, viewing each person as an economic agent acting in his own best interest. Suppose that object j has a price p_j and that the person who receives the object must pay the price p_j. Then, the (net) value of object j for person i is $a_{ij} - p_j$ and each person i would logically want to be assigned to an object j_i with maximal value, that is, with

$$a_{ij_i} - p_{j_i} = \max_{j=1,\ldots,n} \{a_{ij} - p_j\}. \tag{1}$$

We will say that a person i is *happy* if this condition holds, and we will say that an assignment and a set of prices are at *equilibrium* when all persons are happy.

Equilibrium assignments and prices are naturally of great interest to economists, but there is also a fundamental relation with the assignment problem; it turns out that an equilibrium assignment offers maximum total benefit (and thus solves the assignment problem), while the corresponding set of prices solves an associated dual optimization problem. This is a consequence of the celebrated duality theorem of linear programming (see for example Dantzig [1963], Papadimitriou and Steiglitz [1982], Rockafellar [1984]; in the terminology of linear programming, the "happiness" relation (1) is known as *complementary slackness*). I provide a simple, first principles, proof of the relation of equilibria to optimal assignments and dual optimization in the appendix, but for simplicity, I will not emphasize linear programming and duality in this paper.

An Auction Process

Let us consider now a natural process for finding an equilibrium assignment. I will call this process the *naive auction algorithm*, because it has a serious flaw. Nonetheless, this flaw will help motivate a more sophisticated and correct algorithm.

The naive auction algorithm proceeds in "rounds" (or "iterations") starting with any assignment and any set of prices. There is an assignment and a set of prices at the beginning of each round, and if all persons are happy with these, the process terminates. Otherwise some person who is not happy is selected. This person, call him i, finds an object j_i which offers maximal value, that is,

$$j_i \in \arg \max_{j=1,\dots,n} \{a_{ij} - p_j\}, \tag{2}$$

and then

(a) Exchanges objects with the person assigned to j_i at the beginning of the round,

(b) Sets the price of the best object j_i to the level at which he is indifferent between j_i and the second best object, that is, he sets p_{j_i} to

$$p_{j_i} + \gamma_i, \tag{3}$$

where

$$\gamma_i = v_i - w_i, \tag{4}$$

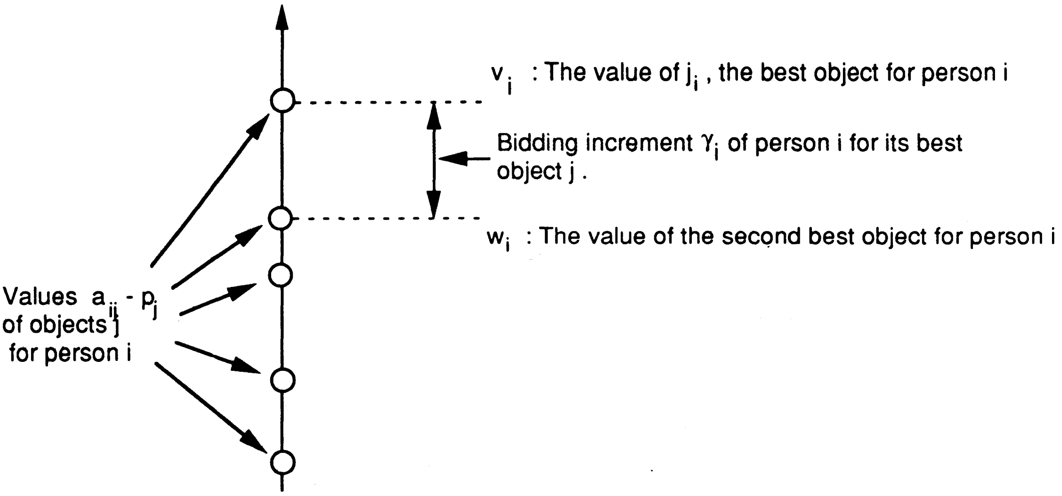

v_i : The value of j_i, the best object for person i
Bidding increment γ_i of person i for its best object j.
w_i : The value of the second best object for person i
Values $a_{ij} - p_j$ of objects j for person i

Figure 1: In the naive auction algorithm, even after the price of j_i is increased by the bidding increment γ_i, j_i continues to be the preferred object, so the bidder i is happy following the round. However, $\gamma_i = 0$ if there are two objects most preferred by the bidder i.

v_i is the best object value,

$$v_i = \max_{j} \{a_{ij} - p_j\}, \tag{5}$$

and w_i is the second best object value

$$w_i = \max_{j \neq j_i} \{a_{ij} - p_j\}. \tag{6}$$

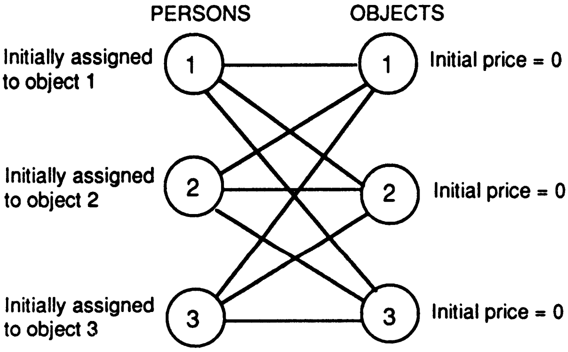
(Note that γ_i is the largest increment by which the best object price p_{j_i} can be increased, with j_i still being the best object for person i.)

This process is repeated in a sequence of rounds until all persons are happy.

We may view this process as an auction; at each round the bidder i raises the price of his or her preferred object by the bidding increment γ_i. Note that γ_i cannot be negative since $v_i \geq w_i$ [compare equations (5) and (6)], so the object prices tend to increase. The choice γ_i is illustrated in Figure 1. Just as in a real auction, bidding increments and price increases spur competition by making the bidder's own preferred object less attractive to other potential bidders.

Does this auction process work? Unfortunately, not always. The difficulty is that the bidding increment γ_i is zero when more than one object offers maximum value for the bidder i. As a result, a situation may be created where several persons contest a smaller number of equally desirable objects without raising their prices, thereby creating a never ending cycle (Figure 2).

To break such cycles, we introduce a


PERSONS
OBJECTS
Initially assigned to object 1
Initially assigned to object 2
Initially assigned to object 3
1
2
3
1
2
3
Initial price = 0
Initial price = 0
Initial price = 0

Here a_{ij} = C > 0 for all (i,j) with i = 1,2,3 and j = 1,2 and a_{ij} = 0 for all (i,j) with i = 1,2,3 and j = 3

At Start of Round #	Object Prices	Assigned Pairs	Happy Persons	Bidder	Preferred Object	Bidding Increment
1	0, 0, 0	(1,1) (2,2) (3,3)	1, 2	3	2	0
2	0, 0, 0	(1,1) (2,3) (3,2)	1, 3	2	2	0
3	0, 0, 0	(1,1) (2,2) (3,3)	1, 2	3	2	0

Figure 2: Illustration of how the naive auction algorithm may never terminate for a three-person and three-object problem. Here objects 1 and 2 offer benefit $C > 0$ to all persons, and object 3 offers benefit 0 to all persons. The algorithm cycles as persons 2 and 3 alternately bid for object 2 without changing its price because they prefer equally object 1 and object 2 ($\gamma_i = 0$; compare Figure 1).

perturbation mechanism, motivated by real auctions where each bid for an object must raise its price by a minimum positive increment, and bidders must on occasion take risks to win their preferred objects. In particular, let us fix a positive scalar ϵ and say that a person i is almost happy with an assignment and a set of prices if the value of its assigned object j_i is within ϵ of being maximal, that is,

$$a_{ij_i} - p_{j_i} \geq \max_{j=1,\ldots,n} \{a_{ij} - p_j\} - \epsilon. \tag{7}$$

We will say that an assignment and a set of prices are almost at equilibrium when all persons are almost happy. The condition (7), introduced first in 1979 in conjunction with the auction algorithm, is known as *ϵ-complementary slackness* and has played a central role in several algorithmic contexts recently (see for example, Ahuja et al. [1988], Bertsekas [1986a], Bertsekas and Castañon [1989b], Bertsekas and Eckstein [1987, 1988], Gabow and Tarjan [1987], and Goldberg and Tarjan [1987]). For $\epsilon = 0$ it reduces to ordinary complementary slackness [compare equation (1)].

We now reformulate the previous auction process so that the bidding increment is always at least equal to ϵ. The resulting method, the auction algorithm, is the same as the naive auction algorithm, except that the bidding increment γ_i is

$$\gamma_i = v_i - w_i + \epsilon, \tag{8}$$

[rather than $\gamma_i = v_i - w_i$ as in equation (4)]. With this choice, the bidder of a round is almost happy at the end of the round (rather than happy), as illustrated in Figure 3. The particular increment $\gamma_i = v_i - w_i + \epsilon$ used in the auction algorithm is the maximum amount with this property. Smaller increments γ_i would also work as long as $\gamma_i \geq \epsilon$, but using the largest possible increment accelerates the algorithm. This is consistent with experience from real auctions, which tend to terminate faster when the bidding is aggressive.

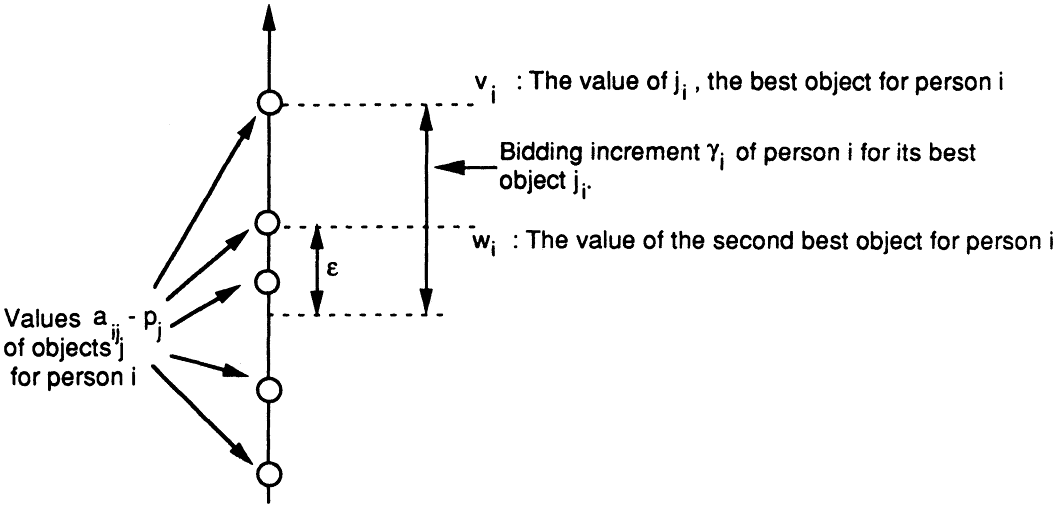

v_i : The value of j_i, the best object for person i
Bidding increment γ_i of person i for its best object j_i.
w_i : The value of the second best object for person i
ε
Values $a_{ij} - p_j$ of objects j for person i

Figure 3: In the auction algorithm, even after the price of the preferred object j_i is increased by the bidding increment γ_i, j_i will be within ϵ from being most preferred, so the bidder i is almost happy following the round.

We can now show that this reformulated auction process terminates in a finite number of rounds, necessarily with an assignment and a set of prices that are almost at equilibrium. To see this, note that once an object receives a bid for the first time, then the person assigned to the object at every subsequent round is almost happy; the reason is that a person is almost happy just after acquiring an object through a bid, and continues to be almost happy as long as he holds the object (since the other object prices cannot decrease in the course of the algorithm). Therefore, the persons that are not almost happy must be assigned to objects that have never received a bid. In particular, once each object receives at least one bid, the algorithm must terminate. Next note that if an object receives a bid in m rounds, its price must exceed its initial price by at least $m\epsilon$. Thus, for sufficiently large m, the object will become "expensive" enough to be judged "inferior" to some object that has not received a bid so far. It follows that only for a limited number of rounds can an object receive a bid while some other object still has not yet received any bid. Therefore, there are two possibilities: either (a) the auction terminates in a finite number of rounds, with all persons almost happy, before every object receives a bid or (b) the auction continues until, after a finite number of rounds, all objects receive at least one bid, at which time the auction terminates. (This argument assumes that any person can bid for any object, but it can be generalized for the case where the set of feasible person-object pairs is limited, as long as at least one feasible assignment exists.) Figure 4 shows how the auction algorithm, based on the bidding increment (8), overcomes the cycling problem of the example of Figure 2.

Optimality Properties at Termination

When the auction algorithm terminates, we have an assignment that is almost at equilibrium, but does this assignment maximize the total benefit? The answer here depends strongly on the size of ϵ. In a real auction, a prudent bidder would not place an excessively high bid for fear that he might win the object at an unnecessarily high price. Consistent with this intuition, we can show that if ϵ is small, then the final assignment will be "almost optimal." In particular, we can show that the total benefit of the final assignment is within $n\epsilon$ of being optimal. A simple self-contained proof of this is given in the appendix; the idea is that an assignment and a set of prices that are almost at equilibrium may be viewed as being at equilibrium for a slightly different problem where all benefits a_{ij} are the same as before, except for the n benefits of the assigned pairs which are modified by an amount no more than ϵ.

Suppose now that the benefits a_{ij} are all integer, which is the typical practical case (if a_{ij} are rational numbers, they can be scaled up to integer by multiplication with a suitable common number). Then, the total benefit of any assignment is integer, so if $n\epsilon < 1$, a complete assignment that is within $n\epsilon$ of being optimal must be optimal. It follows, that if $\epsilon < 1/n$, and the benefits a_{ij} are all integer, then the assignment obtained upon termination of the auction algorithm is optimal. [Actually, with a more careful analysis, we can show that for optimality of the final assignment, it is sufficient that $\epsilon < 1/(n-1)$. This threshold cannot be further improved; my original paper [Bertsekas 1979] gives for every $n \geq 2$ an example where the auction algorithm terminates with a nonoptimal assignment when $\epsilon = 1/(n-1)$.]

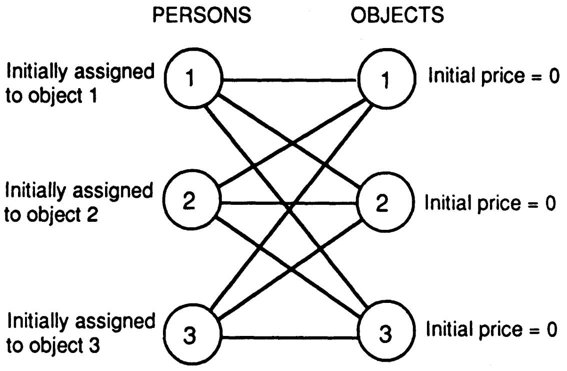
Duality and the Coordinate Descent Interpretation

Just as the final assignment obtained from the auction algorithm is within $n\epsilon$ of being optimal, it turns out that the final set of prices is within $n\epsilon$ of being an optimal solution of a certain dual problem. As shown in the appendix, this dual problem is

$$\text{minimize} \quad \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_{j} \{a_{ij} - p_j\}$$

$$\text{over all prices } p_j, \quad j = 1, \ldots, n. \qquad (9)$$

Figure 5 shows the sequence of generated object prices for the example of Figures 2 and 4 in relation to the contours of the dual cost function of equation (9). It


PERSONS
OBJECTS
Initially assigned to object 1
Initially assigned to object 2
Initially assigned to object 3
1
2
3
1
2
3
Initial price = 0
Initial price = 0
Initial price = 0

Here a_{ij} = C > 0 for all (i,j) with i = 1,2,3 and j = 1,2 and a_{ij} = 0 for all (i,j) with i = 1,2,3 and j = 3

At Start of Round #	Object Prices	Assigned Pairs	Almost Happy Persons	Bidder	Preferred Object	Bidding Increment
1	0, 0, 0	(1,1) (2,2) (3,3)	1, 2	3	2	ε
2	0, ε, 0	(1,1) (2,3) (3,2)	1, 3	2	1	2ε
3	2ε, ε, 0	(1,2) (2,3) (3,1)	2, 3	1	2	2ε
4	2ε, 3ε, 0	(1,2) (2,1) (3,3)	1, 2	3	1	2ε
5	4ε, 3ε, 0	(1,3) (2,1) (3,2)	1, 3	2	2	2ε
6	....	....	....	....	....	....

Figure 4: This figure shows how the auction algorithm overcomes the cycling problem for the example of Figure 2, by making the bidding increment at least equal to ϵ. I give one possible sequence of bids and assignments generated by the auction algorithm, starting with all prices equal to 0. At each round except the last, the person assigned to object 3 bids for either object 1 or 2, increasing its price by ϵ in the first round and by 2ϵ in each subsequent round. In the last round, after the prices of 1 and 2 rise at or above C, object 3 receives a bid and the auction terminates.

can be seen from this figure that each bid has the effect of setting the price of the object receiving the bid nearly equal (within ϵ) to the price that minimizes the dual cost with respect to that price with all other prices held fixed. This observation can be rigorously established [Bertsekas 1988; Bertsekas and Tsitsiklis 1989]. Successive minimization of a cost function along single coordinates is a central feature of coordinate descent and relaxation methods, which are popular for unconstrained mini-

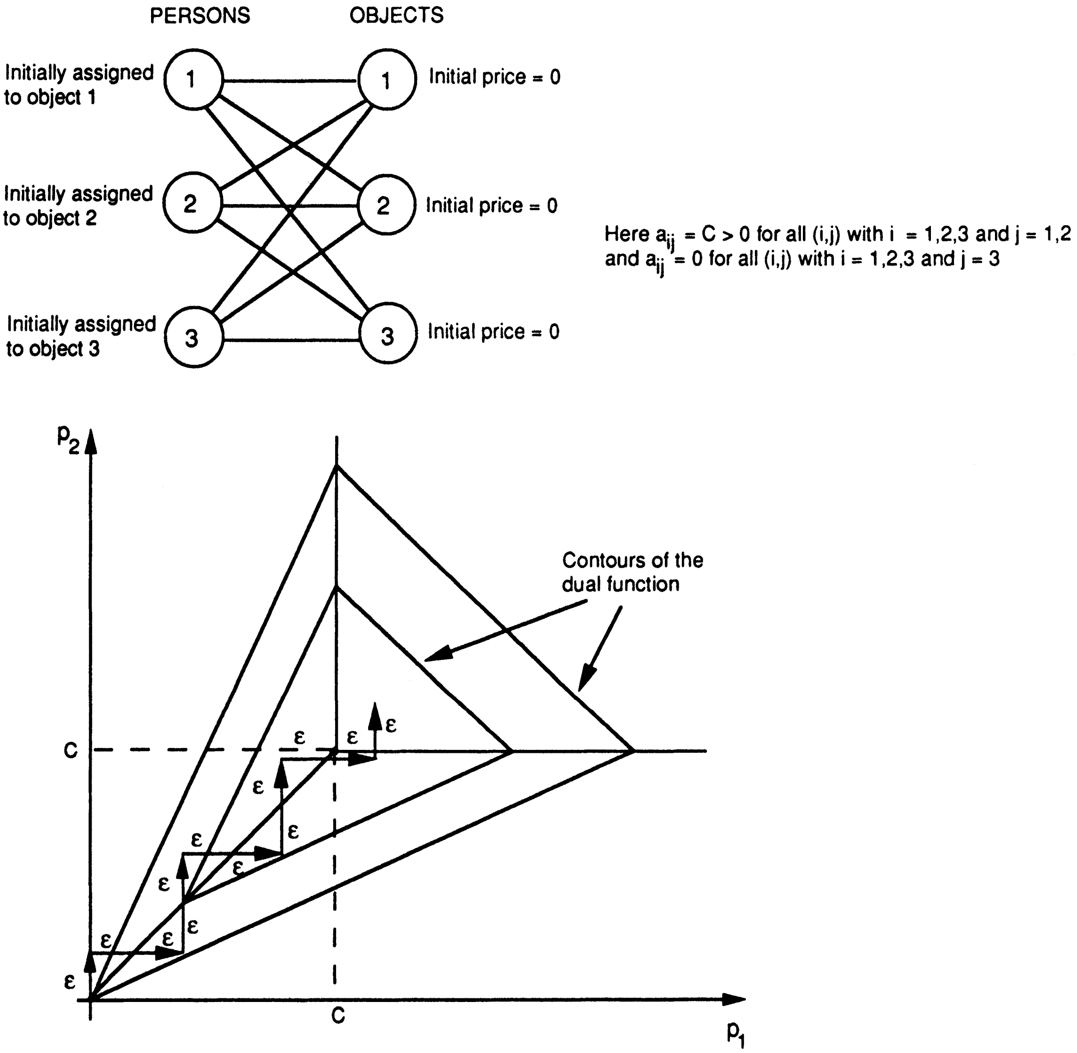

PERSONS
OBJECTS
Initially assigned to object 1
Initially assigned to object 2
Initially assigned to object 3
1
2
3
1
2
3
Initial price = 0
Initial price = 0
Initial price = 0
Here a_{ij} = C > 0 for all (i,j) with i = 1,2,3 and j = 1,2 and a_{ij} = 0 for all (i,j) with i = 1,2,3 and j = 3
p_2
p_1
C
C
Contours of the dual function
ε
ε
ε
ε
ε
ε
ε
ε
ε
ε
ε
ε
ε

Figure 5: Illustration of the sequence of prices generated by the auction algorithm for the example of Figure 4, in relation to the contours of the dual function

$$\sum_{j=1}^{3} p_j + \sum_{i=1}^{3} \max_{j} \{a_{ij} - p_j\},$$

[compare equation (9)] viewed as a function of p_1 and p_2, with p_3 held fixed at its initial price of 0.

mization of smooth functions and for solving systems of smooth equations. The auction algorithm can be interpreted as an approximate coordinate descent method and as such, it is related to relaxation methods for network flow problems [Bertsekas 1982; Bertsekas and Tseng 1985, 1988; Bertsekas and Tsitsiklis 1989], which also resemble coordinate descent methods. There is a fundamental difference here, however; the dual cost function is piecewise linear and thus it is not smooth. It is precisely for this reason that we had to introduce the perturbations implicit in the "almost happiness" or ϵ-complementary slackness condition (7).

In the auction algorithm presented so far, only one person can bid at each round; this version of the auction algorithm is known as the *one-at-a-time* or *Gauss-Seidel implementation,* in view of its similarity with Gauss-Seidel relaxation methods for solving systems of equations [Ortega and Rheinboldt 1970]. An alternative is to calculate at each round the bids of all unassigned persons simultaneously and to raise the prices of objects that receive a bid to the highest bid level. This version is known as the *all-at-once* or *Jacobi implementation,* in view of its similarity to Jacobi relaxation methods for solving systems of equations [Ortega and Rheinboldt 1970]. It is just as valid as the Gauss-Seidel version although it tends to terminate a little slower. It is, however, better suited for parallel computation.

Computational Aspects—ϵ-Scaling

The auction algorithm exhibits interesting computational behavior, and it is essential to understand this behavior to implement the algorithm efficiently.

First note that the amount of work to solve the problem can depend strongly on the value of ϵ and on the maximum absolute object value

$$C = \max_{i,j} |a_{ij}|.$$

Basically, for many types of problems, the number of bidding rounds up to termination tends to be proportional to C/ϵ. This can be seen from the example of Figure 5, where the number of rounds up to termination is roughly C/ϵ, starting from zero initial prices.

Next consider the dependence of the computational requirements on the initial prices; if these prices are "near optimal," we expect that the number of rounds to solve the problem will be relatively small. This can be seen from the example of Figure 5; if the initial prices satisfy $p_1 \approx p_3 + C$ and $p_2 \approx p_3 + C$, the number of rounds up to termination is quite small.

The preceding observations suggest the idea of *ϵ-scaling,* which consists of applying the algorithm several times, starting with a large value of ϵ and successively reducing ϵ up to an ultimate value that is less than some critical value (for example, $1/n$, when the benefits a_{ij} are integer). Each application of the algorithm provides good initial prices for the next application. This is a very common idea in nonlinear programming, encountered, for example, in barrier and penalty function methods. An alternative form of scaling, called *cost scaling,* is based on successively representing the benefits a_{ij} with an increasing number of bits while keeping ϵ at a constant value.

In practice, it is a good idea to at least consider scaling. For sparse assignment problems, that is, problems where the set of feasible assignment pairs is severely restricted, scaling seems almost universally helpful. I convinced myself of this through extensive experimentation when I first proposed the auction algorithm. A related (polynomial) computational complexity analysis of the auction algorithm was given in Bertsekas and Eckstein [1988], using some of the earlier ideas of an ϵ-scaling analysis by Goldberg [1987], and Goldberg and Tarjan [1987], for a different but related method (the ϵ-relaxation method, to be discussed shortly).

A public domain code, called AUCTION, implements the auction algorithm. Roughly, in this code the integer benefits a_{ij} are first multiplied by $n + 1$ and the auction algorithm is applied with progressively lower values of ϵ, to the point where ϵ becomes 1 or smaller (because a_{ij} has been scaled by $n + 1$, it is sufficient for optimality of the final assignment to have $\epsilon \leq 1$). The sequence of ϵ values used is

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

where Δ and θ are parameters set by the user with $\Delta > 0$ and $\theta > 1$. (Typical values for sparse problems are $C/5 \leq \Delta \leq C/2$ and $4 \leq \theta \leq 10$. For nonsparse problems, sometimes $\Delta = 1$, which in effect bypasses ϵ-scaling, works quite well.)

Extensive computational experimentation with the AUCTION code has established that the auction algorithm is very efficient in practice. For sparse problems, it substantially outperforms its principal competitors (see Bertsekas and Castañon [1989b], which contains extensive computational results). Furthermore, the factor of superiority increases with the dimension n, indicating a superior practical computational complexity. For nonsparse problems, the auction algorithm is competitive with its rivals. The practical performance of the auction algorithm is also supported by theoretical computational complexity analysis [Bertsekas and Eckstein 1988; Bertsekas and Tsitsiklis 1989; Bertsekas and Castañon 1989b], which gives it a substantial edge over other popular methods for large and sparse problems.

Figures 6, 7, and 8 give some typical computational results, comparing the AUCTION code with the code of Jonker and Volegnant [1987] (abbreviated as JV), the code APS of Carpaneto, Martello, and Toth [1988], and the code RELAX-II that I developed with P. Tseng [Bertsekas and Tseng 1985, 1988]. Jonker and Volegnant's code has two phases. The first phase is an extensive initialization procedure based on my relaxation method [Bertsekas 1981], and it consists of a sequence of iterations of the naive auction algorithm. To overcome the difficulty with finite termination of the naive auction algorithm, Jonker and Volegnant use a second phase. This phase is based on the Hungarian method and refines the results obtained by the naive auction phase. APS is an efficient implementation of the Hungarian method without the use of the naive auction algorithm in an initialization phase. RELAX-II is an efficient public domain implementation of my general linear network flow relaxation method [Bertsekas 1982; Bertsekas and Tseng 1985]. (RELAX-II is, of course, at a disadvantage here because it treats the assignment problem as a more general network flow problem and ignores much of its special structure.)

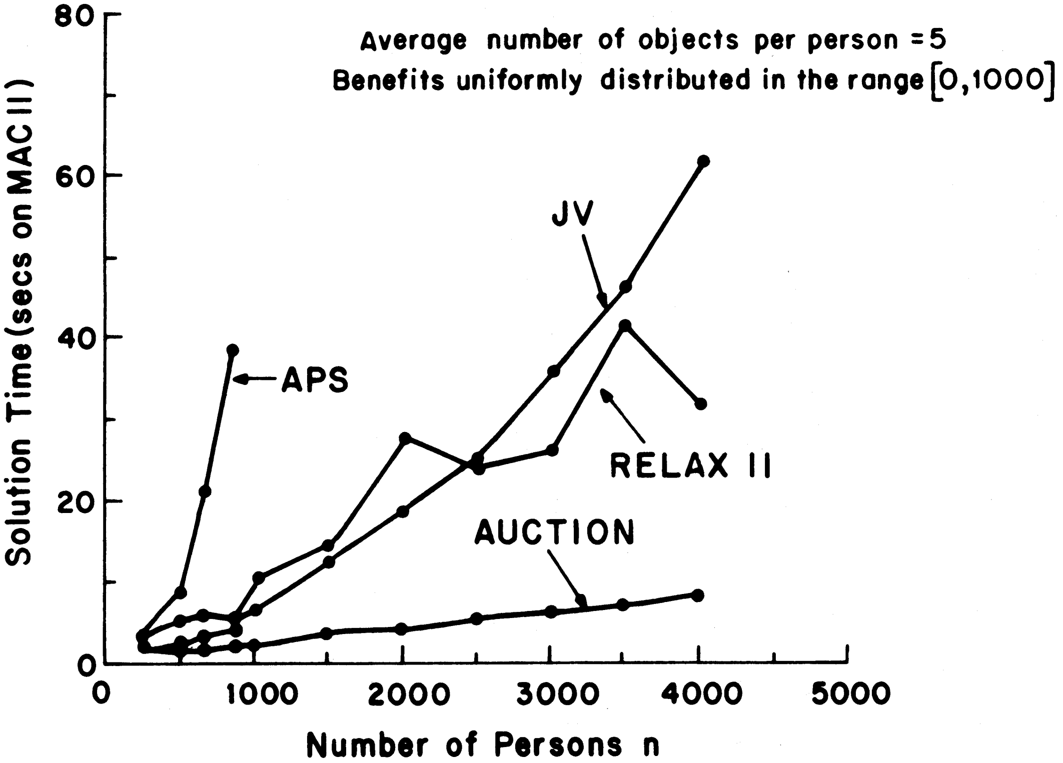


Figure 6: Computational results comparing various codes using a MAC-II on randomly generated problems. For all test problems, the number of feasible assignment pairs is 5*n*, where *n* is the number of persons, and the benefits a_{ij} are integers chosen according to a uniform distribution from the range [0, 1000].

As illustrated in Figures 6–8, AUCTION is almost uniformly faster than the other codes and the factor of superiority increases as the problem becomes more sparse. Note also that JV is greatly superior to APS. Since the Hungarian algorithms used by both codes are essentially equivalent, the superiority of JV must be attributed to the use of the naive auction algorithm for initialization. Indeed, typically, the vast majority of persons (85 to 100 percent) are assigned by the naive auction part of this code.

There have been a number of computational studies involving parallel implementation of the auction algorithm by Bertsekas and Castañon [1989c], Castañon, Smith, and Wilson [forthcoming], Kennington and Wang [1988], Kempa, Kennington, and Zaki [1989], Perry (private communication), and Phillips and Zenios [1988]. Collectively, these studies indicate that the speedup that one can obtain from parallelism is substantial (in the order of four to 10 for sparse problems, and considerably larger for nonsparse problems, depending on the implementation and the machine used). Note also that the Hungarian method has been parallelized recently in an interesting way by Balas et al. [1989]. Their method also admits an asynchronous implementation as

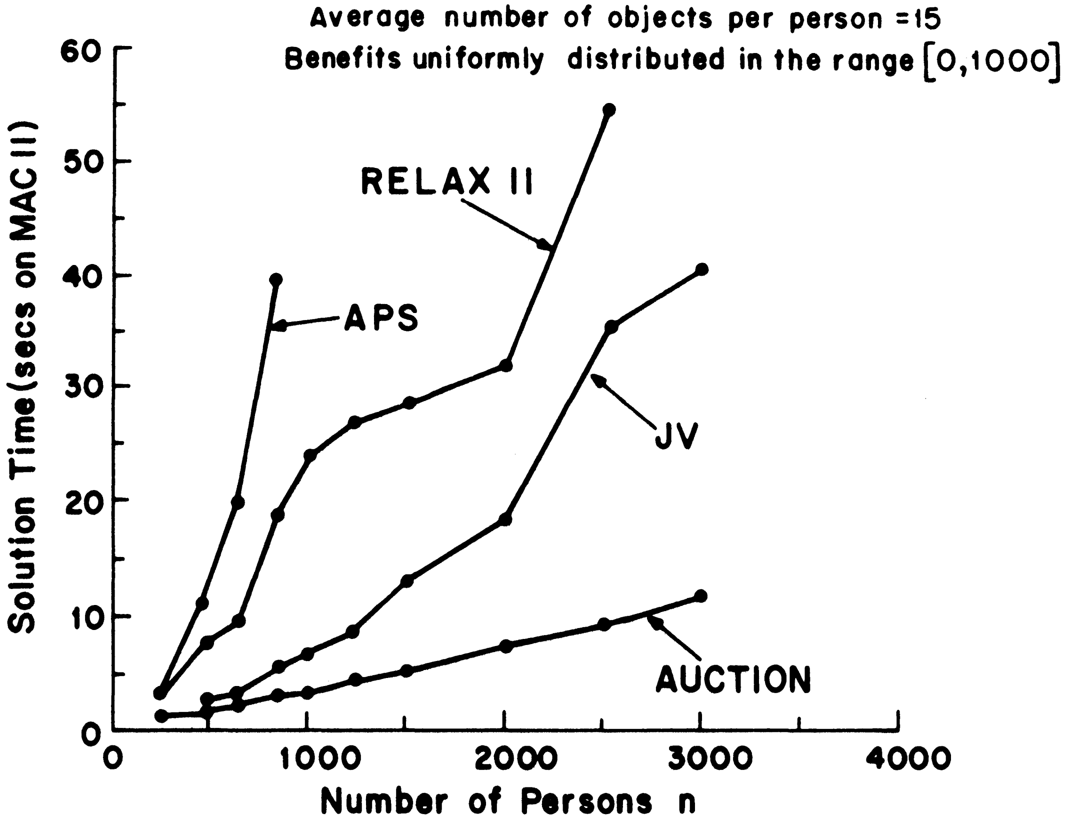


Figure 7: Computational results comparing various codes using a MAC-II on randomly generated problems. For all test problems, the number of feasible assignment pairs is 15*n*, where *n* is the number of persons, and the benefits a_{ij} are integers chosen according to a uniform distribution from the range [0, 1000].

shown in Bertsekas and Castañon [1990]. However, experimental results indicate a generally smaller speedup obtained from parallelization of the Hungarian method than from parallelization of auction.

Variations

A variation of the auction algorithm can be used for asymmetric assignment problems where the number of objects is larger than the number of persons and there is a requirement that all persons be assigned to some object. Naturally, the notion of an assignment must now be modified appropriately. To solve this problem, the auction algorithm need only be modified in the choice of initial conditions. It is sufficient to require that all initial prices be zero. A similar algorithm can be used for the case where there is no requirement that all persons be assigned. Other variations handle efficiently the cases where there are several groups of "identical" persons or objects [Bertsekas and Castañon 1989a].

Parallel and Asynchronous Implementation

Both the bidding and the assignment phases of the auction algorithm are highly parallelizable. This is particularly so for the all-at-once (Jacobi) version of the algorithm, where the bidding and assignment

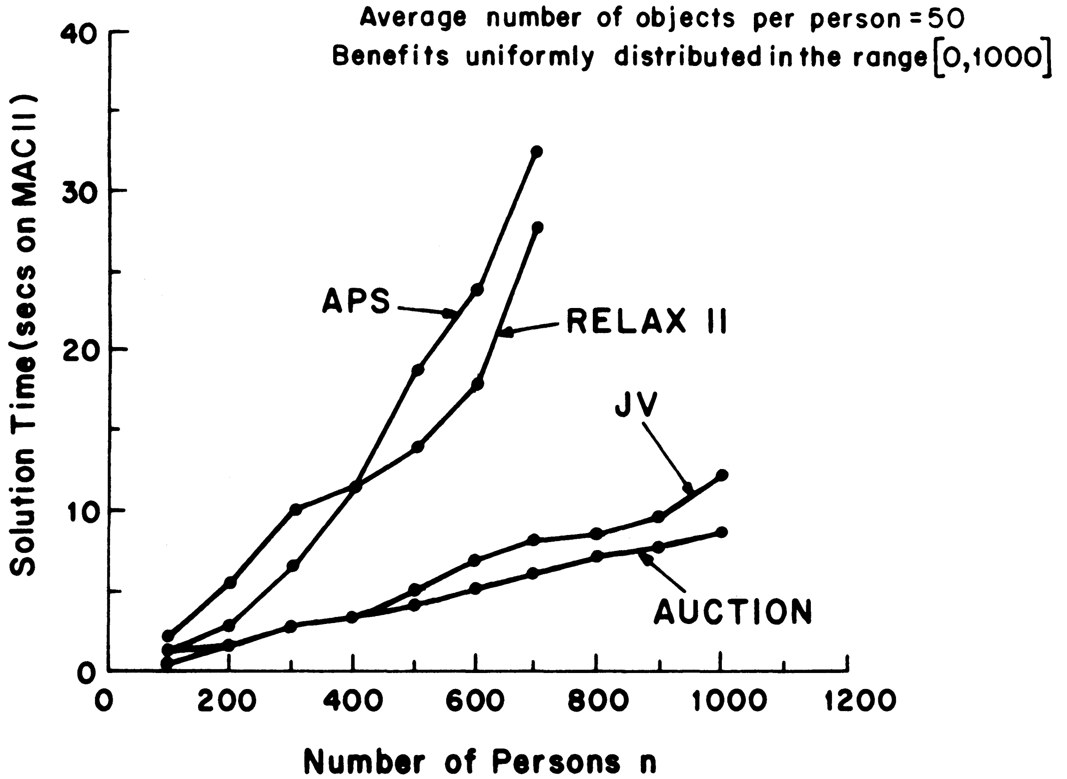


Figure 8: Computational results comparing various codes using a MAC-II on randomly generated problems. For all test problems, the number of feasible assignment pairs is 50*n*, where *n* is the number of persons, and the benefits a_{ij} are integers chosen according to a uniform distribution from the range [0, 1000].

can be carried out for all persons and objects simultaneously. Such an implementation can be termed *synchronous.* There are also *totally asynchronous* implementations of the auction algorithm, which are interesting because they are quite flexible and also tend to result in faster solution in some types of parallel machines. To understand these implementations, it is useful to think of a person as an autonomous decision maker who at unpredictable times obtains information about the prices of the objects. Each person who is not almost happy makes a bid at arbitrary times on the basis of its current object price information (that may be outdated because of communication delays).

Bertsekas and Castañon [1989c] give a careful formulation of the totally asynchronous model, and a proof of its validity. They include also extensive computational results on a shared memory machine, confirming the advantage of asynchronous over synchronous implementations. There are also totally asynchronous models for extensions of the auction algorithms that apply to linear network flow problems [Bertsekas 1986a; Bertsekas and Eckstein 1988; Bertsekas and Tsitsiklis 1989].

Extension to Transportation and Minimum-Cost-Flow Problems

David Castañon and I have extended the

auction algorithm to solve linear transportation problems [Bertsekas and Castañon 1989a]. The basic idea is to convert the transportation problem into an assignment problem by creating multiple copies of persons (or objects) for each source (or sink respectively), and then to modify the auction algorithm to take advantage of the presence of the multiple copies. We give computational results with a code called TRANSAUCTION, showing that this auction algorithm is considerably faster than its chief competitors for important classes of transportation problems. Generally these problems are characterized by two properties, homogeneity and asymmetry. A homogeneous problem is one for which there are only few levels of supply and demand. An asymmetric problem is one for which the number of sources is much larger than the number of sinks. For other types of transportation problems, the auction algorithm is outperformed by, for example, the relaxation code RELAX-II. The computational complexity of transportation-auction is studied by Bertsekas and Castañon [1989b].

There are extensions of the auction algorithm for linear minimum cost flow (transshipment) problems. One such extension is the *ϵ-relaxation method* first proposed in Bertsekas [1986a, 1986b]; see also Bertsekas and Eckstein [1987, 1988], Bertsekas and Tsitsiklis [1989], and Goldberg and Tarjan [1987] for a detailed description and analysis. This method has interesting theoretical properties and, like the auction algorithm, is well suited for parallelization. However, for general transshipment problems, its practical performance has yet to match that of relaxation methods (for example, the RELAX-II code); further research may change this assessment.

Concluding Remarks

The auction algorithm is an intuitive method based on new and interesting computational ideas. It performs very well on serial machines, and it is also well suited for implementation in parallel machines, in both a synchronous and an asynchronous mode. Auction-like algorithms for network flow problems more general than assignment have been developed only recently. Much remains to be done to properly extend them and to realize their full potential.

To foster research in the network optimization area, I have placed the code AUCTION in the public domain. Paul Tseng and I have also placed the code RELAX-II in the public domain. You can obtain these codes from me at no cost.

Acknowledgments

This work was supported by the Army Research Office under Contract No. DAAL 03-86-K-0171. The suggestions of Paul Tseng on the presentation style were very helpful and are greatly appreciated. Thanks are also due to David Castañon for several useful comments.

APPENDIX: Relation of Equilibria with Primal and Dual Optimality

Let us fix $\epsilon \geq 0$. In this appendix, we show that given an assignment $\{(i, j_i) | i = 1, \ldots, n\}$ and a set of prices $\{\bar{p}_j | j = 1, \ldots, n\}$, which are almost at equilibrium (if $\epsilon > 0$) or at equilibrium (if $\epsilon = 0$), then the assignment is within $n\epsilon$ of maximizing the total benefit and is optimal if $\epsilon = 0$. Furthermore, the set of prices is within $n\epsilon$ of minimizing a certain dual cost function.

Let $\epsilon > 0$. The total benefit of *any* assignment $\{(i, k_i) \mid i = 1, \dots, n\}$ satisfies

$$\sum_{i=1}^{n} a_{ik_i} \le \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_{j} \{a_{ij} - p_j\},$$

for any set of prices $\{p_j \mid j = 1, \dots, n\}$, since the second term of the right-hand side is no less than

$$\sum_{i=1}^{n} (a_{ik_i} - p_{k_i}),$$

while the first term is equal to $\sum_{i=1}^{n} p_{k_i}$. Therefore,

$$A^* \le D^*,$$

where A^* is the optimal total assignment benefit

$$A^* = \max_{\substack{k_i, i=1,\dots,n \\ k_i \neq k_m \text{ if } i \neq m}} \sum_{i=1}^{n} a_{ik_i}$$

and

$$D^* = \min_{\substack{p_j \\ j=1,\dots,n}} \left\{ \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_{j} \{a_{ij} - p_j\} \right\}.$$

On the other hand, since all persons are almost happy with the given assignment $\{(i, j_i) \mid i = 1, \dots, n\}$ and set of prices $\{\bar{p}_j \mid j = 1, \dots, n\}$, we have

$$\max_{j=1,\dots,n} \{a_{ij} - \bar{p}_j\} - \epsilon \le a_{ij_i} - \bar{p}_{j_i},$$

and by adding this relation over all i, we see that

$$D^* \le \sum_{i=1}^{n} (\bar{p}_{j_i} + \max_{j} \{a_{ij} - \bar{p}_j\})$$

$$\le \sum_{i=1}^{n} a_{ij_i} + n\epsilon \le A^* + n\epsilon.$$

Since we showed earlier that $A^* \le D^*$, it follows that the total assignment benefit $\sum_{i=1}^{n} a_{ij_i}$ is within $n\epsilon$ of the optimal value A^*.

The function

$$\sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_{j} \{a_{ij} - p_j\},$$

appearing in the definition of D^*, may be viewed as a dual function of the price variables p_j, and its minimization may be viewed as a dual problem in the standard linear programming duality context; see Bertsekas [1988], Bertsekas and Tsitsiklis [1989], Dantzig [1963], Papadimitriou and Steiglitz [1982], and Rockafellar [1984]. It is seen from the preceding analysis, that the prices $\bar{p}_j$ attain within $n\epsilon$ the dual optimal value D^*.

If we let $\epsilon = 0$ in the preceding argument, we see that $A^* = D^*$ and that an assignment and a set of prices that are at equilibrium maximize the total benefit and minimize the dual function, respectively.

References

Ahuja, R. K.; Goldberg, A. V.; Orlin, J. B.; and Tarjan, R. E. 1988, "Finding minimum-cost flows by double scaling," Sloan Working Paper No. 2047-88, Massachusetts Institute of Technology, Cambridge, Massachusetts, (August).

Balinski, M. L. 1985, "Signature methods for the assignment problem," *Operations Research*, Vol. 33, No. 3, pp. 527–537.

Balinski, M. L. 1986, "A competitive (dual) simplex method for the assignment problem," *Mathematical Programming*, Vol. 34, No. 2, pp. 125–141.

Balas, E.; Miller, D.; Pekny, J.; and Toth, P. 1989, "A parallel shortest path algorithm for the assignment problem," Management Science Report MSRR 552, Carnegie-Mellon University, Pittsburgh, Pennsylvania, (April).

Barr, R.; Glover, F.; and Klingman, D. 1977, "The alternating basis algorithm for assignment problems," *Mathematical Programming*, Vol. 13, No. 1, pp. 1–13.

Bertsekas, D. P. 1979, "A distributed algorithm for the assignment problem," Laboratory for Information and Decision Systems Working Paper, Massachusetts Institute of Technology, Cambridge, Massachusetts, (March).

Bertsekas, D. P. 1981, "A new algorithm for the assignment problem," *Mathematical Programming*, Vol. 21, No. 2, pp. 152–171.

Bertsekas, D. P. 1982, "A unified framework for minimum cost network flow problems," Laboratory for Information and Decision Systems Report LIDS-P-1245-A, Massachusetts Institute of Technology, Cambridge, Massachusetts; also in *Mathematical Programming,* 1985, Vol. 32, No. 2, pp. 125–145.

Bertsekas, D. P. 1985, "A distributed asynchronous relaxation algorithm for the assignment problem," *Proceedings 24th IEEE Conference on Decision and Control,* pp. 1703–1704.

Bertsekas, D. P. 1986a, "Distributed asynchronous relaxation methods for linear network flow problems," LIDS Report P-1606, Massachusetts Institute of Technology, Cambridge, Massachusetts, (November).

Bertsekas, D. P. 1986b, "Distributed relaxation methods for linear network flow problems," *Proceedings of 25th IEEE Conference on Decision and Control,* pp. 2101–2106.

Bertsekas, D. P. 1988, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research,* Vol. 14, pp. 105–123.

Bertsekas, D. P. and Castañon, D. A. 1989a, "The auction algorithm for transportation problems," *Annals of Operations Research,* Vol. 20, pp. 67–96.

Bertsekas, D. P. and Castañon, D. A. 1989b, "The auction algorithm for the minimum cost network flow problem," Laboratory for Information and Decision Systems Report LIDS-P-1925, Massachusetts Institute of Technology, Cambridge, Massachusetts, (November).

Bertsekas, D. P. and Castañon, D. A. 1989c, "Parallel synchronous and asynchronous implementations of the auction algorithm," Alphatech Report, Burlington, Massachusetts, (November), to appear in *Parallel Computing.*

Bertsekas, D. P. and Castañon, D. A. 1990, "Parallel asynchronous Hungarian methods for the assignment problem," Alphatech Report, Burlington, Massachusetts, (January).

Bertsekas, D. P. and Eckstein, J. 1987, "Distributed asynchronous relaxation methods for linear network flow problems," *Proceedings of IFAC '87,* Munich, Germany, (July).

Bertsekas, D. P. and Eckstein, J. 1988, "Dual coordinate step methods for linear network flow problems," *Mathematical Programming,* Series B, Vol. 42, No. 2, pp. 203–243.

Bertsekas, D. P. and Tseng, P. 1985, "Relaxation methods for minimum cost ordinary and generalized network flow problems," LIDS Report P-1462, Massachusetts Institute of Technology, Cambridge, Massachusetts, (May); also *Operations Research,* 1988, Vol. 36, No. 1, pp. 93–114.

Bertsekas, D. P. and Tseng, P. 1988, "RELAX: A computer code for minimum cost network flow problems," *Annals of Operations Research,* Vol. 13, pp. 127–190.

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

Carpaneto, G.; Martello, S.; and Toth, P. 1988, "Algorithms and codes for the assignment problem," *Annals of Operations Research,* Vol. 13, pp. 193–223.

Castañon, D.; Smith, B.; and Wilson, A. forthcoming, "Performance of parallel assignment algorithms on different multiprocessor architectures," Argonne National Laboratory Report.

Dantzig, G. B. 1963, *Linear Programming and Extensions,* Princeton University Press, Princeton, New Jersey.

Derigs, U. 1985, "The shortest augmenting path method for solving assignment problems—Motivation and computational experience," *Annals of Operations Research,* Vol. 4, pp. 57–102.

Engquist, M. 1982, "A successive shortest path algorithm for the assignment problem," *INFOR,* Vol. 20, No. 4, pp. 370–384.

Ford, L. R., Jr. and Fulkerson, D. R. 1962, *Flow in Networks,* Princeton University Press, Princeton, New Jersey.

Gabow, H. N. and Tarjan, R. E. 1987, "Faster scaling algorithms for graph matching," working paper.

Glover, F.; Glover, R.; and Klingman, D. 1982, "Threshold assignment algorithm," Center for Business Decision Analysis Report CBDA 107, Graduate School of Business, University of Texas at Austin, (September).

Goldberg, A. V. 1987, "Efficient graph algorithms for sequential and parallel computers," Tech. Report TR-374, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, (February).

Goldberg, A. V. and Tarjan, R. E. 1987, "Solving minimum cost flow problems by successive approximation," *Proceedings 19th ACM STOC,* (May).

Goldfarb, D. 1985, "Efficient dual simplex methods for the assignment problem," *Mathematical Programming,* Vol. 33, No. 2, pp. 187–203.

Hall, M., Jr. 1956, "An algorithm for distinct representatives," *American Mathematical Monthly,* Vol. 51, No. 9, pp. 716–717.

Held, M. and Karp, R. M. 1970, "The traveling salesman problem and minimal spanning trees," *Operations Research,* Vol. 18, No. 6, pp. 1138–1162.

Held, M. and Karp, R. M. 1971, "The traveling salesman problem and minimal spanning trees: Part II," *Mathematical Programming,* Vol. 1, No. 1, pp. 6–25.

Hung, M. 1983, "A polynomial simplex method for the assignment problem," *Operations Research,* Vol. 31, No. 3, pp. 595–600.

Jonker, R. and Volegnant, A. 1987, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing,* Vol. 38, No. 4, pp. 325–340.

Kempa, D.; Kennington, J.; and Zaki, H. 1989, "Performance characteristics of the Jacobi and Gauss-Seidel versions of the auction algorithm on the Alliant FX/8," Report OR-89-008, Department of Mechanical and Industrial Engineering, University of Illinois, Champaign–Urbana, Illinois.

Kennington, J. and Wang, Z. 1988, "Solving dense assignment problems on a shared memory multiprocessor," Tech. Report 88-OR-16, Department of Operations Research and Applied Science, Southern Methodist University, (October).

Kuhn, H. W. 1955, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly,* Vol. 2, No. 1, pp. 83–97.

McGinnis, L. F. 1983, "Implementation and testing of a primal-dual algorithm for the assignment problem," *Operations Research,* Vol. 31, No. 2, pp. 277–291.

Minty, G. J. 1960, "Monotone networks," *Proceedings Royal Society of London,* A, Vol. 257, No. 1289, pp. 194–212.

Ortega, J. M. and Rheinboldt, W. C. 1970, *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York.

Papadimitriou, C. H. and Steiglitz, K. 1982, *Combinatorial Optimization: Algorithms and Complexity,* Prentice-Hall, Englewood Cliffs, New Jersey.

Perry, E. L., private communication.

Phillips, C. and Zenios, S. A. 1988, "Experiences with large scale network optimization on the Connection Machine," Report 88-11-05, Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia, Pennsylvania, (November).

Rockafellar, R. T. 1984, *Network Flows and Monotropic Programming,* Wiley-Interscience, New York.

Thompson, G. L. 1981, "A recursive method for solving assignment problems," in *Studies on Graphs and Discrete Programming,* ed. P. Hansen, North-Holland Publishing Company, Amsterdam, pp. 319–343.