September 1989 LIDS - P - 1908

THE AUCTION ALGORITHM FOR ASSIGNMENT AND OTHER NETWORK FLOW PROBLEMS[1]

by

Dimitri P. Bertsekas[2]

Abstract

This is a tutorial presentation of the auction algorithm, an intuitive method for solving the classical assignment problem. The algorithm 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. We derive the algorithm from first principles, we explain its computational properties, and we discuss its extensions to transportation and transhipment problems.

[1] Supported by the Army Research Office under Contract No. DAAL 03-86-K-0171. Thanks are due to David Castanon and Paul Tseng for their helpful comments.
[2] Department of Electrical Engineering and Computer Science, M. I. T., Cambridge, Mass., 02139.

1. THE ASSIGNMENT PROBLEM

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), which is such that the corresponding total benefit $\sum_{i=1}^{n} a_{ij_i}$ is maximal.

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 [HeK70], [HeK71].

The assignment problem is also of great theoretical importance because, despite its simplicity, it embodies a fundamental linear programming structure. It can be shown that what is perhaps the most important class of linear programming problems, linear network flow problems, can be reduced to the assignment problem by means of a simple reformulation (see [BeT89], p. 335, or [PaS82], 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 [FoF62], [Min60], was motivated and developed through Kuhn's Hungarian method [Kuh55], the first specialized method for the assignment problem. (The name of the algorithm honors its connection with the work of the Hungarian mathematician Egervary, dating to 1931.)

There has also been a plethora of algorithms for the assignment problem in the thirty five years since Kuhn's original proposal; for a representative but incomplete sample, see [Bal85], [Bal86], [BGK77], [Ber81], [CMT88], [Der85], [Eng82], [GGK82], [Gol85], [Hal56], [Hun83], [JoV87], [McG83], [Mun56], and [Tho81]. 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, in dual simplex methods, and in relaxation methods).

The auction algorithm, first proposed by the author in [Ber79] and further discussed recently in [Ber85], [Ber88], represents a significant departure 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 which is not quite optimal. The auction algorithm was originally conceived as a method for masssively parallel solution of the assignment problem, but it has also turned out to be very effective for serial computation.

Our intent in this paper is to provide an intuitive understanding of the auction algorithm. To this end, we introduce an economic equilibrium problem that turns out to be equivalent to the assignment problem as we proceed to explain.

2. PRICES AND EQUILIBRIA

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 e.g. [Dan63], [PaS82]; in the terminology of linear programming, the "happiness" relation (1) is known as *complementary slackness*). We provide a simple, first principles, proof of the relation of equilibria to optimal assignments and dual optimization in Appendix A, but for simplicity, we will not emphasize linear programming and duality in this paper.

3. AN AUCTION PROCESS

Let us consider now a natural process for finding an equilibrium assignment. We will call this process the *naive auction algorithm*, because we will soon discover that 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,

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

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 is the best object value,

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

and w_i is the second best object value

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

that is, the best value over objects other than j_i. (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, where at each round the bidder i raises the price of his preferred object by the ***bidding increment*** γ_i. Note that γ_i cannot be negative since $v_i \geq w_i$ [cf. Eqs. (5) and (6)], so the object prices tend to increase. The choice γ_i is illustrated in Fig. 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.

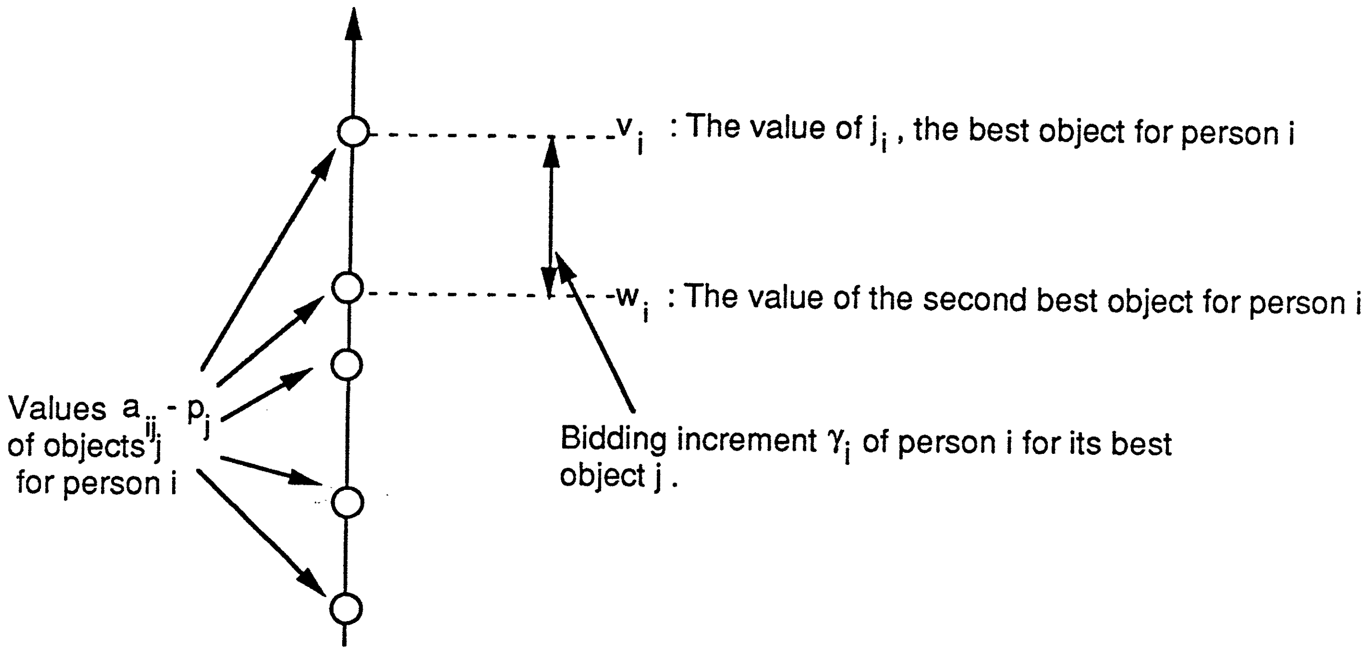
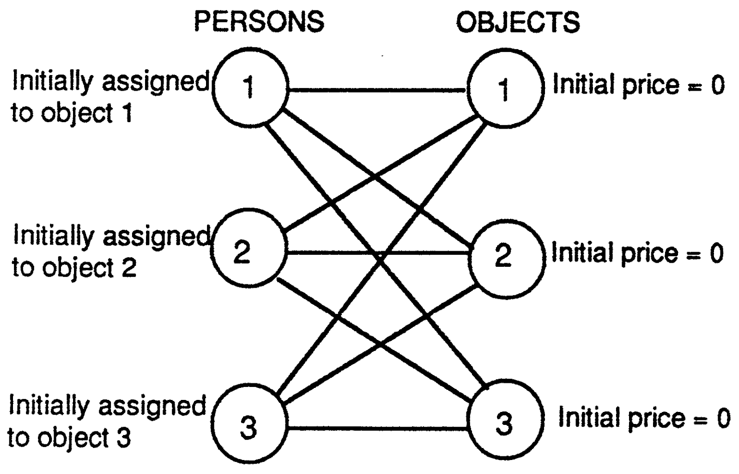

v_i : The value of j_i, the best object for person i
w_i : The value of the second best object for person i
Values a_{ij} - p_j of objects j for person i
Bidding increment γ_i of person i for its best object j.

Figure 1: Bidding increment γ_i in the naive auction algorithm. Even after the price of j_i is increased by this amount, j_i continues to be the preferred object, so the bidder i is happy following the round. Note that we have $\gamma_i = 0$ if there are two objects most preferred by the bidder i.

Does this auction process work? Unfortunately, not always. The difficulty is that the bidding increment γ_i is zero when there are more than one objects offering maximum value for the bidder i.


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 ($\gamma_i = 0$, cf. Fig. 1).

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; see Fig. 2.

To break such cycles, we introduce a 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 occassion 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 e.g. [Ber86a], [BeC89b], [BeE88], [GaT87], and [GoT87]). For $\epsilon = 0$ it reduces to ordinary complementary slackness [cf. Eq. (1)].

We now reformulate the previous auction process so that the bidding increment is always at least equal to ϵ. The resulting method, which we henceforth call 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 Eq. (4)]. With this choice, the bidder of a round is almost happy at the end of the round (rather than happy), as illustrated in Fig. 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.

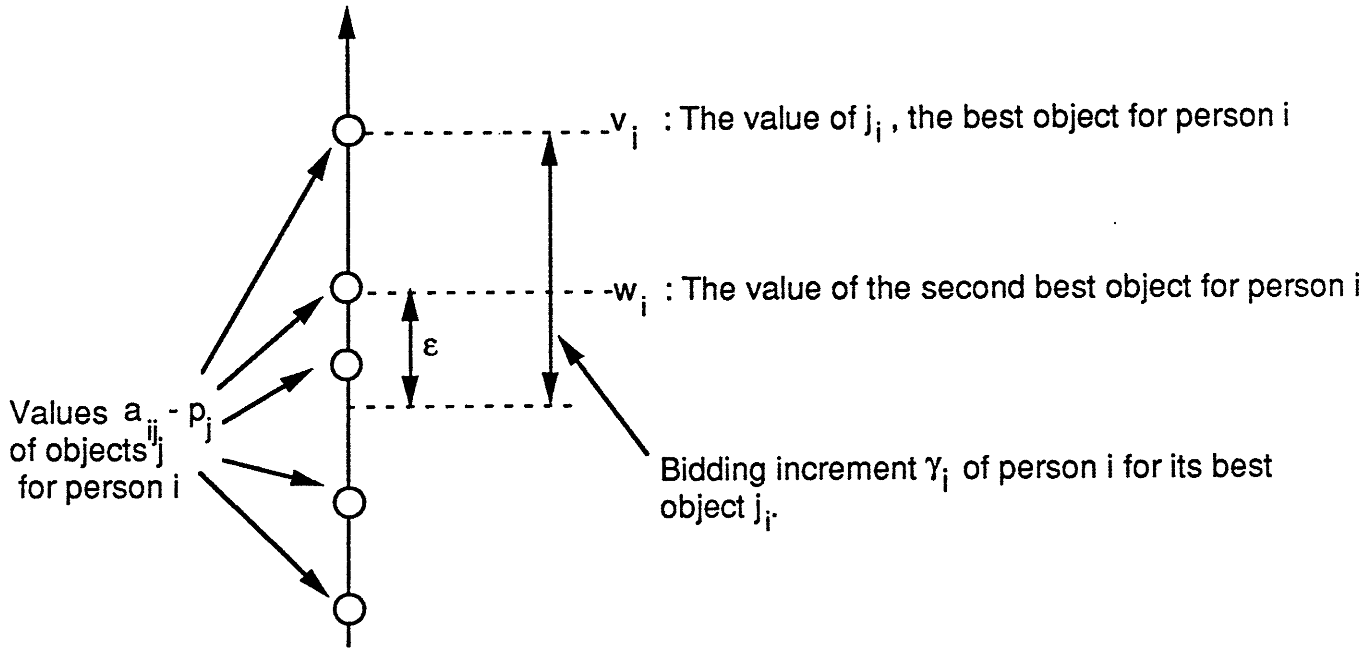

v_i : The value of j_i, the best object for person i
w_i : The value of the second best object for person i
ε
Values $a_{ij} - p_j$ of objects j for person i
Bidding increment γ_i of person i for its best object j_i.

Figure 3: Illustration of the bidding increment γ_i in the auction algorithm. Even after the price of the preferred object j_i is increased by this amount, 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 which are almost at equilibrium. To see this, we 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). We 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 there is a limited number of rounds at which an object can receive a bid, while there is still some other object that has not yet received any bid. Therefore, there are two possibilities: either (a) the auction terminates (in a finite number of rounds) or (b) the auction will continue until all objects receive at least one bid, at which time all persons will be almost happy, and the auction will terminate. (This argument uses the assumption 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 there exists at least one feasible assignment.) Figure 4 shows how the auction algorithm, based on the bidding increment (8), overcomes the cycling problem of the example of Fig. 2.

Optimality Properties at Termination

When the auction algorithm terminates, we have an assignment which 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 the object might be won 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 Appendix A; 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*

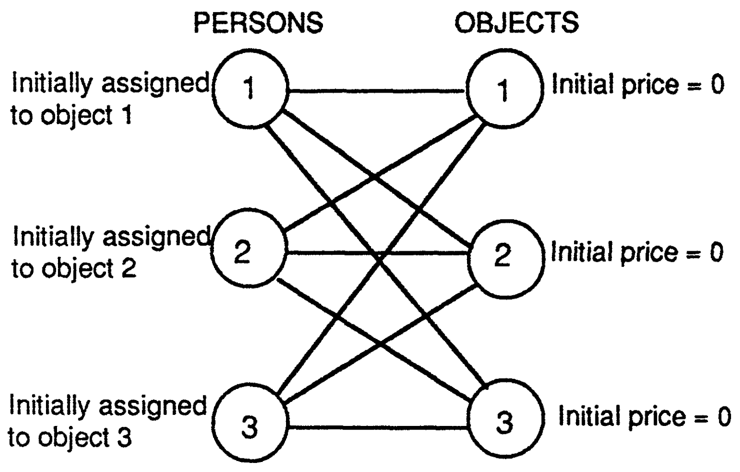
$$\epsilon < \frac{1}{n},$$

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; the original paper [Ber79] gives for every $n \geq 2$, an example where the auction algorithm terminates with a nonoptimal assignment when $\epsilon = 1/(n-1)$.]

4. 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 Appendix A, this dual problem is

$$\min_{\substack{p_j \\ j=1,\ldots,n}} \left\{ \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \max_j \{a_{ij} - p_j\} \right\}. \tag{9}$$


PERSONS
OBJECTS
Initially assigned to object 1
Initially assigned to object 2
Initially assigned to object 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: Illustration of how the auction algorithm overcomes the cycling problem for the example of Fig. 2, by making the bidding increment at least equal to ϵ. We 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.

Figure 5 shows the sequence of generated object prices for the example of Figs. 2 and 4 in relation to the contours of the dual cost function of Eq. (9). It 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 and generalized (we refer the reader to [Ber88] and [BeT89] for the technical details). Successive minimization of a cost function along single coordinates is a central feature of coordinate descent and relaxation methods, which are popular for unconstrained minimization 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 [Ber82], [BeT85], [BeT88], [BeT89], 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 [OrR70]. 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 with Jacobi relaxation methods for solving systems of equations [OrR70]. 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.

5. COMPUTATIONAL ASPECTS – ϵ-SCALING

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

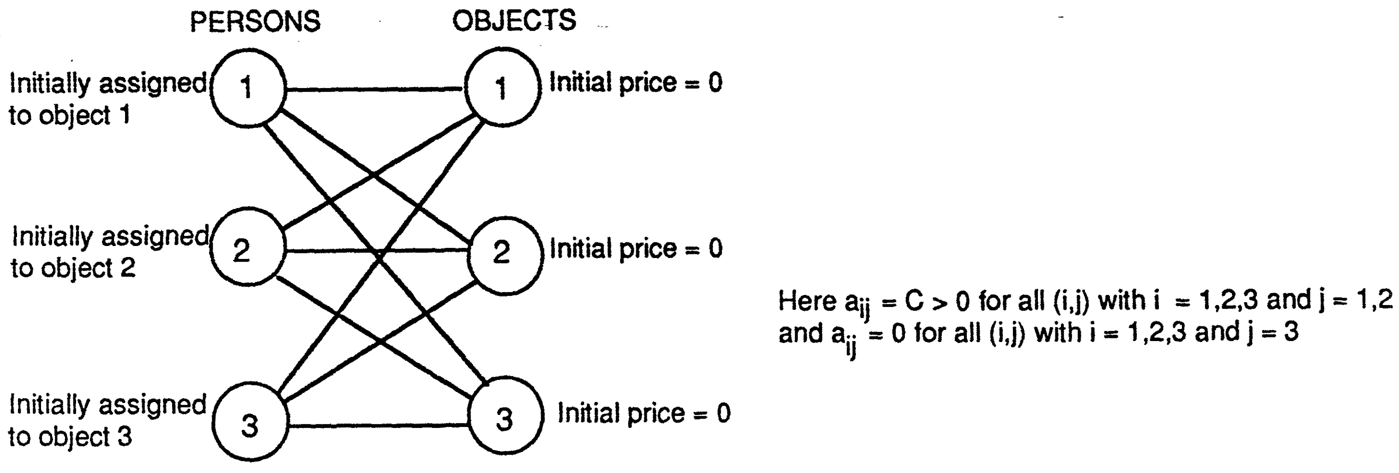
We first note that the amount of work to solve the problem can depend strongly on the value of ϵ and on the maximum absolute object value

$$C = \max_{i,j} |a_{ij}|. \tag{9}$$

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 Fig. 5, where the number of rounds up to termination is roughly C/ϵ, starting from zero initial prices.

We note also that there is a dependence on the initial prices; if these prices are "near optimal", it can be expected that the number of rounds to solve the problem will be relatively small. This can be seen from the example of Fig. 5; if the initial prices satisfy $p_1 \approx p_3 + C$ and $p_2 \approx p_3 + C$, it can be seen that 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 which is less than the critical value $1/n$. 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 penalty function methods. An alternative form of scaling, called *cost scaling*, is


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

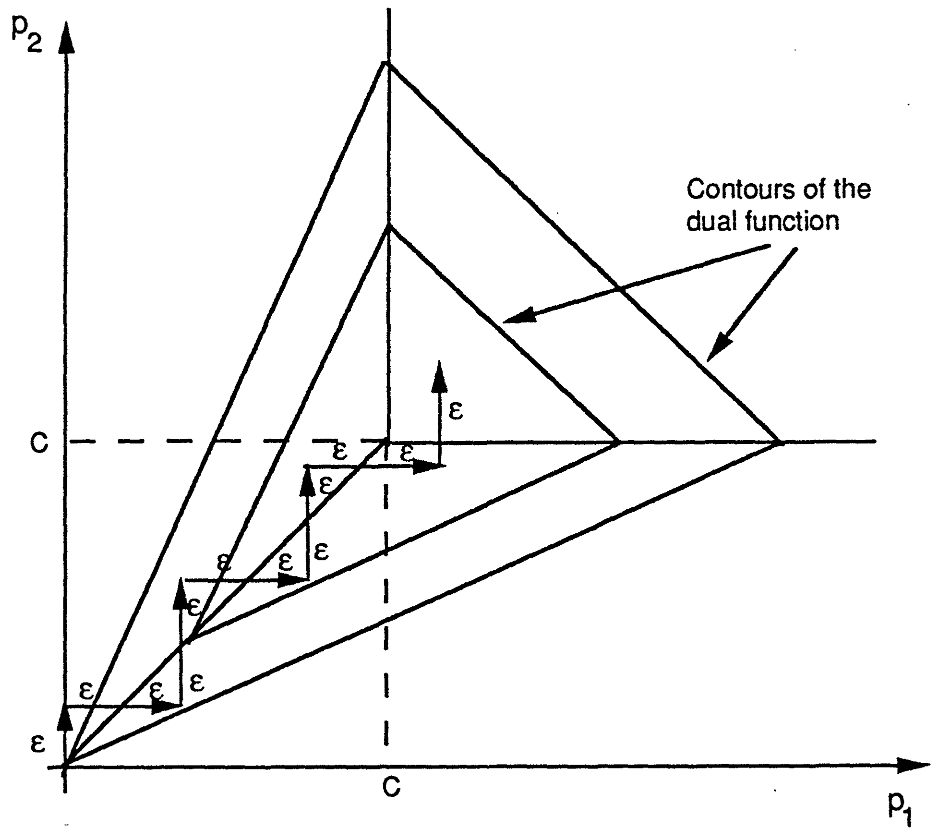


Figure 5: Illustration of the sequence of prices generated by the auction algorithm for the example of Fig. 4, in relation to the contours of the dual function

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

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

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. This was established experimentally at the time of the original proposal of the auction algorithm. A related (polynomial) computational complexity analysis of the auction algorithm was given in [BeE88], [BeT89], using some of the earlier ideas of an ϵ-scaling analysis [Gol87], [GoT87], for a different but related method (the ϵ-relaxation method, to be discussed shortly).

There is a public domain code, called AUCTION, which implements the auction algorithm and is available at no cost from the author. Roughly, in this code the integer benefits a_{ij} are first multiplied by $n+1$ and the auction algorithm is applied with progressively lower value of ϵ, up 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

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

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 outperforms substantially its principal competitors (see [BeC89a] and [BeC89b], which contain 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 [BeE88], [BeT89], [BeC89b], 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 [JoV87] (abbreviated as JV), the code APS of [CMT88], and the code RELAX-II developed by P. Tseng and the author [BeT88]. JV is a two-phase method; the first phase is an extensive initialization procedure based on the relaxation method of [Ber81] and the second phase is an implementation of the Hungarian method based on the use of sequential shortest paths. APS is an efficient implementation of the Hungarian method without the use of sequential shortest paths. RELAX-II is an efficient public domain implementation of the general linear network flow relaxation method of [Ber82] and [BeT85]. (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.) Note from Figs. 6-8 that AUCTION is almost uniformly faster than the other codes and that the factor of superiority increases as the problem becomes more sparse.

There have been also a number of computational studies involving parallel implementation of the auction algorithm by among others, D. Castanon, L. Hatai, J. Kennington, E. L. Perry, H. Zaki, and S. Zenios. Collectively, these studies indicate that the speeedup that one can obtain from parallelism is substantial (in the order of four to ten for sparse problems and considerably larger for nonsparse

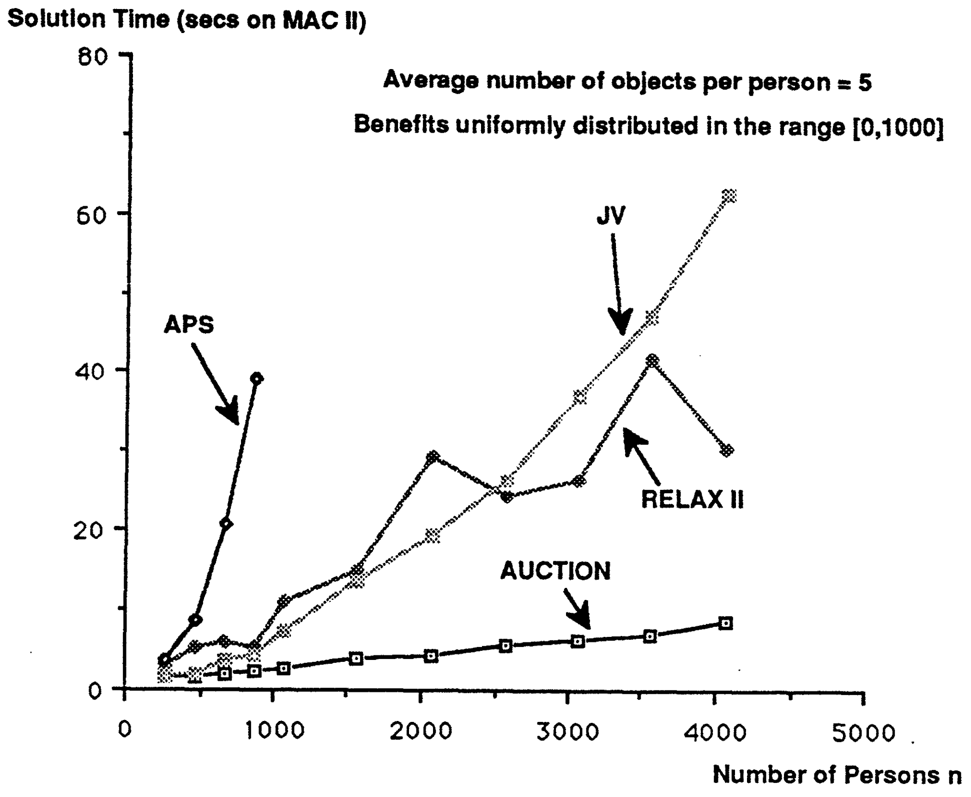


Figure 6: Computational results comparing various codes on a MAC-II on randomly generated problems. For all test problems, the number of feasible assignment pairs is $5n$, where n is the number of persons, and the benefits a_{ij} are integers chosen according to a uniform distribution from the range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

problems, depending on the implementation and the machine used).

6. VARIATIONS AND EXTENSIONS

The auction algorithm can be extended for solution of a number of important linear network flow problems.

Asymmetric Assignment Problems

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. There is also a similar algorithm for the case where there is no requirement that all persons be assigned.

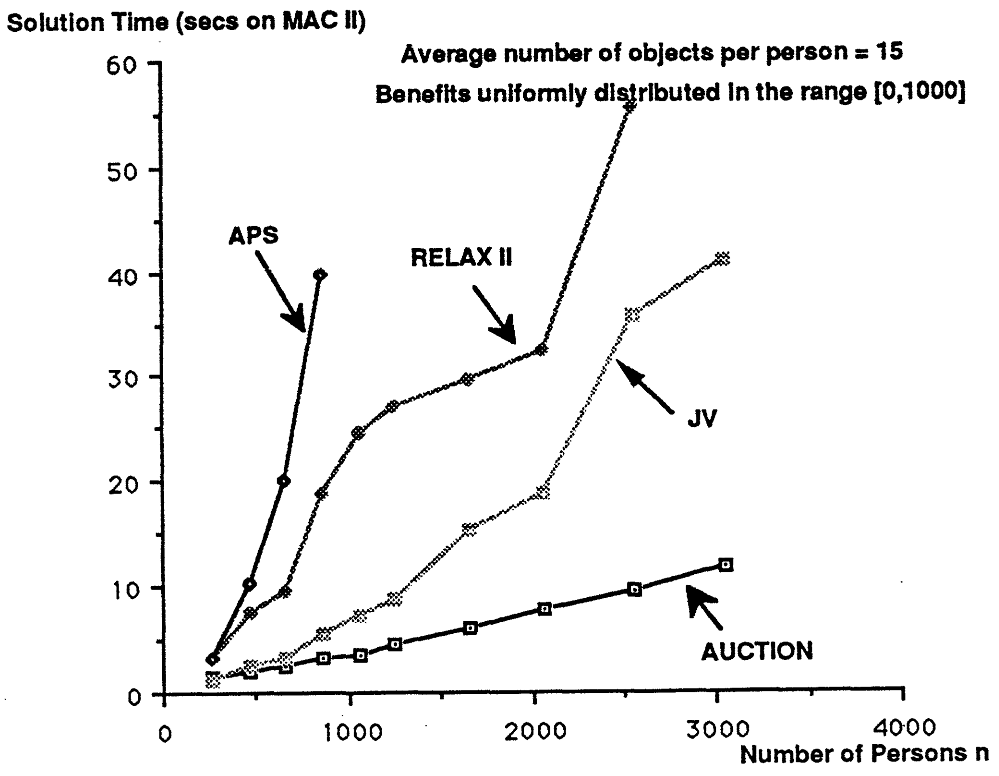


Figure 7: Computational results comparing various codes on a MAC-II on randomly generated problems. For all test problems, the number of feasible assignment pairs is $15n$, where n is the number of persons, and the benefits a_{ij} are integers chosen according to a uniform distribution from the range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

Parallel and Asynchronous Implementation

It is clear that 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 can be carried out simultaneously 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 that obtains at unpredictable times 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). A careful formulation of the totally asynchronous model, and a proof of its validity is given in [Ber86a], [BeE88], and [BeT89].

Extension to Transportation and Minimum Cost Flow Problems

The auction algorithm has been extended by D. Castanon and the author to solve linear transportation problems [BeC89a]. 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

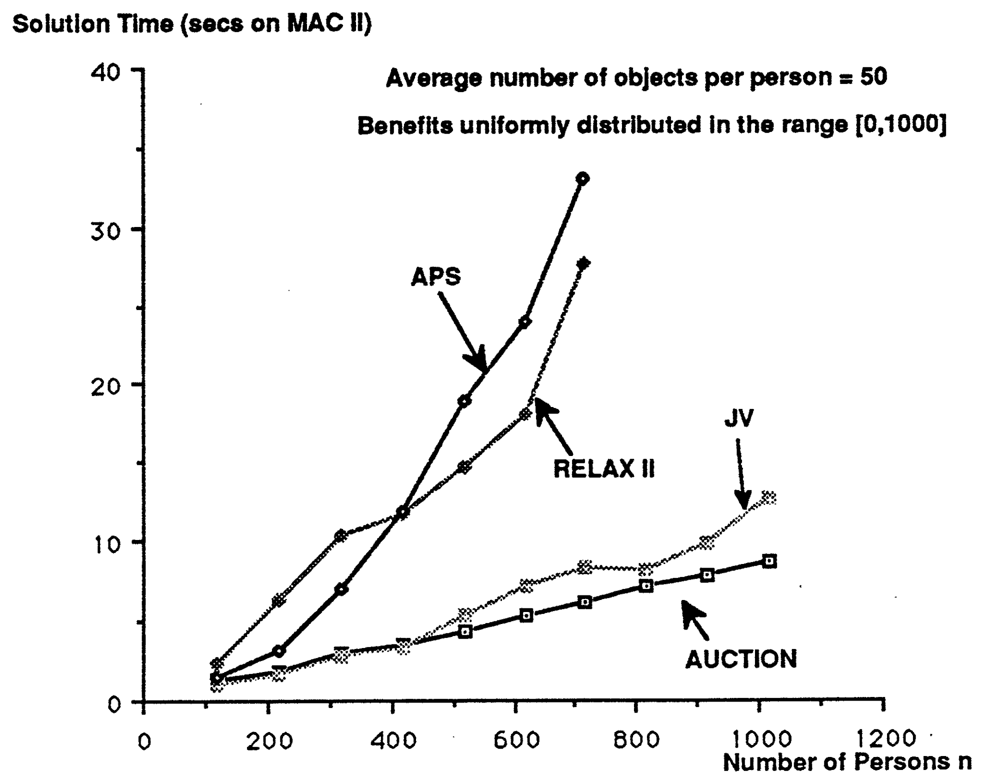


Figure 8: Computational results comparing various codes on a MAC-II on randomly generated problems. For all test problems, the number of feasible assignment pairs is $50n$, where n is the number of persons, and the benefits a_{ij} are integers chosen according to a uniform distribution from the range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

respectively), and then to modify the auction algorithm to take advantage of the presence of the multiple copies. Computational results given in [BeC89a] with a code called TRANSAUCTION, show 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, which we call *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.

We finally note that there are extensions of the auction algorithm for linear minimum cost flow (transhipment) problems. The first such extension is the *ϵ-relaxation method*, originally given in [Ber86a], and [Ber86b] (see [BeT89] for a detailed description of this method). A subsequent extension is the *network auction algorithm* of [BeC89b], which is typically more efficient in practice than the ϵ-relaxation method. These methods have interesting theoretical properties and like the auction algorithm, are well suited for parallelization. However, for general transhipment problems, their practical performance has yet to match the one of relaxation methods (e.g. the RELAX-II code); this is an area where further research may reverse currently prevailing opinion.

7. CONCLUDING REMARKS

The auction algorithm is an intuitive method based on new and interesting computational ideas. Its performance on serial machines is excellent 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, the codes AUCTION, TRANSAUCTION, and RELAX-II have been placed in the public domain and are available from the author at no cost.

REFERENCES

[Bal85] Balinski, M. L., "Signature Methods for the Assignment Problem", Operations Research J., Vol. 33, 1985, pp. 527-537.

[Bal86] Balinski, M. L., "A Competitive (Dual) Simplex Method for the Assignment Problem", Math. Programming, Vol. 34, 1986, pp. 125-141.

[BGK77] Barr, R., Glover, F., and Klingman, D., "The Alternating Basis Algorithm for Assignment Problems", Math. Programming, Vol. 13, 1977, pp. 1-13.

[Ber79] Bertsekas, D. P., "A Distributed Algorithm for the Assignment Problem", Lab. for Information and Decision Systems Working Paper, M.I.T., March 1979.

[Ber81] Bertsekas, D. P., "A New Algorithm for the Assignment Problem", Math. Programming, Vol. 21, 1981, pp. 152-171.

[Ber82] Bertsekas, D. P., "A Unified Framework for Minimum Cost Network Flow Problems", Laboratory for Information and Decision Systems Report LIDS-P-1245-A, M.I.T., Cambridge, MA, 1982; also in Math. Programming, Vol. 32, 1985, pp. 125-145.

[BeC85] Bertsekas, D. P., "A Distributed Asynchronous Relaxation Algorithm for the Assignment Problem", Proc. 24th IEEE Conf. Dec. & Contr., 1985, pp. 1703-1704.

[Ber86a] Bertsekas, D. P., "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems", LIDS Report P-1606, M.I.T., revision of Nov. 1986.

[Ber86b] Bertsekas, D. P., "Distributed Relaxation Methods for Linear Network Flow Problems", Proceedings of 25th IEEE Conference on Decision and Control, 1986, pp. 2101-2106.

[Ber88] Bertsekas, D. P., "The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem", Annals of Operations Research, Vol. 14, 1988, pp. 105-123.

[BeC89a] Bertsekas, D. P., and Castanon, D. A., "The Auction Algorithm for Transportation Problems", LIDS Report P-1850, M.I.T., Feb. 1989, to appear in Annals of Operations Research.

[BeC89b] Bertsekas, D. P., and Castanon, D. A., "The Auction Algorithm for the Minimum Cost Network Flow Problem", May 1989, submitted for publication.

[BeE87] Bertsekas, D. P., and Eckstein, J., "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems", Proc. of IFAC '87, Munich, Germany, July 1987.

[BeE88] Bertsekas, D. P., and Eckstein, J., "Dual Coordinate Step Methods for Linear Network Flow Problems", Laboratory for Information and Decision Systems Report LIDS-P-1768, M.I.T., Cambridge, MA, 1988; also in Math. Progr., Series B, Vol. 42, 1988, pp. 203-243.

[BeT85] 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; also Operations Research J., Vol. 36, 1988, pp. 93-114.

[BeT88] Bertsekas, D. P., and Tseng, P., "RELAX: A Computer Code for Minimum Cost Network Flow Problems", Annals of Operations Research, Vol. 13, 1988, pp. 127-190.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J., 1989.

[CMT88] Carpaneto, G., Martello, S., and Toth, P., "Algorithms and Codes for the Assignment Problem", Annals of Operations Research, Vol. 13, 1988, pp. 193-223.

[Dan63] Dantzig, G. B., Linear Programming and Extensions, Princeton Univ. Press, Princeton, N. J., 1963.

[Der85] Derigs, U., "The Shortest Augmenting Path Method for Solving Assignment Problems – Motivation and Computational Experience", Annals of Operations Research, Vol. 4, 1985, pp. 57-102.

[Eng82] Engquist, M., "A Successive Shortest Path algorithm for the Assignment Problem", INFOR, Vol. 20, 1982, pp. 370-384.

[FoF62] Ford, L. R., Jr., and Fulkerson, D. R., Flow in Networks, Princeton Univ. Press, Princeton, N. J., 1962.

[GaT87] Gabow, H. N., and Tarjan, R. E., "Faster Scaling Algorithms for Graph Matching", Revised Abstract, 1987.

[GGK82] 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.

[Gol87] Goldberg, A. V., "Efficient Graph Algorithms for Sequential and Parallel Computers", Tech. Report TR-374, Laboratory for Computer Science, M.I.T., Feb. 1987.

[GoT87] Goldberg, A. V., and Tarjan, R. E., "Solving Minimum Cost Flow Problems by Successive Approximation", Proc. 19th ACM STOC, May 1987.

[Gol85] Goldfarb, D., "Efficient Dual Simplex Methods for the Assignment Problem", Math. Programming, Vol. 33, 1985, pp. 187-203.

[Hal56] Hall, M., Jr., "An Algorithm for Distinct Representatives", Amer. Math. Monthly, Vol. 51, 1956, pp. 716-717.

[HeK70] Held, M., and Karp, R. M., "The Traveling Salesman Problem and Mimimal Spanning Trees", Operations Research J., Vol. 18, 1970, pp. 1138-1162.

[HeK71] Held, M., and Karp, R. M., "The Traveling Salesman Problem and Mimimal Spanning Trees: Part II", Math. Programming, Vol. 1, 1971, pp. 6-25.

[Hun83] Hung, M., "A Polynomial Simplex Method for the Assignment Problem", Operations Research, Vol. 31, 1983, pp. 595-600.

[JoV87] Jonker, R., and Volegnant, A., "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems", Computing, Vol. 38, 1987, pp. 325-340.

[Kuh55] Kuhn, H. W., "The Hungarian Method for the Assignment Problem", Naval Research Logistics Quarterly, Vol. 2, 1955, pp. 83-97.

[McG83] McGinnis, L. F., "Implementation and Testing of a Primal-Dual Algorithm for the Assignment Problem", Operations Research J., Vol. 31, 1983, pp. 277-291.

[Min60] Minty, G. J., "Monotone Networks", Proc. Roy. Soc. London, A, Vol. 257, 1960, pp. 194-212.

[Mun56] Munkres, J., "Algorithms for the Assignment and Transportation Problems", SIAM J., 1956.

[OrR70] Ortega, J. M., and Rheinboldt, W. C., "Iterative Solution of Nonlinear Equations in Several Variables", Academic Press, N. Y., 1970.

[PaS82] Papadimitriou, C. H., and Steiglitz, K., Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, N. J., 1982.

[Roc84] Rockafellar, R. T., Network Flows and Monotropic Programming, Wiley-Interscience, N. Y., 1984.

[Tho81] Thompson, G. L., "A Recursive Method for Solving Assignment Problems", in Studies on Graphs and Discrete Programming, P. Hansen (ed), North-Holland Publ. Co., 1981, pp. 319-343.

APPENDIX A: RELATION OF EQUILIBRIA WITH PRIMAL AND DUAL OPTIMALITY

Let us fix $\epsilon \geq 0$. In this appendix, we show that given an assignment $\{(i, j_i) \mid i = 1, \ldots, n\}$ and a set of prices $\{\bar{p}_j \mid j = 1, \ldots, n\}$, 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 $\epsilon > 0$. We first note that the total benefit of *any* assignment $\{(i, k_i) \mid i = 1, \ldots, n\}$ satisfies

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

for any set of prices $\{p_j \mid j = 1, \ldots, n\}$, 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,

$$A^* \leq D^*,$$

where A^* is the optimal total assignment benefit

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

and

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

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

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

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

$$D^* \leq \sum_{i=1}^{n} \left(\bar{p}_{j_i} + \max_j \{a_{ij} - \bar{p}_j\} \right) \leq \sum_{i=1}^{n} a_{ij_i} + n\epsilon \leq A^* + n\epsilon.$$

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

Note that 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 [Ber88], [BeT89], [Dan62], and [PaS82]. 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.