Computational Optimization and Applications, 1, 277–297 (1992)
© Kluwer Academic Publishers. Manufactured in the Netherlands.

A Forward/Reverse Auction Algorithm for Asymmetric Assignment Problems

DIMITRI P. BERTSEKAS
Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass., 02139.

DAVID A. CASTAÑON
Department of Electrical Engineering, Boston University, and ALPHATECH, Inc., Burlington, Mass., 01803.

Received April 19, 1992, Revised August 12, 1992.

Abstract. In this paper we consider the asymmetric assignment problem and we propose a new auction algorithm for its solution. The algorithm uses in a novel way the recently proposed idea of reverse auction, where, in addition to persons bidding for objects by raising their prices, we also have objects competing for persons by essentially offering discounts. In practice, the new algorithm apparently deals better with price wars than the currently existing auction algorithms. As a result, it tends to terminate substantially (and often dramatically) faster than its competitors.

Keywords: Assignment, auction algorithm, network programming, optimization.

1. Introduction

We consider the classical asymmetric assignment problem where we want to match m persons with m out of n objects ($m < n$). The benefit for matching a person with an object is given, and we want to assign all the persons to distinct objects so as to maximize the total benefit. There are a number of methods for solving this problem, including primal-simplex and primal-dual (or sequential shortest path) methods ([5], [11], [12], and [14]). In this paper we will focus on auction algorithms, first proposed in [1] for both symmetric and asymmetric problems, and subsequently developed in several other papers ([2], [3], [7], and [8]). The textbook [5] contains an extensive discussion of these methods and their extensions to other network flow problems. Recent experimental evidence suggests that auction algorithms outperform their competitors by a substantial margin, particularly for sparse assignment problems ([4], [7], and [8]), and are also well suited for parallel computation ([6], [10], [13], [15], [16], and [17]).

In the original proposal of the auction algorithm there is a price for each object, and at each iteration, one or more unassigned persons bid simultaneously for their "best" objects (the ones offering maximum benefit minus price), thereby raising the corresponding prices. Objects are then awarded to the highest bidder. The bidding increments must be at least equal to a positive parameter ϵ, and are chosen so as to preserve an ϵ-complementary slackness condition. For good practical (as well as theoretical) performance, it may be important to use 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 threshold ($1/m$ when a_{ij} are integer). Each scaling phase provides good initial prices for the next. The original proposal of the auction algorithm for asymmetric assignment problems had a deficiency: It required that the initial object prices be zero, thereby precluding the use of ϵ-scaling. As a result the method was susceptible to "price wars," that is, protracted sequences of small price rises resulting from groups of persons competing for a smaller number of roughly equally desirable objects. Thus, in order to use auction algorithms to solve asymmetric problems where price wars are likely, one had to convert the problem to a symmetric one by adding $n - m$ artificial persons that can be assigned to any object at zero benefit. There are specialized versions of the auction algorithm–the auction algorithm with similar persons ([6])—that can take advantage of the structure induced by the artificial persons. However, the approach of converting the problem to a symmetric problem introduces an undesirable increase in the problem's dimension and to our knowledge has not seen much use.

In part to address the difficulty with price wars of the original asymmetric auction algorithm, an alternative algorithm, called *reverse auction,* was recently developed in [7]. Here, roughly speaking, the objects compete for persons by lowering their prices. In particular, objects decrease their prices to a level that is sufficiently low to lure a person away from its currently held object. One can show that forward and reverse auctions are mathematically equivalent, but their combination has resulted in algorithms that can solve various assignment-like problems much faster than forward or reverse auction can by themselves. In particular, an ϵ-scaled version of a combined forward/reverse auction was developed for asymmetric problems that can deal effectively with price wars. This method operates principally as a forward auction and uses reverse auction only near the end to rectify violations in the optimality conditions. According to computational results given in [7], the solution times of this method for $m \times n$ asymmetric problems are quite reasonable and do not exceed the solution times of the original (forward only) auction algorithm for similar symmetric $m \times m$ problems by a factor larger than the natural ratio n/m.

However, as demonstrated in [7], by frequently switching between forward and reverse auction, a substantial performance improvement can be obtained for *symmetric* assignment problems. A natural question therefore arises whether a similar improvement can be realized for asymmetric assignment problems by similarly combining forward and reverse auction. The purpose of this paper is to develop auction algorithms for asymmetric assignment problems that switch frequently between forward and reverse auction. Our computational results show that for difficult problem structures, some of which typically arise in important data association problems, the new methods are much faster than the asymmetric assignment method of [7].

In Section 2, we define the asymmetric problem, and we develop ϵ-complementary slackness conditions in a form suitable for our purposes. In Section 3, we introduce the new combined forward/reverse auction algorithm and we develop its basic properties. Finally, in Section 4 we provide computational results.

2. Asymmetric assignment problems

In the asymmetric assignment problem there are m persons and n objects ($m < n$). The benefit or value for assigning person i to object j is a_{ij}. The set of objects to which person i can be assigned is a nonempty set denoted $A(i)$. The set of arcs of the underlying bipartite graph is denoted by $\mathcal{A}$

$$\mathcal{A} = \{(i,j) | j \in A(i), i = 1,\dots,m\}.$$

The set of persons to which object j can be assigned is assumed nonempty and is denoted by $B(j) = \{i | j \in A(i)\}$. 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 can be at most one pair $(i,j) \in S$; and for every object j there can be at most one pair $(i,j) \in S$. Given an assignment S, we say that person i is *assigned* if there exists a pair $(i,j) \in S$; otherwise we say that i is *unassigned.* We use similar terminology for objects. An assignment is said to be *feasible* if it contains m pairs, so that every person is assigned; otherwise the assignment is called *partial.* The problem is said to be feasible if there exists at least one feasible assignment. We want to find an assignment $\{(1,j_1),\dots,(m,j_m)\}$ with maximum total benefit $\sum_{i=1}^{m} a_{ij_i}$.

A dual problem can be defined by introducing a price variable p_j for each object j and a profit variable π_i for each person i. It was shown in [7] (see also [5]) that a corresponding dual problem is

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{m} \pi_i + \sum_{j=1}^{n} p_j - (n-m) \min_{j=1,\dots,n} p_j \\ &\text{subject to } \pi_i + p_j \geq a_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}. \end{aligned} \tag{1}$$

We denote by p the vector of prices $(p_1,\dots,p_n)$, and by π the vector of profits $(\pi_1,\dots,\pi_m)$. The following condition was introduced in [7] for an assignment S and a pair (π,p).

DEFINITION 1. *An assignment S and a pair (π,p) are said to satisfy ϵ-complementary slackness (ϵ-CS for short) if*

$$\pi_i + p_j \geq a_{ij} - \epsilon, \qquad \forall\ (i,j) \in \mathcal{A}, \tag{2a}$$

$$\pi_i + p_j = a_{ij}, \qquad \forall\ (i,j) \in S, \tag{2b}$$

$$p_j \leq \min_{k:\ \text{assigned under } S} p_k, \quad \forall\ j: \text{ unassigned under } S. \tag{2c}$$

The following proposition, proved in [7], clarifies the significance of ϵ-CS.

PROPOSITION 1. *If a feasible assignment S satisfies the ϵ-CS conditions (2a)–(2c) together with a pair (π, p), then S is within $m\epsilon$ of being optimal for the asymmetric assignment problem. In particular, if the benefits a_{ij} are all integer and $\epsilon < 1/m$, S is an optimal assignment.*

3. A forward/reverse auction algorithm for asymmetric assignment problems

In this section we consider auction algorithms that use a fixed $\epsilon > 0$, and maintain an assignment S and a pair (π, p) satisfying together with S the first two ϵ-CS conditions (2a) and (2b). The algorithms strive to obtain a feasible assignment S, that also satisfies the last ϵ-CS condition (2c). The novel feature of the algorithms, which distinguishes them from other auction algorithms, is the mechanism used to satisfy the last ϵ-CS condition (2c). The key idea is to maintain a scalar λ such that

$$p_j \geq \lambda, \quad \forall\ j \text{ that are assigned under } S. \tag{3}$$

As in earlier auction algorithms, any unassigned person can perform a forward bid, but this person will be assigned to his/her preferred object only if the corresponding bid is no less than λ. Furthermore, for an unassigned object to perform a reverse auction bid, it is additionally required that its price exceeds λ. The algorithms terminate when S becomes feasible and, in addition, all unassigned objects j satisfy $p_j \leq \lambda$. Thus upon termination, in view of (3), the last ϵ-CS condition (2c) is satisfied, and by Proposition 1, the assignment S is optimal if $\epsilon < 1/m$ and the benefits a_{ij} are all integer.

The level λ may be viewed as a *profitability threshold* below which we cannot drop the price of any assigned object. In the course of the algorithm, λ may be reduced if it is set initially so high that there is at least one unassigned person that cannot submit a bid that is greater or equal to λ. The various algorithms to be presented differ among each other in the precise mechanism used to adjust λ. Some algorithms adjust λ only after a feasible assignment has been obtained, some algorithms adjust λ only at the beginning of a scaling phase, and some algorithms adjust λ as frequently as is necessary to have at least one unassigned object at a price no less than λ at all times prior to termination. The last two types of algorithms proved most successful in the computational experiments reported in Section 4.

Note that we can initially select S to be empty, λ and p to be arbitrary, and π_i to be sufficiently large so that the conditions (2a) and (2b) are satisfied. Thus, in particular, we can try to use a favorable price vector such as one obtained from a scaling phase corresponding to a larger value of ϵ. This is the feature that allows the use of ϵ-scaling in connection with the algorithms of this paper.

3.1. Forward and reverse auction iterations

There are two types of iterations, forward and reverse. Forward iterations can be performed only as long as there is an unassigned person, and reverse iterations can be performed only as long as there is an unassigned object j with $p_j > \lambda$. Both types of iterations start with an assignment S, a pair (π, p), and a scalar λ satisfying conditions (2a), (2b), and (3).

Forward iteration:

Find an unassigned person i, its best object j_i

$$j_i = \arg \max_{j \in A(i)} \{a_{ij} - p_j\}, \tag{4}$$

the corresponding values

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}, \tag{5}$$

and

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

[If j_i is the only object in $A(i)$, we define w_i to be $-\infty$ or, for computational purposes, a number that is much smaller than v_i.] Set

$$p_{j_i} := \max \{\lambda, a_{ij_i} - w_i + \epsilon\}, \tag{7}$$

$$\pi_i := w_i - \epsilon. \tag{8}$$

If $\lambda \leq a_{ij_i} - w_i + \epsilon$, add (i, j_i) to S, and if j_i was assigned to some i' at the start of the iteration, remove from S the pair (i', j_i).

Reverse iteration:

Find an unassigned object j with $p_j > \lambda$, its best person i_j

$$i_j = \arg \max_{i \in B(j)} \{a_{ij} - \pi_i\}, \tag{9}$$

the corresponding values

$$\beta_j = \max_{i \in B(j)} \{a_{ij} - \pi_i\}, \tag{10}$$

and

$$\gamma_j = \max_{i \in B(j), i \neq i_j} \{a_{ij} - \pi_i\}. \tag{11}$$

[If i_j is the only object in $B(j)$, we define γ_j to be $-\infty$ or, for computational purposes, a number that is much smaller than β_j.] Proceed according to the following two cases:

(1) $\beta_j \geq \lambda + \epsilon$. In this case set

$$p_j := \max \{\lambda, \gamma_j - \epsilon\}, \tag{12}$$

$$\pi_{i_j} := a_{i_j j} - \max \{\lambda, \gamma_j - \epsilon\}, \tag{13}$$

add (i_j, j) to S, and if i_j was assigned to some j' at the start of the iteration, remove from S the pair (i_j, j').
(2) $\beta_j < \lambda + \epsilon$. In this case, set

$$p_j := \beta_j - \epsilon \tag{14}$$

and if the objects k with $p_k < \lambda$ are now more than $n - m$, set

$$\lambda := \min \{\xi \mid p_k \leq \xi \text{ for } n - m \text{ or more objects } k\}. \tag{15}$$

Note that λ remains unchanged in a forward iteration, and it can either decrease or stay unchanged in a reverse iteration. Note also that the "bidding person" i in the forward iteration may not get assigned to the best object j_i; this happens when $\lambda > a_{ij_i} - w_i + \epsilon$. In this case, however, it will be shown in Proposition 3(a) that the price p_{j_i} will be increased by at least ϵ. Furthermore, the "bidding object" j in the reverse iteration may not get assigned to the best person i_j; this happens when the "best value" β_j is low relative to λ, in which case the price p_j is reduced below λ [cf. (14)], and the object cannot bid again until λ decreases from its current level. Figure 1 illustrates the two cases that can arise in the reverse iteration.

The next proposition establishes a basic property of the forward and reverse iterations.

PROPOSITION 2. *Suppose that at the beginning of a forward or a reverse iteration, (π, p) satisfies together with S the first two ϵ-CS conditions (2a) and (2b), and λ satisfies condition (3). The same is true for (π, p), S, and λ at the end of the iteration.*

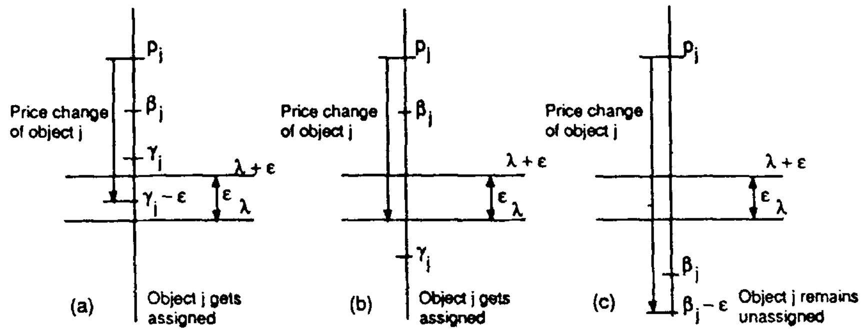

Figure 1. Illustration of the possible cases that can arise in the reverse iteration. These are: (a) $\beta_j \geq \lambda + \epsilon$ and $\gamma_j - \epsilon > \lambda$. Then j gets assigned to i_j, p_j is set to $\gamma_j - \epsilon$, and π_{i_j} is set to $a_{i_j j} - \gamma_j + \epsilon$. (b) $\beta_j \geq \lambda + \epsilon$ and $\gamma_j - \epsilon \leq \lambda$. Then j gets assigned to i_j, p_j is set to λ, and π_{i_j} is set to $a_{i_j j} - \lambda$. (c) $\beta_j < \lambda + \epsilon$. Then j stays unassigned and p_j is set to $\beta_j - \epsilon$, while π_{i_j} remains unchanged.

Proof. Suppose that the iteration starts with S, (π, p), and λ satisfying (2a), (2b), and (3). Let $\overline{S}, (\overline{\pi}, \overline{p})$, and $\overline{\lambda}$ be the corresponding quantities at the end of the iteration.

Consider first a forward iteration and let i be the corresponding unassigned person that submits the bid as per (4)–(8). We will first verify that the pair $(\overline{\pi}, \overline{p})$ satisfies (2a) for each arc. From (4)–(6), we see that $a_{ij_i} - w_i \geq p_{j_i}$, so by (7), we have

$$\overline{p}_{j_i} \geq p_{j_i} + \epsilon. \tag{16}$$

By adding this relation to the relation $\pi_k + p_{j_i} \geq a_{kj_i} - \epsilon$ [cf. (2a)], and by using the fact $\pi_k = \overline{\pi}_k$ for all $k \neq i_j$, we obtain

$$\overline{\pi}_k + \overline{p}_{j_i} \geq a_{kj_i} - \epsilon, \qquad \forall\ k \in B(j_i), k \neq i. \tag{17}$$

On the other hand, since $p_j = \overline{p}_j$ for all $j \neq j_i$, we have

$$\overline{\pi}_i = w_i - \epsilon \geq a_{ij} - p_j - \epsilon = a_{ij} - \overline{p}_j - \epsilon, \qquad \forall\ j \in A(i), j \neq j_i,$$

while from (7) and (8), we have

$$\overline{\pi}_i = w_i - \epsilon \geq a_{ij_i} - p_{j_i} - \epsilon.$$

Combining the last two relations, we obtain

$$\overline{\pi}_i + \overline{p}_j \geq a_{ij} - \epsilon, \qquad \forall\ j \in A(i). \tag{18}$$

Finally, for arcs (k, j) with $k \neq i, j \neq j_i$, we have $\pi_k = \overline{\pi}_k$ and $p_j = \overline{p}_j$, so by (2a), we obtain

$$\overline{\pi}_k + \overline{p}_j \geq a_{kj} - \epsilon, \qquad \forall\ (k, j) \in \mathcal{A}, k \neq i, j \neq j_i. \tag{19}$$

By combining (17)–(19), we see that

$$\bar{\pi}_k + \bar{p}_j \geq a_{kj} - \epsilon, \qquad \forall\ (k,j) \in \mathcal{A},$$

that is, the ϵ-CS condition (2a) holds at the end of the iteration.

We next show that (2b) is preserved by the iteration, that is,

$$\bar{\pi}_k + \bar{p}_j = a_{kj}, \qquad \forall\ (k,j) \in \bar{S}. \tag{20}$$

Note that if $(k,j) \in \bar{S}$ and $j \neq j_i$, we must have $k \neq i, (k,j) \in S$, and $\pi_k = \bar{\pi}_k, p_j = \bar{p}_j$, so by using the hypothesis [cf. (2b)], we see that (20) holds. If, on the other hand, $(k, j_i) \in \bar{S}$ for some k, we claim that $k = i$, since otherwise, by the rules of the iteration, we would have $\lambda > a_{ij_i} - w_i + \epsilon$, so that by (7) and (16),

$$\lambda = \bar{p}_{j_i} \geq p_{j_i} + \epsilon,$$

contradicting the condition (3). Now if $(i, j_i) \in \bar{S}$, we must have by the rules of the iteration, $\lambda \leq a_{ij_i} - w_i + \epsilon$ and $\bar{p}_{j_i} = a_{ij_i} - w_i + \epsilon$, so that

$$\bar{\pi}_i = w_i - \epsilon = a_{ij_i} - \bar{p}_{j_i}. \tag{21}$$

We see therefore that (20) holds for the case where $j = j_i$ as well.

Finally, to show that condition (3) is preserved by the iteration, note that $\lambda = \bar{\lambda}$, that $p_j = \bar{p}_j$ for all $j \neq j_i$, and that $\bar{p}_{j_i} \geq \lambda$ [cf. (7)]. Since the only object that can become assigned during the iteration is j_i, we see that

$$\bar{p}_j \geq \bar{\lambda}, \qquad \forall\ j \text{ that are assigned under } \bar{S}.$$

The proof of the proposition for the case of a forward iteration is thus complete.

Consider next the case of a reverse iteration. Let j be the corresponding unassigned object that submits the bid as per (9)–(15).

In the case where $\beta_j \geq \lambda + \epsilon$, we have $\max\{\lambda, \gamma_j - \epsilon\} \leq \beta_j$, so by (12) and (13),

$$\bar{\pi}_{i_j} = a_{i_j j} - \max\{\lambda, \gamma_j - \epsilon\} \geq a_{i_j j} - \beta_j + \epsilon = \pi_{i_j} + \epsilon \qquad \text{if } \beta_j \geq \lambda + \epsilon. \tag{22}$$

By using also the relation $\pi_{i_j} + p_j \geq a_{i_j j} - \epsilon$, we have

$$\bar{p}_j = \max\{\lambda, \gamma_j - \epsilon\} \leq \beta_j - \epsilon = a_{i_j j} - \pi_{i_j} - \epsilon \leq p_j \qquad \text{if } \beta_j \geq \lambda + \epsilon. \tag{23}$$

In the case where $\beta_j < \lambda + \epsilon$ we have $\gamma_j \leq \beta_j < \lambda + \epsilon$ and by using also the fact $p_j > \lambda$, we obtain

$$\bar{p}_j = \beta_j - \epsilon < \lambda < p_j \quad \text{if } \beta_j < \lambda + \epsilon, \tag{24}$$

$$\bar{\pi}_{i_j} = \pi_{i_j} \quad \text{if } \beta_j < \lambda + \epsilon. \tag{25}$$

To prove that the ϵ-CS condition (2a) is preserved by the reverse iteration, consider first the arcs (i,j) with $i \neq i_j$. Since $\overline{\pi}_i = \pi_i$ and $p_j \geq \gamma_j - \epsilon$, we have

$$\overline{\pi}_i + \overline{p}_j \geq \pi_i + \gamma_j - \epsilon \geq \pi_i + a_{ij} - \pi_i - \epsilon = a_{ij} - \epsilon, \qquad \forall\ i \in B(j), i \neq i_j. \quad (26)$$

Consider next the arcs (i_j,k) with $k \neq j$. We have $\pi_{i_j} + p_k \geq a_{i_jk} - \epsilon$ [cf. (2a)], and since $p_k = \overline{p}_k$ for $k \neq j$, we obtain [cf. (22)]

$$\overline{\pi}_{i_j} + \overline{p}_k \geq \pi_{i_j} + \epsilon + p_k \geq a_{i_jk} \qquad \text{if } \beta_j \geq \lambda + \epsilon,\ k \neq j, \quad (27)$$

and [cf. (25)]

$$\overline{\pi}_{i_j} + \overline{p}_k = \pi_{i_j} + p_k \geq a_{i_jk} - \epsilon \qquad \text{if } \beta_j < \lambda + \epsilon,\ k \neq j. \quad (28)$$

Finally for the arc (i_j, j), we have by (13),

$$\overline{\pi}_{i_j} + \overline{p}_j = a_{i_jk} \qquad \text{if } \beta_j \geq \lambda + \epsilon, \quad (29)$$

and by (14),

$$\overline{\pi}_{i_j} + \overline{p}_j = \pi_{i_j} + \beta_j - \epsilon = a_{i_jj} - \epsilon \qquad \text{if } \beta_j < \lambda + \epsilon. \quad (30)$$

By combining (26)–(30), we see that

$$\overline{\pi}_i + \overline{p}_k \geq a_{ik} - \epsilon, \qquad \forall\ (i,k) \in \mathcal{A},$$

so the condition (2a) is preserved by the reverse iteration.

To show that (2b) is preserved by the reverse iteration, that is,

$$\overline{\pi}_i + \overline{p}_k = a_{ik}, \qquad \forall\ (i,k) \in \overline{S}, \quad (31)$$

note that if $(i,k) \in \overline{S}$ and $i \neq i_j$, we must have $\pi_i = \overline{\pi}_i, p_k = \overline{p}_k$, and $(i,k) \in S$, so by using the hypothesis [cf. (2b)], we see that (31) holds. If, on the other hand, $(i_j, k) \in \overline{S}$ for some k, then either $k = j$, in which case we must have $\overline{\pi}_{i_j} + \overline{p}_j = a_{i_jj}$ by (12) and (13), or else $k \neq j$, in which case $(i_j,k) \in S, \overline{\pi}_{i_j} = \pi_{i_j}$, and $\overline{p}_k = p_k$, so by (2b) and the induction hypothesis we have $\overline{\pi}_{i_j} + \overline{p}_k = a_{i_jk}$. Thus (31) holds in all cases and the condition (2b) is preserved by the reverse iteration.

Finally to show that (3) is preserved by the reverse iteration, note that $\lambda \geq \overline{\lambda}$, while the only object that can become assigned during the iteration and whose price can change is j. On the other hand, if j becomes assigned, we must have $\overline{p}_j \geq \lambda$ by (12), so at the end of the iteration, we will have $\overline{p}_j \geq \overline{\lambda}$, thereby preserving (3). **Q.E.D.**

As a corollary of the preceding proof, we obtain the following proposition.

PROPOSITION 3. *Suppose that S, (π, p), and λ satisfy conditions (2a), (2b), and (3). Then:*

(a) *In a forward iteration, p_{j_i} increases by at least ϵ. Furthermore, either j_i is assigned to i during the iteration and p_{j_i} is increased to a level no less than λ, or else p_{j_i} is increased to the level λ.*
(b) *In a reverse iteration, either π_{i_j} increases by at least ϵ and j becomes assigned, or else j remains unassigned and p_j decreases to a level below λ.*
(c) *If all persons are assigned (S is feasible), the reverse iteration leaves λ unchanged.*

Proof.

(a) See (16) and (7).
(b) See (22) and (24).
(c) If all persons are assigned, the number of assigned objects is m and each assigned object k satisfies $p_k \geq \lambda$ by (3). Therefore, the objects k with $p_k < \lambda$ cannot be more than $n - m$, which is the only situation where λ can change. **Q.E.D.**

We will now use the results obtained so far to analyze several possible algorithms.

3.2. Purely forward algorithm

It is possible to consider a forward auction algorithm that consists exclusively of forward iterations. In such an algorithm it is essential to choose initially $\lambda \geq p_j$ for all unassigned objects j. Then, since λ will remain unchanged, by using Proposition 3(a), it can be seen that in the course of the algorithm, we will have

$$\max_{j:\text{ unassigned under } S} p_j \leq \lambda \leq \min_{k:\text{ assigned under } S} p_k, \tag{32}$$

so by using also Proposition 2, we see that all three ϵ-CS conditions (2a)–(2c) will be satisfied. Furthermore, by Proposition 3(a), the price p_{j_i} is increased by at least ϵ at each forward iteration. Using this fact and standard arguments (see e.g., [3] and [5]), it can be shown that this forward algorithm will terminate with a feasible assignment S that satisfies ϵ-CS together with (π, p) (and is optimal if $\epsilon < 1/m$ and the problem data are integer).

Unfortunately, even though this forward algorithm will work with arbitrary initial prices, it is not suitable for use in conjunction with ϵ-scaling because of the requirement that initially we have $\lambda \geq p_j$ for all unassigned objects j. Since for an object to be assigned, its price must rise to at least the level λ, the advantage of approximately optimal initial prices that ϵ-scaling attempts to carry from one ϵ-scaling phase to the next is largely diminished.

3.3. Purely reverse algorithm

It is also possible to consider a purely reverse auction algorithm that consists exclusively of reverse iterations, provided that the initial assignment is feasible and the initial λ is such that condition (3) is satisfied ($p_j \geq \lambda$ for all assigned objects j). The following proposition establishes the validity of the algorithm.

PROPOSITION 4. *For a feasible problem, the purely reverse algorithm starting from a feasible assignment, a pair (π, p), and a scalar λ satisfying conditions (2a), (2b), and (3) terminates. The assignment obtained satisfies ϵ-CS together with (π, p).*

Proof. From Proposition 3(c), we have that λ will remain unchanged and that at each iteration there are two possibilities: (1) π_{i_j} will increase by ϵ and the selected unassigned object j will get assigned to i_j; or (2) the number of unassigned objects whose price exceeds λ will decrease by one. Therefore, after some iteration, case (1) will occur exclusively. By (13) we have

$$\overline{\pi}_{i_j} = a_{i_j j} - \max\{\lambda, \gamma_j - \epsilon\} \leq a_{i_j j} - \lambda, \tag{33}$$

so π_{i_j} cannot exceed $\max_{(i,k)\in\mathcal{A}} a_{ik} - \lambda$. It follows that the algorithm cannot execute an infinite number of iterations and must therefore terminate. **Q.E.D.**

The disadvantage of the purely reverse algorithm is that it requires an initial feasible assignment. The reason is that if the initial assignment is infeasible, through a poor choice of λ, we may have $p_j \leq \lambda$ for all unassigned objects j, in which case the algorithm can make no further progress. Furthermore, it may occur that in the course of the algorithm, following several iterations in which the prices of some unassigned objects get strictly below λ, we have $p_j \leq \lambda$ for all unassigned objects j, while we have $p_j < \lambda$ for no more than $n - m$ unassigned objects. Then the purely reverse algorithm will leave λ unchanged and will terminate without finding a feasible solution. A possible remedy is to start with an arbitrary assignment, but to reduce λ by some positive increment whenever the difficulty just described occurs. Unfortunately, however, it is not easy to determine the proper size of the increment for fast termination.

Another possible way to circumvent the need for an initial feasible assignment is to combine the forward and reverse algorithms, so that the forward part guarantees that a feasible assignment will be obtained, while the reverse part is capable of dealing with essentially arbitrary starting values of λ. In particular, one may use the purely forward algorithm first to obtain a feasible assignment, and then switch to the reverse algorithm after setting

$$\lambda := \min_{k:\ \text{assigned under } S} .$$

In fact, this is precisely the algorithm proposed in [7]; it is suitable for ϵ-scaling, but it does not take advantage of the beneficial effect of mixing the forward and the reverse algorithms that was demonstrated for symmetric problems in [7]. The following algorithm switches several times between the two algorithms, aiming at less reliance on ϵ-scaling and faster termination.

3.4. Combined forward/ reverse algorithm

The combined forward/reverse algorithm that we now introduce switches between forward and reverse auction until all persons are assigned. Then it executes reverse iterations exclusively, aiming to satisfy the final remaining optimality condition ($p_j \leq \lambda$ for all unassigned objects j). The initial S, (π, p), and λ must satisfy the ϵ-CS conditions (2a), (2b), and the condition $\lambda \leq p_j$ for all assigned objects j. Thus, if the initial assignment is empty, any initial p and λ can be used. We assume that initially there is at least one unassigned person (otherwise the forward part of the algorithm is inapplicable and unnecessary).

Combined forward/reverse auction algorithm:

Step 1 **(forward auction cycle)**: Execute iterations of the forward auction algorithm until at least one more person becomes assigned. If there is an unassigned person left, go to step 2; else go to step 3.

Step 2 **(reverse auction cycle)**: Execute several iterations of the reverse auction algorithm until at least one more object becomes assigned, or until we have $p_j \leq \lambda$ for all unassigned objects j. If there is an unassigned person left, go to step 1; else go to step 3.

Step 3 **(reverse auction)**: Execute successive iterations of the reverse auction algorithm until the algorithm terminates with $p_j \leq \lambda$ for all unassigned objects j.

The following proposition establishes the validity of the algorithm.

PROPOSITION 5. *For a feasible problem, the combined forward/reverse algorithm terminates with a feasible assignment S and a pair (π, p) satisfying the ϵ-CS conditions (2a)–(2c).*

Proof. We will assume that the algorithm does not terminate and will arrive at a contradiction. When the algorithm obtains a feasible assignment, it gets reduced to the purely reverse algorithm and terminates by Proposition 4. Assume therefore that the algorithm never obtains a feasible assignment. Since the cardinality of the assignment must increase before switching from a forward to a reverse cycle, there are two possibilities: (1) The algorithm will execute forward iterations exclusively after some iterations; or (2) the algorithm will execute reverse iterations exclusively after some iteration, and we will always have $p_j > \lambda$ for some unassigned object j.

In case 1 the algorithm will be reduced to the purely forward algorithm, and, as discussed earlier, it must terminate for a feasible problem. This contradicts our earlier hypothesis that the algorithm does not terminate.

In case 2, since whenever a profit variable increases, it increases by at least ϵ, there are two possibilities:

(a) After some iteration, all profit variables π_i stay constant and no object changes assignment.
(b) Some profit variable increases to ∞, in which case, by the argument given in the proof of Proposition 4 [cf. (33)], λ decreases to $-\infty$.

In case a, the variables β_j stay constant after some iteration, so in view of (14), the object prices cannot change after some iteration. This contradicts Proposition 3(b), which states that $\overline{p}_j < p_j$ at each reverse iteration [see also (24)]. In case b, let

$$J_\infty = \{j \mid p_j \to -\infty\}, \quad \overline{J}_\infty = \{j \mid j \notin J_\infty\},$$
$$I_\infty = \{i \mid \pi_i \to \infty\}, \quad \overline{I}_\infty = \{i \mid i \notin I_\infty\}.$$

By the ϵ-CS condition (2a), we must have $\pi_i + p_j \geq a_{ij} - \epsilon$ for all $(i,j) \in \mathcal{A}$, so

$$i \in \overline{I}_\infty \;\Rightarrow\; j \in \overline{J}_\infty, \quad \forall\, j \in A(i). \tag{34}$$

We claim that after some iteration, each of the objects in $\overline{J}_\infty$ must be assigned at all times to the same person from $\overline{I}_\infty$. To see this, note that if some object $j \in \overline{J}_\infty$ bids an infinite number of times for some person i_j, then π_{i_j} will increase by at least ϵ an infinite number of times, in view of Proposition 3(b), the definition of $\overline{J}_\infty$, and the fact $\lambda \to -\infty$. On the other hand, by (34), we must have $i_j \in \overline{I}_\infty$ so π_{i_j} must remain bounded and we have a contradiction.

Thus, $\overline{I}_\infty$ contains the set of persons that are assigned to $\overline{J}_\infty$. However, $\overline{I}_\infty$ contains some additional persons, namely the persons that are unassigned throughout the last reverse cycle (a person that becomes assigned in a reverse cycle remains assigned for the duration of the cycle). Therefore, the number of persons in $\overline{I}_\infty$ exceeds the number of objects in $\overline{J}_\infty$. In view of (34), this contradicts the hypothesis that the problem is feasible. **Q.E.D.**

A careful examination of the preceding proof shows that there are other valid variations of the combined/forward reverse algorithm, corresponding to variations of the reverse iterations and/or the scheme for switching from a reverse to a forward cycle. What is important is that: (a) λ should remain unchanged at all forward iterations and at all reverse iterations where the current assignment is feasible; (b) λ should not increase during all reverse iterations (here again only unassigned objects j with $p_j > \lambda$ should be allowed to bid); and (c) a mechanism is provided whereby the combined method is guaranteed to eventually exit from a reverse cycle if the current assignment is not feasible.

Consider now what happens if the problem is infeasible. Then, eventually the number of unassigned persons will stop decreasing and the method will get caught in either a forward cycle (step 1) or in a reverse cycle (step 2). Infeasibility will then be detected in the standard way for auction algorithms, that is, some price or some profit will exceed a certain precomputable upper bound, as described in [5]. Another way to detect infeasibility is to periodically check (through a breadth-first search) whether, in the current assignment, there is an augmenting path starting at some unassigned person and ending at some unassigned object. It is also possible to deal with infeasibility by adding a sufficient number of artificial arcs to convert the problem to a feasible problem. These arcs must have sufficiently small values to guarantee that they are not part of an optimal assignment unless the original problem is infeasible (see [5]).

3.5. An alternative reverse iteration and combined forward/reverse algorithm

A variation of the reverse iteration is obtained if we keep λ constant, even if the number of objects k with $p_k < \lambda$ becomes greater than $n - m$. Thus, this alternative iteration is defined to be identical to the one given earlier, except that we forego the change of λ in case (2) [cf. (15)]. For this iteration, Propositions 2 and 3 still hold, but the purely reverse algorithm may terminate with some persons still unassigned because λ was set to a value so high that the number of unassigned objects with price less or equal to λ exceeds $n - m$. Nonetheless, if the alternative iteration is combined with the forward iteration as in the algorithm given earlier, the resulting combination is valid because forward iterations will continue as long as there are some unassigned persons, even if no reverse iterations can be executed.

Note that λ remains unchanged throughout this alternative combined forward/reverse algorithm, and can only change at the beginning of each scaling phase. Thus, the choice of λ at each scaling phase is critical for the algorithm's performance. A reasonable scheme is to choose λ at the beginning of each scaling phase except the first as

$$\lambda = \min_{j:\ \textit{assigned under } S} p_j,$$

where S is the assignment obtained at the end of the preceding scaling phase. At the first scaling phase one may start with the empty assignment, zero object prices, and $\lambda = 0$. With these choices, no reverse iterations will be executed in the first scaling phase, since the prices of the unassigned objects as well as λ will remain at zero throughout the phase.

4. Computational results

In order to evaluate the relative performance of the new forward/reverse auction algorithms, we used the following FORTRAN codes:

- **ASFR1**: An implementation of the combined forward/reverse algorithm discussed in Section 3.4.
- **ASFR2**: An implementation of the combined forward/reverse algorithm discussed in Section 3.5.
- **AS**: An implementation of the forward/reverse algorithm of [7] discussed briefly at the end of Section 3.3.

All three codes solve assignment problems as minimization problems, that is, they work with arc costs, which are the negatives of the arc benefits. In all codes we used ϵ-scaling with the same starting value of ϵ. Following each scaling phase, we reduced ϵ by a certain factor, which was 10 for the two ASFR codes and 5 for the AS code. These values were found to work reasonably well over the range of problems solved, and are consistent with the values used for forward and forward/reverse auction codes tested in [7]. It is possible to use a larger ϵ-reduction factor (and consequently execute fewer scaling phases) for the ASFR codes than for the AS code because forward/reverse auction apparently tends to resolve price wars faster than forward auction.

The three codes were evaluated on the following classes of inequality-constrained assignment problems:

1. Randomly generated problems, obtained with the DIMACS assign.c problem generator ([9]), which also include a number of high-cost arcs.
2. Geometric matching problems, where a list of two-dimensional points must be matched with a randomly perturbed copy of the same list.
3. Clustered geometric matching problems, where a list of clustered two-dimensional points must be matched with a randomly perturbed copy of the same list.

The last two classes are representative of an important class of applications that motivated this research: data association in multiobject tracking. In these problems, new sensor measurements at each time frame must be associated with the predicted position of existing tracks. Because of the presence of false alarms (due to clutter or other effects), missed detections, and sensor measurement inaccuracies, the set of measurement values will be a random perturbation of the set of predicted positions. The maximum likelihood problem of determining which measurement-track associations are most likely is equivalent to an asymmetric assignment problem.

4.1. *Results on random problems*

Table 1 summarizes the results of our random experiments for asymmetric

Table 1. Average run time in seconds on the NeXTStation 68040 for 2,000-person, degree-8 random problems. Each data point is an average over 10 sample problems.

Problem	2,000 × 2,020	2,000 × 2,050	2,000 × 2,100	2,000 × 2,200
AS	3.05	2.50	2.00	0.53
ASFR1	1.13	0.84	0.69	0.50
ASFR2	1.14	0.83	0.67	0.50

Table 2. Average run time in seconds on the NeXTStation 68040 for 4,000-person, degree-8 random problems. Each data point is an average over 10 sample problems.

Problem	4,000 × 4,040	4,000 × 4,100	4,000 × 4,200	4,000 × 4,400
AS	9.86	7.85	6.54	2.39
ASFR1	4.43	3.57	2.90	1.14
ASFR2	4.45	3.56	2.92	1.16

assignment problems with 2,000 persons. In these experiments, an initial random problem is generated with eight arcs per person, with cost range $[1,200]$. Based on the results of [7], purely random problems are often easy to solve and require no scaling. In order to make scaling necessary, we modified the problems to increase the costs of 20% of the arcs by a factor of 100. The resulting problems have a difficult structure, which requires scaled auction algorithms, as discussed in [7].

Table 2 summarizes the results of random experiments with 4,000-person, degree-8 problems, with cost range $[1,200]$, with 20% of the arc costs increased by a factor of 100. The results in these two tables indicate little difference in the performance of the ASFR1 and ASFR2 algorithms. The results also indicate the superiority of the new forward/reverse auction algorithms over the previous algorithm of [7].

Table 3 contains the results of experiments with 4,000 × 4,400 person assignment problems as a function of increasing density for two classes of problems: "easy" problems where the arc costs are randomly selected uniformly in $[1,20{,}000]$, and "hard" problems where the arc costs are selected uniformly in $[1,200]$, and 20% of the arc costs are increased by a factor of 100. As the times indicate, the ASFR1 and ASFR2 algorithms offer little advantage over the old AS algorithm for easy problems; these problems do not create "price wars" among persons, and can be solved without the use of scaling. For the "hard" problems, the new forward/reverse auction algorithms are much faster than the AS algorithm.

Table 3. Average run times in seconds on the NeXTStation 68040 for 4,000-person, 4,400-object random problems with increasing degree. Each data point is an average over 10 sample problems.

Problem	Degree-8	Degree-16	Degree-32	Degree-64
AS – easy	0.36	0.57	1.17	2.40
ASFR1 – easy	0.40	0.68	1.46	2.91
ASFR2 – easy	0.42	0.72	1.43	2.91
AS – hard	2.39	8.27	17.32	38.95
ASFR1 – hard	1.18	2.48	5.17	10.42
ASFR2 – hard	1.16	2.50	5.18	10.40

4.2. Results on geometric matching problems

The geometric matching problems were generated to simulate the structure of data association problems arising in multiobject tracking. An initial number of points was randomly generated using a uniform distribution on the square 10^5 by 10^5. From these initial points, two lists of points were generated according to the following rules.

1. The first list was generated by accepting each point of the initial list with probability 0.95, independently of the selection of other points. This effect was chosen to simulate a missed detection rate of 5%. Thus, the first list is a reduced version of the initial list of points.
2. The second list was generated by first accepting each point of the initial list with probability 0.95, independently of the selection of other points in the second as well as the first list. This effect was chosen to simulate some false alarms in the data set, corresponding to points not included in the first list. Then, the locations of all the points in the second list were shifted by a constant bias, which was randomly generated from a bivariate Gaussian distribution with a specified bias standard deviation. Subsequently, each point in the second list was shifted by an independent bivariate Gaussian random variable, representing measurement noise, with a specified measurement standard deviation.
3. The list with the least number of points was selected to be the persons in the asymmetric assignment problem. The other list was selected to be the objects. Arcs were created between each person-object pair for which the Euclidean distance between the corresponding points was less than three times

Table 4. Average run times in seconds on the NeXTStation 68040 for geometric problems with 2,000 points per list, probability of detection 0.95. Each data point is an average over 10 sample problems.

Bias/Measurement SD	5/50	10/100	15/150	20/200
AS	0.26	30.37	378.42	374.03
ASFR1	0.32	1.67	3.78	9.34
ASFR2	0.34	1.62	3.04	5.99

the measurement standard deviation. The cost assigned to each arc was an integer between 1 and 1,000, which was proportional to the Euclidean distance of the corresponding person-object pair.

4. In order to guarantee feasibility of the asymmetric assignment problem, an extra object node was introduced for each person node, with a corresponding arc cost of 20,000. This large cost encouraged the problem to find a feasible assignment without using the extra nodes.

Table 4 summarizes our results with random geometric experiments corresponding to 2,000 points in the initial list. Four different combinations of bias/measurement standard deviation were tested. For each combination, Table 4 lists the average run times across 10 different problems with similar statistics for each of the three algorithms. For small bias/measurement standard deviations, the points in the person lists and object lists are far apart, and thus the solution of the assignment problem is trivial. As the standard deviation increases, object-person groups form with an unbalanced number of persons or real objects in the group (because of the missed detection and false-alarm probability 0.95). This creates long price wars to determine which extra persons in the group will be assigned to artificial objects, or which objects will remain unassigned. The size of these groups increases with bias/measurement standard deviations, leading to longer price wars.

As the results in Table 4 indicate, the new forward/reverse algorithms are much more efficient than the AS algorithm of [7]. The reason for this efficiency is that forward and reverse iterations are interleaved at each scaling phase; in contrast, the AS algorithm performs only forward iterations at most scaling phases, and then switches to an unscaled reverse-only algorithm. Most of the computation time (over 95%) is spent in this unscaled reverse-only algorithm trying to enforce the condition $p_j \leq \lambda$ for all unassigned objects j [cf. (3)] within groups with more objects than persons. The new forward/reverse algorithms use both forward and reverse iterations in all scaling phases, resulting in more robust performance for this class of problems.

Table 5. Average run times in seconds on the NeXTStation 68040 for clustered geometric problems with 2,000 points per list, probability of detection 0.95. Each data point is an average over 10 sample problems.

Spread/Measurement/Bias SD	500/50/5	1000/100/10	2500/150/15	2000/200/20
AS	1.04	28.33	225.60	813.92
ASFR1	0.45	1.70	2.21	3.68
ASFR2	0.49	1.74	1.94	3.53

4.3. Results on clustered geometric matching problems

The clustered geometric matching problems were generated to simulate a different type of data association problem arising in multiobject tracking: groups of objects moving close together, but with significant distance among the groups. The principal difference between this class of problems and the geometric class of problems is the location of the initial number of points, which are generated as follows.

1. An initial number of cluster centers are generated with a uniform distribution on the square 10^5 by 10^5.
2. For each cluster center, a fixed number of points is generated by adding to the cluster center an independent bivariate Gaussian random variable with a specified cluster spread standard deviation.

Once the initial list of points is available, generation of the asymmetric assignment problem follows identically steps 1–5 of the geometric matching problems of the previous section.

Table 5 summarizes our results with random geometric experiments corresponding to 50 clusters of 40 points each in the initial list. Four different combinations of spread/measurement/bias standard deviation were tested. For each combination, Table 5 lists the average run times across 10 different problems with similar statistics for each of the three algorithms. For small spread/measurement/bias standard deviations, the assignment problem decouples by cluster and thus corresponds to solution of 50 small problems. As the spread standard deviation increases, the assignment problems become coupled across clusters, and finding an optimal assignment becomes harder and more susceptible to long price wars.

The results in Table 5 agree closely with the results from Table 4. The performance of the new forward/reverse algorithms is much more robust to scenario variations than the performance of the AS algorithm of [7]. This is due largely to the use of scaling both in forward and reverse iterations, and the mixing of forward and reverse iterations at each scaling phase. In essence, this mixing of forward and reverse iterations forces the object prices and person profits to satisfy the complementary slackness condition (3) locally for each cluster and at each scaling phase. In contrast, the AS algorithm tries to enforce this condition only at the last scaling phase, while using a single value of λ for all the clusters; if the prices in one cluster rise much higher than the prices in other clusters (because of price wars), the reverse-only part of the AS algorithm may require an excessive number of bids to satisfy condition (3).

Acknowledgments

This work was supported in part by the BM/C3 Technology branch of the U.S. Army Strategic Defense Command.

References

[1] D.P. Bertsekas, "A distributed algorithm for the assignment problem," Lab. for Information and Decision Systems Working Paper, M.I.T., Cambridge, MA, March, 1979.

[2] D.P. Bertsekas, "A distributed asynchronous relaxation algorithm for the assignment problem," Proc. 24th IEEE Conf. Dec. & Contr., pp. 1703–1704, 1985.

[3] D.P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," Ann. Oper. Res. vol. 14, pp. 105–123, 1988.

[4] D.P. Bertsekas, "The auction algorithm for assignment and other network flow problems: A tutorial," Interfaces, vol. 20, pp. 133–149, 1990.

[5] D.P. Bertsekas, Linear network optimization: Algorithms and codes, M.I.T. Press, Cambridge, MA, 1991.

[6] D.P. Bertsekas, and D.A. Castañon, "Parallel synchronous and asynchronous implementations of the auction algorithm," Alphatech Report, Burlington, MA, November, 1989 (also Parallel Comput. vol. 17, pp. 707–732, 1991).

[7] D.P. Bertsekas, D.A. Castañon, and H. Tsaknakis, "Reverse auction and the solution of inequality constrained assignment problems," Alphatech Report, Burlington, MA, March 1991; to appear in SIAM J. on Optimization.

[8] D.P. Bertsekas, and J. Eckstein, "Dual coordinate step methods for linear network flow problems," Math. Progr., Series B, vol. 42, pp. 203–243, 1988.

[9] D.A. Castañon, "Reverse auction algorithms for assignment problems," to appear in DIMACS Series in Discrete Math. and Comput. Sci.

[10] D. Kempa, J. Kennington, and H. Zaki, "Performance characteristics of the Jacobi and Gauss-Seidel versions of the auction algorithm on the Alliant FX/8," Report OR-89-008, Dept. of Mech. and Ind. Engg., Univ. of Illinois, Champaign-Urbana, IL, 1989.

[11] J. Kennington, and R. Helgason, Algorithms for network programming, Wiley, New York, NY, 1980.

[12] C.H. Papadimitriou, and K. Steiglitz, Combinatorial optimization: Algorithms and complexity, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[13] C. Phillips, and S.A. Zenios, "Experiences with large scale network optimization on the connection machine", Report 88-11-05, Dept. of Decision Sci., The Wharton School, Univ. of Pennsylvania, Philadelphia, PA, November, 1988.

[14] R.T. Rockafellar, Network flows and monotropic programming, Wiley-Interscience, New York, NY, 1984.

[15] J. Wein, and S.A. Zenios, "Massively parallel auction algorithms for the assignment problem," Proc. of 3rd Symp. on the Frontiers of Massively Parallel Computation, MD., pp. 90–99, November, 1990.
[16] J. Wein, and S.A. Zenios, "On the massively parallel solution of the assignment problem," J. of Parallel and Distributed Comput., vol. 13, pp. 228–236, 1991.
[17] H. Zaki, "A comparison of two algorithms for the assignment problem," Report ORL 90-002, Dept. of Mech. and Ind. Eng., Univ. of Illinois, Urbana, Ill.