Auction Algorithms for Path Planning, Network Transport, and Reinforcement Learning

by

Dimitri P. Bertsekas

Chapter 2
Auction Algorithms for the Assignment Problem

This monograph represents "work in progress," and will be periodically updated. It more than likely contains errors (hopefully not serious ones). Furthermore, the references to the literature are incomplete. Your comments and suggestions to the author at dbertsek@asu.edu are welcome.

October 15, 2022

2

Auction Algorithms for the Assignment Problem

Contents

2.1. The Auction Algorithm for Symmetric Assignment Problems . . . . p. 2
2.1.1. The Main Auction Algorithm . . . . . p. 3
2.1.2. Approximate Coordinate Descent Interpretation . . p. 7
2.1.3. ϵ-Scaling . . . . . p. 7
2.1.4. Dealing with Infeasibility . . . . . p. 9
2.2. Extensions of the Auction Algorithm . . . . . p. 10
2.2.1. Reverse Auction . . . . . p. 11
2.2.2. Auction Algorithms for Asymmetric Assignment p. 16
2.2.3. Auction Algorithms with Similar Persons . . . . p. 23
2.2.4. Combinations with Dual Ascent Methods . . . . p. 27
2.3. Theoretical Aspects . . . . . p. 27
2.4. Notes and Sources . . . . . p. 34

In this chapter we will discuss auction algorithms for various types of assignment problems. These algorithms aim to solve simultaneously the primal and the dual problems. However, contrary to other network optimization algorithms such as simplex methods and dual ascent methods (see [Ber98]), they do not rely on cost improvement. At any one iteration, they may deteriorate both the primal and the dual cost. On the other hand, they can be interpreted as *approximate coordinate ascent* methods for solving he dual problem, as will be discuss in this chapter.

In Section 2.1, we develop some of the major characteristics of the algorithm for symmetric assignment problems. In Section 2.2, we develop auction algorithms for special types of assignment problems, as well as algorithmic variations and combinations with other types of assignment algorithms. In Section 2.3, we discuss theoretical aspects of the methods developed in Sections 2.1 and 2.2, including issues of computational complexity.

2.1 THE AUCTION ALGORITHM FOR THE SYMMETRIC ASSIGNMENT PROBLEM

In this section we consider the assignment problem whereby we want to match n persons and n objects on a one-to-one basis. We are given a "value" a_{ij} for matching person i with object j, and we want to assign persons to objects so as to maximize the total value. The set of objects to which person i can be assigned is a nonempty set denoted $A(i)$. The set of all possible pairs that can be assigned is denoted by $\mathcal{A}$,

$$\mathcal{A} = \big\{(i,j) \mid j \in A(i),\, i = 1, \ldots, n\big\}.$$

Note that $\mathcal{A}$ is the set of arcs of the underlying bipartite assignment graph.

Our terminology is as follows. An *assignment* S is a (possibly empty) set of person-object pairs (i,j) such that $j \in A(i)$ for all $(i,j) \in S$; moreover, we require that 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* or *complete* if it contains n pairs, so that every person and every object is assigned; otherwise the assignment is called *partial*.

We call the problem just described the *symmetric* assignment problem, to distinguish it from the *asymmetric* assignment problem where the number of persons is smaller than the number of objects. We will discuss the asymmetric problem and associated auction algorithms later in Section 2.2.

2.1.1 The Main Auction Algorithm

We recall the auction algorithm, described in Section 1.4.1. It was motivated by the simpler but flawed naive auction algorithm. A key notion, which made possible the correct operation of the algorithm was the ϵ-*complementary slackness* property (ϵ-CS for short), which relates a partial assignment S and a price vector $p = (p_1, \ldots, p_n)$. In particular, we say that S and p satisfy ϵ-CS if for every pair $(i, j) \in S$, object j is within ϵ of being the "best" object for person i, i.e.,

$$a_{ij} - p_j \geq \max_{k \in A(i)} \{a_{ik} - p_k\} - \epsilon, \qquad \forall\ (i, j) \in S. \tag{2.1}$$

In this section, we will consider a more general version of the auction algorithm, which is also well-suited for parallel or distributed implementation, as *it allows simultaneous bids by multiple unassigned persons*, instead of a bid by a single unassigned person, as in Section 1.4.1. In fact, a distributed implementation was the original motivation for introducing the algorithm in the paper [Ber79].

The algorithm proceeds iteratively and terminates when a complete assignment is obtained. At the start of the generic iteration we have a partial assignment S and a price vector p that satisfy ϵ-CS. As an initial choice, we may use an arbitrary set of prices together with the empty assignment, which trivially satisfies ϵ-CS. We will show later that the iteration preserves the ϵ-CS condition. The iteration consists of two phases: the *bidding phase* and the *assignment phase*, which we now describe.

Bidding Phase of the Auction Iteration

Let I be a nonempty subset of persons that are unassigned under the assignment S. For each person $i \in I$:

1. Find a "best" object j_i having maximum value, i.e.,

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

and the corresponding value

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}, \tag{2.2}$$

and find the best value offered by objects other than j_i

$$w_i = \max_{j \in A(i),\, j \neq j_i} \{a_{ij} - p_j\}. \tag{2.3}$$

[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.]

2. Compute the "bid" of person i given by

$$b_{ij_i} = p_{j_i} + v_i - w_i + \epsilon = a_{ij_i} - w_i + \epsilon. \tag{2.4}$$

(Abusing terminology somewhat, we say that "person i bid for object j_i," and that "object j_i received a bid from person i.")

Assignment Phase of the Auction Iteration

For each object j, let $P(j)$ be the set of persons from which j received a bid in the bidding phase of the iteration. If $P(j)$ is nonempty, increase p_j to the highest bid,

$$p_j := \max_{i \in P(j)} b_{ij}, \tag{2.5}$$

remove from the assignment S any pair (i, j) (if j was assigned to some i under S), and add to S the pair (i_j, j), where i_j is a person in $P(j)$ attaining the maximum above.

Note that there is some freedom in choosing the subset of persons I that bid during an iteration. One possibility is to let I consist of a single unassigned person, as in the algorithm of Section 1.4.1. We call this version the *Gauss-Seidel version* because of its similarity with Gauss-Seidel methods for solving systems of nonlinear equations; it usually works best in a serial computing environment. The version where I consists of all unassigned persons is the one best suited for parallel computation; it is known as the *Jacobi version* because of its similarity with Jacobi methods for solving systems of nonlinear equations.

During an iteration, the objects whose prices are changed are the ones that received a bid during the iteration. Each price change involves an increase of at least ϵ. To see this, note that if person i bids for object j_i, from Eqs. (2.2)-(2.4) the corresponding bid is

$$b_{ij_i} = a_{ij_i} - w_i + \epsilon \geq a_{ij_i} - v_i + \epsilon = p_{j_i} + \epsilon,$$

and exceeds the object's current price by at least ϵ. At the end of the iteration, we have a new assignment that differs from the preceding one in

that each object that received a bid is now assigned to some person that was unassigned at the start of the iteration. However, the assignment at the end of the iteration need not have more pairs than the one at the start of the iteration, because it is possible that all objects that received a bid were assigned at the start of the iteration.

The choice of bidding increment [cf. Eq. (2.4)] is such that ϵ-CS is preserved by the algorithm, as shown by the following proposition (in fact, it can be seen that it is the largest bidding increment for which this is so).

Proposition 2.1.1: The auction algorithm preserves ϵ-CS throughout its execution; that is, if the assignment and the price vector available at the start of an iteration satisfy ϵ-CS, the same is true for the assignment and the price vector obtained at the end of the iteration.

Proof: Let p_j and p'_j be the object prices before and after a given iteration, respectively. Suppose that object j^* received a bid from person i and was assigned to i during the iteration. Then we have [see Eqs. (2.4) and (2.5)]

$$p'_{j^*} = a_{ij^*} - w_i + \epsilon.$$

Using this equation, we obtain

$$a_{ij^*} - p'_{j^*} = w_i - \epsilon = \max_{j \in A(i),\, j \neq j^*} \{a_{ij} - p_j\} - \epsilon.$$

Since $p'_j \geq p_j$ for all j, this equation implies that

$$a_{ij^*} - p'_{j^*} \geq \max_{j \in A(i)} \{a_{ij} - p'_j\} - \epsilon, \tag{2.6}$$

which shows that the ϵ-CS condition (2.1) continues to hold after the assignment phase of an iteration for all pairs (i, j^*) that entered the assignment during the iteration.

Consider also any pair (i, j^*) that belonged to the assignment just before the iteration, and also belongs to the assignment after the iteration. Then, j^* must not have received a bid during the iteration, so $p'_{j^*} = p_{j^*}$. Therefore, Eq. (2.6) holds in view of the ϵ-CS condition that held prior to the iteration and the fact $p'_j \geq p_j$ for all j. Hence, the ϵ-CS condition (2.1) holds for all pairs (i, j^*) that belong to the assignment after the iteration, proving the result. **Q.E.D.**

The next proposition establishes the validity of the algorithm. The proof relies on the following observations:

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

(b) Each time an object receives a bid, its price increases by at least ϵ [see Eqs. (2.4) and (2.5)]. Therefore, if the object receives a bid an infinite number of times, its price increases to ∞.

(c) Every $|A(i)|$ bids by person i, where $|A(i)|$ is the number of objects in the set $A(i)$, the scalar v_i defined by the equation

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\} \tag{2.7}$$

decreases by at least ϵ. The reason is that a bid by person i either decreases v_i by at least ϵ, or else leaves v_i unchanged because there is more than one object j attaining the maximum in Eq. (2.7). However, in the latter case, the price of the object j_i receiving the bid will increase by at least ϵ, and object j_i will not receive another bid by person i until v_i decreases by at least ϵ. The conclusion is that if a person i bids an infinite number of times, v_i must decrease to $-\infty$.

Proposition 2.1.2: If at least one feasible assignment exists, the auction algorithm terminates with a feasible assignment that is within $n\epsilon$ of being optimal (and is optimal if the problem data are integer and $\epsilon < 1/n$).

Proof: We argue by contradiction. If termination did not occur, the subset J^∞ of objects that received an infinite number of bids is nonempty. Also, the subset of persons I^∞ that bid an infinite number of times is nonempty. As argued in (b) above, the prices of the objects in J^∞ must tend to ∞, while as argued in (c) above, the scalars $v_i = \max_{j \in A(i)}\{a_{ij} - p_j\}$ must decrease to $-\infty$ for all persons $i \in I^\infty$. In view of ϵ-CS, this implies that

$$A(i) \subset J^\infty, \qquad \forall\, i \in I^\infty, \tag{2.8}$$

and that after a finite number of iterations, each object in J^∞ will be assigned to a person from I^∞. Since after a finite number of iterations at least one person from I^∞ will be unassigned at the start of each iteration, it

follows that the number of persons in I^∞ is strictly larger than the number of objects in J^∞. This contradicts the existence of a feasible assignment, since by Eq. (2.8), persons in I^∞ can only be assigned to objects in J^∞. Therefore, the algorithm must terminate. The feasible assignment obtained upon termination satisfies ϵ-CS by Prop. 2.1.1, so by Prop. 1.4.1 of Section 1.4, this assignment is within $n\epsilon$ of being optimal. **Q.E.D.**

2.1.2 Approximate Coordinate Descent Interpretation

The Gauss-Seidel version of the auction algorithm resembles coordinate descent algorithms, because it involves the change of a single dual variable/price, with all other prices held fixed. However, such a price change may worsen strictly the value of the dual function

$$q(p) = \sum_{i=1}^{n} \max_{j \in A(i)} \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j, \tag{2.9}$$

which was introduced in Section 1.3.

Generally we can interpret the bidding and assignment phases as a simultaneous "approximate" coordinate descent step for all price coordinates that increase during the iteration. The coordinate steps are aimed at minimizing approximately the dual function. In particular, it can be shown that *the price p_j of each object j that receives a bid during the assignment phase is increased to either a value that minimizes $q(p)$ when all other prices are kept constant or else exceeds the largest such value by no more than ϵ.*

Figure 2.1.2 proves this property and suggests that the amount of deterioration of the dual cost is at most ϵ. Indeed, for the Gauss-Seidel version of the algorithm this can be deduced from the argument given in Fig. 2.1.2 and is left as an exercise for the reader.

2.1.3 Variants of the Auction Algorithm

There are several variants of the auction algorithm that differ from each other in small details. For example, as mentioned earlier, one or several persons may bid simultaneously with objects being awarded to the highest bidders, the price increment may be slightly different than the one of Eq. (2.5), etc. The important ingredients of the method are that for each iteration:

(a) ϵ-CS is maintained.

(b) At least one unassigned person gets assigned to some object, and the price of this object is increased by at least $\beta\epsilon$, where β is some fixed positive constant. Furthermore, the person previously assigned to

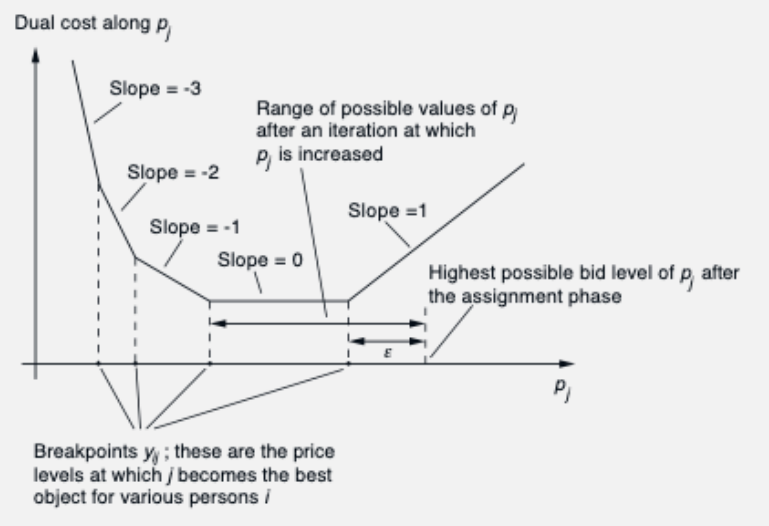

Figure 2.1.1: Form of the dual cost along the price coordinate p_j. From the definition (2.9) of the dual cost q, the right directional derivative of q along p_j is

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

where

$$y_{ij} = a_{ij} - \max_{k \in A(i),\, k \neq j} \{a_{ik} - p_k\}$$

is the level of p_j below which j is the best person for person i. The break points are y_{ij} for all i such that $j \in A(i)$. Let $\bar{i}$ be a person that attains the maximum in $\max_{\{i | j \in A(i)\}} \{a_{ij} - p_j\}$ and let $\overline{y} = y_{\bar{i}j}$. Let also $\hat{i}$ be a person that attains the maximum in $\max_{\{i | j \in A(i),\, i \neq \bar{i}\}} \{a_{ij} - p_j\}$ and let $\hat{y} = y_{\hat{i}j}$. Note that the interval $[\hat{y}, \overline{y}]$ is the set of points that minimize q along the coordinate p_j.

Let p_j be the price of j just before an iteration at which j receives a bid and let p'_j be the price of j after the iteration. We claim that $\hat{y} \leq p'_j \leq \overline{y} + \epsilon$. Indeed, if i is the person that bids and wins j during the iteration, then $p'_j = y_{ij} + \epsilon$, implying that $p'_j \leq \overline{y} + \epsilon$. To prove that $p'_j \geq \hat{y}$, we note that if $p_j \geq \hat{y}$, we must also have $p'_j \geq \hat{y}$, since $p'_j \geq p_j$. On the other hand, if $p'_j < \hat{y}$, there are two possibilities:

(1) At the start of the iteration, $\bar{i}$ was not assigned to j. In this case, either $\bar{i}$ was unassigned in which case i will bid for j so that $p'_j = \overline{y} + \epsilon$, or else $\bar{i}$ was assigned to an object $\bar{j} \neq j$, in which case by ϵ-CS,

$$a_{\bar{i}j} - p_j - \epsilon \leq a_{\bar{i}\,\bar{j}} - p_{\bar{j}} \leq \max_{k \in A(\bar{i}),\, k \neq j} \{a_{\bar{i}k} - p_k\} = a_{\bar{i}j} - \overline{y}.$$

Thus, $p_j \geq \overline{y} - \epsilon$, implying that $p'_j \geq \overline{y}$ (since a bid increases a price by at least ϵ). In both cases we have $p'_j \geq \overline{y} \geq \hat{y}$.

(2) At the start of the iteration, $\bar{i}$ was assigned to j. In this case, $\hat{i}$ was not assigned to j, so by repeating the argument of the preceding paragraph with $\hat{i}$ and $\hat{y}$ replacing $\bar{i}$ and $\overline{y}$, respectively, we obtain $p'_j \geq \hat{y}$.

an object that receives a bid during the iteration (if any) becomes unassigned.

(c) No price is decreased and every object that was assigned at the start of the iteration remains assigned at the end of the iteration (although the person assigned to it may change).

Any variant of the auction algorithm that obeys these three rules can be readily shown to have the termination property given in Prop. 2.1.2.

For example, in Section 2.2.3, we will focus on a special type of assignment problem, which involves groups of persons that are indistinguishable in the sense that they can be assigned to the same objects and with the same corresponding values. We will develop there a special variant of the auction algorithm that combines many bids into a "collective" bid for an entire group of similar persons. Not only this improves the efficiency of the method, but it also provides the vehicle for extending the auction algorithm to other problems, such as max-flow, transportation, and transhipment, as we will discuss in Chapter 4.

2.1.4 Computational Complexity – ϵ-Scaling

As noted in Section 1.3.3, the running time of the auction algorithm can depend strongly on the value of ϵ as well as the maximum absolute object value

$$C = \max_{(i,j)\in\mathcal{A}} |a_{ij}|.$$

In practice, the dependence of the running time on ϵ and C can be significant, as can be seen in the examples of Section 1.4.

The practical performance of the auction algorithm is often considerably improved by using the idea of *ϵ-scaling*, which was briefly discussed in Section 1.3.3. ϵ-scaling consists of applying the algorithm several times, starting with a large value of ϵ and successively reducing ϵ up to some final value $\bar{\epsilon}$ such that $n\bar{\epsilon}$ is deemed sufficiently small (cf. Prop. 2.1.2). Each application of the algorithm, called a *scaling phase*, provides good initial prices for the next application. The value of ϵ used for the $(k+1)$st scaling phase is denoted by ϵ^k. The sequence ϵ^k is generated by

$$\epsilon^{k+1} = \frac{\epsilon^k}{\theta}, \qquad k = 0, 1, \ldots, \tag{2.10}$$

where ϵ^0 is a suitably chosen starting value of ϵ, and θ is an integer with $\theta > 1$.†

In Section 2.3 we will derive an estimate of the worst-case running time of the auction algorithm with ϵ-scaling. This estimate is $O\big(nA\ln(\epsilon^0/\bar{\epsilon})\big)$,

† In practice, if a_{ij} are integer, they are usually first multiplied by $n+1$ and the auction algorithm is applied with progressively lower values of ϵ, to the point where ϵ becomes 1 or smaller. In this case, typical values for sparse problems,

where A is the number of arcs in the underlying graph of the assignment problem, and ϵ^0 and $\bar{\epsilon}$ are the initial and final values of ϵ, respectively. Our analysis requires a few assumptions about the way the auction algorithm and the scaling process are implemented. In particular:

(a) We assume that a Gauss-Seidel implementation is used, where only one person submits a bid at each iteration.

(b) We require that each scaling phase begins with the empty assignment.

(c) We require that the initial prices for the first scaling phase are 0, and the initial prices for each subsequent phase are the final prices of the preceding phase. Furthermore, at each scaling phase, we introduce a modification of the scalars a_{ij}, which will be discussed later.

(d) We introduce a data structure, which ensures that the bid of a person is efficiently computed.

The above requirements are essential for obtaining a favorable worst-case estimate of the running time. It is doubtful, however, that strict adherence to these requirements is essential for good practical performance. Moreover, it is possible that in a given problem, ϵ-scaling may not lead to more efficient computation. A typical case where this can happen is when good initial prices are already known through a combination of heuristics, preliminary computations, or learned experience with similar assignment problems. In such cases a small value of ϵ may lead to high quality and even optimal solutions at small computational cost.

2.2 EXTENSIONS OF THE AUCTION ALGORITHM

The auction algorithm can be extended to deal effectively with the special features of modified versions of the assignment problem. In this section, we develop several such extensions.

---

where $A << n^2$, are

$$\frac{nC}{5} \leq \epsilon^0 \leq nC, \qquad 4 \leq \theta \leq 10.$$

For nonsparse problems, sometimes $\epsilon^0 = 1$, which in effect bypasses ϵ-scaling, works quite well. Note also that practical implementations of the auction algorithm sometimes use an *adaptive* form of ϵ-scaling, whereby, within the kth scaling phase, the value of ϵ is gradually *increased* to the value ϵ^k given above, starting from a relatively small value, based on the results of the computation.

2.2.1 Reverse Auction

In the auction algorithm, persons compete for objects by bidding and raising the price of their best object. It is possible to use an alternative form of the auction algorithm, called *reverse auction*, where, roughly, the *objects* compete for persons by essentially offering discounts.

To describe this algorithm, we introduce a *profit* variable π_i for each person i. Profits play for persons a role analogous to the role prices play for objects. We can describe reverse auction in two equivalent ways: one where unassigned objects lower their prices as much as possible to attract an unassigned person or to lure a person away from its currently held object without violating ϵ-CS, and another where unassigned objects select a best person and raise his or her profit as much as possible without violating ϵ-CS. For analytical convenience, we will adopt the second description rather than the first, leaving the proof of their equivalence as an exercise for the reader.

Let us consider the following ϵ-CS condition for a (partial) assignment S and a profit vector $\pi = (\pi_1, \ldots, \pi_n)$:

$$a_{ij} - \pi_i \geq \max_{k \in B(j)} \{a_{kj} - \pi_k\} - \epsilon, \qquad \forall\ (i,j) \in S, \tag{2.11}$$

where $B(j)$ is the set of persons that can be assigned to object j,

$$B(j) = \big\{ i \mid (i,j) \in \mathcal{A} \big\}.$$

We assume that this set is nonempty for all j, which is of course required for feasibility of the problem. Note the symmetry of this condition with the corresponding one for prices; cf. Eq. (2.1). The reverse auction algorithm starts with and maintains an assignment and a profit vector π satisfying the above ϵ-CS condition. It terminates when the assignment is feasible. At the beginning of each iteration, we have an assignment S and a profit vector π satisfying the ϵ-CS condition (2.11).

Iteration of Reverse Auction

Let J be a nonempty subset of objects j that are unassigned under the assignment S. For each object $j \in J$:

1. Find a "best" person i_j such that

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

and the corresponding value

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

and find

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

[If i_j is the only person in $B(j)$, we define ω_j to be $-\infty$ or, for computational purposes, a number that is much smaller than β_j.]

2. Each object $j \in J$ bids for person i_j an amount

$$b_{i_j j} = \pi_{i_j} + \beta_j - \omega_j + \epsilon = a_{i_j j} - \omega_j + \epsilon.$$

3. For each person i that received at least one bid, increase π_i to the highest bid,

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

where $P(i)$ is the set of objects from which i received a bid; remove from the assignment S any pair (i, j) (if i was assigned to some j under S), and add to S the pair (i, j_i), where j_i is an object in $P(i)$ attaining the maximum above.

Note that reverse auction is identical to (forward) auction with the roles of persons and objects, and the roles of profits and prices interchanged. Thus, by using the corresponding (forward) auction result (cf. Prop. 2.1.2), we have the following proposition.

Proposition 2.2.1: If at least one feasible assignment exists, the reverse auction algorithm terminates with a feasible assignment that is within $n\epsilon$ of being optimal (and is optimal if the problem data are integer and $\epsilon < 1/n$).

Combined Forward and Reverse Auction

One of the reasons we are interested in reverse auction is to construct algorithms that switch from forward to reverse auction and back. Such algorithms must simultaneously maintain a price vector p satisfying the

ϵ-CS condition (2.1) and a profit vector π satisfying the ϵ-CS condition (2.11). To this end we introduce an ϵ-CS condition for the *pair* (π, p), which (as we will see) implies the other two. Maintaining this condition is essential for switching gracefully between forward and reverse auction.

Definition 2.2.1: An assignment S and a pair (π, p) are said to satisfy ϵ-CS if

$$\pi_i + p_j \geq a_{ij} - \epsilon, \qquad \forall\ (i,j) \in \mathcal{A}, \tag{2.12}$$

$$\pi_i + p_j = a_{ij}, \qquad \forall\ (i,j) \in S. \tag{2.13}$$

We have the following proposition.

Proposition 2.2.2: Suppose that an assignment S together with a profit-price pair (π, p) satisfy ϵ-CS. Then:

(a) S and π satisfy the ϵ-CS condition

$$a_{ij} - \pi_i \geq \max_{k \in B(j)} \{a_{kj} - \pi_k\} - \epsilon, \qquad \forall\ (i,j) \in S. \tag{2.14}$$

(b) S and p satisfy the ϵ-CS condition

$$a_{ij} - p_j \geq \max_{k \in A(i)} \{a_{ik} - p_k\} - \epsilon, \qquad \forall\ (i,j) \in S. \tag{2.15}$$

(c) If S is feasible, then S is within $n\epsilon$ of being an optimal assignment.

Proof: (a) In view of Eq. (2.13), for all $(i,j) \in S$, we have $p_j = a_{ij} - \pi_i$, so Eq. (2.12) implies that $a_{ij} - \pi_i \geq a_{kj} - \pi_k - \epsilon$ for all $k \in B(j)$. This shows Eq. (2.14).

(b) The proof is similar to part (a), with the roles of π and p interchanged.

(c) Since by part (b) the ϵ-CS condition (2.15) is satisfied, Prop. 1.4 of Section 1.3.3 implies that S is within $n\epsilon$ of being optimal. **Q.E.D.**

We now introduce a combined forward/reverse auction algorithm. The algorithm starts with and maintains an assignment S and a profit-price

pair (π, p) satisfying the ϵ-CS conditions (2.12) and (2.13). It terminates when the assignment is feasible.

Combined Forward/Reverse Auction Algorithm

Step 1 (Run forward auction): Execute a finite number of iterations of the forward auction algorithm (subject to the termination condition), and at the end of each iteration (after increasing the prices of the objects that received a bid) set

$$\pi_i = a_{ij_i} - p_{j_i} \tag{2.16}$$

for every person-object pair (i, j_i) that entered the assignment during the iteration. Go to Step 2.

Step 2 (Run reverse auction): Execute a finite number of iterations of the reverse auction algorithm (subject to the termination condition), and at the end of each iteration (after increasing the profits of the persons that received a bid) set

$$p_j = a_{i_j j} - \pi_{i_j} \tag{2.17}$$

for every person-object pair (i_j, j) that entered the assignment during the iteration. Go to Step 1.

Note that the additional overhead of the combined algorithm over the forward or the reverse algorithm is minimal; just one update of the form (2.16) or (2.17) is required per iteration for each object or person that received a bid during the iteration. An important property is that these updates maintain the ϵ-CS conditions (2.12) and (2.13) for the pair (π, p), and therefore, by Prop. 2.2.2, maintain the required ϵ-CS conditions (2.14) and (2.15) for π and p, respectively. This is shown in the following proposition.

Proposition 2.2.3: If the assignment and the profit-price pair available at the start of an iteration of either the forward or the reverse auction algorithm satisfy the ϵ-CS conditions (2.12) and (2.13), the same is true for the assignment and the profit-price pair obtained at the end of the iteration, provided Eq. (2.16) is used to update π (in the case of forward auction), and Eq. (2.17) is used to update p (in the case of reverse auction).

Proof: Assume for concreteness that forward auction is used, and let (π, p) and $(\overline{\pi}, \overline{p})$ be the profit-price pair before and after the iteration, respectively. Then, $\overline{p}_j \geq p_j$ for all j (with strict inequality if and only if j received a bid during the iteration). Therefore, we have $\overline{\pi}_i + \overline{p}_j \geq a_{ij} - \epsilon$ for all (i, j) such that $\pi_i = \overline{\pi}_i$. Furthermore, we have $\overline{\pi}_i + \overline{p}_j = \pi_i + p_j = a_{ij}$ for all (i, j) that belong to the assignment before as well as after the iteration. Also, in view of the update (2.16), we have $\overline{\pi}_i + \overline{p}_{j_i} = a_{ij_i}$ for all pairs (i, j_i) that entered the assignment during the iteration. What remains is to verify that the condition

$$\overline{\pi}_i + \overline{p}_j \geq a_{ij} - \epsilon, \qquad \forall\ j \in A(i), \tag{2.18}$$

holds for all persons i that submitted a bid and were assigned to an object, say j_i, during the iteration. Indeed, for such a person i, we have, by Eq. (2.4),

$$\overline{p}_{j_i} = a_{ij_i} - \max_{j \in A(i),\, j \neq j_i} \{a_{ij} - p_j\} + \epsilon,$$

which implies that

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

This shows the desired relation (2.18). **Q.E.D.**

Note that during forward auction the object prices p_j increase while the profits π_i decrease, but exactly the opposite happens in reverse auction. For this reason, the termination proof that we used for forward and for reverse auction does not apply to the combined method. Indeed, it is possible to construct examples of feasible problems where the combined method never terminates if the switch between forward and reverse auctions is done arbitrarily. However, it is easy to provide a device guaranteeing that the combined algorithm terminates for a feasible problem; it is sufficient to ensure that some "irreversible progress" is made before switching between forward and reverse auction. One easily implementable possibility is to refrain from switching until the number of assigned person-object pairs increases by at least one.

The combined forward/reverse auction algorithm often works substantially faster than the forward version. It seems to be affected less by "price wars," that is, protracted sequences of small price rises by a number of persons bidding for a smaller number of objects. Price wars can still occur in the combined algorithm, but they arise through more complex and unlikely problem structures than in the forward algorithm. For this reason the combined forward/reverse auction algorithm depends less on ϵ-scaling for good performance than its forward counterpart; in fact, starting with $\epsilon = 1/(n+1)$, thus bypassing ϵ-scaling, is sometimes the best choice.

2.2.2 Auction Algorithms for Asymmetric Assignment

Reverse auction can be used in conjunction with forward auction to provide algorithms for solving the asymmetric assignment problem, where the number of objects n is larger than the number of persons m. Here we still require that each person be assigned to some object, but we allow objects to remain unassigned. As before, 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 $(i,j) \in S$. The assignment S is said to be feasible if all persons are assigned under S.

The corresponding linear programming problem is

$$\begin{aligned}
\text{maximize} \quad & \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j\in A(i)} x_{ij} = 1, \qquad \forall\ i = 1,\ldots,m, \\
& \sum_{i\in B(j)} x_{ij} \le 1, \qquad \forall\ j = 1,\ldots,n, \\
& 0 \le x_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}.
\end{aligned}$$

We can convert this program to the transhipment problem

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in\mathcal{A}} \big(-a_{ij}\big)x_{ij} \\
\text{subject to} \quad & \sum_{j\in A(i)} x_{ij} = 1, \qquad \forall\ i = 1,\ldots,m, \\
& \sum_{i\in B(j)} x_{ij} + x_{sj} = 1, \qquad \forall\ j = 1,\ldots,n, \\
& \sum_{j=1}^{n} x_{sj} = n - m, \\
& 0 \le x_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}, \\
& 0 \le x_{sj}, \qquad \forall\ j = 1,\ldots,n,
\end{aligned}$$

by replacing maximization by minimization, by reversing the sign of a_{ij}, and by introducing a supersource node s, which is connected to each object node j by an arc (s,j) of zero cost and feasible flow range $[0,\infty)$ (see Fig. 2.2.1).

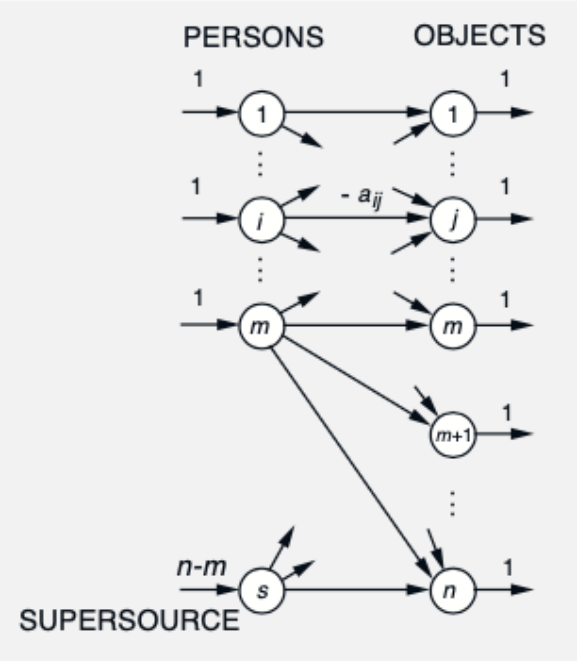

Figure 2.2.1: Converting an asymmetric assignment problem into a minimum cost flow problem involving a supersource node s and a zero cost artificial arc (s, j) with feasible flow range $[0, \infty)$ for each object j.

Using the duality theory of Section 1.3, it can be seen that the corresponding dual problem is

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} \pi_i + \sum_{j=1}^{n} p_j - (n-m)\lambda \\ \text{subject to} \quad & \pi_i + p_j \geq a_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}, \\ & \lambda \leq p_j, \qquad \forall\ j = 1, \ldots, n, \end{aligned} \tag{2.19}$$

where we have converted maximization to minimization, we have used $-\pi_i$ in place of the price of each person node i, and we have denoted by λ the price of the supersource node s.

We now introduce an ϵ-CS condition for an assignment S and a pair (π, p).

Definition 2.2.2: An assignment S and a pair (π, p) are said to satisfy ϵ-CS if

$$\pi_i + p_j \geq a_{ij} - \epsilon, \qquad \forall\ (i,j) \in \mathcal{A}, \tag{2.20}$$

$$\pi_i + p_j = a_{ij}, \qquad \forall\ (i,j) \in S, \tag{2.21}$$

$$p_j \leq \min_{k:\ \text{assigned under } S} p_k, \qquad \forall\ j \text{ that are unassigned under } S. \tag{2.22}$$

The following proposition clarifies the significance of the preceding ϵ-CS condition.

Proposition 2.2.4: If a feasible assignment S satisfies the ϵ-CS conditions (2.20)-(2.22) together with a pair (π, p), then S is within $m\epsilon$ of being optimal for the asymmetric assignment problem. The triplet $(\hat{\pi}, \hat{p}, \lambda)$, where

$$\lambda = \min_{k:\ \text{assigned under } S} p_k, \tag{2.23}$$

$$\hat{\pi}_i = \pi_i + \epsilon, \qquad \forall\ i = 1, \ldots, m, \tag{2.24}$$

$$\hat{p}_j = \begin{cases} p_j & \text{if } j \text{ is assigned under } S, \\ \lambda & \text{if } j \text{ is unassigned under } S, \end{cases} \qquad \forall\ j = 1, \ldots, n, \tag{2.25}$$

is within $m\epsilon$ of being an optimal solution of the dual problem (2.19).

Proof: For any feasible assignment $\big\{(i, k_i) \mid i = 1, \ldots, m\big\}$ and for any triplet $(\overline{\pi}, \overline{p}, \lambda)$ satisfying the dual feasibility constraints $\overline{\pi}_i + \overline{p}_j \geq a_{ij}$ for all $(i,j) \in \mathcal{A}$ and $\lambda \leq \overline{p}_j$ for all j, we have

$$\sum_{i=1}^{m} a_{ik_i} \leq \sum_{i=1}^{m} \overline{\pi}_i + \sum_{i=1}^{m} \overline{p}_{k_i} \leq \sum_{i=1}^{m} \overline{\pi}_i + \sum_{j=1}^{n} \overline{p}_j - (n-m)\lambda.$$

By maximizing over all feasible assignments $\big\{(i, k_i) \mid i = 1, \ldots, m\big\}$ and by minimizing over all dual-feasible triplets $(\overline{\pi}, \overline{p}, \lambda)$, we see that

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

where A^* is the optimal assignment value and D^* is the minimal dual cost.

Let now $S = \{(i, j_i) \mid i = 1, \ldots, m\}$ be the given assignment satisfying ϵ-CS together with (π, p), and consider the triplet $(\hat{\pi}, \hat{p}, \lambda)$ defined by Eqs. (2.23)-(2.25). Since for all i we have $\hat{\pi}_i + \hat{p}_{j_i} = a_{ij} + \epsilon$, we obtain

$$\begin{aligned} A^* &\geq \sum_{i=1}^{m} a_{ij_i} \\ &= \sum_{i=1}^{m} \hat{\pi}_i + \sum_{i=1}^{m} \hat{p}_{j_i} - m\epsilon \\ &\geq \sum_{i=1}^{m} \hat{\pi}_i + \sum_{j=1}^{n} \hat{p}_j - (n-m)\lambda - m\epsilon \\ &\geq D^* - m\epsilon, \end{aligned}$$

where the last inequality holds because the triplet $(\hat{\pi}, \hat{p}, \lambda)$ is feasible for the dual problem. Since we showed earlier that $A^* \leq D^*$, the desired conclusion follows. **Q.E.D.**

Consider now trying to solve the asymmetric assignment problem by means of auction. We can start with any assignment S and pair (π, p) satisfying the first two ϵ-CS conditions (2.20) and (2.21), and perform a forward auction (as defined earlier for the symmetric assignment problem) up to the point where each person is assigned to a distinct object. For a feasible problem, by essentially repeating the proof of Prop. 2.1.2 for the symmetric case, it can be seen that this will yield, in a finite number of iterations, a feasible assignment S satisfying the first two conditions (2.20) and (2.21). However, this assignment may not be optimal, because the prices of the unassigned objects j are not minimal; that is, they do not satisfy the third ϵ-CS condition (2.22).

To remedy this situation, we introduce a modified form of reverse auction to lower the prices of the unassigned objects so that, after several iterations in which persons may be reassigned to other objects, the third condition, (2.22), is satisfied. We will show that the assignment thus obtained satisfies all the ϵ-CS conditions (2.20)-(2.22), and by Prop. 2.2.4, is optimal within $m\epsilon$ (and thus optimal if the problem data are integer and $\epsilon < 1/m$).

The modified reverse auction starts with a feasible assignment S and with a pair (π, p) satisfying the first two ϵ-CS conditions (2.20) and (2.21). [For a feasible problem, such an S and (π, p) can be obtained by regular forward or reverse auction, as discussed earlier.] Let us denote by λ the minimal assigned object price under the initial assignment,

$$\lambda = \min_{\substack{j: \text{ assigned under the} \\ \text{initial assignment } S}} p_j.$$

The typical iteration of modified reverse auction is the same as the one of reverse auction, except that only unassigned objects j with $p_j > \lambda$ participate in the auction. In particular, the algorithm maintains a feasible assignment S and a pair (π, p) satisfying Eqs. (2.20) and (2.21), and terminates when all unassigned objects j satisfy $p_j \leq \lambda$, in which case it will be seen that the third ϵ-CS condition (2.22) is satisfied as well. The scalar λ is kept fixed throughout the algorithm.

Iteration of Reverse Auction for Asymmetric Assignment

Select an object j that is unassigned under the assignment S and satisfies $p_j > \lambda$ (if no such object can be found, the algorithm terminates). Find a "best" person i_j such that

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

and the corresponding value

$$\beta_j = \max_{i \in B(j)} \{a_{ij} - \pi_i\}, \tag{2.26}$$

and find

$$\omega_j = \max_{i \in B(j),\, i \neq i_j} \{a_{ij} - \pi_i\}. \tag{2.27}$$

[If i_j is the only person in $B(j)$, we define ω_j to be $-\infty$.] If $\lambda \geq \beta_j - \epsilon$, set $p_j := \lambda$ and go to the next iteration. Otherwise, let

$$\delta = \min\{\beta_j - \lambda, \beta_j - \omega_j + \epsilon\}. \tag{2.28}$$

Set

$$p_j := \beta_j - \delta, \tag{2.29}$$

$$\pi_{i_j} := \pi_{i_j} + \delta, \tag{2.30}$$

add to the assignment S the pair (i_j, j), and remove from S the pair (i_j, j'), where j' is the object that was assigned to i_j under S at the start of the iteration.

Note that the formula (2.28) for the bidding increment δ is such that the object j enters the assignment at a price which is no less that λ [and is equal to λ if and only if the minimum in Eq. (2.28) is attained by the first term]. Furthermore, when δ is calculated (that is, when $\lambda > \beta_j - \epsilon$) we have $\delta \geq \epsilon$, so it can be seen from Eqs. (2.29) and (2.30) that, throughout the algorithm, prices are monotonically decreasing and profits

are monotonically increasing. The following proposition establishes the validity of the method.

Proposition 2.2.5: The preceding reverse auction algorithm for the asymmetric assignment problem terminates with an assignment that is within $m\epsilon$ of being optimal.

Proof: In view of Prop. 2.2.4, the result will follow once we prove the following:

(a) The modified reverse auction iteration preserves the first two ϵ-CS conditions (2.20) and (2.21), as well as the condition

$$\lambda \leq \min_{\substack{j:\ \text{assigned under the} \\ \text{current assignment } S}} p_j, \tag{2.31}$$

so upon termination of the algorithm (necessarily with the prices of all unassigned objects less or equal to λ) the third ϵ-CS condition, (2.22), is satisfied.

(b) The algorithm terminates.

We will prove these facts in sequence.

We assume that the conditions (2.20), (2.21), and (2.31) are satisfied at the start of an iteration, and we will show that they are also satisfied at the end of the iteration. First consider the case where there is no change in the assignment, which happens when $\lambda \geq \beta_j - \epsilon$. Then Eqs. (2.21) and (2.31) are automatically satisfied at the end of the iteration; only p_j changes in the iteration according to

$$p_j := \lambda \geq \beta_j - \epsilon = \max_{i \in B(j)} \{a_{ij} - \pi_i\} - \epsilon,$$

so the condition (2.20) is also satisfied at the end of the iteration.

Next consider the case where there is a change in the assignment during the iteration. Let (π, p) and $(\overline{\pi}, \overline{p})$ be the profit-price pair before and after the iteration, respectively, and let j and i_j be the object and person involved in the iteration. By construction [cf. Eqs. (2.29) and (2.30)], we have $\overline{\pi}_{i_j} + \overline{p}_j = a_{i_j j}$, and since $\overline{\pi}_i = \pi_i$ and $\overline{p}_k = p_k$ for all $i \neq i_j$ and $k \neq j$, we see that the condition (2.21) ($\overline{\pi}_i + \overline{p}_k = a_{ik}$) is satisfied for all assigned pairs (i, k) at the end of the iteration.

To show that Eq. (2.20) holds at the end of the iteration, i.e.,

$$\overline{\pi}_i + \overline{p}_k \geq a_{ik} - \epsilon, \qquad \forall\ (i,k) \in \mathcal{A}, \tag{2.32}$$

consider first objects $k \neq j$. Then, $\overline{p}_k = p_k$ and since $\overline{\pi}_i \geq \pi_i$ for all i, the above condition holds, since our hypothesis is that at the start of the iteration we have $\pi_i + p_k \geq a_{ik} - \epsilon$ for all (i,k). Consider next the case $k = j$. Then condition (2.32) holds for $i = i_j$, since $\overline{\pi}_{i_j} + \overline{p}_j = a_{i_j j}$. Also using Eqs. (2.26)-(2.29) and the fact $\delta \geq \epsilon$, we have for all $i \neq i_j$

$$\begin{aligned}\overline{\pi}_i + \overline{p}_j &= \pi_i + \overline{p}_j \\ &\geq \pi_i + \beta_j - (\beta_j - \omega_j + \epsilon) \\ &= \pi_i + \omega_j - \epsilon \\ &\geq \pi_i + (a_{ij} - \pi_i) - \epsilon \\ &= a_{ij} - \epsilon,\end{aligned}$$

so condition (2.32) holds for $i \neq i_j$ and $k = j$, completing its proof. To see that condition (2.31) is maintained by the iteration, note that by Eqs. (2.26), (2.27), and (2.29), we have

$$\overline{p}_j = \beta_j - \delta \geq \beta_j - (\beta_j - \lambda) = \lambda.$$

Finally, to show that the algorithm terminates, we note that in the typical iteration involving object j and person i_j there are two possibilities:

(1) The price of object j is set to λ without the object entering the assignment; this occurs if $\lambda \geq \beta_j - \epsilon$.

(2) The profit of person i_j increases by at least ϵ [this is seen from the definition (2.28) of δ; we have $\lambda < \beta_j - \epsilon$ and $\beta_j \geq \omega_j$, so $\delta \geq \epsilon$].

Since only objects j with $p_j > \lambda$ can participate in the auction, possibility (1) can occur only a finite number of times. Thus, if the algorithm does not terminate, the profits of some persons will increase to ∞. This is impossible, since when person i is assigned to object j, we must have by Eqs. (2.21) and (2.31)

$$\pi_i = a_{ij} - p_j \leq a_{ij} - \lambda,$$

so the profits are bounded from above by $\max_{(i,j)\in\mathcal{A}} a_{ij} - \lambda$. Thus the algorithm must terminate. **Q.E.D.**

Note that one may bypass the modified reverse auction algorithm by starting the forward auction with all object prices equal to zero. Upon termination of the forward auction, the prices of the unassigned objects will still be at zero, while the prices of the assigned objects will be nonnegative. Therefore the ϵ-CS condition (2.22) will be satisfied, and the modified reverse auction will be unnecessary.

Unfortunately the requirement of zero initial object prices is incompatible with ϵ-scaling. The principal advantage offered by the modified reverse auction algorithm is that it allows arbitrary initial object prices for the forward auction, thereby also allowing the use of ϵ-scaling. This can

be shown to improve the theoretical worst-case complexity of the method, and is often beneficial in practice.

The method for asymmetric assignment problems just described operates principally as a forward algorithm and uses reverse auction only near the end, after the forward algorithm has terminated, to rectify violations of the ϵ-CS conditions. An alternative is to switch more frequently between forward and reverse auction, similar to the algorithm described earlier in this section for symmetric problems. We refer to Bertsekas and Castañon [1992] for methods of this type, together with computational results suggesting a more favorable practical performance over the asymmetric assignment method given earlier.

Reverse auction can also be used in the context of other types of network flow problems. One example is the variation of the asymmetric assignment problem where persons (as well as objects) need not be assigned if this degrades the assignment's value. Another assignment-like problem where reverse auction finds use is the multiassignment problem.

2.2.3 Auction Algorithms with Similar Persons

In this section, we develop an auction algorithm to deal efficiently with assignment problems that involve groups of persons that are indistinguishable in the sense that they can be assigned to the same objects and with the same corresponding benefits. This algorithm provides a general approach to extend the auction algorithm to the minimum cost flow problem and some of its special cases, such as the max-flow and the transportation problems, as we will show in Section 2.3.3.

We introduce the following definition in the context of the asymmetric or the symmetric assignment problem:

Definition 2.2.3: We say that two persons i and i' are *similar*, if

$$A(i) = A(i'), \qquad \text{and} \qquad a_{ij} = a_{i'j} \quad \forall\, j \in A(i).$$

For each person i, the set of all persons similar to i is called the *similarity class of* i.

When there are similar persons, the auction algorithm can get bogged down into a long sequence of bids (known as a "price war"), whereby a number of similar persons compete for a smaller number of objects by making small incremental price changes. An example is given in Fig. 2.2.2. It turns out that if one is aware of the presence of similar persons, one can "compress" a price war within a similarity class into a single iteration.

It is important to note that the corresponding algorithm is still a special case of the auction algorithms of Section 2.1; the computations are merely streamlined by combining many bids into a "collective" bid by the persons of a similarity class.

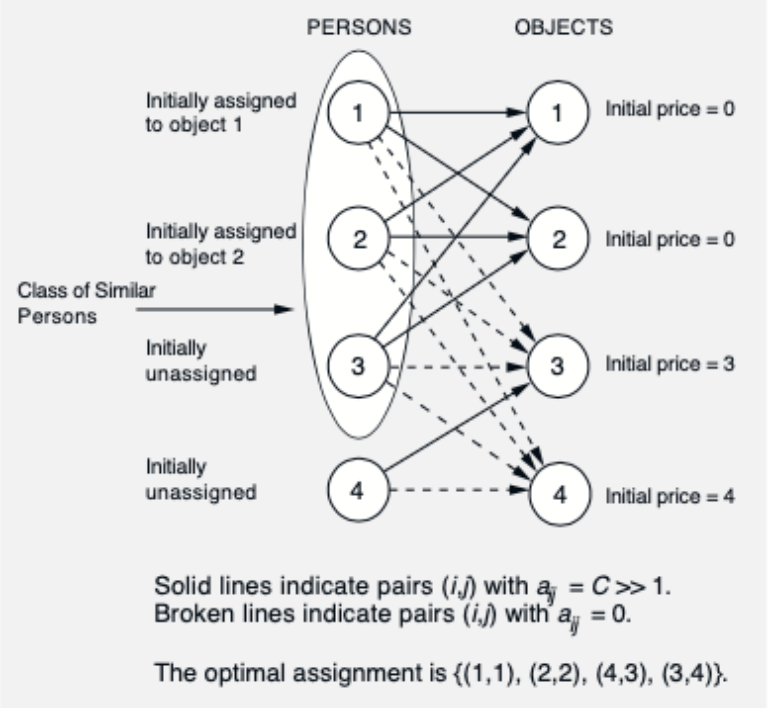

Figure 2.2.3: An example of an assignment problem with similar persons. Here the persons 1, 2, and 3 form a similarity class. This structure induces a price war in the auction algorithm. The persons 1, 2, and 3 will keep on bidding up the prices of objects 1 and 2 until the prices p_1 and p_2 reach a sufficiently high level (at least $C+3$), so that either object 3 or object 4 receives a bid from one of these persons. The price increments will be at most 2ϵ.

The method to resolve a price war within a similarity class is to let the auction algorithm run its course, then look at the final results and see how they can be essentially reproduced with less calculation. In particular, suppose that we have an assignment-price pair (S, p) satisfying ϵ-CS, and that a similarity class M has m persons, only $q < m$ of which are assigned under S. Suppose that we restrict the auction algorithm to run within M; that is, we require the bidding person to be from M, until all persons in M are assigned. We call this the *M-restricted auction*.

The final results of an M-restricted auction are quite predictable. In

particular, the set

$$A_{new} = \text{The } m \text{ objects that are assigned to persons in } M \text{ at the end of the } M\text{-restricted auction}$$

consists of the set

$$A_{old} = \text{The } q \text{ objects that were assigned to persons in } M \text{ at the beginning of the } M\text{-restricted auction}$$

plus $m - q$ extra objects that are not in A_{old}. These extra objects are those objects not in A_{old} that offered the best value $a_{ij} - p_j$ for the persons $i \in M$ (under the price vector p that prevailed at the start of the M-restricted auction). For a more precise description, let us label the set of objects not in A_{old} in order of decreasing value, that is,

$$\{j \mid j \notin A_{old}\} = \{j_1, \ldots, j_{m-q}, j_{m-q+1}, \ldots, j_{n-q}\}, \tag{2.33}$$

where for all persons $i \in M$,

$$a_{ij_r} - p_{j_r} \geq a_{ij_{r+1}} - p_{j_{r+1}}, \qquad r = 1, \ldots, n-q-1. \tag{2.34}$$

Then

$$A_{new} = A_{old} \cup \{j_1, \ldots, j_{m-q}\}. \tag{2.35}$$

The price changes of the objects as a result of the M-restricted auction can also be predicted to a great extent. In particular, the prices of the objects that are not in A_{new} will not change, since these objects do not receive any bid during the M-restricted auction. The ultimate prices of the objects $j \in A_{new}$ will be such that the corresponding values $a_{ij} - p_j$ for the persons $i \in M$ will all be within ϵ of each other, and will be no less than the value $a_{ij_{m-q+1}} - p_{j_{m-q+1}}$ of the next best object j_{m-q+1} minus ϵ. At this point, to simplify the calculations, we can just raise the prices of the objects $j \in A_{new}$ so that their final values $a_{ij} - p_j$ for persons $i \in M$ are exactly equal to the value $a_{ij_{m-q+1}} - p_{j_{m-q+1}}$ of the next best object j_{m-q+1} minus ϵ; that is, we set

$$p_j := a_{ij} - \left(a_{ij_{m-q+1}} - p_{j_{m-q+1}}\right) + \epsilon, \qquad \forall\, j \in A_{new}, \tag{2.36}$$

where i is any person in M. It can be seen that this maintains the ϵ-CS property of the resulting assignment-price pair, and that the desirable termination properties of the algorithm are maintained (see the discussion of the variants of the auction algorithm in Section 2.1.3).

To establish some terminology, consider the operation that starts with an assignment-price pair (p, S) satisfying ϵ-CS and a similarity class M that has m persons, only q of which are assigned under S, and produces through

an M-restricted auction an assignment-price pair specified by Eqs. (2.33)-(2.36). We call this operation an *M-auction iteration*. Note that when the similarity class M consists of a single person, an M-auction iteration produces the same results as the simpler auction iteration given earlier. Thus the algorithm that consists of a sequence of M-auction iterations generalizes the auction algorithm given earlier, and deals effectively with the presence of similarity classes. The table of Fig. 2.2.4 illustrates this algorithm.

Suppose now that this algorithm is started with an assignment-price pair for which the following property holds:

> If A_M is the set of objects assigned to persons of a similarity class M, the values
>
> $$a_{ij} - p_j, \qquad i \in M,\ j \in A_M,$$
>
> are all equal, and no less than the values offered by all other objects $j \notin A_M$ minus ϵ.

Then it can seen from Eqs. (2.33)-(2.36) that throughout the algorithm this property is maintained. Thus, if in particular the benefits a_{ij} of the objects in a subset $A'_M \subset A_M$ are equal, the prices p_j, $j \in A'_M$ must all be equal. This property will be useful in Section 2.3.3, where we will develop the connection between the auction algorithm and some other price-based algorithms for the max-flow and the minimum cost flow problems.

At Start of Iteration #	**Object Prices**	**Assigned Pairs**	**Bidder Class M**	**Preferred Object(s)**
1	0,0,3,4	(1,1),(2,2)	$\{1,2,3\}$	1,2,3
2	$C+4+\epsilon, C+4+\epsilon, 4+\epsilon, 4$	(1,1),(2,2),(3,3)	$\{4\}$	3
3	$C+4+\epsilon, C+4+\epsilon, C+4+\epsilon, 4$	(1,1),(2,2),(4,3)	$\{1,2,3\}$	1,2,4
Final	$2C+4+2\epsilon, 2C+4+2\epsilon$ $C+4+\epsilon, C+4+2\epsilon$	(1,1),(2,2) (4,3),(3,4)		

Figure 2.2.4: Illustration of the algorithm based on M-auction iterations for the problem of Fig. 2.2.2. In this example, the initial price vector is $(0,0,3,4)$ and the initial partial assignment consists of the pairs $(1,1)$ and $(2,2)$. We first perform an M-auction iteration for the similarity class $\{1,2,3\}$. We then perform an iteration for person 4, and then again an M-auction iteration for the similarity class $\{1,2,3\}$. The last iteration assigns the remaining object 4, and the algorithm terminates without a price war of the type discussed in Fig. 2.2.2.

2.2.4 Combinations with Dual Ascent Methods

2.3 THEORETICAL ASPECTS

In this section we derive an estimate of the worst-case running time of the auction algorithm with ϵ-scaling. This estimate is $O\big(nA\ln(\epsilon^0/\overline{\epsilon})\big)$, where A is the number of arcs in the underlying graph of the assignment problem, and ϵ^0 and $\overline{\epsilon}$ are the initial and final values of ϵ, respectively. Our analysis requires a few requirements about the way the auction algorithm and the scaling process are implemented. These requirements were mentioned in Section 2.1 and are repeated here for convenience:

(a) We assume that a Gauss-Seidel implementation is used, where only one person submits a bid at each iteration.

(b) We require that each scaling phase begins with the empty assignment.

(c) We require that the initial prices for the first scaling phase are 0, and the initial prices for each subsequent phase are the final prices of the preceding phase. Furthermore, at each scaling phase, we introduce a modification of the scalars a_{ij}, which will be discussed later.

(d) We introduce a data structure, which ensures that the bid of a person is efficiently computed.

We first focus on the case where ϵ *is fixed*. For the data structure mentioned in (d) above to work properly, we must assume that the values $a_{ij} - p_j$ are integer multiples of ϵ throughout the auction algorithm. This will be so if the a_{ij} and the initial prices p_j are integer multiples of ϵ, since in this case it is seen that the bidding increment, as given by Eq. (2.4), will be an integer multiple of ϵ. (We will discuss later how to fulfill the requirement that ϵ evenly divides the a_{ij} and the initial p_j.) To motivate the data structure, suppose that each time a person i scans all the objects $j \in A(i)$ to calculate a bid for the best object j_i, he/she records in a list denoted $Cand(i)$ all the objects $j \neq j_i$ that are tied for offering the best value; that is, they attain the maximum in the relation [cf. Eq. (2.2)]

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}. \tag{2.37}$$

Along with each object $j \in Cand(i)$, the price p'_j of j that prevailed for j *at the time of the last scan of* j is also recorded. The list $Cand(i)$ is called the *candidate list* of i, and can be used to save some computation in iterations where there are ties in the best object calculation of Eq. (2.37). In particular, if node i is unassigned and its candidate list $Cand(i)$ contains an object j whose current price p_j is equal to the price p'_j, we know that j

is the best object for i. Furthermore, the presence of a second object j in the list with $p_j = p'_j$ indicates that the bidding increment is exactly equal to ϵ. This suggests the following implementation for a bid of a person i, which will be assumed in the subsequent Prop. 2.3.1.

Bid Calculation

Step 1: Choose an unassigned person i.

Step 2: Examine the pairs (j, p'_j) corresponding to the candidate list $Cand(i)$, starting at the top. Discard any for which $p'_j < p_j$. Continue until reaching the end of the list, or the *second* element for which $p'_j = p_j$. If the end is reached, empty the candidate list and go to Step 4.

Step 3: Let j_i be the *first* element on the list for which $p'_j = p_j$. Discard the contents of the list up to, but not including, the *second* such element. Place a bid on j_i at price level $p_{j_i} + \epsilon$, assigning i to j_i and breaking any prior assignment of j_i.

Step 4: Scan the objects in $A(i)$, determining an object j_i of maximum value, the next best value w_i, as given by Eq. (2.3), and all objects (other than j_i) tied at value level w_i, and record these objects in the candidate list together with their current prices. Submit a bid for j_i at price level b_{ij_i}, as given by Eq. (2.4), assigning i to j_i and breaking any prior assignment of j_i.

We note that candidate lists are often used in the calculations of various auction algorithms to improve theoretical efficiency. For example they will also be used later in the algorithms of Sections 2.3 and 2.4.

The complexity analysis of the auction algorithm is based on the following proposition, which estimates the amount of computation needed to reduce the violation of CS by a given factor $r > 1$; that is, to obtain a feasible assignment and price vector satisfying ϵ-CS, starting from a feasible assignment and price pair satisfying $r\epsilon - CS$. Because each price increase is of size at least ϵ, the value

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

decreases by at least ϵ each time the prices p_j of all the objects $j \in A(i)$ that attain the maximum above increase by at least ϵ. The significance of the preceding method for bid calculation is that for v_i to decrease by at least ϵ, it is sufficient to scan the objects in $A(i)$ in Step 4 *only once*. Assuming that the problem is feasible, we will provide in the following proposition an

upper bound on the amount by which v_i can decrease, thereby bounding the number of bids that a person can submit in the course of the algorithm, and arriving at a running time estimate.

Proposition 2.3.2: Let the auction algorithm be applied to a feasible assignment problem, with a given $\epsilon > 0$ and with the bid calculation method just described. Assume that:

(1) All the scalars a_{ij} and all the initial object prices are integer multiples of ϵ.

(2) For some scalar $r \geq 1$, the initial object prices satisfy $r\epsilon$-CS together with some feasible assignment.

Then the running time of the algorithm is $O(rnA)$.

Proof: Let p^0 be the initial price vector and let S^0 be the feasible assignment together with which p^0 satisfies $r\epsilon$-CS. Let also (S, p) be an assignment-price pair generated by the algorithm *prior* to termination (so that S is infeasible). Define for all persons i

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}, \qquad v_i^0 = \max_{j \in A(i)} \{a_{ij} - p_j^0\}.$$

The values v_i are monotonically nonincreasing in the course of the algorithm. We will show that the differences $v_i^0 - v_i$ are upper bounded by $(r+1)(n-1)\epsilon$.

Let i be a person that is unassigned under S. We claim that there exists a path of the form

$$(i, j_1, i_1, \ldots, j_m, i_m, j_{m+1})$$

where $m \geq 0$ and:

(1) j_{m+1} is unassigned under S.

(2) If $m > 0$, then for $k = 1, \ldots, m$, i_k is assigned to j_k under S and is assigned to j_{k+1} under S^0.

This can be shown constructively using the following algorithm: Let j_1 be the object assigned to i under S^0. If j_1 is unassigned under S, stop; else let i_1 be the person assigned to j_1 under S, and note that $i_1 \neq i$. Let j_2 be the person assigned to i_1 under S^0, and note that $j_2 \neq j_1$ since j_1 is assigned to i under S^0 and $i_1 \neq i$. If j_2 is unassigned under S, stop; else continue similarly. This procedure cannot produce the same object twice, so it must terminate with the properties (1) and (2) satisfied after $m+1$ steps, where $0 \leq m \leq n-2$.

Since the pair (S^0, p^0) satisfies $r\epsilon$-CS, we have

$$v_i^0 = \max_{j \in A(i)} \{a_{ij} - p_j\} \leq a_{ij_1} - p_{j_1}^0 + r\epsilon,$$

$$a_{i_1 j_1} - p_{j_1}^0 \leq a_{i_1 j_2} - p_{j_2}^0 + r\epsilon,$$

$$\cdots$$

$$a_{i_m j_m} - p_{j_m}^0 \leq a_{i_m j_{m+1}} - p_{j_{m+1}}^0 + r\epsilon.$$

Since the pair (S, p) satisfies ϵ-CS, we have

$$v_i \geq a_{ij_1} - p_{j_1} - \epsilon,$$

$$a_{i_1 j_1} - p_{j_1} \geq a_{i_1 j_2} - p_{j_2} - \epsilon,$$

$$\cdots$$

$$a_{i_m j_m} - p_{j_m} \geq a_{i_m j_{m+1}} - p_{j_{m+1}} - \epsilon.$$

Since j_{m+1} is unassigned under S, we have $p_{j_{m+1}} = p_{j_{m+1}}^0$, so by adding the preceding inequalities, we obtain the desired relation

$$v_i^0 - v_i \leq (r+1)(m+1)\epsilon \leq (r+1)(n-1)\epsilon, \qquad \forall\ i. \tag{2.38}$$

We finally note that because a_{ij} and p_j^0 are integer multiples of ϵ, all subsequent values of p_j, $a_{ij} - p_j$, and $v_i = \max_{j \in A(i)}\{a_{ij} - p_j\}$ will also be integer multiples of ϵ. Therefore, with the use of the candidate list $Cand(i)$, the typical bid calculation, as given earlier, scans only once the objects in $A(i)$ in Step 4 to induce a reduction of v_i by at least ϵ. It follows that the total number of computational operations for the bids of node i is proportional to $(r+1)(n-1)|A(i)|$, where $|A(i)|$ is the number of objects in $A(i)$. Thus, the algorithm's running time is $(r+1)(n-1)\sum_{i=1}^n |A(i)| = O(rnA)$, as claimed. **Q.E.D.**

Complexity with ϵ-Scaling

We will now estimate the running time of the auction algorithm with ϵ-scaling. A difficulty here is that in order to use the estimate of Prop. 7.3, the a_{ij} and p_j at each scaling phase must be integer multiples of the prevailing ϵ for that phase. We bypass this difficulty as follows:

(a) We start the first scaling phase with $p_j = 0$ for all j.

(b) We use the final prices of each scaling phase as the initial prices for the next scaling phase.

(c) We choose $\bar{\epsilon}$, the final value of ϵ, to divide evenly all the a_{ij}. [We assume that such a common divisor can be found. This will be true

if the a_{ij} are rational. Otherwise, the a_{ij} may be approximated arbitrarily closely, say within some $\delta > 0$, by rational numbers, and the final assignment will be within $n(\bar{\epsilon} + \delta)$ of being optimal. If the a_{ij} are integer, we choose $\epsilon = 1/(n+1)$, which also guarantees optimality of the final assignment.] Furthermore, we choose ϵ^0 to be equal to a fraction of the range

$$C = \max_{(i,j)\in\mathcal{A}} |a_{ij}|,$$

which is fixed and independent of the problem data.

(d) We replace each a_{ij} at the beginning of the $(k+1)$st scaling phase with a corrected value a_{ij}^k that is divisible by ϵ^k. The correction is of size at most ϵ^k. In particular, we may use in place of a_{ij},

$$a_{ij}^k = \left\lceil \frac{a_{ij}}{\epsilon^k} \right\rceil \epsilon^k, \qquad \forall\ (i,j) \in \mathcal{A},\ k = 0, 1, \ldots$$

However, no correction is made in the last scaling phase, since each a_{ij} is divisible by $\bar{\epsilon}$ [cf. (c) above].

It can be seen that since the a_{ij}^0 and the initial (zero) p_j used in the first scaling phase are integer multiples of ϵ^0, the final prices of the first scaling phase are also integer multiples of ϵ^0, and thus also integer multiples of $\epsilon^1 = \epsilon^0/\theta$ (since θ is integer). Therefore, the a_{ij}^1 and initial p_j used in the second scaling phase are integer multiples of ϵ^1, which similarly guarantees that the final prices of the second scaling phase are also integer multiples of $\epsilon^2 = \epsilon^1/\theta$. Continuing in this manner (or using induction), we see that the object benefits and prices are integer multiples of the prevailing value of ϵ throughout the algorithm.

Thus, we can use Prop. 7.3 to estimate the complexity of the $(k+1)$st scaling phase as $O(r^k nA)$, where r^k is such that the initial prices p_j^k of the scaling phase satisfy $r^k\epsilon^k$-CS with some feasible assignment S^k, and with respect to the object benefits a_{ij}^k. Take S^k to be the final assignment of the preceding (the kth) scaling phase, which must satisfy ϵ^{k-1}-CS (or $\theta\epsilon^k$-CS) with respect to the object benefits a_{ij}^{k-1}. Since, for all $(i,j) \in \mathcal{A}$ and k, we have

$$|a_{ij}^k - a_{ij}^{k-1}| \leq |a_{ij}^k - a_{ij}| + |a_{ij} - a_{ij}^{k-1}| \leq \epsilon^k + \epsilon^{k-1} = (1+\theta)\epsilon^k,$$

it can be seen, using the definition of ϵ-CS, that S^k and p_j^k must satisfy $\big(\theta + 2(1+\theta)\big)\epsilon^k$-CS. It follows that we can use $r^k = \theta + 2(1+\theta)$ in the complexity estimate $O(r^k nA)$ of the $(k+1)$st scaling phase. Thus the running time of all scaling phases except for the first is $O(nA)$. Because ϵ^0 is equal to a fixed fraction of the range C, the initial scaling phase will also have a running time $O(nA)$, since then the initial (zero) price vector will satisfy $r\epsilon^0$-CS with any feasible assignment, where r is some fixed constant. Since $\epsilon^k = \theta\epsilon^{k-1}$ for all $k = 0, 1, \ldots$, the total number of scaling phases is

$O\big(\log(\epsilon^0/\overline{\epsilon})\big)$, and it follows that the running time of the auction algorithm with ϵ-scaling is $O\big(nA\log(\epsilon^0/\overline{\epsilon})\big)$.

Suppose now that the a_{ij} are integer, and that we use $\overline{\epsilon}$ equal to $1/(n+1)$ and ϵ^0 equal to a fixed fraction of the benefit range C. Then $\epsilon^0/\overline{\epsilon} = O(nC)$, and an optimal assignment will be found with $O\big(nA\log(nC)\big)$ computation. This is a worst-case estimate. In practice, the average running time of the algorithm with ϵ-scaling seems to grow proportionally to something like $A\log n\log(nC)$.

We note that the implementation using the candidate lists was important for the proof of Prop. 7.3 and the $O\big(nA\log(\epsilon^0/\overline{\epsilon})\big)$ running time of the method with ϵ-scaling. However, it is doubtful that the overhead for maintaining the candidate lists is justified. In practice, a simpler implementation is usually preferred, whereby each person scans all of its associated objects at each bid, instead of using candidate lists. Also the approach of modifying the a_{ij} to make them divisible by the prevailing value of ϵ, while important for the complexity analysis, is of questionable practical use. It is simpler and typically as effective in practice to forego this modification. An alternative approach to the complexity analysis, which uses a slightly different method for selecting the object that receives a bid, is described in Section 9.6, in the context of auction algorithms for separable convex problems.

Since termination of the auction algorithm can only occur with a feasible assignment, when the problem is infeasible, the auction algorithm will keep on iterating, as the user is wondering whether the problem is infeasible or just hard to solve. Thus for problems where existence of a feasible assignment is not known a priori, one must supplement the auction algorithm with a mechanism to detect infeasibility. There are several such mechanisms, which we will now discuss.

One criterion that can be used to detect infeasibility is based on the maximum values

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

It can be shown that if the problem is feasible, then in the course of the auction algorithm, all of these values will be bounded from below by a precomputable bound, but if the problem is infeasible, some of these values will be eventually reduced below this bound. In particular, suppose that the auction algorithm is applied to a symmetric assignment problem with initial object prices $\{p_j^0\}$. Then it can be shown (see the proof of Prop. 2.3.2) that if person i is unassigned with respect to the current assignment S and the problem is feasible, then there is an augmenting path with respect to S that starts at i. Furthermore, by adding the ϵ-CS condition along the augmenting path, as in the proof of Prop. 2.3.2, we obtain

$$v_i \geq -(2n-1)C - (n-1)\epsilon - \max_j\{p_j^0\}, \tag{2.39}$$

where $C = \max_{(i,j)\in\mathcal{A}} |a_{ij}|$. If the problem is feasible, then as discussed earlier, there exists at all times an augmenting path starting at each unassigned person, so the lower bound (2.39) on v_i will hold for all unassigned persons i throughout the auction algorithm. On the other hand, if the problem is infeasible, some persons i will be submitting bids infinitely often, and the corresponding values v_i will be decreasing towards $-\infty$. Thus, we can apply the auction algorithm and keep track of the values v_i as they decrease. Once some v_i gets below its lower bound, we know that the problem is infeasible.

Unfortunately, it may take many iterations for some v_i to reach its lower bound, so the preceding method may not work well in practice. An alternative method to detect infeasibility is to convert the problem to a feasible problem by adding a set of artificial pairs $\overline{\mathcal{A}}$ to the original set $\mathcal{A}$. The benefits a_{ij} of the artificial pairs (i,j) should be very small, so that none of these pairs participates in an optimal assignment unless the problem is infeasible. In particular, it can be shown that if the original problem is feasible, no pair $(i,j) \in \overline{\mathcal{A}}$ will participate in the optimal assignment, provided that

$$a_{ij} < -(2n-1)C, \qquad \forall\ (i,j) \in \overline{\mathcal{A}}, \tag{2.40}$$

where $C = \max_{(i,j)\in\mathcal{A}} |a_{ij}|$. To prove this by contradiction, assume that by adding to the set $\mathcal{A}$ the set of artificial pairs $\overline{\mathcal{A}}$ we create an optimal assignment S^* that contains a nonempty subset $\overline{S}$ of artificial pairs. Then, for every assignment S consisting exclusively of pairs from the original set $\mathcal{A}$ we must have

$$\sum_{(i,j)\in\overline{S}} a_{ij} + \sum_{(i,j)\in S^*-\overline{S}} a_{ij} \geq \sum_{(i,j)\in S} a_{ij},$$

from which

$$\sum_{(i,j)\in\overline{S}} a_{ij} \geq \sum_{(i,j)\in S} a_{ij} - \sum_{(i,j)\in S^*-\overline{S}} a_{ij} \geq -(2n-1)C.$$

This contradicts Eq. (2.40). Note that if $a_{ij} \geq 0$ for all $(i,j) \in \mathcal{A}$, the preceding argument can be modified to show that it is sufficient to have $a_{ij} < -(n-1)C$ for all artificial pairs (i,j).

On the other hand, the addition of artificial pairs with benefit $-(2n-1)C$ as per Eq. (2.40) expands the cost range of the problem by a factor of $(2n-1)$. In the context of ϵ-scaling, this necessitates a much larger starting value for ϵ and correspondingly large number of ϵ-scaling phases. If the problem is feasible, these extra scaling phases are wasted. Thus for problems which are normally expected to be feasible, it may be better to introduce artificial pairs with benefits that are of the order of $-C$, and then gradually scale downward these benefits towards the $-(2n-1)C$ threshold if artificial pairs persist in the assignments obtained by the auction algorithm.

This procedure of scaling downward the benefits of the artificial pairs can be embedded in a number of ways within the ϵ-scaling procedure.

A third method to deal with infeasibility is based on the notion of maximally feasible flows and the decomposition method discussed in Section 3.1.4. It uses the property that even when the problem is infeasible, the auction algorithm will find an assignment of maximal cardinality in a finite number of iterations (this can be seen by a simple modification of the proof of Prop. 2.3.2). The idea now is to modify the auction algorithm so that during the first scaling phase we periodically check for the existence of an augmenting path from some unassigned person to some unassigned object. Once the cardinality of the current assignment becomes maximal while some person still remains unassigned, this check will establish that the problem is infeasible. With this modification, the auction algorithm will either find a feasible assignment and a set of prices satisfying ϵ-CS, or it will establish that the problem is infeasible and simultaneously obtain an assignment of maximal cardinality. In the former case, the algorithm will proceed with subsequent scaling phases of the algorithm, but with the breadth-first feature suppressed. In the latter case, we can use the maximal cardinality assignment obtained to decompose the problem into two or three component problems, as will be discussed later. Each of these problems is either a symmetric or an asymmetric assignment problem, which can be solved separately.

Note a nice feature of the approach just described: In the case of a feasible problem, it involves little additional computation (the breadth-first searches of the first scaling phase) over the unmodified algorithm. In the case of an infeasible problem, the computation of the first scaling phase is not wasted, since it provides good starting prices for the subsequent scaling phases.

2.4 NOTES AND SOURCES

The auction algorithm, and the notions of ϵ-complementary slackness and ϵ-scaling were first proposed by the author (Bertsekas [1979a]; see also Bertsekas [1988]). Another paper by the author, [Ber81], has focused on the naive auction algorithm, and its combination with a primal-dual method as a means to over come its convergence difficulties (cf. Section 1.4.1). The worst-case complexity of the auction algorithm was given by Bertsekas and Eckstein [1988], who used an alternative method of scaling whereby ϵ is kept constant and the values a_{ij} are successively scaled to their final values; see also the complexity analysis in Bertsekas and Tsitsiklis [1989]. Tutorial presentations of auction algorithms that supplement this chapter are given in Bertsekas [1990], [1992a]. A comprehensive presentation, in-

cluding several additional auction variants, is given in Chapter 7 of the book [Ber98].

Auction algorithms are particularly well-suited for parallel computation because both the bidding and the assignment phases are highly parallelizable. In particular, the bids can be computed simultaneously and in parallel for all persons participating in the auction. Similarly, the subsequent awards to the highest bidders can be computed in parallel by all objects that received a bid. In fact these operations maintain their validity in an asynchronous environment where the bidding phase is executed with price information that is outdated because of communication delays between the processors of the parallel computing system. The parallel computation aspects of the auction algorithm have been explored by Bertsekas and Tsitsiklis [1989], Bertsekas and Castañon [1991], Wein and Zenios [1991], Amini [1994], and Bertsekas, Castañon, Eckstein, and Zenios [1995].

The reverse auction algorithm and its application in asymmetric assignment problems is due to Bertsekas, Castañon, and Tsaknakis [1993]. An extensive computational study of forward and reverse auction algorithms is given in Castañon [1993]. Still another auction algorithm of the forward-reverse type for asymmetric assignment problems is given by Bertsekas and Castañon [1992]. An extension of the auction algorithm to transportation problems based on the notion of similar persons is given in Bertsekas and Castañon [1989].