Auction Algorithms

Dimitri P. Bertsekas
bertsekas@lids.mit.edu

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

The auction algorithm is an intuitive method for solving the classical assignment problem. It outperforms substantially its main competitors for important types of problems, both in theory and in practice, and is also naturally well suited for parallel computation. In this article, we will skecth the basic principles of the algorithm, we will explain its computational properties, and we will discuss its extensions to more general network flow problems. For a detailed presentation, we refer to the survey paper [Ber92] and the author's textbooks [Ber91], [Ber98]. For an extensive computational study, we refer to Castañon [Cas93]. The algorithm was first proposed in a 1979 report by the author [Ber79].

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

The assignment problem is important in many practical contexts. The most obvious ones are resource allocation problems, such as assigning personnel to jobs, machines to tasks, and the like. There are also situations where the assignment problem appears as a subproblem in various methods for solving more complex problems.

The assignment problem is also of great theoretical importance because, despite its simplicity, it embodies a fundamental linear programming structure. The most important type of linear programming problems, the linear network flow problem, can be reduced to the assignment problem by means of a simple reformulation. Thus, any method for solving the assignment problem can be generalized to solve the linear network flow problem, and in fact this approach is particularly helpful in understanding the extension of auction algorithms to network flow problems that are more general than assignment.

The classical methods for assignment are based on iterative improvement of some cost function; for example a primal cost (as in primal simplex methods), or a dual cost (as in Hungarian-like methods, dual simplex methods, and relaxation methods). The auction algorithm departs significantly from the cost improvement idea; at any one iteration, it may deteriorate both the primal and the dual cost, although in the end it finds an optimal assignment. It is based on a notion of approximate optimality, called *ϵ-complementary slackness*, and while it implicitly tries to solve a dual problem, it actually attains a dual solution that is not quite optimal.

The Auction Process

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

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

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

Equilibrium assignments and prices are naturally of great interest to economists, but there is also a fundamental relation with the assignment problem; it turns out that an equilibrium assignment offers maximum total benefit (and thus solves the assignment problem), while the corresponding set of prices solves an associated dual optimization problem. This is a consequence of the celebrated duality theorem of linear programming.

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

The naive auction algorithm proceeds in “rounds” (or “iterations”) starting with *any* assignment and *any* set of prices. There is an assignment and a set of prices at the beginning of each round, and if all persons are happy with these, the process terminates. Otherwise some person who is not happy is selected.

This person, call him i, finds an object j_i which offers maximal value, that is,

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

and then:

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

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

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

where

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

v_i is the best object value,

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

and w_i is the second best object value

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

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

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

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

Does this auction process work? Unfortunately, not always. The difficulty is that the bidding increment γ_i is zero when more than one object offers maximum value for the bidder i [cf. Eqs. (4), (6)]. As a result, a situation may be created where several persons contest a smaller number of equally desirable objects without raising their prices, thereby creating a never ending cycle.

To break such cycles, we introduce a perturbation mechanism, motivated by real auctions where each bid for an object must raise its price by a minimum positive increment, and bidders must on occasion take risks to win their preferred objects. In particular, let us fix a positive scalar ϵ and say that a person i

is *almost happy* with an assignment and a set of prices if the value of its assigned object j_i is within ϵ of being maximal, that is,

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

We will say that an assignment and a set of prices are *almost at equilibrium* when all persons are almost happy. The condition (7), introduced first in 1979 in conjunction with the auction algorithm, is known as *ϵ-complementary slackness* and plays a central role in several optimization contexts. For $\epsilon = 0$ it reduces to ordinary complementary slackness [compare Eq. (1)].

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

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

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

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

Optimality Properties at Termination

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

Suppose now that the benefits a_{ij} are all integer, which is the typical practical case (if a_{ij} are rational numbers, they can be scaled up to integer by multiplication with a suitable common number). Then, the total benefit of any assignment is integer, so if $n\epsilon < 1$, a complete assignment that is within $n\epsilon$ of being optimal must be optimal. It follows, that *if*

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

and the benefits a_{ij} are all integer, then the assignment obtained upon termination of the auction algorithm is optimal. Let us also note that the final set of prices is within $n\epsilon$ of being an optimal solution of the dual problem

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

This leads to the interpretation of the auction algorithm as a dual algorithm (in fact an approximate coordinate ascent algorithm; see the cited literature).

Computational Aspects – ϵ-Scaling

The auction algorithm exhibits interesting computational behavior, and it is essential to understand this behavior to implement the algorithm efficiently. First note that the amount of work to solve the problem can depend strongly on the value of ϵ and on the maximum absolute object value

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

Basically, for many types of problems, the number of bidding rounds up to termination tends to be proportional to C/ϵ. Note also that there is a dependence on the initial prices; if these prices are "near optimal," we expect that the number of rounds to solve the problem will be relatively small.

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

In practice, it is a good idea to at least consider scaling. For sparse assignment problems, that is, problems where the set of feasible assignment pairs is severely restricted, scaling seems almost universally helpful. In theory, scaling leads to auction algorithms with a particularly favorable polynomial complexity (without scaling, the algorithm is pseudopolynomial; see the cited literature).

Parallel and Asynchronous Implementation

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

Bertsekas and Castañon [BeC91] give a careful formulation of the totally asynchronous model, and a proof of its validity. They include also extensive computational results on a shared memory machine, confirming the advantage of asynchronous over synchronous implementations.

Variations and Extensions

The auction algorithm can be extended to solve a number of variations of the assignment problem, such as the asymmetric assignment problem where the number of objects is larger than the number of persons and there is a requirement that all persons be assigned to some object. Naturally, the notion of an assignment must now be modified appropriately. To solve this problem, the auction algorithm need only be modified in the choice of initial conditions. It is sufficient to require that all initial prices be zero. A similar algorithm can be used for the case where there is no requirement that all persons be assigned. Other variations handle efficiently the cases where there are several groups of "identical" persons or objects (Bertsekas and Castañon [BeC89]).

There have been extensions of the auction algorithm for other types of linear network optimization problems. The general approach for constructing auction algorithms for such problems is to convert them to assignment problems, and then to suitably apply the auction algorithm and streamline the computations. In particular, the classical shortest path problem can be solved correctly by the naive auction algorithm described earlier, once the method is streamlined. Similarly, auction algorithms can be cosntructed for the max-flow problems, and are very efficient. These algorithms bear a close relation to preflow-push algorithms for the max-flow problem, which were developed independently of auction ideas.

The auction algorithm has been extended to solve linear transportation problems (Bertsekas and Castañon [Ber89]). The basic idea is to convert the transportation problem into an assignment problem by creating multiple copies of persons (or objects) for each source (or sink respectively), and then to modify the auction algorithm to take advantage of the presence of the multiple copies.

There are extensions of the auction algorithm for linear minimum cost flow (transshipment) problems, such as the so called ϵ-relaxation method, and the auction/sequential shortest path algorithm algorithm (see the cited literature for a detailed description). These methods have interesting theoretical properties and like the auction algorithm, are well suited for parallelization (see the survey by Bertsekas, Castañon, Eckstein, and Zenios [BCE95], and the textbook by Bertsekas and Tsitsiklis [BeT89]).

Let us finally note that there have been proposals of auction algorithms for convex separable network optimization problems with and without gains (but with a single commodity and without side constraints); see Tseng and Bertsekas [TsB96].

REFERENCES

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

[Ber91] Bertsekas, D. P., 1991. Linear Network Optimization: Algorithms and Codes, MIT Press, Cambridge, MA.

[Ber92] Bertsekas, D. P., 1992. "Auction Algorithms for Network Flow Problems: A Tutorial Introduction," Computational Optimization and Applications, Vol. 1, pp. 7-66.

[Ber91] Bertsekas, D. P., 1991. Linear Network Optimization: Algorithms and Codes, MIT Press, Cambridge, MA.

[Ber98] Bertsekas, D. P., 1998. Network Optimization: Continuous and Discrete Problems, Athena Scientific, Belmont, MA.

[BeC89] Bertsekas, D. P., and Castañon, D. A., 1989. "The Auction Algorithm for Transportation Problems," Annals of Operations Research, Vol. 20, pp. 67-96.

[BCE95] Bertsekas, D. P., Castañon, D. A., Eckstein, J., and Zenios, S., 1995. "Parallel Computing in Network Optimization," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 7, North-Holland, Amsterdam, pp. 331-399.

[BeC89] Bertsekas, D. P., and Tsitsiklis, J. N., 1989. Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J. (republished in 1997 by Athena Scientific, Belmont, MA).

[Cas93] Castañon, D. A., 1993. "Reverse Auction Algorithms for Assignment Problems," in Algorithms for Network Flows and Matching, Johnson, D. S., and McGeoch, C. C. (eds.), American Math. Soc., Providence, RI, pp. 407-429.

[TsB96] Tseng, P., and Bertsekas, D. P., 1996. "An Epsilon-Relaxation Method for Separable Convex Cost Generalized Network Flow Problems," Lab. for Information and Decision Systems Report P-2374, M.I.T., Cambridge, MA; to appear in Math. Programming.