

Auction Algorithms for Network Flow Problems: A Tutorial Introduction¹

by

Dimitri P. Bertsekas²

Abstract

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. It operates like a real auction where persons compete for objects by raising their prices through competitive bidding. The algorithm represents a significant departure from the cost improvement idea that underlies primal simplex and dual ascent methods; at any one iteration, it may deteriorate both the primal and the dual cost, although in the end it does find an optimal primal solution. The auction algorithm can be extended to solve a variety of other linear network flow problems such as transportation, transshipment, max-flow, and shortest path. This paper provides a tutorial introduction to the basic auction algorithm and surveys its various extensions; for textbooks presentations, see [BeT89] and [Ber91a].

¹ Based on the plenary lecture of the author at the Twenty-Ninth Annual Allerton Conference, Allerton Park, Ill., Oct. 1991. Supported by the National Science Foundation under Grant DDM-8903385 and Grant CCR-9103804, and the Army Research Office under Grant DAAL03-86-K-0171.

² Department of Electrical Engineering and Computer Science, Rm 35-210, M. I. T., Cambridge, Mass., 02139.

1. THE ASSIGNMENT PROBLEM

The classical algorithms for solving linear network flow problems are primal cost improvement methods, including simplex methods, which iteratively improve the primal cost by moving flow around simple cycles, and dual ascent methods, which iteratively improve the dual cost by changing the prices of a subset of nodes by equal amounts. These methods are dual to each other when viewed in the context of Rockafellar's monotropic programming theory [Roc84]; they are based on cost improvement along elementary directions of the circulation space (in the primal case) or the differential space (in the dual case).

Auction algorithms are fundamentally different; they depart from the cost improvement idea and at any one iteration, they may deteriorate both the primal and the dual cost, although in the end they find an optimal primal solution. Their origin lies in nondifferentiable optimization (where nonmonotonic subgradient-based algorithms are common), and in the ϵ -subgradient method of [BeM73] (where the progress of the method depends on gradually reducing an ϵ parameter to an acceptable tolerance level). Auction algorithms are also highly intuitive and easy to understand; they can be explained in terms of economic competition concepts, which are couched on everyday experience.

Auction algorithms originated in 1979 with an algorithm proposed by the author for solving the classical assignment problem [Ber79]. The motivation was to solve the problem by using parallelism in a natural way. It turned out, however, that the resulting method was very fast in a serial environment as well. Subsequent work extended the auction algorithm to other linear network flow problems. The purpose of this paper is to provide a tutorial introduction to auction algorithms. We will initially focus on the basic algorithm for the assignment problem, and then discuss various extensions to other problems.

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

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/her 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

$$a_{ij_i} - p_{j_i} = \max_{j=1, \dots, n} \{a_{ij} - p_j\}. \quad (1)$$

We say that an assignment and a set of prices satisfy *complementary slackness (or CS for short)* when this condition holds for all persons i . The economic system would then be at equilibrium, in the sense that no person would have an incentive to act unilaterally, seeking another object.

A basic linear programming result is that if a feasible assignment and a set of prices satisfy the complementary slackness conditions (1) for all persons i , then the assignment is optimal and the prices are an optimal solution of the following problem

$$\min_{p_j} \left\{ \sum_{j=1}^n p_j + \sum_{i=1}^n \max_j \{a_{ij} - p_j\} \right\}, \quad (2)$$

called the *dual problem*. Furthermore, the value of the optimal assignment and the optimal cost of the dual problem are equal.

2. THE NAIVE AUCTION ALGORITHM

Let us consider a natural process for finding an equilibrium assignment and price vector. We 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 iterations and generates a sequence of price vectors and partial assignments. By a *partial assignment* we mean an assignment where only a subset of the persons have been matched with objects. At the beginning of each iteration, the CS condition

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

is satisfied for all pairs (i, j_i) of the partial assignment.

If all persons are assigned, the algorithm terminates. Otherwise a nonempty subset I of persons i that are unassigned is selected. Each of these persons, call him/her i , finds an object j_i which offers maximal value, that is,

$$j_i \in \arg \max_{j=1, \dots, n} \{a_{ij} - p_j\}, \quad (3)$$

and computes a bidding increment

$$\gamma_i = v_i - w_i, \quad (4)$$

where v_i is the best object value,

$$v_i = \max_j \{a_{ij} - p_j\}, \quad (5)$$

and w_i is the second best object value

$$w_i = \max_{j \neq i} \{a_{ij} - p_j\}. \quad (6)$$

Each object j that is selected as best object by a nonempty subset $P(j)$ of persons in I , determines the highest bidder

$$i_j = \max_{i \in P(j)} \gamma_i, \quad (7)$$

raises its prices by the highest bidding increment $\max_{i \in P(j)} \gamma_i$, and gets assigned to the highest bidder i_j ; the person that was assigned to j at the beginning of the iteration (if any) becomes unassigned.

This process is repeated in a sequence of iterations until all persons have an assigned object.

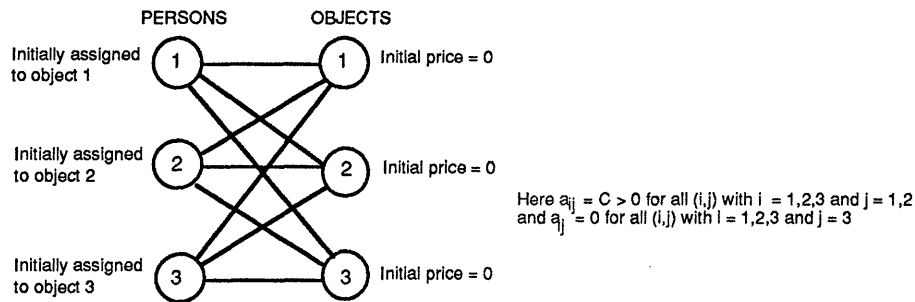
Note that γ_i cannot be negative since $v_i \geq w_i$ [compare Eqs. (5) and (6)], so the object prices tend to increase. Just as in a real auction, bidding increments and price increases spur competition by making the bidder's own preferred object less attractive to other potential bidders.

Note also 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. This version, known as the *Gauss-Seidel version* because of its similarity with Gauss-Seidel methods for solving systems of nonlinear equations, 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.

3. ϵ -COMPLEMENTARY SLACKNESS

Unfortunately, the naive auction algorithm does not always work (although it is an excellent initialization procedure for other methods, e.g. primal-dual or relaxation). The difficulty is that the bidding increment γ_i is zero when more than one object offers maximum value for the bidder i . As a result, a situation may be created where several persons contest a smaller number of equally desirable objects without raising their prices, thereby creating a never ending cycle; see Fig. 1.

To break such cycles, we introduce a perturbation mechanism, motivated by real auctions where each bid for an object must raise the object's 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



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

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

ϵ and say that a partial assignment and a price vector p satisfy ϵ -complementary slackness (or ϵ -CS for short) if

$$a_{ij_i} - p_{j_i} \geq \max_{j=1,\dots,n} \{a_{ij} - p_j\} - \epsilon, \tag{8}$$

for all assigned pairs (i, j_i) . In words, to satisfy ϵ -CS, all assigned persons of the partial assignment must be assigned to objects that are within ϵ of being best.

4. THE AUCTION ALGORITHM

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

$$\gamma_i = v_i - w_i + \epsilon, \tag{9}$$

[rather than $\gamma_i = v_i - w_i$ as in Eq. (4)]. With this choice, the ϵ -CS condition is satisfied. 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.

It can be shown that this reformulated auction process terminates in a finite number of iterations, necessarily with a feasible assignment and a set of prices that satisfy ϵ -CS. To get a sense of this, note that if an object receives a bid in m iterations, 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 iterations can an object receive a bid while some other object still has not yet received any bid. On the other hand, once all objects receive at least one bid, the auction terminates. (This argument assumes that any person can bid for any object, but it can be generalized for the case where the set of feasible person-object pairs is limited, as long as at least one feasible assignment exists.) Figure 2 shows how the auction algorithm, based on the bidding increment $\gamma_i = v_i - w_i + \epsilon$ [cf. Eq. (9)], overcomes the cycling problem of the example of Fig. 1.

At Start of Iteration #	Object Prices	Assigned Pairs	Bidder	Preferred Object	Bidding Increment
1	0,0,0	(1,1), (2,2)	3	2	ϵ
2	0, ϵ ,0	(1,1), (3,2)	2	1	2ϵ
3	2ϵ , ϵ ,0	(2,3), (3,1)	1	2	2ϵ
4	2ϵ , 3ϵ ,0	(1,2), (2,1)	3	1	2ϵ
5	4ϵ , 3ϵ ,0	(1,3), (3,2)	2	2	2ϵ
6

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

When the auction algorithm terminates, we have an assignment satisfying ϵ -CS, but is this assignment optimal? 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/she might win the object at an unnecessarily high price. Consistent with this intuition, we will show that if ϵ is small, then the final assignment will be “almost optimal.” In particular, the following proposition (proved in several sources, e.g. [Ber79], [Ber88], [BeT89], [Ber91]) shows that *the total cost of the final assignment is within $n\epsilon$ of being optimal*. The idea is that a feasible assignment and a set of prices satisfying ϵ -CS also satisfy CS (and are therefore primal and dual optimal, respectively) for a *slightly perturbed* problem where all costs a_{ij} are the same as before, except for the costs of the n assigned pairs which are modified by an amount no more than ϵ .

Proposition 1: A feasible assignment satisfying ϵ -complementary slackness together with some price vector is within $n\epsilon$ of being optimal.

Suppose now that the costs 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 $n\epsilon < 1$, any complete assignment that is within $n\epsilon$ of being optimal must be optimal. It follows, that *if*

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

and the costs a_{ij} are all integer, then the assignment obtained upon termination of the auction algorithm is optimal.

Figure 3 shows the sequence of generated object prices for the example of Fig. 1 in relation to the contours of the dual cost function. It can be seen from this figure that with each bid, the dual cost is approximately minimized (within ϵ) with respect to the price of the object receiving the bid. Successive minimization of a cost function along single coordinates is a central feature of coordinate descent and relaxation methods, which are popular for unconstrained minimization of smooth functions and for solving systems of smooth equations. Thus, the auction algorithm can be interpreted as an approximate coordinate descent method and as such, it is related to the relaxation method discussed in the previous subsection.

5. COMPUTATIONAL ASPECTS – ϵ -SCALING

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

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

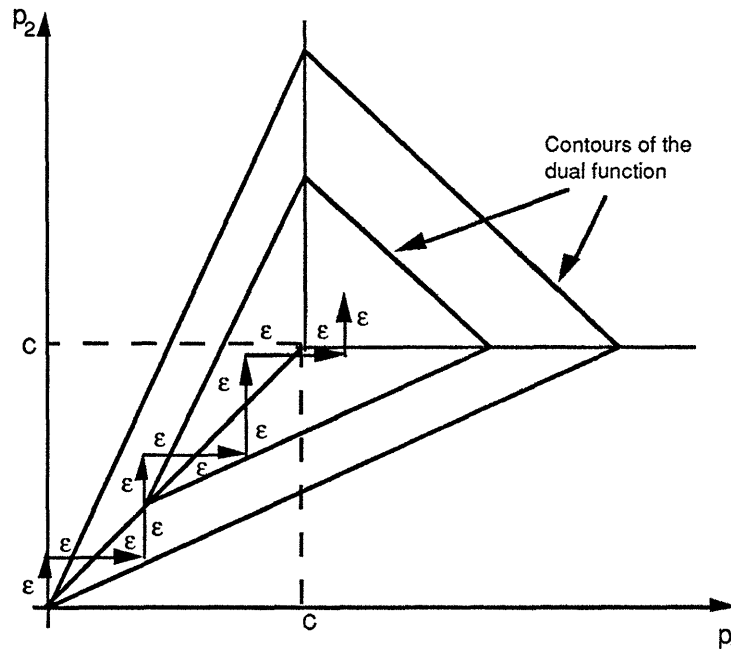


Figure 3: A sequence of prices p_1 and p_2 generated by the auction algorithm for the example of Fig. 1. The figure shows the equal dual cost surfaces in the space of p_1 and p_2 , with p_3 fixed at 0.

Basically, for many types of problems, the number of iterations up to termination tends to be proportional to C/ϵ . This can be seen from the example of Fig. 3, where the number of iterations up to termination is roughly C/ϵ , starting from zero initial prices.

Note also that there is a dependence on the initial prices; if these prices are “near optimal,” we expect that the number of iterations to solve the problem will be relatively small. This can be seen from the example of Fig. 3; if the initial prices satisfy $p_1 \approx p_3 + C$ and $p_2 \approx p_3 + C$, the number of iterations up to termination is quite small.

The preceding observations suggest the idea of ϵ -scaling, which consists of applying the algorithm several times, starting with a large value of ϵ and successively reducing ϵ up to an ultimate value that is less than some critical value (for example, $1/n$, when the benefits a_{ij} are integer). Each application of the algorithm provides good initial prices for the next application. ϵ -scaling was suggested in the original proposal of the auction algorithm [Ber79], based on extensive experimentation, which established its effectiveness for many types of assignment problems. In particular, ϵ -scaling is typically beneficial for sparse problems, that is, problems where the set of feasible assignment pairs is severely restricted. The cost structure of the problem is also important in determining whether ϵ -scaling is needed.

6. VARIATIONS AND EXTENSIONS

Extension to Asymmetric Assignment Problems

A slight variation of the auction algorithm can be used for asymmetric assignment problems where the number of objects is larger than the number of persons and there is a requirement that each person be assigned to some object. 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.

Reverse Auction

Unfortunately, the variation just described for the asymmetric assignment problem cannot be embedded in a scaling procedure because of the zero initial price restriction; to do ϵ -scaling, we must use the nonzero final prices obtained for a large value of ϵ to initialize the algorithm for a smaller value of ϵ . As a result the method performs poorly for problems where price wars (protracted sequences of small price changes of the type illustrated in Fig. 3) are likely to arise; such problems include sparse nearly symmetric assignment problems with moderately or very large cost range.

An interesting approach that circumvents this difficulty involves the idea of *reverse auction*, which was recently proposed in [BCT91]. While in the auction algorithm, persons compete for objects by bidding and raising the price of their best object, in reverse auction the objects compete for persons by essentially offering discounts (lowering their prices). One can show that forward and reverse auctions are mathematically equivalent, but their combination results in algorithms that can solve the symmetric and asymmetric assignment problems, as well as a variety of inequality-constrained assignment-like problems, more efficiently than forward auction alone.

The combination of forward and reverse auction is also far less susceptible to price wars than forward auction in solving symmetric assignment problems. 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 often the best choice.

Parallel and Asynchronous Implementation

Both the bidding and the assignment phases of the auction algorithm 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. 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. In such implementations, a person may be viewed as an autonomous decision maker who at unpredictable times obtains information about the prices of the objects. Each unassigned person bids at an arbitrary time on the basis of his/her current object price information (that may be outdated because of communication delays).

A careful formulation of the totally asynchronous model, and a proof of its validity is given in [BeC89c], which includes also extensive computational results on a shared memory machine, confirming the advantage of asynchronous over synchronous implementations. Other parallel implementations of the auction algorithm have been presented in several studies [PhZ88], [KKZ89], [WeZ90], [Zak90].

Extension to Transportation and Minimum Cost Flow Problems

The auction algorithm can be extended to solve a variety of linear network flow problems. This is not surprising since the general linear minimum cost network flow problem can be transformed to an assignment problem (see e.g. [Ber91a], p. 17). By applying the auction algorithm to this problem and by streamlining the computations (sometimes in creative ways) one can obtain auction algorithms for network flow problems such as max-flow, shortest path, transportation, min-cost flow, etc.

In particular, the auction algorithm has been extended to solve linear transportation problems [BeC89a]. The basic idea is to convert the transportation problem into an assignment problem by creating multiple copies of persons (or objects) for each source (or sink respectively), and then to modify the auction algorithm to take advantage of the presence of the multiple copies. Computational results show that this auction algorithm is considerably faster than its chief competitors for important classes of transportation problems. Generally these problems are characterized by two properties, *homogeneity* and *asymmetry*. A homogeneous problem is one for which there are only few levels

of supply and demand. An asymmetric problem is one for which the number of sources is much larger than the number of sinks. For other types of transportation problems, the auction algorithm is outperformed by state-of-the-art codes based on relaxation methods; see [BeC89a], [Ber91].

There are extensions of the auction algorithm for linear minimum cost flow (transshipment) problems. The first such extension is the ϵ -relaxation method due to [Ber86a], [Ber86b]; see [BeT89] and [Ber91a] for a detailed description. The computational complexity of the scaled version of this algorithm was first studied in [Gol87], where particularly favorable polynomial running time estimates were derived; see [BeE87], [BeE88], [GoT90] for additional results along these lines. Serial implementations of the ϵ -relaxation method for general minimum cost flow problems are not yet competitive with implementations of other methods such as relaxation; see [Ber91]. However, the ϵ -relaxation method is well suited for parallel computation. In fact the method admits a totally asynchronous implementation, as first shown in [Ber86a]; see also [BeE88], [BeT89]. A synchronous massively parallel implementation of the ϵ -relaxation method is presented in [LiZ91].

A general algorithm of the auction type for linear minimum cost flow problems was also given in [BeC89b]; it includes as special cases the auction algorithm for assignment and transportation problems, as well as the ϵ -relaxation method. By specializing this general algorithm to network flow problems with special structure, one may obtain efficient methods; for example, an interesting and intuitive algorithm for the k node-disjoint shortest path problem is developed in [BeC91].

Application to Max-Flow Problems

The ϵ -relaxation method can be applied to the max-flow problem, once this problem is converted to the minimum cost flow format, involving a feedback arc connecting the sink with the source, and having cost -1 (see [BeT89], [Ber91]). When this is done, the ϵ -relaxation method bears a close resemblance with a max-flow algorithm first proposed in [Gol85] and [GoT86]. These max-flow algorithms were derived from a different point of view that is unrelated to duality or ϵ -CS. They use node “labels,” which in the context of the ϵ -relaxation approach can be viewed as prices. The max-flow version of the ϵ -relaxation method, first given in [Ber86a], [Ber86b] is simpler than the algorithms of [Gol85] and [GoT86] in that it obtains a maximum flow in one phase rather than two. It can also be initialized with arbitrary prices, whereas in the max-flow algorithms of [Gol85], [GoT86] the initial prices must satisfy $p_i \leq p_j + 1$ for all arcs (i, j) . This can be significant, for example if one wants to use scaling or more generally, in a reoptimization setting where one wants to capitalize on the results of the solution of a slightly different max-flow problem.

In practice, the ϵ -relaxation method often finds a minimum cut very quickly – much faster than

classical methods such as the Ford-Fulkerson algorithm. However, the method finds a maximum flow much faster than the Ford-Fulkerson algorithm for fairly dense problems and much slower for very sparse problems. This is due to the price war phenomenon referred to earlier. If one is interested in just a minimum cut or just the value of the maximum flow, it is worth testing periodically to see whether a minimum cut has already been obtained. A method for doing this is described in [Ber91] (Exercise 5.4) and is used in a max-flow code given in Appendix A.6 of [Ber91].

Extension to Shortest Path Problems

A new and simple auction algorithm for finding shortest paths in a directed graph was recently developed in [Ber91a], [Ber91b]. In the single origin/single destination case, the algorithm maintains a single path starting at the origin, which is either extended or contracted by a single node at each iteration. Simultaneously, at most one price variable is adjusted at each iteration so as to either improve or maintain the value of a dual function. The algorithm may be viewed as a special case of the naive auction algorithm applied to a special type of assignment problem, which is derived from the shortest path problem. While, as discussed earlier, the naive auction algorithm may not be successful in solving general assignment problems, it is guaranteed to solve this particular type of assignment problem thanks to its special structure.

Based on experiments with randomly generated problems on a serial machine, the auction/shortest path algorithm outperforms substantially its closest competitors for problems with few origins and a single destination (the computation time is reduced often by an order of magnitude or more). Also, for the case of multiple origins, the algorithm is better suited for parallel computation than other shortest path algorithms. The thesis [Pol91] discusses parallel asynchronous implementations of the algorithm.

7. CONCLUSIONS

Much progress has been made in the last few years to extend and apply auction algorithms to a variety of linear network flow problems, and to place them on an equal footing with the classical primal and dual cost improvement methods. At present, it appears that for simple problems such as assignment, shortest path, and max-flow, auction algorithms are at least competitive and often far superior to the classical methods; they are also better suited for parallel computation. Auction algorithms are also new and not fully developed. With further research, they should become more

broadly applicable and more competitive with the classical methods.

REFERENCES

- [BCT91] Bertsekas, D. P., Castañon, D. A., and Tsaknakis, H., 1991. "Reverse Auction and the Solution of Inequality Constrained Assignment Problems," Unpublished Report.
- [BeC89a] Bertsekas, D. P., and Castañon, D. A., "The Auction Algorithm for Transportation Problems," *Annals of Operations Research*, Vol. 20, pp. 67-96.
- [BeC89b] Bertsekas, D. P., and Castañon, D. A., "The Auction Algorithm for the Minimum Cost Network Flow Problem," *Laboratory for Information and Decision Systems Report LIDS-P-1925*, M.I.T., Cambridge, MA, (November).
- [BeC89c] Bertsekas, D. P., and Castañon, D. A., "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," *Alphatech Report*, Burlington, MA, (November), to appear in *Parallel Computing*.
- [BeC90] Bertsekas, D. P., and Castañon, D. A., "Parallel Asynchronous Hungarian Methods for the Assignment Problem," *Alphatech Report*, Burlington, MA, Jan. 1990.
- [BeC91] Bertsekas, D. P., and Castañon, D. A., "A Generic Auction Algorithm for the Minimum Cost Network Flow Problem," *Alphatech Report*, Burlington, MA, Sept. 1991.
- [BeE87] Bertsekas, D. P., and Eckstein, J., "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems," *Proc. of IFAC '87*, Munich, Germany, July 1987.
- [BeE88] Bertsekas, D. P., and Eckstein, J., "Dual Coordinate Step Methods for Linear Network Flow Problems," *Math. Progr., Series B*, Vol. 42, 1988, pp. 203-243.
- [BeM73] Bertsekas, D. P., and Mitter, S. K., "Descent Numerical Methods for Optimization Problems with Nondifferentiable Cost Functions," *SIAM Journal on Control*, Vol. 11, pp. 637-652.
- [BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, N. J., 1989.
- [Ber79] Bertsekas, D. P., "A Distributed Algorithm for the Assignment Problem," *Lab. for Information and Decision Systems Working Paper*, M.I.T., (March).
- [Ber85] Bertsekas, D. P., "A Distributed Asynchronous Relaxation Algorithm for the Assignment

- Problem,” Proc. 24th IEEE Conf. Dec. & Contr., pp. 1703-1704.
- [Ber86a] Bertsekas, D. P., “Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems,” LIDS Report P-1606, M.I.T., (November).
- [Ber86b] Bertsekas, D. P., “Distributed Relaxation Methods for Linear Network Flow Problems,” Proceedings of 25th IEEE Conference on Decision and Control, pp. 2101-2106.
- [Ber88] Bertsekas, D. P., “The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem,” Annals of Operations Research, Vol. 14, pp. 105-123.
- [Ber91a] Bertsekas, D. P., Linear Network Optimization: Algorithms and Codes, M.I.T. Press, Cambridge, Mass.
- [Ber91b] Bertsekas, D. P., “An Auction Algorithm for Shortest Paths,” SIAM J. on Optimization, Vol. 1, (October).
- [Gol85] Goldberg, A. V., “A New Max-Flow Algorithm,” Tech. Mem. MIT/LCS/TM-291, Laboratory for Computer Science, M.I.T., Cambridge, MA.
- [GoT86] Goldberg, A. V., and Tarjan, R. E., “A New Approach to the Maximum Flow Problem,” Proc. 18th ACM STOC, pp. 136-146.
- [GoT90] Goldberg, A. V., and Tarjan, R. E., “Solving Minimum Cost Flow Problems by Successive Approximation,” Math. of Operations Research, Vol. 15, pp. 430-466.
- [Gol87] Goldberg, A. V., “Efficient Graph Algorithms for Sequential and Parallel Computers,” Tech. Report TR-374, Laboratory for Computer Science, M.I.T., Cambridge, MA.
- [KKZ89] Kempa, D., Kennington, J., and Zaki, H., “Performance Characteristics of the Jacobi and Gauss-Seidel Versions of the Auction Algorithm on the Alliant FX/8,” Report OR-89-008, Dept. of Mech. and Ind. Eng., Univ. of Illinois, Champaign-Urbana, 1989.
- [LiZ91] Li, X., and Zenios, S. A., “Data Parallel Solutions of Min-Cost Network Flow Problems Using ϵ -Relaxations,” Report 1991-05-20, Dept. of Decision Sciences, The Wharton School, Univ. of Pennsylvania, Phil., Penn.
- [PhZ88] Phillips, C., and Zenios, S. A., “Experiences with Large Scale Network Optimization on the Connection Machine,” Report 88-11-05, Dept. of Decision Sciences, The Wharton School, Univ. of Pennsylvania, Phil., Penn., Nov. 1988.
- [Pol91] Polymenakos, L., “Analysis of Parallel Asynchronous Schemes for the Auction Shortest Path

Algorithm,” MS Thesis, EECS Dept., M.I.T., Cambridge, MA.

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

[WeZ90] Wein, J., and Zenios, S. A., “Massively Parallel Auction Algorithms for the Assignment Problem,” *Proc. of 3rd Symposium on the Frontiers of Massively Parallel Computation*, Md.

[Zak90] Zaki, H., “A Comparison of Two Algorithms for the Assignment Problem,” Report ORL 90-002, Dept. of Mechanical and Industrial Engineering, Univ. of Illinois, Urbana, Ill.