Noname manuscript No.
(will be inserted by the editor)

General auction method for real-valued optimal transport

J.D. Walsh · Luca Dieci

the date of receipt and acceptance should be inserted later

Abstract The auction method developed by Bertsekas in the late 1970s is a relaxation technique for solving integer-valued assignment problems. It resembles a competitive bidding process, where unsatisfied persons (bidders) attempt to claim the objects (lots) offering the best value. By transforming integer-valued transport problems into assignment problems, the auction method can be extended to compute optimal transport solutions. We propose a more general auction method that can be applied directly to real-valued transport problems. We prove termination and provide *a priori* error bounds for the general auction method. Our numerical results indicate that the complexity of the general auction is roughly comparable to that of the original auction method, when the latter is applicable.

Keywords auction, network programming, optimization, transportation

1 Introduction

The auction method was first proposed by Dimitri Bertsekas in the late 1970s [1]. He developed the method for the assignment problem, as an alternative to the Hungarian method [5]. In 1989 Bertsekas and Castañón extended the original auction method to solve minimal cost flow and optimal transport problems by taking into account

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE–1148903.

J.D. Walsh
School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0160 USA
Tel.: +1-404-894-6441, Fax: +1-404-894-4409
E-mail: jdwalsh03@gatech.edu

Luca Dieci
School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0160 USA
Tel.: +1-404-894-9209, Fax: +1-404-894-4409
E-mail: dieci@math.gatech.edu

what they call "similar persons and objects" [4]. Their extended auction method decomposes the transport problem into an equivalent assignment problem by splitting each supply and demand vertex into multiple identical vertices of unitary weight. The number of copies is equal to the supply or demand "weight." The extended auction method then solves the assignment problem and combines the resulting assignments to provide the desired transport solution. These two auction methods offer worst-case error and complexity bounds, as well as conditions under which an optimal solution is guaranteed.

However, these auction methods put restrictions on the data inherent to the problems they can solve. The complexity argument for the assignment auction method requires that the cost of each matching is an integer. If any rational costs are required, all costs must be transformed into integers using a common denominator. This can impact the worst-case time complexity of the algorithm. If irrational costs are required, as can occur when (for instance) cost is given by a p-norm with $p \neq 1$, the method offers no guarantee of termination or convergence.

When the extended auction method is used to solve minimal cost flow or optimal transport, then another, more significant, data restriction arises. The extended auction requires all source and sink weights to be integers. As with the costs, if any rational weights are required, the entire set of weights must be transformed into integers using a common denominator. Here, though, the common denominator directly impacts both the storage requirements and the worst-case time complexity of the resulting assignment problem. If any irrational weights are required, the extended auction method cannot be applied at all.

We propose a more general auction method, one developed specifically for the transport problem (rather than the assignment problem) and capable of handling real-valued costs and weights. To support this extension, we also prove convergence and provide *a priori* error bounds for transport problems with real-valued data. Finally, we compare auction methods using a series of standard problems and show results indicating complexity roughly comparable to that of the original auction method, in cases where the latter can be applied.

2 Background

Our general auction method is best understood by contrasting it with the assignment auction method and its extensions. In order to highlight these contrasts, we use a common framework: the real-valued optimal transport problem. This common frame allows us to unify notation and interrelate key concepts as we briefly describe the auction and extended auction methods. When algorithmic restrictions limit us to a special-case of the transport problem, we describe the required conditions.

2.1 Transport problem

Consider the transport problem $\mathcal{T}$, which we define as follows:

Suppose we are given a *demand* vector $\{d_i\}_{i=1}^{M}$ and a *supply* vector $\{s_j\}_{j=1}^{N}$, whose demand coefficients d_i and supply coefficients s_j are positive scalars such that

$$L := \sum_{i=1}^{M} d_i = \sum_{j=1}^{N} s_j > 0. \tag{2.1}$$

We refer to L as the *total weight* of the transport problem. In the underlying transport graph, the vertex i associated with demand coefficient d_i is a *sink*, and the vertex j associated with supply coefficient s_j is a *source*. We denote the set of sinks by I, and the set of sources by J.

Furthermore, suppose for each sink i we are given the nonempty set $A(i)$ of sources to which the sink i is adjacent. The set of all possible transport pairs is equal to

$$\mathcal{A} := \{\, (i, j) \mid j \in A(i), i \in \{1, \ldots, M\} \,\}. \tag{2.2}$$

Thus, $\mathcal{A}$ is the set of arcs of the underlying transport graph, a bipartite graph with $M + N$ vertices and $|\mathcal{A}| \leq MN$ arcs.

For each $(i, j) \in \mathcal{A}$, let $c_{ij} < 0$ be the given *cost coefficient* (or simply *cost*). Our goal is to

$$\text{maximize} \quad \sum_{(i,j)\in\mathcal{A}} c_{ij} f_{ij} \tag{2.3a}$$

$$\text{subject to} \quad \sum_{\{\, j \mid j\in A(i) \,\}} f_{ij} = d_i \qquad \forall\, i \in \{1, \ldots, M\}, \tag{2.3b}$$

$$\sum_{\{\, i \mid j\in A(i) \,\}} f_{ij} = s_j \qquad \forall\, j \in \{1, \ldots, N\}, \text{ and} \tag{2.3c}$$

$$0 \leq f_{ij} \leq \min\{d_i, s_j\} \qquad \forall\, (i, j) \in \mathcal{A}. \tag{2.3d}$$

We refer to f_{ij} as the *flow* along (i, j), because it gives the amount transported (i.e. "flowing") from source j to sink i.

We have assumed negative costs and formulated the transport problem as a maximization problem, in accordance with the standard implementation of the auction method. Because $c_{ij} < 0$, the maximization equation Equation (2.3a) provides a minimum overall cost (obtained by reversing the sign on each c_{ij}). We refer to

$$\sum_{(i,j)\in\mathcal{A}} c_{ij} f_{ij} \tag{2.4}$$

as the *primal cost*. The solution to Equation (2.3a) is called the *optimal primal cost*, or *optimal cost*, of the transport problem, and is denoted by P^*.

2.2 Transport plan

A *transport plan* (or *transport map*) T is a multiset of triples $(i, j; q_{ij})$ such that $(i, j) \in \mathcal{A}$ and the transported *quantity*, q_{ij}, is non-negative. Note that T may be

empty. While the elements of T are not necessarily unique, for each $(i, j) \in \mathcal{A}$ we can compute the unique flow f_{ij} given by T as

$$f_{ij} = \sum_{\{\,(k,l;q_{kl})\in T \;|\; (k,l)=(i,j)\,\}} q_{kl}. \tag{2.5}$$

In order to apply the plan to our transport problem, we require that T satisfies $f_{ij} \leq \min\{d_i, s_j\}$ for all $(i, j) \in \mathcal{A}$. By a minor abuse of notation, we may say $(i, j) \in T$ to signify that $(i, j; q_{ij}) \in T$ for some $q_{ij} > 0$. We may also say $T \in \mathcal{T}$ to refer to some transport plan T associated with the transport problem $\mathcal{T}$.

Given any transport plan T, we say that sink i is *satisfied* if

$$\sum_{\{\,q_{ij} \;|\; (i,j,q_{ij})\in T\,\}} q_{ij} = d_i. \tag{2.6}$$

Otherwise, we say that i is *unsatisfied*. (Alternatively, we may say i has unsatisfied demand D_i, where $0 \leq D_i \leq d_i$.)

Similarly, when

$$\sum_{\{\,q_{ij} \;|\; (i,j,q_{ij})\in T\,\}} q_{ij} = s_j, \tag{2.7}$$

we say the source j is *unavailable*. Otherwise, we say that j is *available*, or that j has available supply S_j, where $0 \leq S_j \leq s_j$.

A transport plan is said to be *feasible* or *complete* when all sinks are satisfied; otherwise the plan is called *partial*.

If T is such that the pair (i, j) appears at most once, and $(i, j; q_{ij}) \in T$ implies $q_{ij} > 0$, we refer to T as a *simplified* transport plan. In this case $q_{ij} = f_{ij}$, and we may refer to flow and quantity interchangeably. Simplified transport plans, while not strictly necessary, greatly improve clarity of notation. We will generally assume T is simplified when stating definitions and proofs.

2.3 Dual problem

We can write the dual transport problem as

$$\min_{u_i,\, p_j} \left\{ \sum_{i=1}^{M} d_i u_i + \sum_{j=1}^{N} s_j p_j \right\}, \tag{2.8}$$

with the restriction that $u_i + p_j \leq c_{ij}$ for all $(i, j) \in \mathcal{A}$. We call the dual variable p_j a *price* of j, and the vector $p = \{p_j\}_{j=1}^{N}$ a price vector of $\mathcal{T}$. Assume $p_j \geq 0$ for all j.

Given some price vector p, the *expense* associated with the arc $(i, j) \in \mathcal{A}$ is

$$x_{ij} := c_{ij} - p_j \tag{2.9}$$

and the *expense* for the sink i is

$$x_i := \max_{j\in A(i)} x_{ij} = \max_{j\in A(i)} \{c_{ij} - p_j\}. \tag{2.10}$$

Because $c_{ij} < 0$, the maximization of the expense x_i actually generates the least overall expense (obtained by reversing the signs of the x_is).

Suppose we have a simplified complete transport plan T and a price vector p. From linear programming theory, we know (T, p) is simultaneously primal and dual optimal if and only if

$$u_i = \max_{k \in A(i)} \{c_{ik} - p_k\} = c_{ij} - p_j \qquad \forall\, (i, j;\, f_{ij}) \in T. \tag{2.11}$$

In other words, the expense for each sink is minimized by transport to the least expensive source(s). This is known as the *complementary slackness condition*, or *complementary slackness*.

Among other things, complementary slackness implies that the dual problem can only be minimized when $u_i = x_i$ for all i. Thus, we can view the prices p_j as the only variables in our dual problem, which we can rewrite as

$$\min_{p=\{p_j\}_{j=1}^N} \left\{ \sum_{i=1}^{M} d_i \max_{j \in A(i)} \{c_{ij} - p_j\} + \sum_{j=1}^{N} s_j p_j \right\}. \tag{2.12}$$

Because $c_{ij} < 0$, the minimization equation Equation (2.12) provides a maximum overall profit (obtained by reversing the sign of each c_{ij}). We refer to

$$\sum_{i=1}^{M} d_i \max_{j \in A(i)} \{c_{ij} - p_j\} + \sum_{j=1}^{N} s_j p_j \tag{2.13}$$

as the *dual profit*. The solution to Equation (2.12) is called the *optimal dual profit*, or *optimal profit*, of the transport problem, and is denoted by D^*.

2.4 Auction fundamentals

The auction method solves the dual problem described above, using a relaxation technique inspired by the open ascending price process commonly used for real-world auctions. Each sink i is a *bidder* in the auction, seeking to satisfy its demand d_i. Each source is a *lot* containing the supply s_j. Each unsatisfied bidder i offers a *bid amount*, b_{ij}, for some lot j. Naturally, the bid b_{ij} must exceed the current price, given by p_j. The bidders each want to minimize their *loss*: the quantity

$$c_{ij} - b_{ij}, \tag{2.14}$$

representing the cost-price total for bidder i to obtain lot j. The best possible loss for bidder i is the amount closest to zero, as given by the negative scalar

$$\max_{j \in A(i)} \{c_{ij} - b_{ij}\}. \tag{2.15}$$

Each bid can be "outbid"; that is, superseded by another bidder offering a higher price. So long as b_{ij} is greater than the highest previous price for lot j, designated p_j, we know that some bidder (either i or a competitor) will claim lot j. Once prices are

sufficiently high, lot costs becomes irrelevant, so every lot eventually receives a bid. At that point, the auction ends.

Knowing that we require $b_{ij} > p_j$, the natural question to ask is: how much larger than p_j should we make b_{ij}? As Bertsekas so eloquently explains in [3, p. 29–30], we need to set a minimum *bidding increment*, or *step size*: some $\varepsilon > 0$, such that $b_{ij} \geq p_j + \varepsilon$. Otherwise, the auction risks stalling if two options are equally optimal.

The assignment problem, as formulated for the original auction method, assumes that we have N sources and N sinks, each of which has weight 1, and that all costs are integer-valued. Thus, we can assume that the auction method for the assignment problem operates on a special case of our transport problem: one with integer costs and unit weights, where $M = N$. Applying the terminology used by Bertsekas, we call a sink of weight 1 a *person* and a source of weight 1 an *object*.

2.5 ε-complementary slackness

We can relax the complementary slackness condition, allowing flow to sinks from sources whenever the loss comes within ε of attaining the maximum. This is called *ε-complementary slackness*, or *ε-CS*, and it can be considered for any transport plan, complete or not.

Formally: Given some $\varepsilon > 0$, a simplified transport plan T and price vector p satisfy ε-complementary slackness if

$$x_i - \varepsilon = \max_{k \in A(i)} \{c_{ik} - p_k\} - \varepsilon \leq c_{ij} - p_j \qquad \forall\, (i,\, j;\, f_{ij}) \in T. \tag{2.16}$$

As Bertsekas proved in [3, p. 255–257], the auction method maintains ε-CS, and the resulting transport plan is guaranteed to be optimal if $\varepsilon < 1/N$.

2.5.1 ε-scaling

Bertsekas found that the number of iterations of the auction algorithm depends on ε and the number of possible values, or *range*, that the cost can take. When we restrict costs to the negative (or positive) integers, as Bertsekas does, we can safely assume that the cost range C is equal to the maximum absolute cost,

$$C = \max_{(i,\, j) \in \mathcal{A}} |c_{ij}| = \max_{(i,\, j) \in \mathcal{A}} \{-c_{ij}\}. \tag{2.17}$$

(It is possible for the cost range to be smaller; for example, when $\gcd\{c_{ij}\} > 1$.) As Bertsekas concluded, for many assignment problems the number of iterations required for termination is proportional to C/ε; see [3, p. 34].

Of course, the number of iterations is also dependent on the initial price vector; when the initial prices are close to "ε-optimal," the number of iterations required is relatively small. This suggests that it may be advantageous to use a scaling technique, similar to that used in penalty and barrier methods. For the auction algorithm, Bertsekas calls this technique *ε-scaling*.

To perform ε-scaling, we apply the auction algorithm multiple times. Each iteration of the algorithm is called a *scaling phase*. In the first iteration, we use a simple

set of initial prices, along with an initial ε. For each successive iteration, we use the resulting price vector from the previous phase, along with an altered ε-value. The ε-scaling technique terminates when ε reaches some critical value. When ε-scaling is used, the auction method has worst-case time complexity $O(NA \log(NC))$, where $A = |\mathcal{A}|$, the number of arcs in the underlying transport graph; see [3, p. 265].

2.6 Extended auction

The extended auction for the integer-valued transport problem was initially described by Bertsekas and Castañón in 1989; [4]. They wanted to extend the assignment auction method to one that could handle transport problems. Their idea was to transform the integer-valued transport problem into an assignment problem by creating multiple copies: construct an assignment problem with a number of identical persons (or objects) equal to the weight at each sink (or source, respectively), preserving the adjacencies and costs of the original vertices. After solving the assignment problem, redundant arcs with positive flow could be combined to generate a simplified optimal transport plan.

When a sink or source is replaced with multiple weight-1 copies, the resulting persons and objects retain an identical underlying structure: the adjacencies and cost coefficients of the originals. Such persons and objects are called *similar*. A *similarity class* is the equivalence class of persons or objects with the same adjacencies and costs. Persons and objects in the same similarity class can engage in protracted "bidding wars" unless their similarity is taken into account.

Bertsekas and Castañón address the similarity relationships in multiple ways. Thus, their extended auction method is best understood as three distinct algorithms:

- The *auction algorithm for the assignment problem*, or simply **AUCTION**, expands the sinks and sources of the transport problem, creating an assignment problem that it solves without considering the underlying structure.
- The *auction algorithm for similar objects*, or **AUCTION–SO**, considers the impact of similar objects when increasing prices.
- The *auction algorithm for similar objects and persons*, or **AUCTION–SOP**, treats similar persons as a unit during each bidding phase. At every iteration, each unsatisfied similarity class of persons bids collectively for a number of objects equal to its total demand d_i. Each similarity class of persons shares a single price increase, determined similarly to the technique of the AUCTION–SO.

For all three algorithms, the costs and weights must be integers. If $\varepsilon < 1/\min\{M, N\}$, the solution resulting from any of these is guaranteed to be optimal; [4, p. 85].

3 General auction for the transport problem

Our general auction method uses the real-valued transport problem as its basis, rather than the integer-valued assignment problem. (Thus, the method's name: it is designed around a more general problem than other auction methods.) To avoid redundancy,

we define and explain only those terms and ideas which differ from other auction methods.

3.1 Description and terminology

The general auction uses a variant form of lot bidding, similar to "times the money" bidding. Bidder i has unsatisfied demand D_i, so bidder i makes a bid of bid amount b_{ij} on lot j. The quantity desired from lot j is set to $q_{ij} = \min\{D_i, s_j\}$. We can write the bid as the pair (b_{ij}, q_{ij}). The actual bid is understood to be price b_{ij} per q_{ij} items, for a total bid value of $b_{ij}q_{ij}$.

Suppose that lot j has available supply S_j. If $q_{ij} \leq S_j$, the desired quantity is immediately available, so bidder i is awarded a *claim* on lot j of quantity q_{ij} at bid price b_{ij}. This claim, represented by the triple $(i; b_{ij}, q_{ij})$, is added to the *claim list* for lot j, which we denote by C_j.

Such claims can still be outbid. If $q_{ij} > S_j$, we compare bidder i's offer to those already on the claim list. The difference between q_{ij} and S_j is made up by taking the required amount from the lowest priced claim(s) with bid price less than b_{ij}. Only if insufficient low-priced claims exist will bidder i claim less than q_{ij}.

Even so, as long as we ensure that b_{ij} is greater than the lowest bid price on lot j's current claim list, we know that bidder i will be able to claim some quantity in lot j. To guarantee the occurrence of such a claim, for each lot j we must first determine a *lot price* p_j, defined as

$$p_j := \min_{(i;\, b_{ij},\, q_{ij}) \in C_j} \{b_{ij}\}. \tag{3.1}$$

(If C_j is empty, let p_j be equal to some initial price p_j^0.) When bidding, we require that bid prices satisfy $b_{ij} \geq p_j + \varepsilon$ for some fixed $\varepsilon > 0$.

The lot price vector $p = \{p_j\}_{j=1}^N$ corresponds to the price vector used in the dual profit equation. Thus, like the assignment auction, the general auction attempts to solve the dual problem. The general auction also uses ε-complementary slackness, which is defined identically to the assignment auction.

3.2 Iteration

Assume without loss of generality that $M \geq N$; that is, there are at least as many sinks as sources. To initialize the general auction method, one must have a bidding step size $\varepsilon > 0$ and an initial lot price vector. Once initialized, the auction is performed in iterations. Each iteration consists of two phases: a bidding phase (Section 3.2.1) and a claims phase (Section 3.2.2).

3.2.1 Bidding phase of the general auction

Let $\tilde{I}$ be a nonempty subset of sinks i that are unsatisfied under the current transport plan T. For each sink $i \in \tilde{I}$:

1. Find:

(a) The lot j_i offering best expense, given by

$$j_i := \arg\max_{j \in A(i)} \{c_{ij} - p_j\}. \tag{3.2}$$

(b) The second-best expense, chosen by considering lots other than j_i,

$$w_i := \max_{j \in A(i),\, j \neq j_i} \{c_{ij} - p_j\}. \tag{3.3}$$

If j_i is the only source in $A(i)$, define w_i to be $-\infty$. (For computational purposes, this can be any value satisfying $w_i \ll c_{ij_i} - p_{j_i}$.)

2. Compute the bid (b_{ij_i}, q_{ij_i}), where the bid price b_{ij_i} is given by

$$b_{ij_i} := c_{ij_i} - w_i + \varepsilon \tag{3.4}$$

and the quantity claimed q_{ij_i} is equal to

$$q_{ij_i} := \min\{D_i, s_{j_i}\}. \tag{3.5}$$

3.2.2 Claims phase of the general auction

For each source j, let I_j be the set of sinks from which j received a bid in the bidding phase of the iteration. If I_j is nonempty, for each $i_j \in I_j$ with bid $(b_{i_j j}, q_{i_j j})$:

1. While $S_j < q_{i_j j}$ and $p_j \leq b_{i_j j}$:
   (a) Find the lowest-priced claim $c = (k;\, b_{kj}, q_{kj})$ given by

$$c := \arg\min_{(i;\, b_{ij},\, q_{ij}) \in C_j} \{p_{ij}\}. \tag{3.6}$$

   (b) If $k = i_j$, add q_{kj} to $q_{i_j j}$. [HC rule; see Section 3.4]
   (c) Find the quantity in c to be claimed by bidder i_j,

$$q := \min\{q_{i_j j}, q_{kj}\}. \tag{3.7}$$

   (d) Make the quantity q available:
      i. Add q to S_j.
      ii. Subtract q from q_{kj}.
      iii. If $q_{kj} = 0$, remove c from C_j and update p_j.
      iv. Add q to D_k.
2. Let $q_{i_j j} = \min\{q_{i_j j}, S_j\}$, and if $q_{i_j j} > 0$:
   (a) Insert $(i_j;\, b_{i_j j}, q_{i_j j})$ into C_j.
   (b) Subtract $q_{i_j j}$ from D_{i_j}.
   (c) Update p_j.

3.3 Resulting transport plan and cost

If all sinks have been satisfied, the general auction terminates. The resulting complete transport plan is equal to

$$T = \bigcup_{j=1}^{n} \{ (i, j; q_{ij}) \mid (i; b_{ij}, q_{ij}) \in C_j \} \tag{3.8}$$

and the primal cost of T is equal to

$$\sum_{j=1}^{n} \sum_{(i; b_{ij}, q_{ij}) \in C_j} c_{ij} q_{ij}. \tag{3.9}$$

If we want to represent T as a simplified transport plan, there is still one more step to perform. Using the claim lists, we can determine the simplified flow for each (i, j) as

$$f_{ij} := \sum_{(i; b_{ij}, q_{ij}) \in C_j} q_{ij}. \tag{3.10}$$

(If C_j does not contain a claim by bidder i, we assume $f_{ij} = 0$.) Using these flow values, the simplified complete transport plan $\widetilde{T}$ equals

$$\widetilde{T} := \{ (i, j; f_{ij}) \mid (i, j) \in \mathcal{A}, f_{ij} > 0 \}. \tag{3.11}$$

The primal cost of the simplified transport plan equals

$$\sum_{(i, j) \in \mathcal{A}} c_{ij} f_{ij}, \tag{3.12}$$

which is exactly the form used in the transport problem.

3.4 Hungry cannibals

The Hungry Cannibal rule is a modification to the process of handling claim lists, motivated by a potential slow-down during the iterative process. In the bidding phase, each sink attempts to maximize the quantity it acquires. However, it is possible for a sink i to bid on a lot j where it already has a claim, and for the new claim to supersede some or all of the old one. When this happens, the old claim is "cannibalized" by the new one, and the quantity acquired by i is not maximal. (It may even be zero.) We call any claim that supersedes an earlier claim by the same bidder a *cannibal* claim.

It is possible to speed up the bidding process, guaranteeing the maximum possible quantity is acquired by each bid, even when confronted by cannibal claims. We can deal with cannibals by implementing the *Hungry Cannibal* (or *HC*) rule: if a new claim by i cannibalizes a quantity q, then during the claims phase q is added to the quantity desired by the new claim. Thus, i can cannibalize itself during the claims phase, and the "hunger" of i remains maximal. As a side effect, the HC rule can unify adjacent bids by the same bidder.

It is worth pointing out that the HC rule is an optimization of the general auction. The rule is not necessary to guarantee termination or bound error, but it can significantly speed computation. The speedup occurs because the HC rule requires little overhead to implement, while reducing the number of iterations required for convergence. Numerical tests suggest that the transport plan resulting from the general auction will be nearly identical, whether or not the HC rule is used, but that time improves measurably when the rule is in place. The HC rule is labeled as Step 1-(b) of the claims phase of the general auction.

For the proofs given below, we assume that the HC rule is in place, ensuring that the quantity acquired by every new claim is maximal. This assumption helps simplify our arguments.

4 Mathematical Results

The proofs given in [4,3] for the assignment auction and its extensions rely heavily on the integral nature of the data. Our generalization to transport problems with real-valued data requires a different approach.

4.1 Termination of the general auction

Assuming real-valued data invalidates many of the standard assumptions for auction algorithms. For example, unlike Bertsekas and Castañón in [4], we cannot assume that any quantities claimed are bounded away from zero. Thus, we must consider the possibility that claimed quantities tend to zero as the number of bids goes to infinity. We also cannot assume that lot price increases are bounded away from zero, so we must consider the possibility that price increases tend to zero as the number of bids goes to infinity. Finally, given the definition of lot prices as a minimum, we cannot assume that lot prices change at all. We must consider the possibility that lot prices remain fixed over infinitely many bids. However, our key result, Theorem 3, conclusively shows that none of these possibilities can occur; the general auction behaves well when applied to real-valued data, and terminates after a finite number of iterations. Because of the above considerations, our approach to proving termination bears little resemblance to that used by Bertsekas and Castañón in [4].

4.1.1 General auction prices are nondecreasing

Theorem 1 *When applying the general auction method with step size $\varepsilon > 0$, lot prices are nondecreasing and each bid price exceeds the current lot price by at least ε.*

Proof Assume $\varepsilon > 0$ is given. Fix the lot j^* and let its lot price before and lot price after iteration be given by p_{j^*} and p'_{j^*}, respectively. If no sink bids on j^*, then the lot price of j^* does not change, and we know that $p'_{j^*} = p_{j^*}$. Suppose instead that some

sink i bids on j^*, with bid price b_{ij^*}. From Equations (2.10) and (3.2), we know that the expense of sink i's bid is given by

$$x_i = \max_{j \in A(i)} \{c_{ij} - p_j\} = c_{ij^*} - p_{j^*}. \tag{4.1}$$

Thus, the bid price b_{ij^*} is given by

$$b_{ij^*} = c_{ij^*} - w_i + \varepsilon = p_{j^*} + x_i - w_i + \varepsilon. \tag{4.2}$$

By Equation (3.3), $x_i - w_i \geq 0$, so

$$b_{ij^*} = p_{j^*} + x_i - w_i + \varepsilon \geq p_{j^*} + \varepsilon > p_{j^*}. \tag{4.3}$$

Since the new bid b_{ij^*} is at least as high as the current lot price p_{j^*}, by Equation (3.1),

$$p'_{j^*} \geq \min\{p_{j^*}, b_{ij^*}\} \geq p_{j^*}. \tag{4.4}$$

Since this is true for all i that bid on j^*, we know $p'_{j^*} \geq p_{j^*}$. Therefore, lot prices are nondecreasing.

4.1.2 Steady price implies satisfaction

Theorem 2 *If a bid by sink i on lot j does not increase the lot price p_j, then i becomes satisfied. Furthermore, if bidder k was satisfied by lot j after the last increase of p_j, then k remains satisfied after i's bid is resolved. If i should become unsatisfied and p_j has not increased, then i will bid on a lot of price p_j again.*

Proof Fix lot j^*. Let p_{j^*} be the price of j^* prior to the bid by i, and assume the second-highest bid price on the claim list C_{j^*} is

$$\hat{p}_{j^*} > p_{j^*}. \tag{4.5}$$

If $|C_j| < 2$, it is sufficient to assume $\hat{p}_{j^*} = +\infty$.

Suppose p'_{j^*} is the lot price of j^* at some later time, and that $p'_{j^*} = p_{j^*}$.

Assume first that there are no satisfied sinks on the claim list of j^*, and let i_1 be the sink currently bidding $(b_{i_1 j^*}, q_{i_1 j^*})$ on j^*. Let q_1 be equal to

$$q_1 = \sum_{\{(k; b_{kj^*}, q_{kj^*}) \mid b_{kj^*} = p_{j^*}\}} q_{kj^*}. \tag{4.6}$$

Because of the HC rule, we can assume without loss of generality that $k \neq i_1$ for all claims of price p_{j^*}.

As shown in Theorem 1, we have $b_{i_1 j^*} \geq p_{j^*} + \varepsilon$. Thus, if $q_1 \leq q_{i_1 j^*}$, all claims of price p_{j^*} have been overbid and the price after i_1's claim satisfies

$$p'_{j^*} \geq \min\{b_{i_1 j^*}, \hat{p}_{j^*}\} > p_{j^*}. \tag{4.7}$$

This contradicts our supposition that $p'_{j^*} = p_{j^*}$, and so we must have $q_1 > q_{i_1 j^*}$. By definition, $q_{i_1 j^*} = \min\{D_{i_1}, s_{j^*}\}$. If $q_{i_1 j^*} = s_{j^*}$, then we have $q_1 > s_{j^*}$, which contradicts the definition of q_1 as a quantity claimed on C_{j^*}. Hence, we know $q_{i_1 j^*} =$

D_{i_1}. Because i_1 claimed the quantity D_{i_1} and the HC rule is in place, it must be the case that sink i_1 is satisfied and

$$\hat{p}'_{j^*} = \min\{b_{i_1 j^*}, \hat{p}_{j^*}\} > p_{j^*}. \tag{4.8}$$

Next, suppose instead that j^* has one satisfied sink, and that the sink was satisfied by bidding on j^* without increasing the price p_{j^*}. Without loss of generality, assume the sink is i_1. Suppose some sink i_2 bids $(b_{i_2 j^*}, q_{i_2 j^*})$ on lot j^*. Because i_1 is satisfied, we know $i_2 \neq i_1$. The remaining quantity available at price p_{j^*} equals

$$q_2 = \sum_{\{(k; b_{kj^*}, q_{kj^*}) \mid b_{kj^*} = p_{j^*}\}} q_{kj^*}. \tag{4.9}$$

Because of the HC rule, we can assume $k \neq i_2$ for all claims of price p_{j^*}. (From the steps above, we also know $k \neq i_1$ for all such claims.)

Once again, we have $b_{i_2 j^*} \geq p_{j^*} + \varepsilon$. Note that the bid price associated with the quantity claimed by i_1 exceeds p_{j^*}, so i_2 cannot claim any quantity from i_1 unless $q_2 \leq q_{i_2 j^*}$. However, in that case all claims of price p_{j^*} have been overbid and the price after i_2's claim satisfies

$$p'_{j^*} \geq \min\{b_{i_2 j^*}, \hat{p}_{j^*}\} > p_{j^*}. \tag{4.10}$$

This contradicts our supposition that $p'_{j^*} = p_{j^*}$, and so we must have $q_2 > q_{i_2 j^*}$, and i_1 remains satisfied.

By definition, $q_{i_2 j^*} = \min\{D_{i_2}, s_{j^*}\}$. If $q_{i_2 j^*} = s_{j^*}$, once again we have a contradiction because the total quantity in C_{j^*} exceeds s_{j^*}. Hence, we know $q_{i_2 j^*} = D_{i_2}$. Because i_2 claimed the quantity D_{i_2} and the HC rule is in place, it must be the case that sink i_2 is also satisfied and

$$\hat{p}'_{j^*} > p_{j^*}. \tag{4.11}$$

Continuing inductively, we find that up to $M - 1$ distinct bidders can be satisfied while maintaining $p'_{j^*} = p_{j^*}$, and that

$$\hat{p}'_{j^*} > p_{j^*} \tag{4.12}$$

for each of them. Because the quantity with price p_{j^*} must be owned by at least one bidder, it is not possible for more than $M - 1$ distinct bidders to satisfy our initial assumption that $p'_{j^*} = p_{j^*}$.

Suppose now that i_k becomes unsatisfied for some $k = 1, \ldots, M - 1$, and $p'_{j^*} = p_{j^*}$. Because expenses are nonincreasing and $p'_{j^*} = p_{j^*}$, $x_{i_k j^*}$ must still equal the best expense for bidder i_k. Therefore, during its next bidding phase, i_k will bid on a lot with expense equal to $x_{i_k j^*}$.

4.1.3 General auction terminates

Theorem 3 *Given $\varepsilon > 0$, if at least one feasible transport plan exists, then the general auction method terminates after finitely many iterations.*

Proof Let I be the set of bidders (sinks) and J be the set of lots (sources) for a feasible transport problem $\mathcal{T}$. As a consequence of Theorem 1, we can partition J into four subsets:

- J^F, the set of lots which receive finitely many bids.
- J^M, the set of lots whose prices achieve maximum values, while receiving infinitely many bids.
- J^A, the set of lots whose prices asymptotically approach finite limits, without ever achieving those limits.
- J^∞, the set of lots whose prices increase without bound.

Now consider the sets

$$I^F = \{\, i \in I \mid i \text{ bids finitely many times } \}, \tag{4.13}$$

$$I^M = \{\, i \in I \mid i \text{ bids on some } j \in J^M \text{ infinitely many times } \}, \tag{4.14}$$

$$I^A = \{\, i \in I \mid i \text{ bids on some } j \in J^A \text{ infinitely many times } \}, \text{ and} \tag{4.15}$$

$$I^\infty = \{\, i \in I \mid i \text{ bids on some } j \in J^\infty \text{ infinitely many times } \}. \tag{4.16}$$

We will show that these four sets partition I.

Suppose $i \in I$ such that $A(i) \setminus J^\infty$ is nonempty. Then there exists $j^* \in A(i) \setminus J^\infty$, and by definition p_{j^*} is bounded above. Thus, after a finite number of iterations

$$x_i = \max_{j \in A(i)} \{c_{ij} - p_j\} \geq c_{ij^*} - p_{j^*} > c_{ij^\infty} - p_{j^\infty} \qquad \forall\, j^\infty \in J^\infty. \tag{4.17}$$

Therefore, if $A(i) \setminus J^\infty$ is nonempty, we must have $i \notin I^\infty$. By the contrapositive,

$$A(i) \subseteq J^\infty, \qquad \forall\, i \in I^\infty. \tag{4.18}$$

This implies that I^∞ is pairwise disjoint with I^F, I^A, and I^M.

Suppose there exists $i \in I \setminus I^F$ such that $J^A \cap A(i) \neq \varnothing$ and $J^M \cap A(i) \neq \varnothing$. By definition, i bids infinitely many times on lots from J^A and/or J^M. Let p_j^n be the lot price of j at the end of the n-th iteration. For all $j \in J^A$, define

$$l_j := \lim_{n \to \infty} p_j^n. \tag{4.19}$$

After a finite number of iterations k, all lots in J^F no longer receive bids and all lots in J^M have reached their fixed prices. Thus, for all $j \in J^M \cup J^F$, and all iterations $n \geq k$,

$$p_j^n = p_j^k. \tag{4.20}$$

Define

$$j^M := \underset{j \in A(i) \cap J^M}{\arg\max} \{c_{ij} - p_j^k\} \tag{4.21}$$

and

$$j^A := \underset{j \in A(i) \cap J^A}{\arg\max} \{c_{ij} - l_j\}. \tag{4.22}$$

One of two possibilities must exist:

1. If $c_{ij^M} - p^k_{j^M} \leq c_{ij^A} - l_{j^A}$, then for any $n \geq k$ we have $c_{ij^M} - p^{\hat{k}}_{j^M} < c_{ij^A} - p^n_{j^A}$. Thus, after the k-th iteration, i will not bid on lots in J^M, and so $i \notin I^M$.
2. If $c_{ij^M} - p^k_{j^M} > c_{ij^A} - l_{j^A}$, then there exists iteration $\hat{k} \geq k$ such that $c_{ij^M} - p^k_{j^M} > c_{ij^A} - p^{\hat{k}}_{j^A}$. Because lot prices are nondecreasing, after the $\hat{k}$-th iteration i will not bid on lots in J^A. Therefore, $i \notin I^A$.

Therefore, I^M and I^A are pairwise disjoint. By definition, I^F must be pairwise disjoint with both I^M and I^A, because a sink cannot bid both finitely many and infinitely many times. Thus, all four sets are pairwise disjoint, and since their union is I they constitute a partition.

We will now consider possible elements in the three sets I^M, I^A, and I^∞, to show that these sets are in fact empty.

1. Suppose I^M is nonempty. After a finite number of iterations all prices in J^M are fixed and all bidders in I^M no longer bid on lots outside of J^M. From that iteration on, by Theorem 2, each bidder $i \in I^M$ must be satisfied after its bid. Thus, some other bidder(s) must cause each $i \in I^M$ to become unsatisfied infinitely many times. By Theorem 2 and the definition of I^M, each time i becomes unsatisfied it will rebid on a lot in J^M with the exact same price as the one it chose on its previous bid. Therefore, after a finite number of additional iterations, all $i \in I^M$ will only have claims in J^F and J^M, and sinks in I^M only become unsatisfied as a result of bids from other members of I^M for lots in J^M. As a result, after a finite number of additional iterations the quantity claimed from $j \in J^M$ at lot price p_j will only be claimed by bidders in I^M.
   Assume without loss of generality that all this has occurred by the start of the k-th iteration. Let D^k be the total unsatisfied demand at the start of the k-th iteration,

$$D^k = \sum_{i \in I^M} D_i, \tag{4.23}$$

   and let Q^k be the total supply available at the fixed prices:

$$Q^k = \sum_{j \in J^M} \sum_{\substack{(i;\, b_{ij},\, q_{ij}) \in C_j \\ b_{ij} = p^k_j}} q_{ij}. \tag{4.24}$$

   Assume without loss of generality that $Q^k = u^k D^k$ for some u^k. Because the prices for lots in J^M remain unchanged, we know $u^k > 1$. During the round, each unsatisfied bidder in I^M must bid on some lot in J^M. Because the prices of those lots do not increase, we know that all those bidders must be satisfied. Thus, the total quantity acquired by bidders in I^M during the k-th round must equal D^k. Since the prices of the lots do not increase, the claimed quantities must

have been taken from Q^k, and so Q^k must have been reduced by D^k. Because the new amounts claimed could come only from previous claims by bidders in I^M, at the start of the $(k+1)$-st round the unsatisfied demand, D^{k+1}, must equal D^k, and the ratio of the available quantity, given by u^{k+1}, must satisfy $u^{k+1} \leq u^k - 1$. Therefore, $\lceil u^k \rceil$ rounds after the k-th, the available quantity at the current prices must have been exhausted, and the price of some lot in J^M must have increased. This contradicts the definition of J^M, and therefore I^M must be empty.

2. Suppose I^A is nonempty. By the definition of J^A, for each $j_r \in J^A$, there exists an associated iteration k_r such that the price at the start of that iteration, $p_{j_r}^{k_r}$, satisfies
$$l_r - p_{j_r}^{k_r} < \varepsilon. \tag{4.25}$$
Let $k = \max_r k_r$.
Assume without loss of generality that by the start of iteration k all lots in J^F have also received their final bids. This implies that all bidders in I^A are now bidding exclusively on lots in J^A. Recall from Theorem 1 that each new bid exceeds the current lot price by at least ε. Thus, after the k-th iteration, for all $i \in I^A$ and all $j \in J^A$, the new bid price b_{ij} must exceed l_j.
Let D^k be the total unsatisfied demand at the start of the k-th iteration,
$$D^k = \sum_{i \in I^A} D_i, \tag{4.26}$$
and let Q^k be the total supply available at prices not exceeding the asymptotic limits:
$$Q^k = \sum_{j \in J^A} \sum_{\substack{(i;\, b_{ij},\, q_{ij}) \in C_j \\ b_{ij} \leq l_j}} q_{ij}. \tag{4.27}$$
Assume without loss of generality that $Q^k = u^k D^k$ for some $u^k > 0$.
Let i be some unsatisfied bidder in I^A, bidding on $j \in J^A$. The bid price offered by i, by exceeding l_j, exceeds the price of all quantities claimed in Q^k. We know from the definition of J^A that the price of lot j does not exceed l_j. Thus, all bidders in I^A must be satisfied at the end of their bids. This implies that the claimed quantities must have been taken from Q^k, and that
$$Q^{k+1} = (u^k - 1) D^k. \tag{4.28}$$
Because no bidders outside I^A will bid again on lots in J^A, the bidders who become unsatisfied due to the new claims on Q^k must be members of I^A. Thus, at the start of the $(k+1)$-st round the unsatisfied demand, D^{k+1}, must equal D^k and the ratio of the available quantity u^{k+1}, must satisfy $u^{k+1} \leq u^k - 1$. Therefore, $\lceil u^k \rceil$ rounds after the k-th, the quantity claimed at prices below asymptotic bounds must have been exhausted, and the price of some lot $j \in J^A$ exceeds l_j. This contradicts the definition of J^A, and therefore I^A must be empty.

3. Suppose I^∞ is nonempty. From Items 1 and 2, we know $I = I^F \cup I^\infty$. Let $i \in I^\infty$. Equation (4.18) implies
$$\min_{j \in A(i)} p_j \geq \min_{j \in J^\infty} p_j, \tag{4.29}$$

and since $p_j \to +\infty$ for all $j \in J^\infty$, it must be the case that

$$\min_{j \in A(i)} p_j \to +\infty. \tag{4.30}$$

Therefore,

$$x_i = \max_{j \in A(i)} \{c_{ij} - p_j\} \to -\infty. \tag{4.31}$$

Thus, after a finite number of iterations, each lot in J^∞ will be satisfied exclusively by the bidders in I^∞. Otherwise, some bidder in I^F would become unsatisfied infinitely many times, causing them to bid infinitely often. This would contradict the definition of I^F.
Furthermore, because I^∞ is nonempty, we know the algorithm must not terminate. Thus, after a finite number of iterations there exists at least one bidder in I^∞ that is not satisfied, while all bidders in I^F have been satisfied by the lots in J^F. It follows that the total demand of the bidders in I^∞ must be strictly larger than the total supply of the lots in J^∞. However, by Equation (4.18), the demand in I^∞ can only be satisfied by supply in J^∞. This contradicts the assumption that a feasible transport plan exists. Therefore, I^∞ must be empty.

Because I^M, I^A, and I^∞ are all empty, we know $I = I^F$. Therefore, the general auction algorithm must terminate after finitely many bids.

4.2 Optimality of the general auction

When a real-valued transport problem is solved by computer, some error is inevitable, if only from the limits of machine precision. We show below that the error associated with the general auction method can be bounded *a priori* with respect to ε. As a consequence, the method is convergent with respect to ε-scaling.

4.2.1 Minimum price increase for general auction

Corollary 1 *Given a feasible transport problem $\mathcal{T}$, there exists a finite $k \in \mathbb{N}$ and $\delta > 0$ such that every k bids the price of a lot is guaranteed to change, and each time a lot price changes it increases by at least δ.*

Proof This follows from Theorem 3, specifically that $J^M \cup J^A = \varnothing$.

4.2.2 General auction preserves ε-CS

Theorem 4 *If a transport plan and lot price vector satisfy ε-CS for the general auction method at the start of an iteration, the same is true of the transport plan and lot price vector obtained at the end of that iteration.*

Proof Suppose ε-CS holds at the start of the iteration. Fix lot j^* and let its price before and price after iteration be given by p_{j^*} and p'_{j^*}, respectively.

Suppose that sink i bids on lot j^* during the iteration, and the lot price of j^* changes as a result. By Equations (3.2) and (3.4)

$$b_{ij^*} = c_{ij^*} - w_i + \varepsilon, \tag{4.32}$$

which implies, by applying Equation (3.3), that

$$c_{ij^*} - b_{ij^*} = w_i - \varepsilon = \max_{j \in A(i),\, j \neq j^*} \{c_{ij} - p_j\} - \varepsilon. \tag{4.33}$$

Equation (3.1) guarantees that $p'_{j^*} \leq b_{ij^*}$, so

$$c_{ij^*} - p'_{j^*} \geq c_{ij^*} - b_{ij^*} = \max_{j \in A(i),\, j \neq j^*} \{c_{ij} - p_j\} - \varepsilon. \tag{4.34}$$

By Theorem 1, $p'_j \geq p_j$ for all j. Thus,

$$c_{ij^*} - p'_{j^*} \geq \max_{j \in A(i), j \neq j^*} \{c_{ij} - p'_j\} - \varepsilon. \tag{4.35}$$

Because $c_{ij^*} - p'_{j^*} \geq c_{ij^*} - p'_{j^*} - \varepsilon$,

$$c_{ij^*} - p'_{j^*} \geq \max_{j \in A(i)} \{c_{ij} - p'_j\} - \varepsilon, \tag{4.36}$$

and ε-CS is satisfied.

Suppose that i has a claim on j^*, but the lot price p_{j^*} has not changed during the iteration. Because ε-CS held prior to the iteration and $p_j \leq p'_j$ for all j,

$$c_{ij^*} - p'_{j^*} = c_{ij^*} - p_{j^*} \geq \max_{j \in A(i)} \{c_{ij} - p_j\} - \varepsilon \geq \max_{j \in A(i)} \{c_{ij} - p'_j\} - \varepsilon. \tag{4.37}$$

Thus, ε-CS holds for all claims in every claim list.

Therefore, ε-CS holds for all $(i, j; q_{ij})$ in the transport plan T.

4.2.3 General auction error bound

Theorem 5 *Let $\varepsilon > 0$ be the step size of the general auction method. If a feasible transport plan exists, then when the general auction method terminates, the resulting feasible transport plan is within $L\varepsilon$ of optimal, where*

$$L = \sum_{i=1}^{M} d_i = \sum_{j=1}^{N} s_j. \tag{4.38}$$

Proof Assume the transport problem is feasible. Let P^* be the optimal primal solution to the transport problem,

$$P^* = \max_{\{\, T \in \mathcal{T} \;|\; T \text{ complete} \,\}} \sum_{(i,\, j) \in \mathcal{A}} c_{ij} f_{ij}, \tag{4.39}$$

and D^* be the optimal dual solution

$$D^* = \min_{p = \{p_j\}_{j=1}^{N}} \left\{ \sum_{i=1}^{M} d_i \max_{j \in A(i)} \{c_{ij} - p_j\} + \sum_{j=1}^{N} s_j p_j \right\}. \tag{4.40}$$

Let T^* be the simplified transport plan when the auction terminates, and let $p^* = (p_1^*, \ldots, p_N^*)$ be the resulting price vector. Let i be any sink. Suppose $(i, j_r;\ f_{ij_r}) \in T^*$, with $r \in \{1, \ldots, t_i\}$. Because (T^*, p^*) satisfies ε-CS,

$$\max_{j \in A(i)} \{c_{ij} - p_j^*\} - \varepsilon \leq c_{ij_r} - p_{j_r}^*. \tag{4.41}$$

Therefore, by rearranging and summing terms for all j_r,

$$\max_{j \in A(i)} \{c_{ij} - p_j^*\} + p_{j_r}^* \leq \varepsilon + c_{ij_r} \tag{4.42}$$

$$f_{ij_r} \left(\max_{j \in A(i)} \{c_{ij} - p_j^*\} + p_{j_r}^* \right) \leq f_{ij_r} (\varepsilon + c_{ij_r}) \tag{4.43}$$

$$\sum_{\substack{(i, j_r; f_{ij_r}) \in T^* \\ r \in \{1, \ldots, t_i\}}} f_{ij_r} \left(\max_{j \in A(i)} \{c_{ij} - p_j^*\} + p_{j_r}^* \right) \leq \sum_{\substack{(i, j_r; f_{ij_r}) \in T^* \\ r \in \{1, \ldots, t_i\}}} (f_{ij_r} \varepsilon + f_{ij_r} c_{ij_r}) \tag{4.44}$$

$$d_i \max_{j \in A(i)} \{c_{ij} - p_j^*\} + \sum_{\substack{(i, j_r; f_{ij_r}) \in T^* \\ r \in \{1, \ldots, t_i\}}} f_{ij_r} p_{j_r}^* \leq d_i \varepsilon + \sum_{\substack{(i, j_r; f_{ij_r}) \in T^* \\ r \in \{1, \ldots, t_i\}}} f_{ij_r} c_{ij_r}. \tag{4.45}$$

Summing over all sinks i, this gives us

$$\sum_{i=1}^{M} d_i \max_{j \in A(i)} \{c_{ij} - p_j^*\} + \sum_{(i, j; f_{ij}) \in T^*} f_{ij} p_j^* \leq \sum_{i=1}^{M} d_i \varepsilon + \sum_{(i, j; f_{ij}) \in T^*} f_{ij} c_{ij}. \tag{4.46}$$

Given any sink j, we have $(i_u, j;\ f_{i_u j}) \in T^*$ for $u \in \{1, 2, \ldots, v_j\}$. Thus, summing first over the i_u for j, and then over all j, we have

$$\sum_{\substack{(i_u, j; f_{i_u j}) \in T^* \\ u \in \{1, \ldots, v_j\}}} f_{i_u j} p_j^* = s_j p_j^* \tag{4.47}$$

$$\sum_{(i, j; f_{ij}) \in T^*} f_{ij} p_j^* = \sum_{j=1}^{N} s_j p_j^*. \tag{4.48}$$

Substituting this into Equation (4.46), we have

$$\sum_{i=1}^{M} d_i \max_{j \in A(i)} \{c_{ij} - p_j^*\} + \sum_{j=1}^{N} s_j p_j^* \leq \sum_{i=1}^{M} d_i \varepsilon + \sum_{(i, j; f_{ij}) \in T^*} f_{ij} c_{ij}. \tag{4.49}$$

Therefore,

$$P^* = D^* \tag{4.50}$$

$$\leq \sum_{i=1}^{M} d_i \max_{j \in A(i)} \{c_{ij} - p_j^*\} + \sum_{j=1}^{N} s_j p_j^* \tag{4.51}$$

$$\leq \sum_{i=1}^{M} d_i \varepsilon + \sum_{(i, j; f_{ij}) \in T^*} f_{ij} c_{ij} \tag{4.52}$$

$$\leq L\varepsilon + P^* \tag{4.53}$$

$$\leq L\varepsilon + D^*. \tag{4.54}$$

4.3 Essential characteristics of the general auction

In *Network Optimization: Continuous and Discrete Models*, Dimitri Bertsekas describes what he considers the "important ingredients" of auction methods [3]. Here, in the same form used by Bertsekas, are what we consider the essential elements of the general auction method:

Given $\varepsilon > 0$:

(a) ε-CS is maintained.
(b-1) During each iteration, at least one bidder with unsatisfied demand claims supply in one lot.
(b-2) The bid price of any claimed supply is increased by at least $\beta\varepsilon$, where β is some fixed positive constant.
(b-3) Any previously-claimed supply that is needed to satisfy a higher priced claim (if any) becomes unclaimed.
(c-1) No bid price is decreased.
(c-2) Any supply that was claimed at the start of an iteration remains claimed at the end of that iteration (although the bidder claiming it may change).

With the exception of (a), all of these characteristics are essential to our argument that the general auction algorithm terminates after a finite number of iterations. Characteristic (a) allows us to relate the the resulting price vector to the optimal price vector we seek, in order to establish a worst-case bound on the distance from optimality.

4.4 AUCTION–SO is a special case of the general auction

Given an integer-valued transport problem, we can relate the general auction method to the extended auction method. This relationship bypasses the AUCTION–SOP algorithms, and establishes the AUCTION–SO as a special case of the general auction. In [4], Bertsekas and Castañón showed that the original assignment auction method is a special case of the AUCTION–SO. Thus, it follows that the assignment auction developed by Bertsekas in [1] is a special case of the general auction.

Theorem 6 *If $\mathcal{T}$ is a feasible integer-valued transport problem and $d_i = 1$ for all sinks i in $\mathcal{T}$, then the general auction algorithm is equivalent to the auction algorithm for similar objects.*

Proof Let $\mathcal{T}$ be any feasible integer-valued transport problem such that for all sinks i, $d_i = 1$. Thus, for purposes of the AUCTION–SO we have the similarity class $S(i) = \{i\}$ for all sinks i. The lot represented by $S(i)$ is unsatisfied if and only if there exists exactly one person for the AUCTION–SO that is unsatisfied. Consider the bidding phase for such a person i.

If the object j_i offers best expense for the AUCTION–SO, then the lot represented by $S(j_i)$ also offers best expense, so the object j_i chosen by the AUCTION–SO corresponds to the choice of lots in the general auction. Let j' be the object such that for the AUCTION–SO

$$w_i = \max_{j \in A(i) \setminus S(j_i)} \{c_{ij} - p_j\} = c_{ij'} - p_{j'}. \tag{4.55}$$

As a consequence, it must be that

$$p_{j'} = \min_{j \in S(j')} p_j, \tag{4.56}$$

and so the second-best expense computed by the general auction method must equal that computed by the AUCTION–SO. Therefore, the bid prices for the SO and general auction must be equal. Because $\mathcal{T}$ is an integer-valued transport problem, $S(i)$ is unsatisfied, and $d_i = 1$, the quantity desired by the general auction must be

$$q_{ij_i} = \min\{D_i, s_{j_i}\} = 1. \tag{4.57}$$

Therefore, the two bidding phases are equivalent.

Let j be some object who receives one or more bids during the claims phase, and let i_j be the person that made the highest bid on j.

Suppose the object j has already been claimed by some person k. Because j is the lowest-priced object in $S(j)$, and each bid increases the price of an object by at least ε, this implies that the lot $S(j)$ is unavailable. The AUCTION–SO "claim" on object j corresponds to making a claim on the lot $S(j)$, and

$$b_{i_j j} = c_{i_j j} - w_{i_j} + \varepsilon \geq p_j + x_{i_j} - w_{i_j} + \varepsilon \geq p_j + \varepsilon. \tag{4.58}$$

Thus, in the general auction the lowest priced claim on the current claim list corresponding to $S(j)$ will be removed.

Assume without loss of generality that the order of the claims in the claim list of the general auction matches the sorted order of the expenses determined in the AUCTION–SO algorithm. Then the lowest priced claim corresponds to the object claimed by person k. As shown in the bidding phase, the quantity claimed by i is 1, so both methods add 1 to D_k. Removing the claim $(k;\, b_{kj},\, 1)$ from the claim list corresponding to $S(j)$ is equivalent to removing $(k,\, j;\, 1)$ from T.

Appending $(i_j,\, j;\, 1)$ to T is equivalent to inserting $(i_j;\, b_{i_j j},\, 1)$ into the claim list corresponding to $S(j)$, and both methods subtract 1 from D_{i_j}. The price update for the general auction is the same as determining the lowest-priced object in $S(j)$ after the price increase on object j. Thus, the two claims phases are equivalent, and so the two auction algorithms are equivalent.

5 Numerical results

We created two different versions of the general auction. The first is integer-based and stores all arc costs; we used it for comparison with existing auction methods.

The second uses floating-point numbers and computes arc costs as needed; we used it to approximate time and memory scaling for the general auction.

We implemented four additional methods for testing our numerical results: extended method's AUCTION, AUCTION–SO, and AUCTION–SOP algorithms, and the network simplex method. The assignment and extended auction methods were used for benchmarking and comparison, while the network simplex method was used to test real-valued solutions for optimality.

All implementations were written in C++, and rely heavily on the C++11 Standard Library. We wrote, compiled, and tested all the programs ourselves, in order to minimize possible confounds in our results. As shown below, the AUCTION–SOP algorithm generated the most favorable results in our tests on non-assignment problems, so we focused on comparisons involving that algorithm.

All of these methods were implemented to solve the general transport problem as described in Section 2.1, applying only those restrictions necessary to satisfy the minimal requirements of the algorithm. One could improve on our results for any particular method, simply by customizing the code to handle specific purposes or environments. However, our goal was to evaluate the average-case effectiveness of the underlying methods themselves, aside from any potential time savings due to specialized design.

The transport problems we used for comparison testing were initialized by creating assignment problems with NETGEN; see [6]. The only modifications to the NETGEN code were increased array sizes (to generate the large problems desired) and alterations to the input/output routines. The network generation code was not altered. When different weights or costs were desired, the existing nodes and arcs were modified using the random uniform distribution functions from the C++11 Standard Library.

In order to obtain accurate comparisons, we used identical conditions for the creation and implementation of all programs, and ran all tests in large blocks. To minimize variation due to underlying problem structure, we solved at least 10 distinct transport problems for each result. Solution times are averages. When comparing algorithms, our programs loaded the data and ran multiple algorithms in sequence, minimizing changes in conditions over time. We also ran multiple iterations of each test to ensure that results were typical.

Computation times under one second were obtained by forcing multiple runs to bring the total time over one second, and dividing the total by the number of runs to obtain an average. For example, if a time of 0.9 seconds is given, that indicates the algorithm was run ten times, and the total resulting time divided by ten. In order to obtain accurate results over multiple runs, we wrote our programs to recreate all data structures and computations from scratch. We also compared our multi-run results to single-run times, to ensure that our multi-run averages closely approximated the times given by single runs, which they did.

5.1 Comparison of auction methods for assignment

When Bertsekas and Castañón compared their implementation of the extended auction to the performance of the assignment auction on standard assignment problems [4, p. 92], they found that the additional overhead required by the extended auction measurably slowed computation time. For this reason, we wished to see how the performance of the general auction compared to that of the assignment and extended auction methods when applied to assignment problems of increasing size. The comparison done by Bertsekas and Castañón in 1989 used problems with 150 to 500 pairs of sources and sinks. Given the improvements in computation power since that time, our comparison starts at 3000 sinks and ends at 10000 (with an equal number of sources at each size). As in [4], the number of arcs in each problem is 12.5% of the maximum possible. The cost range is also relevant, as it influences complexity scaling; see [3, p. 34]. We used the fixed cost range $C = 100$. The resulting times are given in Table 1. Approximating the time complexity of each algorithm with the power regression aN^b, we find that the unoptimized AUCTION–SOP has $b = 2.9$, but each of the other three algorithms has a nearly identical power: $b = 2.4$.

N	Assignment auction	General auction	SOP auction	SOP (unoptimized)
3000	1.23	1.21	1.71	9.06
4000	2.65	2.68	3.52	18.63
5000	4.73	4.82	6.23	35.33
6000	6.96	7.06	8.82	60.55
7000	8.90	8.97	10.85	90.82
8000	12.60	12.64	15.98	132.93
9000	17.00	16.83	20.73	193.30
10000	18.88	19.05	23.04	249.03

Table 1: Time in seconds for assignment scaling
N sinks and N sources, $N^2/8$ arcs

Over all tested problem sizes, the computed times for the assignment and general auction methods are nearly identical, differing by less than 2%. The largest difference in the times required by these two methods was less than two-tenths of one second. This result suggests that the overhead required for the implementation of claim lists scales acceptably for relatively simple problems such as assignments.

Storage requirements were dominated by the need to store an explicit cost value for each arc, and so were nearly identical for all three algorithms. Storage scaled quadratically with respect to the number of sinks.

5.2 Comparison of extended and general auction

As Bertsekas explains in [2, p. 58–59], the extended auction method works best when applied to problems characterized by two properties: *homogeneity* and *asymmetry*.

Homogeneity means the sinks and sources have a small range of weights, while asymmetric problems have far more sinks than sources (or sources than sinks).

We replicated these ideal conditions for our tests, using NETGEN to construct systems with 400 sinks and N sources. Each of the N sources has unit weight. The 400 sinks have a 90%/10% weight distribution: 50% of the weight is uniformly distributed over 90% of the sinks, while the other 50% is uniformly distributed over the other 10%. The density of the arcs is 14%. Computation times for all four auction algorithms, with various values of N, are given in Table 2. Approximating the time complexity of each algorithm with the power regression aN^b gives us the results shown in Table 3.

N	General auction	SOP auction	SO auction	Assignment (unoptimized)
3000	0.16	0.13	1.45	1.38
6000	0.56	0.32	6.41	6.29
9000	1.06	0.67	15.51	13.30
12000	1.72	0.67	28.37	27.89
15000	2.37	0.99	40.13	39.82
18000	3.17	1.18	61.41	55.42
21000	3.80	1.45	80.09	86.85
24000	5.34	1.80	118.66	114.38

Table 2: Time in seconds for asymmetric problem with unit weights
400 sinks with 90%/10% weights, N sources with unit weight
14% arc density

aN^b coefficients	General auction	SOP auction	SO auction	Assignment (unoptimized)
a	0.029	0.036	0.15	0.14
b	1.63	1.22	2.07	2.10

Table 3: Power regression aN^b for asymmetric problem with unit weights

The assignment problem results given in Table 1 illustrate what happens when we apply the AUCTION–SOP to a homogeneous problem without asymmetry. To evaluate what happens with an asymmetric problem that is not homogeneous, we implemented the same $400 \times N$ problem as above, but gave the N sources weights in the integer range [1, 19]. We used a uniform integer distribution, giving us an average source weight of 10. The sinks kept the 90%/10% weight distribution. Even this slight increase in weight resulted in a significant change in the relationship between the general auction and AUCTION–SOP, as shown in Table 4.

N	General auction	SOP auction
3000	1.15	1.88
6000	2.64	4.89
9000	5.40	8.06
12000	6.82	12.18
15000	8.93	19.12
18000	12.08	25.13
21000	8.72	35.58
24000	10.71	43.81

Table 4: Time in seconds for asymmetric problem with increased weights
400 sinks with 90%/10% weights, N sources with [1, 19] weights
14% arc density

Approximating the time complexity of each algorithm with the power regression aN^b gives us:

$a = 0.38, \quad b = 1.11$ for the general auction, and
$a = 0.32, \quad b = 1.51$ for the AUCTION–SOP.

Interestingly, the scaling behavior of the general auction improved with the added weight, while that of the AUCTION–SOP degraded.

Generally speaking, the transport-to-assignment transformation used by the algorithms of the extended auction method degrades performance on transport problems with arbitrarily large total weights. To assess the overall impact, we generated a small transport problem and scaled only the total weight, comparing the time required by the extended and general auction methods.

In every case, the structure of the transport problem was identical. It had 500 sources, 500 sinks, and the same 225000 arcs (90% of maximum). The cost range was $C = 100$. Using a uniform integer distribution, the weight was spread among the sources and sinks. Each was given a minimum weight of one, but no maximum was assumed. The total weights and computation times for the two algorithms are given in Table 5.

The data collected suggests that, as weight increases, time for the AUCTION–SOP algorithm scales on the order of $O(L^b)$, where $b = 1.73$ and L is the total transport weight. Time increases for the general auction algorithm as well, but it appears to be scaling with $b = 0.63$. The storage requirements for the AUCTION–SOP scale almost linearly: $O(L^b)$, where $b = 0.74$. This is not surprising, given that the AUCTION–SOP algorithm explicitly transforms the set of sinks into L persons. Storage requirements for the general auction are approximately constant.

The time increases given by both algorithms suggest that the weight range, or number of possible weight values, may be relevant to complexity calculations. The impact of increasing weight was not mentioned in [4], and weight changes do not occur in the assignment problem. However, it seems natural to consider such changes in the context of the transport problems. The larger the weight range, the more likely it becomes that each sink will have positive flows over multiple arcs, increasing the complexity of the optimal transport plan, and thus the time required to calculate it.

Weight L	General auction	Extended auction
500	0.09	0.11
600	0.15	6.35
700	0.16	17.72
800	0.13	23.16
900	0.15	24.85
1000	0.15	29.65
2000	0.25	51.44
3000	0.31	65.29
4000	0.40	82.20
5000	0.47	102.07

(a) Time in seconds

Weight L	General auction	Extended auction
500	10.78	10.72
600	10.79	11.78
700	10.80	12.81
800	10.81	13.84
900	10.81	14.88
1000	10.82	15.91
2000	10.88	26.28
3000	10.97	36.70
4000	11.00	47.02
5000	11.08	57.54

(b) Storage in megabytes

Table 5: Results for weight-only scaling
500 sinks and 500 sources, 225000 arcs

Given the degree to which range increases adversely affect the performance of the extended auction method, we also compared the scaling behaviors of the AUCTION–SOP and general auction methods in problems where the relative total weight was fixed. We randomly generated transport problems with N sources and N sinks, where the total weight of each problem was $L = 2N$. The number of arcs for each problem was fixed at 90% of maximum, and the cost range was $C = 100$. The results are shown in Table 6.

N	General auction	Extended auction
500	0.13	8.32
600	0.19	37.28
700	0.31	63.27
800	0.42	177.86
900	0.61	233.96
1000	0.81	291.29
1100	1.08	920.57

(a) Time in seconds

N	General auction	Extended auction
500	10.82	15.91
600	20.15	27.53
700	25.23	35.26
800	30.77	43.86
900	36.48	53.05
1000	42.89	63.36
1100	70.52	95.38

(b) Storage in megabytes

Table 6: Results for fixed weight-ratio scaling
N sinks and N sources, $0.9N^2$ arcs, total weight $2N$

The time required by the general auction method scales like $O(N^b)$ with $b = 2.69$, whereas the extended auction scales with $b = 5.40$. Because of the need to store an explicit cost value for each arc, both methods scale quadratically with respect to storage. At any given size, the extended auction uses approximately 150% of the amount of storage needed for the general auction method. The algorithm uses the additional storage to explicitly transform sources into objects.

5.3 General auction performance on real-valued transport

When evaluating the scaling properties of the real-valued general auction method, we wanted to focus on its behavior as problem size increased, with all other variables held constant. To do so, we needed to fix the ranges of both costs and weights. We constructed transport problems using the following method:

- We generate N sinks and an equal number of sources. Sources and sinks are points in the plane, restricted to the integer coordinates on the cube $[0,\ N]\times[0,\ N]$. Points are placed randomly using the uniform distribution.
- The graph underlying our transport problem is given by the complete bipartite graph $\mathbb{K}_{N,\,N}$, so the problem is maximally dense with N^2 arcs. The cost function is a modification of the Euclidean distance. Given sink $i = (x_i,\, y_i)$ and source $j = (x_j,\, y_j)$, the cost of the arc connecting them is

$$c_{ij} = \sqrt{1 + m_{ij}}, \quad \text{where } m_{ij} = \left[(x_i - x_j)^2 + (y_i - y_j)^2\right] \pmod{20}. \tag{5.1}$$

- For $i \in \{1, \ldots, N\}$, the weight of sink i is given by the formula

$$d_i = \sqrt{1 + m_i} \quad \text{where } m_i = [i - 1] \pmod{20}. \tag{5.2}$$

  and for $j \in \{1, \ldots, N\}$, the weight of source j is given by the formula

$$s_j = \sqrt{1 + m_j} \quad \text{where } m_j = [j - 1] \pmod{20}. \tag{5.3}$$

We chose this method because it populates the weights and costs with irrational values, while restricting the range of those values. Because the total weight of the sinks equals the total weight of the sources, and the transport graph is complete, the problem is guaranteed to be feasible.

The integers are closed under addition and subtraction, so for an integral transport problem the smallest deviation from the optimal cost cannot be less than one. Furthermore, the range of possible flow values cannot be greater than the largest absolute weight. Neither of these is true for real-valued problems. To get a sense of how real-valued data affects these characteristics, consider the cost and weight values chosen for our sample problems. Even without weight splitting, it is possible to make a flow adjustment on the smallest possible cycle, four arcs, that changes the cost by

$$1 \cdot \left(\sqrt{19} - \sqrt{20} + \sqrt{19} - \sqrt{18}\right) \approx 0.00302.$$

From this, one can extrapolate the size of the range. Increasing the number of arcs in the cycle would further increase the range and reduce the minimum possible cost.

As the flow adjustment above indicates, cost and weight ranges are determined, not by the 20 distinct irrational values given, but by the number of possible differences that those values can generate. Theoretically, there could be as many as $O(V^N)$ differences, where V is the number of distinct cost/weight values in the problem. The smallest possible cost adjustment could be significantly less than machine precision.

In fact, the cost and weight ranges seemed to have far less negative impact on computation than the description above would suggest. We able to generate optimal

N	General auction
1000	1.11
2000	6.37
3000	12.74
4000	25.87
5000	42.49
6000	61.14
7000	94.12
8000	143.51
9000	179.20
10000	218.59

(a) Time in seconds

N	General auction
1000	3.27
2000	5.81
3000	7.93
4000	14.00
5000	14.69
6000	15.24
7000	16.27
8000	18.38
9000	25.82
10000	29.16

(b) Storage in megabytes

Table 7: Real-valued general auction results
N sinks and N sources, N^2 arcs

solutions in a single iteration of the general auction using $\varepsilon = 0.75$. The results of our tests are given in Table 7.

As described above, it can be difficult to determine *a priori* minimum ε values that guarantee optimal solutions. For these tests, we confirmed the optimality of our solutions *a posteriori*, by solving the same problems using the network simplex method. We first calculated a primal solution using the transport map resulting from the general auction method. We then transformed that transport map into a feasible tree solution, which we used to initialize the network simplex.[1] Finally, we compared the optimal cost resulting from the network simplex to the primal cost determined using the general auction.

Given our computer and the type of calculations performed, we have machine precision equal to

$$\sqrt{\texttt{eps}} := 1.490116 \times 10^{-8}. \tag{5.4}$$

The optimal cost values calculated by our general auction and network simplex methods were identical up to the limit of machine precision. Because the optimal costs all exceeded 10^4, this gave us at least 11 identical digits.

The time complexity with respect to the number of sinks (or sources) resembles:

$$aN^b \text{ seconds}, \quad \text{with} \quad a = 1.62 \times 10^{-7} \text{ and } b = 2.28.$$

This is not surprising, given the dense and complicated nature of the real-valued transport problems we chose to solve. Because our real-valued implementation of the general auction algorithm computed cost values as needed, storage complexity is roughly linear.

[1] The performance of the network simplex method is strongly dependent on the initial network, which must be a feasible transport map whose underlying graph is a spanning tree. The close to optimality the network is (as measured in the number of "pivots"), the better the performance.

6 Conclusions

We have introduced what we call the general auction method to overcome some of the data restrictions imposed by traditional auction algorithms. Our goal was to create an auction technique suitable to compute real-valued optimal transport solutions. As we have shown, the general auction allows us to take the same rationale as the original auction method and apply it directly to real-valued optimal transport problems. The method is guaranteed to terminate after a finite number of iterations, and it offers *a priori* error bounding.

Furthermore, as a discrete transport solver, the general auction method offers many practical advantages. Extending it to parallel computing is relatively straightforward. By applying Theorem 5, it is possible to fine-tune the amount of computation to a predefined level of acceptable error. This is a significant departure from most linear programming methods, which halt only when the exact solution is reached, and can be arbitrarily far from optimal when interrupted. The results of the general auction include the information necessary to compute both transport solutions: the claim lists contain the flows needed for the primal solution, and the price vector allows approximation of the dual solution. Together, these two solutions give improved *a posteriori* error bounds, which can be used in "on-the-fly" ε-scaling schemes.

The general auction should also be easy to extend. In this regard, the efforts of Bertsekas and Castañón provide a road map for the type of extensions that may be possible. In particular, a reverse general auction algorithm seems feasible, as does a method for solving partial transport problems. Given the current interest in fast, accurate techniques for solving optimal transport problems, we invite further exploration of the general auction method.

References

1. Bertsekas, D.P.: A new algorithm for the assignment problem. Mathematical Programming **21**(1), 152–171 (1981)
2. Bertsekas, D.P.: Auction algorithms for network flow problems: a tutorial introduction. Computational Optimization and Applications **1**(1), 7–66 (1992)
3. Bertsekas, D.P.: Network Optimization: Continuous and Discrete Models. Athena Scientific, Belmont, Massachusetts (1998)
4. Bertsekas, D.P., Castañón, D.A.: The auction algorithm for the transportation problem. Annals of Operations Research **20**(1), 67–96 (1989)
5. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1–2), 83–97 (1955)
6. Netlib: The Netlib repository at UTK and ORNL. URL `http://www.netlib.org/lp/generators`