Online Algorithms for Market Clearing*

Avrim Blum† Tuomas Sandholm† Martin Zinkevich†

Abstract

In this paper we study the problem of online market clearing where there is one commodity in the market being bought and sold by multiple buyers and sellers whose bids arrive and expire at different times. The auctioneer is faced with an online clearing problem of deciding which buy and sell bids to match without knowing what bids will arrive in the future. For maximizing *profit*, we present a (randomized) online algorithm with a competitive ratio of $\ln(p_{max}-p_{min})+1$, when bids are in a range $[p_{min}, p_{max}]$, which we show is the best possible. A simpler algorithm has a ratio twice this, and can be used even if expiration times are not known. For maximizing the *number* of trades, we present a simple greedy algorithm that achieves a factor of 2 competitive ratio if no money-losing trades are allowed. Interestingly, we show that if the online algorithm is allowed to *subsidize* matches — match money-losing pairs if it has already collected enough money from previous pairs to pay for them — then it can be 1-competitive with respect to the optimal offline algorithm that is not allowed subsidy. That is, the ability to subsidize is at least as valuable as knowing the future. We also consider the objectives of maximizing buy or sell volume, and present algorithms that achieve a competitive ratio of $2(\ln(p_{max}/p_{min})+1)$, or $\ln(p_{max}/p_{min})+1$ if the online algorithm is allowed subsidization. We show the latter is the best possible competitive ratio for this setting. For social welfare maximization we also obtain an optimal competitive ratio, which is below $\ln(p_{max}/p_{min})$. We present all of these results as corollaries of theorems on online matching in an incomplete interval graph.

Some of our algorithms probabilistically select a threshold that is used later in making their decisions. Even though this produces optimal competitive ratios, we show we can use online learning methods to perform nearly as well as the *best* choice of threshold in hindsight, which may be much closer to the offline optimum in certain stationary settings. We also consider incentive compatibility, and develop a nearly optimal incentive-compatible algorithm for maximizing social welfare. Finally, we show how some of our results can be generalized to settings in which the buyers and sellers themselves have online bidding strategies, rather than just each having individual bids.

1 Introduction

Electronic commerce is becoming a mainstream mode of conducting business. In electronic commerce there has been a significant shift to dynamic pricing via *exchanges* (that is, markets with potentially multiple buyers and multiple sellers). The range of applications includes trading in stock markets, bandwidth allocation in communication networks, as well as resource allocation in operating systems and computational grids. In addition, exchanges play an increasingly important role in business-to-business commerce. Several independent business-to-business exchanges

*An extended abstract of this paper appeared in Proceedings of SODA 2002. This paper contains additional results not present in the conference version.

†Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213-3891. {avrim,sandholm,maz}@cs.cmu.edu

have been founded (e.g., ChemConnect). In addition, large established companies have started to form *buying consortia* (such as Transora, Covisint, Visteon, Exostar, Trade-Ranger, Enporion, and NASBA) for increasing their buying power through aggregation. This sourcing trend means that instead of one company buying from multiple suppliers, we have *multiple* companies buying from the multiple suppliers. In another words, sourcing is moving toward an exchange format.

These trends have led to an increasing need for fast market clearing algorithms for exchanges. Such algorithms have been studied in the offline (batch) context [19, 20, 23, 21]. Also, recent electronic commerce server prototypes such as *eMediator* [18] and *AuctionBot* [22] have demonstrated a wide variety of new market designs, leading to the need for new clearing algorithms.

In this paper we study the ubiquitous setting where there is a market for one commodity, for example, DELL stocks, bonds, pork bellies, electricity, memory chips, or CPU time. For simplicity, we assume that each buy bid and each sell bid is for a single unit of the commodity (to buy or sell multiple units, a bidder could submit multiple bids[1]). In these settings, the auctioneer has to clear the market (match buy and sell bids) without knowing what the future buy/sell bids will be. The auctioneer faces the tradeoff of clearing all possibles matches as they arise versus waiting for additional buy/sell bids before matching. Waiting can lead to a better matching, but can also hurt because some of the existing buy/sell bids might expire or get retracted as the bidders get tired of waiting.

While the Securities Exchange Commission imposes relatively strict rules on the matching process in securities markets like NYSE and NASDAQ [6], most new electronic markets (for example for business-to-business trading) are not securities markets. In those markets the auctioneer has significant flexibility in deciding which buy bids and sell bids to accept. In this paper we will study how well the auctioneer can do in those settings, and with what algorithms.

We formalize the problem faced by the auctioneer as an online problem in which buy and sell bids arrive over time. When a bid is introduced, the auctioneer learns the bid price and expiration time (though some of the simpler algorithms will not need to know the expiration times). At any point in time, the auctioneer can match a live buy bid with a live sell bid, removing the pair from the system. It will be convenient to assume that all bids have integer-valued prices[2](for example, money that cannot be split more finely than pennies) that lie in some range $[p_{min}, p_{max}]$.

Definition 1 *A* ***temporal clearing model*** *consists of a set B of buy bids and a set S of sell bids. Each bid $v \in B \cup S$ has a positive price $p(v)$, is introduced at a time $t_i(v)$, and removed at a time $t_f(v)$. A bid v is said to be alive in the interval $[t_i(v), t_f(v)]$. Two bids $v, v' \in B \cup S$ are said to be concurrent if there is some time when both are alive simultaneously.*

Definition 2 *A* ***legal matching*** *is a collection of pairs $\{(b_1, s_1), (b_2, s_2), \ldots\}$ of buy and sell bids such that b_i and s_i are concurrent.*

An *offline algorithm* receives knowledge of all buy and sell bids up front. An *online algorithm* only learns of bids when they are introduced. Both types of algorithms have to produce a legal

[1] This works correctly if the buyers have diminishing marginal valuations for the units (the first unit is at least as valuable as the second, etc.) and the sellers have increasing marginal valuations for the units. Otherwise, the auctioneer could allocate the bidder's $k+1$'st unit to the bidder without allocating the bidder's k'th unit to the bidder, thus violating the intention of the bid by allocating a certain number of units to the bidder at a different price than the bidder intended. These restrictions seem natural—at least in the limit—as buyers get saturated and sellers run out of inventory and capacity to produce.

[2] Technically, we will assume that the optimal algorithm is incapable of matching two bids which are closer than one unit in value when it is attempting to maximize profit.

matching, that is, a buy bid and sell bid can only be matched if they are concurrent. As usual in competitive analysis [5], we will measure the performance of our online algorithms against the optimal offline solution for the observed bid sequence.[3]

Competitive analysis of auctions (multiple buyers submitting bids to one seller) has been conducted by a number of authors [15, 10, 9, 2]. In this paper, we present the first competitive analysis of *exchanges* (where there can be multiple buyers and multiple sellers submitting bids). Exchanges are a generalization of auctions — one could view an exchange as a number of overlapping auctions — and they give rise to additional issues. For example, if a seller does not accept a buy bid, some other seller might.

Our goal, for each objective function we consider, will be to produce online algorithms with optimal competitive ratios. We consider the following objectives:

- **Maximize profit.** Each pair of buy and sell bids that are matched produces a profit, which is the difference between the buy price and the sell price. The total profit is the sum of these differences, over all matched pairs. Offline, the matching that optimizes profit can be found via weighted bipartite matching. We present a (randomized) online algorithm with competitive ratio $\ln(p_{max} - p_{min}) + 1$, which we show is the best possible. A simpler algorithm has ratio twice this, and can be used even if expiration times are not known. These algorithms build on analysis of [7] for the one-way-trading problem.[4] We also show how online learning results [17, 8, 4] can be used to produce algorithms with even stronger guarantees in certain stationary settings.

- **Maximize liquidity.** Liquidity maximization is important for a marketplace for several reasons. The success and reputation of an electronic marketplace is often measured in terms of liquidity, and this affects the (acquisition) value of the party that runs the marketplace. Also, liquidity attracts buyers and sellers to the marketplace; after all, they want to be able to buy and sell.

  We analyze three common measures of liquidity: 1) number of trades, 2) sum of the prices of the cleared buy bids (*buy volume*), and 3) sum of the prices of the cleared sell bids (*sell volume*). Under criterion 1, the goal is to maximize the number of trades made, rather than the profit, subject to not losing money. We show that a simple greedy algorithm achieves a factor of 2 competitive ratio, if no money-losing trades are allowed. This can be viewed as a variant on the on-line bipartite matching problem [13]. Interestingly, we show that if the online algorithm is allowed to *subsidize* matches — match money-losing pairs if it has already collected enough money from previous pairs to pay for them — then it can be 1-competitive with respect to the optimal offline algorithm that is not allowed subsidy. That is, the ability to subsidize is at least as valuable as knowing the future.

  For the problems of maximizing buy or sell volume, we present algorithms that achieve a competitive ratio of $2(\ln(p_{max}/p_{min}) + 1)$ without subsidization. We also present algorithms that achieve a competitive ratio of $\ln(p_{max}/p_{min}) + 1$ with subsidization with respect to the

[3] This performance measure assumes that actions performed by the auctioneer do not influence the sequence of bids submitted in the future. For instance, we do not worry that had we acted differently, we might have seen a better sequence of bids. We relax this assumption in Section 9.

[4] A somewhat related problem is *online difference maximization* [12]. However, in that setting, the focus is on the average case (bids arrive in a random order) and the goal is to find a single pair of bids whose difference in *rank* is maximized.

optimal offline algorithm that cannot use subsidies. This is the best possible competitive ratio for this setting.

- **Maximize social welfare.** This objective corresponds to maximizing the good of the buyers and sellers in aggregate. Specifically, the objective is to have the items end up in the hands of the agents that value them the most. We obtain an optimal competitive ratio, which is the fixed point of the equation $r = \ln \frac{p_{max}}{r p_{min}}$. The Greedy algorithm achieves a competitive ratio at most twice this.

We develop all of our best algorithms in a more general setting we call the incomplete interval-graph matching problem. In this problem, we have a number of intervals (bids), some of which overlap in time, but only *some* of those may actually be matched (because we can only match a buy to a sell, because the prices must be in the correct order, etc.). By addressing this more general setting, we are able to then produce our algorithmic results as corollaries.

Throughout most of the paper, we assume that the agents are truthful, that is, they bid their true valuations. Later on in Section 8, we remove this assumption.

2 An Abstraction: Online Incomplete Interval Graphs

In this section we introduce an abstraction of the temporal bidding problem that will be useful for producing and analyzing optimal algorithms, and may be useful for analyzing other online problems as well.

Definition 3 *An incomplete interval graph is a graph $G = (V, E)$, together with two functions t_i and t_f from V to $[0, \infty)$ such that:*

1. *For all $v \in V$, $t_i(v) < t_f(v)$.*
2. *If $(v, v') \in E$, then $t_i(v) \leq t_f(v')$ and $t_i(v') \leq t_f(v)$.*

We call $t_i(v)$ the start time of v, and $t_f(v)$ the expiration time of v. For simplicity, we assume that for all $v \neq v' \in V$, $t_i(v) \neq t_i(v')$ and $t_f(v) \neq t_f(v')$.[5]

An incomplete interval graph can be thought of as an abstraction of the temporal bidding problem where we ignore the fact that bids come in two types (buy and sell) and have prices attached to them, and instead we just imagine a black box "E" that given two bids v, v' that overlap in time, outputs an edge if they are allowed to be matched. By developing algorithms for this generalization first, we will be able to more easily solve the true problems of interest.

We now consider two problems on incomplete interval graphs: the online edge-selection problem and the online vertex-selection problem. In the online *edge-selection* problem, the online algorithm maintains a matching M. The algorithm sees a vertex v at the time it is introduced (that is, at time $t_i(v)$). At this time, the algorithm is also told of all edges from v to other vertices which have already been introduced. The algorithm can select an edge only when both endpoints of the edge are alive. Once an edge has been selected, it can never be removed. The objective is to maximize the number of edges in the final matching, $|M|$.

[5] All our results can be extended to settings without this restriction, and where $t_i(v)$ may equal $t_f(v)$. This is accomplished by imposing an artificial total order on simultaneous events. Among the events that occur at any given time, bid introduction events should precede bid expiration events. The introduction events can be ordered, for example, in the order they were received, and so can the expiration events.

In the online *vertex-selection* problem, the online algorithm maintains a set of vertices W, with the requirement that there must *exist* some perfect matching on W. At any point in time, the algorithm can choose two live vertices v and v' and add them into W so long as there exists a perfect matching on $W \cup \{v, v'\}$. Note that there need not exist an edge between v and v'. The objective is to maximize the size of W. So, the vertex-selection problem can be thought of as a less stringent version of the edge-selection problem in that the algorithm only needs to commit to the *endpoints* of the edges in its matching, but not the edges themselves.

It is easy to see that no deterministic online algorithm can achieve a competitive ratio less than 2 for the edge-selection problem.[6] A simple greedy algorithm achieves this ratio:

Algorithm 1 (Greedy) *When a vertex is introduced, if it can be matched, match it (to any one of the vertices to which it can be matched).*

Theorem 1 *The Greedy algorithm achieves a competitive ratio of 2 for the edge-selection problem.*

Proof: Consider an edge (v, v') in the optimal matching M^*. Define v to be the vertex which is introduced first, and v' to be the vertex which is introduced second. Then the algorithm will match either v or v'. In particular, if v is not matched before v' is introduced, then we are guaranteed v' *will* be matched (either to v or some other vertex). Therefore, the number of *vertices* in the online matching M is at least the number of *edges* in M^*, which means $|M| \geq |M^*|/2$. ∎

For the vertex-selection problem, we show the following algorithm achieves a competitive ratio of 1. That is, it is guaranteed to find (the endpoints of) a maximum matching in G.

Algorithm 2 *Let W be the vertices selected so far by the algorithm. When a vertex v is about to expire, consider all the live unmatched vertices v', sorted by expiration time from earliest to latest. Add the first pair $\{v, v'\}$ to W such that there exists a perfect matching on $W \cup \{v, v'\}$. Otherwise, if no unmatched vertex v' has this property, allow v to expire unmatched.*

Theorem 2 (Main Theorem) *Algorithm 2 produces a set of nodes having a perfect matching M which is a maximum matching in G.*

The proof of Theorem 2 appears in the appendix. The core of the proof is to show that if W is the set of selected vertices, then at all times the following invariants hold:

- H_1: For any expired, unmatched vertex w, there does not exist any untaken vertex w' such that there is a perfect matching on $W \cup \{w, w'\}$. (An untaken vertex is a vertex that has been introduced but not matched.)
- H_2: For any matched vertex w, there does not exist an untaken vertex w' such that there is a perfect matching on $W \cup \{w'\} - \{w\}$ and $t_f(w) > t_f(w')$.
- H_3: For any two unexpired vertices $w, w' \in W$, there exists no perfect matching on $W - \{w, w'\}$.

The first invariant says that the algorithm is complete: it lets no matchable vertex expire, and expired vertices do not later become matchable. The second and third invariants say that the algorithm is cautious. The second invariant says that the algorithm favors vertices which expire

[6] Consider the following scenario: vertex u expires first and has edges to v_1 and v_2. Then, right after u expires, a new vertex w arrives with an edge to whichever of v_1 or v_2 the algorithm matched to u.

earlier over those which expire later. The third invariant states that the algorithm only matches vertices that it has to: no subset of the set of vertices chosen has a perfect matching and contains all of the expired vertices[7].

Since an untaken vertex is a vertex which has been introduced but not matched, no untaken vertices exist at the start of the algorithm. Also, W is empty. Therefore, these invariants vacuously hold at the start.

Three events can occur: a vertex is introduced, a vertex expires without being matched, or a vertex expires and is added with some other vertex to W. We establish that if all the invariants hold before any of these events, then they hold afterwards as well. If the first invariant holds at the termination of the algorithm, then no pair could be added to the set selected. The augmenting path theorem [3] establishes that the selected set is therefore optimal. A full proof appears in the appendix.

3 Profit Maximization

We now return to the temporal bidding problem and show how the above results can be used to achieve an optimal competitive ratio for maximizing profit.

We can convert the profit maximization problem to an incomplete interval graph problem by choosing some θ to be the minimum profit which we will accept to match a pair. So, when translating from the temporal bidding problem to the incomplete interval matching problem, we insert an edge between a concurrent buy bid b and a sell bid s if and only if $p(b) \geq p(s) + \theta$.

The Greedy algorithm (Algorithm 1) then corresponds to the strategy: "whenever there exists a pair of bids in the system that would produce a profit at least θ, match them immediately." Algorithm 2 attempts to be more sophisticated: first of all, it waits until a bid is about to expire, and then considers the possible bids to match to in order of their expiration times. (So, unlike Greedy, this algorithm needs to know what the expiration times are.) Second, it can choose to match a pair with profit less than θ if the actual *sets* of matched buy and sell bids could have been paired differently in hindsight so as to produce a matching in which each pair yields a profit of at least θ. This is not *too* bizarre since the sum of surpluses is just the sum of buy prices minus the sum of sell prices, and so doesn't depend on which was matched to which.

Define $M^*(G_\theta)$ to be the maximum matching in the incomplete interval graph G_θ produced in the above manner. Then, from Theorem 1, the greedy edge-selection algorithm achieves a profit of at least $\frac{1}{2}\theta|M^*(G_\theta)|$. Applying Algorithm 2 achieves surplus of at least $\theta|M^*(G_\theta)|$.

So how should we choose θ? If we set θ to 1, then the number of matched pairs will be large, but each one may produce little surplus. If we set θ deterministically any higher than 1, it is possible the algorithms will miss every pair, and have no surplus even when the optimal matching has surplus.

Instead, as in [7] and similar to the Classify-and-Randomly-Select approach [16, 1] (see also [11]), we will choose θ randomly according to an exponential distribution. Specifically, for all $x \in [1, p_{max} - p_{min}]$, let

$$\Pr[\theta \leq x] = \frac{\ln(x) + 1}{\ln(p_{max} - p_{min}) + 1},$$

[7]The paraphrasing of this last point is a bit more extreme than the others, but it turns out that if there exists a perfect matching on W and a perfect matching on $W' \subset W$, then there exists two vertices $v, v' \in W - W'$ such that there is a perfect matching on $W - \{v, v'\}$.

where $Pr[\theta = 1] = \frac{1}{\ln(p_{max} - p_{min}) + 1}$. Observe that this is a valid probability distribution. Let OPT be the surplus achieved by the optimal offline algorithm.

Lemma 1 *If θ is chosen from the above distribution, then* $\mathbf{E}[\theta|M^*(G_\theta)|] \geq \frac{\text{OPT}}{\ln(p_{max} - p_{min}) + 1}$.

Corollary 1 *The algorithm that chooses θ from the above distribution and then applies Greedy to the resulting graph achieves competitive ratio $2(\ln(p_{max} - p_{min}) + 1)$. Replacing Greedy with Algorithm 2 achieves competitive ratio $\ln(p_{max} - p_{min}) + 1$.*

In Section 6 we prove a corresponding lower bound of $\ln(p_{max} - p_{min}) + 1$ for this problem.

Proof (of Lemma 1): Let us focus on a specific pair (b, s) matched by OPT. Let $R_\theta(b, s) = \theta$ if $p(b) - p(s) \geq \theta$ and $R_\theta(b, s) = 0$ otherwise. Observe that $\theta|M^*(G_\theta)| \geq \sum_{(b,s) \in \text{OPT}} R_\theta(b, s)$ because the set of pairs of profit at least θ matched by OPT is a legal matching in the incomplete interval graph. So, it suffices to prove that $\mathbf{E}[R_\theta(b, s)] \geq (p(b) - p(s))/(\ln(p_{max} - p_{min}) + 1)$.

We do this as follows. First, for $x > 1$, $\frac{d}{dx}\Pr[\theta \leq x] = \frac{1}{x(\ln(p_{max} - p_{min}) + 1)}$. So,

$$\begin{aligned} \mathbf{E}[R_\theta(b, s)] &= \Pr[\theta = 1] + \int_1^{p(b)-p(s)} \frac{x\,dx}{x(\ln(p_{max} - p_{min}) + 1)} \\ &= \frac{p(b) - p(s)}{\ln(p_{max} - p_{min}) + 1}. \quad \blacksquare \end{aligned}$$

One somewhat strange feature of Algorithm 2 is that it may recommend matching a pair of buy and sell bids that actually have negative profit. Since this cannot possibly improve total profit, we can always just ignore those recommendations (even though the algorithm will think that we matched them).

4 Liquidity Maximization

In this section we study the online maximization of the different notions of liquidity: number of trades, aggregate price of cleared sell bids, and aggregate price of cleared buy bids.

4.1 Maximizing the Number of Trades

Suppose that instead of maximizing profit, our goal is to maximize the *number* of trades made, subject to the constraint that each matched pair have non-negative profit. This can directly be mapped into the incomplete interval graph edge-matching problem by including an edge for every pair of buy and sell bids that are allowed to be matched together. So, the greedy algorithm achieves competitive ratio of 2, which is optimal for a deterministic algorithm, as we prove in Section 6.3.

However, if the online algorithm can subsidize matches (match a pair of buy and sell bids of negative profit if it has already made enough money to pay for them) then we can use Algorithm 2, and do as well as the optimal solution in hindsight that is not allowed subsidization. Specifically, when Algorithm 2 adds a pair $\{b, s\}$ to W, we match b and s together, subsidizing if necessary. We know that we always have enough money to pay for the subsidized bids because of the property of Algorithm 2 that its set W always has a perfect matching. We are guaranteed to do as well as the best offline algorithm which is not allowed to subsidize, because the offline solution is a matching in the incomplete interval graph.

4.2 Maximizing Buy or Sell Volume

A different important notion of liquidity is the aggregate *size* of the trades.

Definition 4 *Given a matching M the* buy-volume *is* $\sum_{(b,s)\in M} p(b)$. *The* sell-volume *is* $\sum_{(b,s)\in M} p(s)$.

If we wish to maximize buy volume without subsidization, we can use an algorithm based on the greedy profit algorithm.

Algorithm 3 *Choose a buy price threshold θ at random. Specifically, for all $x \in [p_{min}, p_{max}]$, let*

$$\begin{aligned}\Pr[\theta \leq x] &= \frac{\ln(x)+1}{\ln(p_{max}/p_{min})+1} \text{ and let} \\ \Pr[\theta = 1] &= \frac{\ln(p_{min})+1}{\ln(p_{max}/p_{min})+1}.\end{aligned}$$

When a buy bid b is introduced, if $p(b) \geq \theta$, and there exists an untaken, unexpired sell bid that can be matched without subsidy, match them. When a sell bid s is introduced, if there exists an untaken, unexpired buy bid b such that $p(b) \geq \theta$ and the bids can be matched without subsidy, match them.

This algorithm achieves a competitive ratio of $2(\ln(p_{max}/p_{min})+1)$. The proof follows that of Lemma 1. If the online algorithm is allowed to use subsidization, then we can use Algorithm 2 as follows.

Algorithm 4 *Choose a buy price threshold θ at random according to the distribution in Algorithm 3. Convert the online problem into an incomplete interval graph. For each bid b, insert a vertex with an interval $[t_i(b), t_f(b)]$. If a buy bid b and a sell bid s can be matched without subsidy, and $p(b) \geq \theta$, add an edge between their respective vertices.*

Run Algorithm 2 on the constructed graph. If Algorithm 2 chooses a buy bid b and a sell bid s, match them. (If $p(b) < p(s)$, then this match involves a subsidy.)

This achieves a competitive ratio of $\ln(p_{max}/p_{min})+1$ with respect to the offline algorithm which does not use subsidy. This is the best ratio that can be achieved (the proof is by threat-based analysis similar to that in Section 6.1).

Maximizing sell volume is analogous to maximizing buy volume. The best competitive ratio we know without using subsidy is $2(\ln(p_{max}/p_{min})+1)$. The best achievable with subsidy against an offline algorithm not allowed to use subsidy is $\ln(p_{max}/p_{min})+1$.

5 Maximizing Social Welfare

Maximizing social welfare means maximizing the sum of the valuations of the people who are left with an item, that is, matched buyers and unmatched sellers. If B' is the set of buy bids that were matched, and S' is the set of sell bids that were unmatched, then the term which we wish to maximize is:

$$\sum_{b\in B'} p(b) + \sum_{s\in S'} p(s)$$

Equivalently, if M is our matching, and S is the set of *all* sell bids, then what we wish to maximize is:

$$\sum_{(b,s)\in M} (p(b) - p(s)) + \sum_{s\in S} p(s)$$

Note that the second term cannot be affected by the algorithm (because we assume that the bids are predetermined). Furthermore, adding a constant to the offline and online algorithms' objective values can only improve the competitive ratio. Therefore, Corollary 1 immediately implies we can achieve a competitive ratio of $\ln(p_{max} - p_{min}) + 1$. However, we can in fact do quite a bit better, as shown in the following theorem.

Theorem 3 *There exists an online algorithm for maximizing social welfare that achieves a competitive ratio r that is the fixed point of the equation:*

$$r \;=\; \ln \frac{p_{max}}{r p_{min}}$$

and furthermore, this is also a lower bound. Note that this competitive ratio is at most $\ln(p_{max}/p_{min})$.

We prove the lower bound in Section 6. We describe here the basic idea of the algorithm but defer the full description and analysis to Appendix D.

The basic idea is to follow the approach used by Lavi and Nisan [15] for the case of auctions, and to select a price threshold at random according to a specific distribution. All trades occur at this price: if a buy bid is below the price, it is ignored, and if a sell bid is above the price, it is ignored. Unlike the case of auctions, however, multiple sell bids mean that we also have to decide which unignored sell bids to match. As we will show, one can achieve the optimal competitive ratio by using our Algorithm 2 together with an appropriate probability distribution on the threshold. In fact, we derive the results more generally so that they apply to suboptimal online matching algorithms as well, such as our Greedy algorithm. This is useful because, as we show in Section 8.1, the Greedy algorithm is incentive-compatible. Thus we can achieve incentive-compatibility while remaining within a factor of 2 of the optimal competitive ratio. For a full analysis, see Appendix D.

6 Lower Bounds on Competitive Ratios

In this section we establish that our analysis is tight for these algorithms. Specifically we show that no algorithm can achieve a competitive ratio lower than $\ln(p_{max} - p_{min}) + 1$ for the profit maximization problem, no algorithm can achieve competitive ratio lower than the fixed point of $r = \ln \frac{p_{max}}{r p_{min}}$ for social welfare, and no *deterministic* algorithm can achieve a competitive ratio better than 2 for the trade volume maximization problem without subsidization. We also show that no randomized algorithm can achieve a competitive ratio better than 4/3 for the trade volume maximization problem without subsidization, though we believe this is a loose bound. Furthermore, on this problem it is impossible to achieve a competitive ratio better than 3/2 without taking into consideration the expiration times of the bids. Also, we prove that our greedy profit-maximizing algorithm does not achieve a competitive ratio better than $2(\ln(p_{max} - p_{min}) + 1)$ (that is, our analysis of this algorithm is tight).

6.1 A Threat-Based Lower Bound for Profit Maximization

In this analysis, we prove a lower bound for the competitive ratio of any online algorithm by looking at a specific set of temporal bidding problems. We prove that even if the algorithm knows that the problem is in this set, it cannot achieve a competitive ratio better than $\ln(p_{max} - p_{min}) + 1$. This is very similar to the analysis of the continuous version of the one-way trading problem in [7].

For this analysis, assume $p_{min} = 0$. Consider the situation where you have a sell bid at 0 that lasts until the end. First, there is a buy bid at a, and then a continuous stream of increasing buy bids with each new one being introduced after the previous one expired. The last bid occurs at some value y. Define $D(x)$ to be the probability that the sell bid has not been matched before the bid of x dollars expires. Since it is possible there is only one buy bid, if one wanted to achieve a competitive ratio of r, then $D(a) \leq 1 - \frac{1}{r}$. Also, define $Y(y)$ to be the expected profit of the algorithm. The ratio r is achieved if for all $x \in [a, p_{max}]$, $Y(x) \geq x/r$. Observe that $Y'(x) = -D'(x)x$, because $-D'(x)$ is the probability density at x.

Observe that one wants to use no more probability mass on a bid than absolutely necessary to achieve the competitive ratio of r, because that probability mass is better used on later bids if they exist. Therefore, for an optimal algorithm, $D(a) = 1 - \frac{1}{r}$, and $Y(x) = x/r$. Taking the derivative of the latter, $Y'(x) = 1/r$. Substituting, $1/r = -D'(x)x$. Manipulating, $D'(x) = -\frac{1}{rx}$. Integrating:

$$
\begin{aligned}
D(y) &= D(a) + \int_a^y D'(x)dx \\
&= 1 - \frac{1}{r} - \frac{1}{r}\ln\left|\frac{y}{a}\right|
\end{aligned}
$$

For the optimal case, we want to just run out of probability mass as y approaches p_{max}. Therefore:

$$
D(p_{max}) = 1 - \frac{1}{r} - \frac{1}{r}\ln\left|\frac{p_{max}}{a}\right| = 0
$$

$$
r = \ln\frac{p_{max}}{a} + 1
$$

Thus, setting a to 1, and shifting back by p_{min}, one gets a lower bound of $\ln(p_{max} - p_{min}) + 1$.

6.2 Greedy Profit Maximization

The following scenario shows that our analysis of the greedy profit algorithm is tight. Imagine that a buy bid for \$2 is introduced at 1:00 (and is good forever), and a sell bid for \$1 is introduced at 1:01 (that is also good forever). At 2:00, another buy bid for \$2 is introduced, which expires at 3:00. At 3:01, another sell-bid for \$1 is introduced. In this scenario, the optimal offline algorithm achieves a profit of \$2 (matching the first buy bid to the last sell bid, and vice versa).

With a probability of $1 - \frac{1}{\ln(p_{max}-p_{min})+1}$, $\theta > 1$, and the greedy algorithm ignores all of the bids. Otherwise ($\theta = 1$), the greedy algorithm matches the first two bids for a profit of 1 and then cannot match the second two. Therefore, the expected reward is $\frac{1}{\ln(p_{max}-p_{min})+1}$ compared to an optimal of 2.

6.3 Lower Bound for Maximizing the Number of Trades

Here we establish that no deterministic algorithm can achieve a competitive ratio lower than 2 for maximizing the number of trades without subsidization. Also, no randomized algorithm can achieve a competitive ratio lower than 4/3. Furthermore, without observing the expiration times, it is impossible to achieve a competitive ratio less than 3/2.

Imagine a sell bid s^* for \$1, is introduced at 1:00 and will expire at 2:00. At 1:01, a buy bid b is introduced for \$3, and will expire at 3:00. At 1:02, a buy bid b' is introduced for \$2, and will expire at 4:00.

There are two possible sell bids that can be introduced: either a sell bid s for \$2.5 at 2:30, or a sell bid s' for \$1.5 at 3:30. Observe that b can match s, and b' can match s'. So if s is to appear, s^* should match b', and if s' is to appear, s^* should match b. But when s^* expires, the online algorithm does not know which one of s and s' will appear. So while a randomized algorithm can guess the correct match to make with a probability of 1/2, the deterministic algorithm must make a decision of which to take before the adversary chooses the example, and so it will choose the wrong match.

Without observing the expiration times, it is impossible achieve a competitive ratio below 3/2. Imagine that a sell bid is introduced at 9:00 AM for one dollar, and a buy bid is introduced at 9:00 AM for 2 dollars. Should these be matched? Suppose an algorithm A matches them with probability p by 10:00 AM. If $p \leq 2/3$, then they expire at 10:00 AM and no other bids are seen and the expected reward is less than 2/3 when it could have been 1. If $p > 2/3$, then the bids last all day. Moreover, there is another buy bid from 1:00 PM to 2:00 PM for two dollars and another sell bid from 3:00 PM to 4:00 PM for one dollar. If the first two bids were unmatched, then there is the possibility of a profit of 2 whereas the expected reward of the algorithm is $(1)p + 2(1-p) \leq 4/3$. Therefore, no algorithm has a competitive ratio better than 3/2 on these two examples.

6.4 Lower Bound for Maximizing Social Welfare

Consider the derivation of the price threshold distribution given in Appendix D. When $\alpha = 1$, then $N^*(T) = N^{alg}(T)$ (see Appendix D for definitions). Suppose that we observe the sequence described in Section 6.1. For this sequence, the reward of the algorithm exactly equals $R_T(s)$, where s is the single sell bid. Therefore, the price threshold distribution is optimal for this sequence, and no algorithm[8] can achieve a competitive ratio lower than the fixed point of the equation:

$$r = \ln \frac{p_{max}}{r p_{min}}$$

7 Combining Algorithms Online

The algorithm of Corollary 1 begins by picking a threshold θ from some distribution. It turns out that under certain reasonable assumptions described below, we can use standard online learning results to do nearly as well as the *best* value of θ picked in hindsight. This does not violate the optimality of the original algorithm: it could be that *all* thresholds perform a log factor worse than OPT. However, one can imagine that in certain natural settings, the best strategy *would* be to pick some fixed threshold, and in these cases, the modified strategy would be within a $(1+\epsilon)$ factor of optimal.

The basic idea of this approach is to probabilistically combine all the fixed-threshold strategies using the Randomized Weighted Majority (also called Hedge) algorithm of [17, 8], as adapted by [4] for the case of experts with internal state. In particular, at any point in time, for each threshold

[8] The competitive ratio of Lavi and Nisan [15] is lower than this because they compare their algorithm to the offline Vickrey auction, not the optimal offline algorithm.

θ, we can calculate how well we *would* have done had we used this threshold since the beginning of time as a pair $(\mathsf{profit}_\theta, \mathsf{state}_\theta)$, where profit_θ is the profit achieved so far, and state_θ is the set of its current outstanding bids. For example, we might find that had we used a threshold of 5, we would have made \$10 and currently have live bids $\{b_1, b_2, s_1\}$. On the other hand, had we used a threshold of 1, we would have made \$14 but currently have no live unmatched bids.

In order to apply the approach of [4], we need to be able to view the states as points in a metric space of some bounded diameter D. That is, we need to imagine our algorithm can move from $\mathsf{state}_{\theta_1}$ to $\mathsf{state}_{\theta_2}$ at some cost $d(\mathsf{state}_{\theta_1}, \mathsf{state}_{\theta_2}) \leq D$. To do this, we make the following assumption:[9]

Assumption: There is some a priori upper bound B on the number of bids alive at any one time.

We now claim that under this assumption we can view the states as belonging to a metric space of diameter $D \leq Bp_{max}$. Specifically, suppose the overall "master" algorithm tells us to switch from threshold θ_1 to θ_2. We then conservatively only match buy/sell pairs if they are *both* in $\mathsf{state}_{\theta_2}$ (and have profit at least θ_2). This guarantees that at worst we make B fewer matches than had we been in $\mathsf{state}_{\theta_2}$, and therefore, at worst our profit is Bp_{max} less.

We can now plug in the Randomized Weighted-Majority (Hedge) algorithm to get the following theorem.

Theorem 4 *Under the assumption above, for any $\epsilon > 0$ (ϵ is given to the algorithm), we can achieve an expected gain at least*

$$\max_\theta \left[(1-\epsilon)\mathsf{profit}_\theta - \frac{2Bp_{max}}{\epsilon} \log N \right],$$

where N is the number of different thresholds.

The proof follows the exact same lines as [4] (which builds on [17, 8]). Technically, the analysis of [4] is given in terms of losses rather than gains (the goal is to have an expected loss only slightly worse than the loss of the best expert). For completeness, we give the proof of Theorem 4 from first principles in Appendix C.

8 Strategic Agents

Up until this point of the paper, we have assumed that agents bid truthfully. Each agent has one time interval during which it wants to trade and its valuation stays constant in that interval. In this section we discuss bidding strategies that are more sophisticated in either of two ways. First, an agent may bid strategically, that is, it might submit a bid that differs from its true valuation. To address this, we present an incentive-compatible algorithm for social welfare maximization. Second, we consider the case of agents whose bid values are allowed to change over time. We show that the Greedy profit maximization algorithm maintains its competitive ratio even in this more difficult setting.

[9] This assumption can be weakened a bit and still allow the results to go through.

8.1 An Incentive-Compatible Mechanism for Maximizing Social Welfare

We design an algorithm that is incentive-compatible in dominant strategies. That is, each agent's best strategy is to bid truthfully regardless of how others bid. Bidding truthfully means revealing one's true valuation.

We assume that each bidder wants to buy or sell exactly one item. We also assume that there is one window of time for each agent during which the agent's valuation is valid (outside this window, a seller's valuation is infinite, and a buyer's valuation is negative infinity). For example, the window for a seller might begin when the seller acquires the item, and ends when the seller no longer has storage space. A buyer's window might begin when the buyer acquires the space to store the item. We also assume that within this window, the valuation of the item to the buyer or seller is constant.

Let us first consider a simpler setting with no temporal aspects. That is, everyone makes bids upfront to buy or sell the item. In our mechanism, we first decide a price T at which all trades will occur, if at all. We then examine the bids and only consider sell bids below T and buy bids above T. If there are more sell bids than buy bids, then for each buy bid, we select a random seller with a valid bid to match it to (this is different from selecting a random sell bid if some sellers submitted multiple bids). We do the reverse if there are more buy bids than sell bids.

Now, there are three factors that affect the expected reward of an agent:

1. The agent's valuation for the item. This is fixed before the mechanism begins, and cannot be modified by changing the strategy.
2. The price of a possible exchange. This is set upfront, and cannot be modified by changing the strategy.
3. The probability that the agent is involved in an exchange.

The last factor is the only one that the agent has control over. For a buyer, if its valuation is below the price of an exchange, then the buyer does not wish to exchange, and can submit one bid at its true valuation, and it will have zero probability of being matched.

The more interesting case is when the agent's valuation is above the trading price. Now, the agent wishes to maximize the probability of getting to trade. If the other agents submit at least one fewer buy bid above T than the number of sell bids below T, then the agent is guaranteed a trade. On the other hand, if there are as many or more buy bids than sell bids, then the agent would not be *guaranteed* to purchase an item. Regardless of how many bids the agent submits, it will only have as much of a chance as any other buyer. So, the agent loses nothing by submitting a single bid (at its true valuation).

A similar argument applies to sellers. Therefore, all of the agents are motivated to bid truthfully.[10] An important aspect of this is that the trading price does not depend on the bids.

Now, let us apply this technique to the online problem.

Algorithm 5 *Select a price threshold T at random before seeing any bids. Reject any sell bids above T, and any buy bids below T. If at any time, there are more remaining unmatched buy bids than sell bids, then match a buy bid to a random seller (not a random sell bid). If there are more remaining unmatched sell bids than buy bids, then match a sell bid to a random buyer (not*

[10] This scheme is vulnerable to coalitions: if for a buyer b and a seller s, $p(b) > p(s) > T$, then b has a motivation to pay s to decrease his offered price to below T, such that b can receive the item. If we assume there are no side payments, such problems do not occur.

a random buy bid). If there are an equal number of remaining unmatched buy and sell bids, match all of them.

Under this algorithm, each agent is motivated to bid its true price. Furthermore, no agent can benefit from submitting multiple bids or from only bidding for a fraction of its window.

This algorithm is an instance of our Greedy algorithm, because it always matches two bids when it can. For each pair in the optimal matching (where the buy bid is above the price threshold and sell bid is below the threshold), the above algorithm will select at least one bid. If the first bid introduced is not there when the second bid arrives, then it was matched. If the first bid still remains when the second bid arrives, then (since either all buy bids or all sell bids are selected), one bid or the other bid is selected from the pair. This implies that our competitive ratio for the Greedy social welfare maximizing algorithm applies.

8.2 Incentive Compatibility for Maximizing Liquidity or Profit

We do not know if there exists an incentive-compatible algorithm that achieves a good competitive ratio for liqudity or profit. Such an algorithm would have to be drastically different from the algorithm described above. Imagine one sets a price of all transactions before any bids are seen (perhaps selecting the price from a distribution). Now suppose that only one buy and sell bid, separated by a dollar, are submitted. Then if and only if the price threshold is between the buy and sell bid will any profit be made. Since the buy bid can be anywhere between p_{max} and $p_{min}+1$, in the worst case selecting a good price is achieved with probability at most $(p_{max}-(p_{min}+1))^{-1}$. So, no algorithm of this form can achieve a competitive ratio better than this for either liquidity or profit maximization.

9 When Bids Depend on the Algorithm's Actions: Stronger Notions of Competitive Ratio

So far in this paper we have used the usual framework for competitive analysis, in which we compare the algorithm's performance to the optimal offline algorithm on the *same request sequence*. However, if the event stream (bids) that the algorithm sees may depend on the algorithm's past choices (e.g., think of a bidder who increases his buy bid if he hasn't been matched after a certain amount of time) then this notion does not reflect the ratio of our performance to the best we could have done. In other words, if we had behaved differently we might have made a lot more profit because we would have seen a different sequence of bids.[11]

If one allows arbitrary dependence of future bids on the algorithm's actions, any online algorithm will fail miserably under this stronger notion of competitive ratio. To see this, imagine that in the beginning of the sequence, there are several buy bids, and one sell bid. One of the buy bids is special because if it is matched, then there will be many more bids submitted, and if it is not matched, there will be no more bids submitted. The optimal offline algorithm will match this special buy bid and end up making a huge profit, but the online algorithm will not know which one of the buy bids is special.

However, surprisingly there are some natural dependencies that can be handled. Suppose that an agent is interested in buying or selling a single unit, but instead of giving one bid for it, the

[11] This is a different (though related) issue than the notion of an oblivous versus adaptive adversary. In the adaptive-adversary model, the bid sequence may depend on the algorithm's actions, but we still compare our performance to the optimal in hindsight on the *same* request sequence.

agent gives several bids over time (at most one bid active at a time) until the agent buys or sells the unit. For example, a buyer might put in a bid for \$100, and then if that bid has not been matched by time t_1, replace that with a bid for \$200. So, if the online algorithm matches the \$100 bid, then it never sees the \$200 one. A natural example of this is plane tickets. Mary wants to go on a flight, but does not want to pay what the flight is worth to her. So she first puts in a low bid, and then gradually increases that bid over time, removing the bid if she does not receive a flight before she has to commit to the trip. Similarly, the airlines might also vary the price of a single seat on a plane over time.

We show that a competitive ratio of $2(\ln(\Delta p) + 1)$ for profit maximization can still be achieved in this setting by using the Greedy algorithm described earlier in this paper.

The nature of the online graph is different in this model. We can still represent the bids as vertices and the matches as edges. If there is a single agent interested in a single unit, only one vertex will be used to represent all of that agent's bids (which differ based on when they are active and on price). This will guarantee that the agent does not buy or sell more than one item. If there is, at some time, a buy bid made by agent b for $pr(b)$ and a sell bid made by agent s for $pr(s)$ where $pr(b) \geq pr(s)$, then an edge from b to s is inserted into the graph. Because a single vertex can represent multiple bids at different prices, there may be different edges between two vertices at different times. Also, two vertices may enter, and at a later time an edge may appear between them.

The algorithm remains the same: we choose a random profit threshold T according to the same distribution as before and only include edges representing trades that achieve profit level at least T. The algorithm selects from the edges greedily as they appear. This is still guaranteed to get a matching at least half as large as optimal. To see this, observe that at least one vertex from every edge in the optimal matching is selected by the algorithm. When an edge arrives, either one of the vertices has already been taken, or both are free. If both are free, they will be matched to each other, or at least one of them will be matched to another edge that happened to be introduced simultaneously. Thus, after an edge is introduced, at least one of its vertices has been selected.

This technique also works for maximizing liquidity. However, maximizing social welfare in this setting is a bit trickier. One problem is that it is not totally clear how social welfare should be defined when the valuations of the participants change with time. If only the *buyers'* valuations change with time, then one natural definition of social welfare for a matching M is

$$\sum_{b \in B'} p_M(b) + \sum_{s \in S'} p(s),$$

where B' is the set of matched buy bids, S' is the set of unmatched sell bids, and $p_M(b)$ is the value of b's bid at the time the match was made. In that case, the Greedy algorithm still achieves its near-optimal competitive ratio. This is perhaps surprising since the online algorithm is at an extra disadvantage, in that even if it magically knew which buyers and sellers to match, it still would need to decide exactly *when* to make those matches in order to maximize social welfare.

10 Conclusions and Open Questions

In this paper, we derived bounds on competitive ratios for a number of basic online market clearing problems. These include the objectives of maximizing profit, maximizing liquidity according to several natural measures, and maximizing social welfare. Using the abstraction of online incomplete interval graphs, we obtain algorithms that achieve best-possible competitive ratios for most of these problems, and for the rest we are within a factor of two of the best that is achievable online. For the objective of maximizing number of trades, we demonstrate that by allowing the online algorithm to subsidize matches with profits from previous trades, we can perform as well as the optimal offline algorithm without subsidy. Thus the ability to subsidize trades is at least as powerful as knowing the future, in this context.

It is a somewhat unfortunate fact about competitive analysis that no algorithm can guarantee better than a logarithmic ratio for maximizing profit. This lower bound occurs because we are comparing the performance of our online algorithm to an incredible goal: the best performance of any arbitrary algorithm in hindsight. However, if we restrict our comparison class to a fixed, finite set of algorithms (such as the class of all fixed-threshold algorithms), and if we assume the data is not too "bursty", then we can use online learning results to achieve a $1+\epsilon$ ratio, as we demonstrate in Section 7.

We also consider the issue of incentive compatibility, and show that our Greedy algorithm is incentive-compatible for the goal of maximizing social welfare. Thus, losing only a factor of 2 in competitive ratio we can achieve this desirable property. In fact, we show the Greedy algorithm is quite robust in that it maintains its competitive ratio even if bidders are allowed to modify their bids over time, making the task even harder on the online algorithm.

Open problems

As mentioned in Section 8.2, we do not know any incentive-compatible algorithm that achieves good competitive ratio for maximizing liquidity or profit. We know that such an algorithm would need to have a different character than the ones in this paper.

There is also a gap between our best upper bound (2) and our best lower bound (3/2) for algorithms for the incomplete interval graph vertex-selection problem when *expiration times are unknown*. Algorithm 2 achieves an optimal matching but it needs to know the expiration times in advance. Related to this question, we do not know if the Greedy algorithm can be improved upon for the setting of Section 9 in which a bidder may submit a bid whose value varies with time.

There has also been recent interest in offline clearing algorithms for *combinatorial* exchanges where a bid can concern multiple distinguishable items (possibly multiple units of each) [18, 19, 21]. A bid could state, for example, "I want to buy 20 IBM, buy 50 DELL, sell 700 HP, and get paid \$500". As we transformed the online clearing problem to an online matching problem in an incomplete interval graph, the online problem of maximizing profit or liquidity in a combinatorial exchange can be transformed to the problem of finding a matching on an incomplete interval *hypergraph* online. While we can show with a simple example that a maximum matching cannot be achieved online in the hypergraph problem, it might be possible to achieve a matching with an expected size no less than half of the maximum one.

One direction of future research is to extend these results to settings where there is a market maker who can carry inventory of the commodity (sell short and long) rather than simply deciding which bids to match.

Acknowledgements

The authors would like to thank Ke Yang for a number of helpful suggestions. This material is based upon work supported under NSF grants CCR-9732705, CCR-0085982, CAREER Award IRI-9703122, IIS-9800994, ITR IIS-0081246, ITR IIS-0121678, and an NSF Graduate Research

Fellowship. Any opinion, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Baruch Awerbuch, Yair Bartal, and Amos Fiat. Distributed paging for general networks. In *Proc. 7th Symp. on Discrete Algorithms*, pages 574–583, 1996.

[2] A. Bagchi, A. Chaudhary, R. Garg, M. Goodrich, and V. Kumar. Seller-focused algorithms for online auctioning. In *Proceedings of WADS 2001*, pages 135–147, 2001.

[3] C. Berge. Two theorems in graph theory. In *Proceedings of the National Academy of Sciences*, volume 43, pages 842–844, 1957.

[4] A. Blum and C. Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, April 2000.

[5] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[6] Ian Domowitz. Automating the continuous double auction in practice: Automated trade execution systems in financial markets. In Daniel Friedman and John Rust, editors, *The Double Auction Market*, volume 14 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 27–60. Addison-Wesley, 1993.

[7] Ran El-Yaniv, Amos Fiat, Richard M. Karp, and G. Turpin. Competitive analysis of financial games. In *Proc. 33rd Symp. Foundations of Computer Science*, pages 327–333, 1992.

[8] Yoav Freund and Robert Schapire. Game theory, on-line prediction and boosting. In *Proc. 9th Conf. on Computational Learning Theory*, pages 325–332, 1996.

[9] Andrew Goldberg, J Hartline, and A Wright. Competitive auctions and multiple digital goods. Technical report, InterTrust 00-01, 2000.

[10] Andrew Goldberg, J Hartline, and A Wright. Competitive auctions and digital goods. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Washington, DC, 2001.

[11] S. Goldman, J. Parwatikar, and S. Suri. On-line scheduling with hard deadlines. *Journal of Algorithms*, 34:370–389, 2000.

[12] Ming-Yang Kao and Stephen R. Tate. On-line difference maximization. *SIAM Journal of Discrete Mathematics*, 12(1):78–90, 1999.

[13] Richard M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd Symp. Theory of Computing*, pages 352–358, Baltimore, Maryland, 1990.

[14] D. Kozen. *The Design and Analysis of Algorithms*. Springer-Verlag, New York, 1991.

[15] Ron Lavi and Noam Nisan. Competitive analysis of incentive compatible on-line auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 233–241, Minneapolis, MN, 2000.

[16] R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proc. 5th Symp. on Discrete Algorithms*, pages 302–311, 1994.

[17] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[18] Tuomas Sandholm. eMediator: A next generation electronic commerce server. *Computational Intelligence*, 2002. Special issue on Agent Technology for Electronic Commerce. To appear. Early versions appeared in the Conference on Autonomous Agents (AGENTS-00), pp. 73–96, 2000; AAAI-99 Workshop on AI in Electronic Commerce, Orlando, FL, pp. 46–55, July 1999; and as a Washington University, St. Louis, Dept. of Computer Science technical report WU-CS-99-02, Jan. 1999.

[19] Tuomas Sandholm and Subhash Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 90–97, Austin, TX, 2000.

[20] Tuomas Sandholm and Subhash Suri. Optimal clearing of supply/demand curves. In *AAAI-02 workshop on Agent-Based Technologies for B2B Electronic Commerce*, Edmonton, Canada, 2002.

[21] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Bologna, Italy, July 2002. Early version appeared at the AGENTS-01 Workshop on Agent-Based Approaches to B2B, pp. 35–41, Montreal, Canada, May 2001.

[22] Peter R Wurman, Michael P Wellman, and William E Walsh. The Michigan Internet Auction-Bot: A configurable auction server for human and software agents. In *Proceedings of the Second International Conference on Autonomous Agents (AGENTS)*, pages 301–308, Minneapolis/St. Paul, MN, May 1998.

[23] P.R. Wurman, W.E. Walsh, and M.P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24(1):17–27, 1998.

Appendices

A Lemmas on Matchings

In this section, we prove some lemmas required for the proof of the main theorem in Appendix B. We will use some standard facts and definitions from graph theory; for a review, see [14]. Paths will be represented as sets of edges, although sometimes it will be easier to write them as a sequence of vertices. $V(P)$ is the set of vertices of a path P. We will also use some well known lemmas:

Lemma 2 *Given $W \subset V$, a perfect matching M on W, and $v, v' \in V - W$, if there exists some augmenting path P with respect to M from v to v', then $M \oplus P$ is a perfect matching on $W \cup \{v, v'\}$.*

Lemma 3 *The symmetric difference of two matchings M and M' consists of alternating paths and alternating cycles with respect to M.*

Lemma 4 *A vertex $v \in V$ is the endpoint of an alternating path in the symmetric difference of two matchings M and M' if and only if it is in $V(M) \oplus V(M')$.*

Proof: Suppose a vertex is in $V(M) \oplus V(M')$. Then it has an incident edge in either M or M', but not both. Hence, it can only have one incident edge in $M \oplus M'$, making it an endpoint of an alternating path.

Suppose a vertex is at the endpoint of a path in $M \oplus M'$. Then it has one edge in $M \oplus M'$. Observe that if it has an odd number of edges incident to it in $M \oplus M'$, the number of edges in M to it plus the number of edges in M' to it is also odd. But since the latter is bounded by zero and two, it is one. Therefore, the vertex cannot be in both $V(M)$ and $V(M')$. ∎

Augmenting paths capture some of the local properties of matchings. But they do not capture how to remove elements from a matching, or how to replace an element. Since these are important concepts in the online setting, we introduce some other types of paths. These paths have properties very similar to those of augmenting paths.

Definition 5 *An **abridging path** with respect to a matching M is an alternating path whose first and last edges are in M. A **replacement path** with respect to M is an alternating path whose first edge is in M, and whose last endpoint is not in $V(M)$.*

Lemma 5 *Given $v, v' \in W$, and a perfect matching M on W, if there exists some abridging path P with respect to M from v to v', then $M \oplus P$ is a perfect matching on $W - \{v, v'\}$.*

The proof is similar to the proof of Lemma 2.

Lemma 6 *Given $v \in W$ and $v' \in V - W$, and a perfect matching M on W, if there exists some replacement path P with respect to M from v to v', then $M \oplus P$ is a perfect matching on $W \cup \{v'\} - \{v\}$.*

The proof is similar to the proof of Lemma 2.

Lemma 7 *Suppose W and W' are subsets of V, and there exists a perfect matching on W and a perfect matching on W', then there exists a partition of $W \oplus W'$ into sets of size two:*

$$\{\{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_k, b_k\}\}$$

such that for all $1 \leq i \leq k$, there exists a perfect matching on $W \oplus \{a_i, b_i\}$.

Proof: Define M to be a perfect matching on W and M' to be a perfect matching on W'. Then $M \oplus M'$ consists of a set of alternating paths and cycles. Here, we are only concerned with the paths. Since these paths only begin and end at points in $W \oplus W'$, we can partition $W \oplus W'$ where each set is the set of endpoints of a path in $M \oplus M'$. Consider a set in this partition $\{a, b\}$, where P is the path between the two elements of the set. There are three possibilities:

1. Both vertices are in $W' - W$. Then P is an augmenting path, and $M \oplus P$ is a perfect matching on $W \cup \{a, b\} = W \oplus \{a, b\}$.
2. One vertex is in $W - W'$ and one vertex in $W' - W$. In this case, the path P is a replacement path. Without loss of generality, assume $a \in W - W'$. Then $M \oplus P$ is a perfect matching on $W \cup \{b\} - \{a\} = W \oplus \{a, b\}$.

3. Both vertices are in $W - W'$. The first and last edges are not in M', because a and b are not in W'. Also, the first and last edges are in $M \oplus M' \subseteq M \cup M'$. Therefore, the first and last edges are in M, and P is an abridging path. $M \oplus P$ is a perfect matching on $W - \{a, b\} = W \oplus \{a, b\}$.

∎

Corollary 2 *If $W \subseteq V$ has a perfect matching, but it is not one of the largest sets that has a perfect matching, then there exists $v, v' \in V - W$ such that $W \cup \{v, v'\}$ has a perfect matching.*

B Proof of Theorem 2 (Main Theorem)

Assume W is the set of selected vertices, then we define H_1,H_2, and H_3 as follows:

H_1: For any expired, unmatched vertex w, there does not exist any untaken vertex w' such that there is a perfect matching on $W \cup \{w, w'\}$.

H_2: For any matched vertex w, there does not exist an untaken vertex w' such that there is a perfect matching on $W \cup \{w'\} - \{w\}$ and $t_f(w) > t_f(w')$.

H_3: For any two unexpired vertices $w, w' \in W$, there exists no perfect matching on $W - \{w, w'\}$.

Let H denote the conjunction of the three invariants H_1, H_2, and H_3. We will prove that if these invariants of the algorithm hold before an event occurs, they will hold after the event. The possible events that can occur are a vertex being introduced, expiring without being matched, or expiring and being added.

Lemma 8 *If H holds before a vertex w' is introduced, H holds after the event.*

Proof: We prove this for all three parts of H.

H_1: Consider w to be an expired, unmatched vertex. Here we need to prove that there exists no perfect matching on $W \cup \{w, w'\}$. We will prove this by contradiction. Suppose that M is a perfect matching on $W \cup \{w, w'\}$. Define v such that $(w', v) \in M$. Let us define $M' = M - \{(w', v)\}$. Then, M' is a perfect matching on $W - \{v\} + \{w\}$. Since w has expired and v has not expired, $t_f(w) < t_f(v)$. This contradicts the inductive hypothesis H_2.

H_2: Suppose that $w \in W$ and $t_f(w) > t_f(w')$. We need to prove that there does not exist a perfect matching for $W \cup \{w'\} - \{w\}$ by contradiction, assuming there exists a perfect matching M.

Define v such that $(w', v) \in M$. Observe that $t_f(v) > t_i(w')$, so v has not expired. Neither has w, since $t_f(w) > t_f(w') > t_i(w')$. However, $M - (w', v)$ is a perfect matching on $W - \{w, v\}$. This is a contradiction of H_3.

H_3: This cannot have any effect.

∎

Lemma 9 *If H holds before a vertex w expires without being matched, H holds after the event.*

Proof: We prove this for all three parts of H.

H_1: Suppose w' is an untaken vertex. We need to prove that there does not exist a perfect matching on $W \cup \{w, w'\}$.

(a) Assume w' expired. By H_1, there there did not exist a perfect matching on $W \cup \{w, w'\}$.
(b) If there existed an unexpired, unmatched vertex w' such that $W \cup \{w, w'\}$ had a perfect matching, then w would have been matched.

H_2,H_3: This cannot have any effect.

∎

Lemma 10 *If H holds before a vertex v expires and is added with v', H holds after the event.*

Proof: We prove this for all three parts of H.

H_1: Suppose w is an expired, unmatched vertex, and w' is an untaken vertex. We need to prove that there does not exist a perfect matching on $W \cup \{v, v', w, w'\}$. We can prove this by contradiction.

Observe that there exists a perfect matching on W. If there existed a perfect matching on $W \cup \{v, v', w, w'\}$, then by Lemma 7 one of the following conditions holds:

(a) There exists a perfect matching on $W \cup \{v', w\}$. This would contradict H_1.
(b) There exists a perfect matching on $W \cup \{v, w\}$. This would contradict H_1.
(c) There exists a perfect matching on $W \cup \{w, w'\}$. This would contradict H_1.

H_2: Consider an arbitrary vertex w in $W \cup \{v, v'\}$, and an arbitrary untaken vertex w'. Assume that $W \cup \{v, v', w'\} - \{w\}$ is a perfect matching. We must prove that $t_f(w') \geq t_f(w)$.

(a) $w = v$. Then $W \cup \{v, v', w'\} - \{w\} = W \cup \{v', w'\}$. So by H_1, w' has not expired. Then $t_f(w') \geq t_f(w)$, because w is just expiring.
(b) $w = v'$. Then $W \cup \{v, v', w'\} - \{w\} = W \cup \{v, w'\}$. So by H_1, w' has not expired. Thus if $t_f(w) > t_f(w')$, then w' would have been added instead of w.
(c) $w \in W$. Then by Lemma 7 applied to W and $W \cup \{v, v', w'\} - \{w\}$, one of the following conditions holds:
   i. $W \cup \{v\} - \{w\}$ and $W \cup \{v', w'\}$ have perfect matchings. Then by H_1, w' has not expired, and $t_f(w) < t_f(v)$, so w has expired.
   ii. $W \cup \{v'\} - \{w\}$ and $W \cup \{v, w'\}$ have perfect matchings. Then by H_1, w' has not expired, so $t_f(w') \geq t_f(v')$. Also, $t_f(w) \leq t_f(v')$.
   iii. $W \cup \{w'\} - \{w\}$ has a perfect matching. Then by H_2, $t_f(w') \geq t_f(w)$.

H_3: A vertex v expires and is added with v'. Suppose that $w, w' \in W \cup \{v, v'\}$. Assume that $W \cup \{v, v'\} - \{w, w'\}$ has a perfect matching. We must prove that w or w' has expired. Consider three cases:

(a) $w = v$. Then w has expired.
(b) $w = v'$. Then $W \cup \{v, v'\} - \{w, w'\} = W \cup \{v\} - \{w'\}$. Thus by H_2, $t_f(v) \geq t_f(w')$, so w' has already expired(a contradiction).

(c) $w \in W$. Then by Lemma 7 applied to W and $W \cup \{v, v'\} - \{w, w'\}$, one of the following conditions holds:

i. $W \cup \{v\} - \{w\}$ has a perfect matching. By H_2, $t_f(v) \geq t_f(w)$, so w has expired.
ii. $W \cup \{v\} - \{w'\}$ has a perfect matching. By H_1, $t_f(v) \geq t_f(w')$, so w' has expired.
iii. $W - \{w, w'\}$ has a perfect matching. By H_3, either w or w' has expired.

∎

Proof (of Theorem 2): We will prove by induction that H holds at the termination of the algorithm. First, observe that H_1, H_2, and H_3 are all properties of introduced vertices, so when there are no vertices introduced, they cannot be violated. This is the base case. Inductively, by Lemmas 8, 9, 10, if these properties hold before an event, they hold afterward as well. Thus, H holds at the termination of the algorithm. Specifically, H_1 implies that there exists no $\{v, v'\} \subseteq V - W$ such that $W \cup \{v, v'\}$ has a perfect matching. By Corollary 2, W is one of the largest sets with a perfect matching. ∎

C Proof of Theorem 4 (Combining Algorithms Online)

The setting is we have N algorithms ("experts") and we can switch from one to another at cost $\leq D$. At each time step (introduction of a bid) the algorithms experience gains between 0 and g_{max}. (We use g_{max} instead of p_{max} because we will be using "p" for probabilities). At any point in time, our overall "master" algorithm will be a probability distribution over the N base algorithms. Specifically, we use the standard multiplicative-weights algorithm [17, 8] as follows:

- We begin with each base algorithm having a weight $w_i = 1$. Let $W = \sum_i w_i$. Our probability distribution over base algorithms will be $p_i = w_i/W$.
- After each event, we update weights according to the rule:

$$w_i \leftarrow w_i(1 + \hat{\epsilon})^{g_i/g_{max}},$$

  where g_i is the gain of the ith algorithm, and $\hat{\epsilon}$ is defined below. We then update our probability distribution p_i accordingly. Note: if $\vec{p}$ is the old distribution and $\vec{p}'$ is the new distribution, we can do this such that the expected movement cost is at most $\frac{D}{2}L_1(\vec{p}, \vec{p}')$ where $L_1(\vec{p}, \vec{p}')$ is the L_1 distance between the two distributions.

Theorem 5 *If we use $\hat{\epsilon} = \frac{\epsilon g_{max}}{2D}$, and under the assumption that $D \geq g_{max}$, the above algorithm produces an expected gain at least*

$$\text{OPT}(1 - \epsilon) - \frac{2D}{\epsilon} \ln N,$$

where OPT *is the gain of the best of the N base algorithms in hindsight.*

Proof: Suppose the algorithm currently has weights $w_1, \ldots, w_n$, observes a gain vector $g_1, \ldots, g_n$ and then updates the weights to $w'_1, \ldots, w'_n$, where $w'_i = w_i(1 + \hat{\epsilon})^{g_i/g_{max}}$. Let $W = \sum_i w_i$ and let $W' = \sum_i w'_i$. Let $p_i = w_i/W$ and let $p'_i = w'_i/W'$.

The expected gain of the online algorithm from this event is $\sum_i p_i g_i$, which we will call E_t if this is event t. The expected cost due to moving from probability distribution $\vec{p}$ to distribution $\vec{p}'$ is at most

$$D \sum_{i:p'_i > p_i} (p'_i - p_i) \le \frac{D}{W} \sum_{i:p'_i > p_i} (w'_i - w_i) \le \frac{D}{W}(W' - W) = D\left(\frac{W'}{W} - 1\right),$$

where the first inequality above uses the fact that $W' \ge W$, and the second inequality uses the fact that $w'_i \ge w_i$ for *all* i, not just the ones such that $p'_i > p_i$.

The next step in the analysis is to upper bound W' as a function of W. To do this, we use the fact that for $x \le 1$, $(1+\alpha)^x \le 1 + x\alpha$. So,

$$W' \le \sum_i w_i \left(1 + \frac{\hat{\epsilon} g_i}{g_{max}}\right) = W \sum_i p_i \left(1 + \frac{\hat{\epsilon} g_i}{g_{max}}\right) = W\left(1 + \frac{\hat{\epsilon}}{g_{max}} E_t\right).$$

This upper bound on W' is useful in two ways. First, we can now analyze the expected movement cost: $D(W'/W - 1) \le D(\frac{\hat{\epsilon}}{g_{max}} E_t)$. So, our total expected gain for event t is at least $E_t(1 - \frac{D\hat{\epsilon}}{g_{max}})$.

Second, we can use this to give an upper bound on the total weight W_{final} at the end of the process:

$$W_{final} \le N \prod_t \left(1 + \frac{\hat{\epsilon}}{g_{max}} E_t\right).$$

This upper bound can be compared to a lower bound

$$W_{final} \ge (1 + \hat{\epsilon})^{\mathrm{OPT}/g_{max}}$$

which is simply the final weight of the best of the N algorithms. Taking logs on the upper and lower bounds to W_{final} and using the fact that $\ln(1+x) \in [x - x^2/2, x]$ for $x \in [0,1]$, we have:

$$(\mathrm{OPT}/g_{max})(\hat{\epsilon} - \hat{\epsilon}^2/2) \le (\mathrm{OPT}/g_{max}) \ln(1+\hat{\epsilon}) \le \ln N + \sum_t \ln(1 + \frac{\hat{\epsilon}}{g_{max}} E_t) \le \ln N + \frac{\hat{\epsilon}}{g_{max}} \sum_t E_t.$$

So,

$$\sum_t E_t \ge \mathrm{OPT}(1 - \hat{\epsilon}/2) - \frac{g_{max}}{\hat{\epsilon}} \ln N.$$

Finally, using the fact that the total expected gain G_{alg} (with movement costs included) is at least $(1 - \frac{D\hat{\epsilon}}{g_{max}}) \sum_t E_t$, and using our definition $\hat{\epsilon} = \epsilon g_{max}/(2D)$, we have:

$$G_{alg} \ge \mathrm{OPT}(1 - \hat{\epsilon}/2)(1 - \epsilon/2) - (1 - \epsilon/2)\frac{2D}{\epsilon} \ln N \ge \mathrm{OPT}(1 - \epsilon) - \frac{2D}{\epsilon} \ln N.$$

∎

D Proof of Bound for Global Welfare (Theorem 3)

Before describing the algorithm, we begin with a few definitions. First, let us associate utilities from trades with specific sell bids. Define S to be the set of sell bids. Fix an optimal matching

M^*. Let us define $R_{opt} : S \to [p_{min}, p_{max}]$ to be the valuation of the person who received the item from seller s in the optimal matching. If s trades with b, then $R_{opt}(s) = p(b)$. If s does not trade, then $R_{opt}(s) = p(s)$.

For an arbitrary matching M, let us use the notation $M_{b,\bar{s}}(T)$ to indicate the number of pairs in M where the buy bid is greater than or equal to T and the sell bid is less than T. Similarly, define $M_{b,s}(T)$ to be the number of pairs in M where the buy bid and sell bid are both greater than or equal to T. Define $M_s(T)$ to be the number of unmatched sell bids greater than or equal to T.

Now let us restrict our view to algorithms which work by first selecting a price threshold and then reducing the resulting problem to an incomplete interval graph and applying some algorithm. Suppose that α is the reciprocal of the competitive ratio of the underlying online incomplete interval graph matching algorithm, that is, α is a lower bound on the fraction of potential trades actually harnessed. So, for any optimal online matching algorithm (such as Algorithm 2), $\alpha = 1$. For the Greedy algorithm, $\alpha = 0.5$.

Define $N^*(T)$ to be the number of agents (buyers and sellers) that end up with an item in the optimal matching M^*, and have valuation greater than or equal to T. So,

$$N^*(T) = |\{R_{opt}(s) \geq T | s \in S\}| = M^*_{b,\bar{s}}(T) + M^*_{b,s}(T) + M^*_s(T)$$

Let M_{alg} be a matching obtained by an online algorithm. Define $N^{alg}(T)$ to be the number of agents (buyers and sellers) that end up with an item in the matching M_{alg}, and have valuation greater than or equal to T. Then,

$$N^{alg}(T) \geq \alpha M^*_{b,\bar{s}}(T) + M^*_{b,s}(T) + M^*_s(T) \geq \alpha\ N^*(T)$$

The social welfare achieved by the algorithm is at least

$$N^{alg}(T)\ T + (|S| - N^{alg}(T))p_{min} = (N^{alg}(T))(T - p_{min}) + |S|p_{min}.$$

Because $T > p_{min}$, this is an increasing function of $N^{alg}(T)$, so the algorithm achieves social welfare at least

$$(\alpha\ N^*(T))(T - p_{min}) + |S|p_{min}$$

Arithmetic manipulation yields

$$N^*(T)[\alpha T + (1-\alpha)p_{min}] + (|S| - N^*(T))p_{min}$$

By substitution we get

$$|\{R_{opt}(s) \geq T | s \in S\}|\, [\alpha T + (1-\alpha)p_{min}] + |\{R_{opt}(s) < T | s \in S\}|\, p_{min}$$

By defining $R_T(s) = \alpha T + (1-\alpha)p_{min}$ when $R_{opt}(s) \geq T$ and $R_T(s) = p_{min}$ otherwise, the algorithm is guaranteed to achieve social welfare at least

$$\sum_{s \in S} R_T(s)$$

So, we have divided the social welfare on a per sell bid basis. In the rest of this section we show how to design a price threshold distribution so as to maximize the competitive ratio associated with one sell bid in the worst case. This will complete our online algorithm for social welfare maximization.

We will attempt to achieve a competitive ratio of r for some arbitrary $r \geq 1$. Define $D(x)$ to be the probability that the threshold is above x. Define $y = R_{opt}(s)$ and $z = R_{alg}(s)$. Observe that if $y = rp_{min}$, then even if $z = p_{min}$, we achieve our competitive ratio. Therefore, $D(rp_{min}) = 1$. Also, there is no need for the threshold to be above p_{max}, so $D(p_{max}) = 0$. Also, define $Y(y)$ to be the expected value of $R_{alg}(s)$ if $R_{opt}(s) = y$. The ratio r is achieved if for all $x \in [a, p_{max}]$, $Y(x) \geq x/r$. Note that $Y'(x) = -D'(x)(\alpha x + (1-\alpha)p_{min})$, because $-D'(x)$ is the probability density at x.

Observe that one wants to use no more probability mass on a bid than absolutely necessary to achieve the competitive ratio of r, because that probability mass is better used on later bids if they exist. In other words, the algorithm should hold out as long as possible. Therefore, for an optimal algorithm, $D(rp_{min}) = 0$, and $Y(x) = x/r$. Taking the derivative of the latter, we get $Y'(x) = 1/r$. Substituting, we get $1/r = -D'(x)(\alpha x + (1-\alpha)p_{min})$, so $D'(x) = -\frac{1}{r(\alpha x+(1-\alpha)p_{min})}$. Integration yields

$$\begin{aligned} D(y) &= 1 + \int_{rp_{min}}^{y} D'(x)dx \\ &= 1 - \int_{rp_{min}}^{y} \frac{dx}{r(\alpha x + (1-\alpha)p_{min})} \\ &= 1 - \frac{1}{r\alpha}\left[\ln\left|y + \frac{1-\alpha}{\alpha}p_{min}\right| - \ln\left|rp_{min} + \frac{1-\alpha}{\alpha}p_{min}\right|\right] \end{aligned}$$

We want to just run out of probability mass as y approaches p_{max}. Therefore,

$$D(p_{max}) = 1 - \frac{1}{r\alpha}\left[\ln\left|p_{max} + \frac{1-\alpha}{\alpha}p_{min}\right| - \ln\left|rp_{min} + \frac{1-\alpha}{\alpha}p_{min}\right|\right] = 0$$

$$r = \frac{1}{\alpha}\left[\ln\left(p_{max} + \frac{1-\alpha}{\alpha}p_{min}\right) - \ln\left(rp_{min} + \frac{1-\alpha}{\alpha}p_{min}\right)\right]$$

Solving the above equation for r will result in the optimal competitive ratio, that can then be used to calculate the price threshold probability distribution $D(x)$.

So, for Algorithm 2,

$$r = \ln p_{max} - \ln rp_{min}$$

For the Greedy algorithm,

$$r = 2\left[\ln(p_{max} + p_{min}) - \ln((r+1)p_{min})\right]$$

In order to better understand this competitive ratio, we can derive an upper bound on it:

$$\begin{aligned} r &\leq \frac{1}{\alpha}\left[\ln\left(p_{max} + \frac{1-\alpha}{\alpha}p_{min}\right) - \ln\left(p_{min} + \frac{1-\alpha}{\alpha}p_{min}\right)\right] \\ &\leq \frac{1}{\alpha}\left[\ln p_{max} - \ln p_{min}\right] \end{aligned}$$

So, for Algorithm 2,

$$r \leq \ln p_{max} - \ln p_{min}$$

For the Greedy algorithm,

$$r \leq 2\left[\ln p_{max} - \ln p_{min}\right]$$