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 who submit buy and sell bids that 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 *surplus*, we present a (randomized) online algorithm with competitive ratio $\ln(p_{max} - p_{min}) + 1$, when bids are in a range $[p_{min}, p_{max}]$, which we show is the best possible. A simpler algorithm has 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. 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 optimal offline algorithm that cannot use subsidies. This is the best possible competitive ratio for the setting. We present all of our results as corollaries of theorems on online matching in an incomplete interval graph.

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, auctions with potentially multiple buyers and multiple sellers). The range of applications includes trading in stock markets, business-to-business commerce, bandwidth allocation in communication networks, as well as resource allocation in operating systems and computational grids. These trends have led to an increasing need for fast market clearing algorithms [20, 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, bandwidth, oil, 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 (SEC) 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, and are thus not regulated by the SEC. 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

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

[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.

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 matching, i.e., 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 already been conducted by others [15, 10, 9, 2], while we study exchanges where there can be multiple buyers and multiple sellers submitting bids. Our goal, for each objective function we consider, will be to produce online algorithms with optimal competitive ratios. We consider the following objectives:

- **Maximize surplus.** Each pair of buy and sell bids that are matched produces a surplus, which is the difference between the buy price and the sell price. The total surplus is the sum of these differences, over all matched pairs. Maximizing surplus corresponds to maximizing economic value (social welfare).[4] The surplus could be divided in any way among the buyers, sellers, and the auctioneer. If surplus is maximized, there is no way to make all of the parties better off. Offline, the matching that optimizes surplus 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.[5] 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

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

[3] This performance measure implicitly skirts the issue that actions performed by the auctioneer can potentially influence the sequence of bids submitted in the future.

[4] This is the case when buyers and sellers truthfully reveal their valuations as the buy bids and sell bids. In reality they might under-bid and over-ask strategically (the extent to which they would act strategically depends on the mechanism design and is beyond the scope of this paper). Therefore, to be accurate, we should call this measure *revealed surplus*.

[5] A similar problem to our surplus maximization problem is *online difference maximization* [12]. However, in that problem, the focus is on the average case, not the worst case. Furthermore, the goal is to maximize the number of "bids" whose prices fall between the matched "buy bid" and "sell bid", rather than maximizing the monetary difference between the two. Finally, our problem concerns multiple matches.

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

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.

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 in the future.

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 $t_f(v) \neq t_f(v')$.[6]

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', outputs whether or not they can 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: the online edge-selection problem and the online vertex-selection problem. In the online *edge-selection* problem, the online algorithm maintains a matching. The algorithm sees a vertex v at the time it is introduced. 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 size of the final matching.

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.[7] A simple greedy algorithm achieves this ratio:

ALGORITHM 2.1. (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 2.1. *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 2.2. 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

[6] Our results can be extended to settings where $t_i(v)$ may equal $t_f(v)$, $t_i(v)$ may equal $t_i(v')$, and $t_f(v)$ 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.

[7] 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.

on $W \cup \{v, v'\}$. Otherwise, if no unmatched vertex v' works, allow v to expire unmatched.

THEOREM 2.2. (MAIN THEOREM) *Algorithm 2.2 produces a set of nodes having a perfect matching M which is a maximum matching in G.*

The proof of Theorem 2.2 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 $W \cup \{w, w'\}$.

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 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[8].

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.

[8] 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 $W - \{v, v'\}$.

3 Surplus 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 surplus.

We can convert the surplus maximization problem to an incomplete interval graph problem by choosing some θ to be the minimum surplus 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 2.1) then corresponds to the strategy: "whenever there exists a pair of bids in the system that would produce a surplus at least θ, match them immediately." Algorithm 2.2 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 surplus 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 surplus of at least θ. This is not *too* bizarre since, after all, 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 produced in the above manner. Then, from Theorem 2.1, the greedy edge-selection algorithm achieves a surplus at least $\frac{1}{2}\theta|M^*(G_\theta)|$. Applying Algorithm 2.2 achieves surplus 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

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

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 3.1. *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 3.1. *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 2.2 achieves competitive ratio* $\ln(p_{max} - p_{min}) + 1$.

In Section 5 we prove a corresponding lower bound of $\ln(p_{max} - p_{min}) + 1$ for this problem.

Proof (of Lemma 3.1): 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 $\theta|M^*(G_\theta)| \geq \sum_{(b,s)\in \mathrm{OPT}} R_\theta(b, s)$ because the set of pairs of surplus 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 2.2 is that it may recommend matching a pair of buy and sell bids that actually have negative surplus. Since this cannot possibly improve total surplus, we can always just ignore those recommendations (even though the algorithm will think that we did, in fact, match 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 surplus, our goal is to maximize the *number* of trades made, subject to the constraint that each matched pair have non-negative surplus. 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 5.3.

However, if the online algorithm can subsidize matches (match a pair of buy and sell bids of negative surplus if it has already made enough money to pay for them) then we can use Algorithm 2.2, and do as well as the optimal solution in hindsight that is not allowed subsidization. Specifically, when Algorithm 2.2 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 2.2 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

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 surplus algorithm.

ALGORITHM 4.1. 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 is similar to that of Lemma 3.1.

If the online algorithm is allowed to use subsidization, then we can use Algorithm 2.2 as follows.

ALGORITHM 4.2. Choose a buy price threshold θ at random accoring to the same distribution as in Algorithm 4.1. 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 2.2 on the constructed graph. If Algorithm 2.2 chooses a buy bid b and a sell bid s, then they are concurrent. 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 (the proof is similar to that of Lemma 3.1 and its corollary). This is the best ratio that can be achieved (the proof is by threat-based analysis similar to that in Section 5.1).

Maximizing sell volume is analogous to maximizing buy volume. The best competitive ratio achievable without using subsidy we know 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 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 surplus maximization problem, no deterministic algorithm can achieve a competitive ratio better than 2 for the trade volume maximization problem without subsidization, no randomized algorithm can achieve a competitive ratio better than 4/3 for the trade volume maximization problem without subsidization. Also, we prove that the greedy algorithm described in the paper for maximizing surplus does not achieve a competitive ratio better than $2(\ln(p_{max}-p_{min})+1)$.

5.1 A Threat-Based Lower Bound for Surplus

In this analysis, we prove a lower bound for the competitive ratio of any online algorithm by looking at a 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 which 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 surplus 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 $\ln(p_{max}-p_{min})+1$.

5.2 Greedy Surplus Maximization

The following scenario shows that our analysis of the greedy surplus 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 surplus 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 surplus 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. This example shows why the optimal algorithm must consider the past when considering current bids.

5.3 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.

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.

6 Combining Algorithms Online

The algorithm of Corollary 3.1 begins by picking a threshold θ from some (exponential) distribution. It turns out that if there is an upper bound on the "burstiness" of the bids, then 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, in some natural distributional settings, the best strategy *is* to pick some fixed threshold, and in these cases, the modified strategy would be within a $(1+\epsilon)$ factor of optimal.

At any point in time, for each threshold θ, we can encapsulate how well we *would* have done had we used this threshold as a pair $(\mathsf{surplus}_\theta, \mathsf{state}_\theta)$, where $\mathsf{surplus}_\theta$ is the surplus acheived so far, and state_θ is the set of its current outstanding bids. What we can now do is combine all these fixed-threshold algorithms using the Randomized Weighted Majority (Hedge) algorithm of [17, 8], as adapted by [4] for the case of experts with internal state. The important issue here is the following. When the overall "master" algorithm tells us to switch from threshold θ_1 to θ_2, we may not be in as good a state as $\mathsf{state}_{\theta_2}$ (in particular, we may have fewer outstanding bids). However, suppose we have an apriori upper-bound B on the maximum size of any state (this is our bound on "burstiness"). Then, if we conservatively only match buy/sell pairs in our state that are also in $\mathsf{state}_{\theta_2}$ (and that have surplus at least θ_2) then we can guarantee that our surplus in this time period is at worst Bp_{max} less than had we been in $\mathsf{state}_{\theta_2}$. In other words, we can view all the states as belonging to a metric space of diameter at most Bp_{max}. At this point, we can apply a theorem of [4] to say that for any $\epsilon > 0$ (ϵ is given to the algorithm), we can achieve an expected gain at least

$$\max_\theta \left[(1-\epsilon)\mathsf{surplus}_\theta - \frac{4Bp_{max}}{\epsilon}\log N\right],$$

where N is the number of different thresholds. Details are left to the full version of the paper.[9]

[9] One issue we have glossed over is that the results of [4] are given in terms of losses instead of gains. But the arguments carry over for gains, as in the stateless case [8].

7 Conclusions and Future Research

In this paper, we derived a variety of bounds on competitive ratios for several online market clearing problems having to do with maximizing surplus or maximizing different measures of liquidity. Some of our worst-case competitive ratios are the best that can be achieved online, and the rest are within a factor of two of the best that is achievable online.

For some of our metrics, a constant factor is very important since they are monetary measures, not measures of computation speed. The difference between the greedy surplus algorithm and the surplus algorithm based on Algorithm 2.2 is that the latter guarantees twice as much surplus in terms of a competitive ratio.

There has 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]. 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 surplus 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. In *Proceedings of the Fourth International Conference on Autonomous Agents (AGENTS)*, pages 73–96, Barcelona, Spain, June 2000. Early version appeared in the 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. Market clearability. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1145–1151, Seattle, WA, 2001.

[21] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1102–1108, Seattle, WA, 2001.

[22] Peter R Wurman, Michael P Wellman, and William E Walsh. The Michigan Internet AuctionBot: 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.

Appendices

A Lemmas on Matchings

In this section, we prove some lemmas required for the main theorem proof 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 A.1. *Given $v, v' \in V - W$, and that there exists a perfect matching M on 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 A.2. *The symmetric difference of two matchings M and M' consists of alternating paths and alternating cycles with respect to M.*

LEMMA A.3. *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 adjacent edge in either M or M', but not both. Hence, it can only have one adjacent 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 a path whose first and last edges are in M. A* ***replacement path***[10] *with respect to M has one endpoint v in $V(M)$ with the edge incident to v in P also in M, and one endpoint not in $V(M)$.*

LEMMA A.4. *Given $v, v' \in W$, and that there exists 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 A.1.

LEMMA A.5. *Given $v \in W$ and $v' \in V - W$, and that there exists 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 A.1.

LEMMA A.6. *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 A.1. *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 2.2 (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 B.1. *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.

[10] A replacement path can be from a vertex in $V(M)$ to a vertex not in $V(M)$, or from a vertex not in $V(M)$ to a vertex in $V(M)$.

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 B.2. *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 B.3. *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 A.6 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 A.6 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 A.6 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 2.2): 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 B.1, B.2, B.3, 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 A.1, W is one of the largest sets with a perfect matching. ∎