Online and Offline Selling in Limit Order Markets

Kevin L. Chang[1]* and Aaron Johnson[2]**

[1] Yahoo Inc.
klchang@yahoo-inc.com
[2] Yale University
ajohnson@cs.yale.edu

Abstract. Completely automated electronic securities exchanges and algorithms for trading in these exchanges have become very important for modern finance. In [4], Kakade *et al.* introduced the limit order market model, which is a prevalent paradigm in electronic markets. In this paper, we consider both online and offline algorithms for maximizing revenue when selling in limit order markets. We first prove that the standard reservation price algorithm has an optimal competitive ratio for this problem. This ratio is not constant, and so we consider computing solutions offline. We show that the offline optimization problem is **NP**-hard, even for very restricted instances. We complement the hardness result by presenting an approximation scheme that runs in polynomial time for a wide class of instances.

1 Introduction

Electronic exchanges are very important venues for trading many different classes of financial securities. With the widespread use of such markets, a technique known as *algorithmic trading* has become popular among financial institutions and institutional investors who seek to buy or sell large amounts of a particular security. Consider, for instance, a pension fund that would like to sell many shares of a particular stock. Typically, such an investor would not sell large amounts of stock himself by just submitting an order to a stock exchange, but rather would seek the expertise of a broker (*e.g.* an investment bank) to perform the transaction on his behalf. A standard strategy for the broker is to break the large order into many smaller orders and to submit the orders gradually over a given time horizon, with the obvious goal of maximizing his total revenue.

Advances in financial technology have automated this process, and the pension fund's entire trade can now be effected by computer with little human intervention: computer programs can choose how to divide the large order into smaller orders, choose at what time and at what price to submit the smaller

* Work performed while affiliated with the Max Planck Institute for Computer Science.

** Supported by NSF grants 0428422, 0534052, and 0716223, ONR grants N00014-01-1-0795 and N00014-04-1-0725, and ARO grant W911NF-05-1-0417.

orders, and then actually execute the transactions on an electronic market. Sophisticated algorithms for such large trades are offered by most large investment banks and by many smaller firms specializing in financial technology.

The manner in which an electronic market itself executes the various orders from clients can also be complex. One of the most important electronic exchange paradigms is the *electronic communication network* (ECN), which implements a *limit order market*. Prominent ECNs for trading equities and equity derivatives are operated by NASDAQ, Instinet, and NYSE-Euronext.

In a limit order market, buyers and sellers submit limit orders to buy or sell a commodity; these orders have semantics such as "I would like to buy v shares, but I am only willing to pay p USD or less for each share." The market matches buyers and sellers based on a transparent function of the orders submitted. Often, the input order stream is available to market participants, and so algorithms should be conscious of this *market microstructure* in order to effectively trade.

In this paper, we will study natural theoretical computer science questions motivated by algorithmic trading in limit order markets. We consider both online and offline algorithms for placing sell orders in limit order markets, with the goal of maximizing the revenue generated by selling volume N of a particular commodity (*e.g.* a stock). This problem domain was introduced into the theoretical computer science literature by Kakade, Kearns, Mansour and Ortiz in [4], although online trading algorithms in much simpler market models have been studied for many years.

1.1 The Trading Model: The Mechanics of Limit Order Markets

In a limit order market, market participants submit limit orders, which consist of three-tuples: $\sigma = \langle \theta, p, v \rangle$. The parameter θ specifies whether the order is to buy or to sell; p denotes the least competitive price the market participant is willing to accept, that is, the lowest price *per share* that is acceptable for a sell order, or the highest price *per share* for a buy order; and v denotes the volume, that is, the number of shares to transact.

A sell order $\sigma_1 = \langle \mathsf{S}, p_1, v_1 \rangle$ can be "matched" to a buy order $\sigma_2 = \langle \mathsf{B}, p_2, v_2 \rangle$ if $p_1 \leq p_2$. If σ_1 and σ_2 transact and $v_1 > v_2$, then σ_2 will be filled, but σ_1 will only be partially filled: its volume will be reduced to $v_1 \leftarrow v_1 - v_2$. If $v_1 < v_2$, then σ_1 will be filled, and for σ_2, we have $v_2 \leftarrow v_2 - v_1$. Observe that when $p_1 < p_2$, any price $p \in [p_1, p_2]$ would be acceptable to both parties. Limit order markets use the convention that the transaction will occur at the limit price of the order that arrived first. With this convention, it is advantageous to be the second of the two matched orders to arrive.

At any time step, buy and sell orders that have been submitted but that have not yet been paired with suitable counterparties are stored in the *buy order book* and the *sell order book*, respectively. Orders in the book are sorted according to their prices, with the most competitive orders at the "top" of the book (*i.e.* in the buy order book, orders with the highest price are at the top; in the sell order book, orders with the lowest price are at the top). Ties are broken by placing the order that arrives first higher in the book. An important property of the two books is that the prices of all orders in the buy book are lower than the prices of all orders in the sell book, since the books consist of orders that have not yet been matched.

When a new sell order $\sigma_s = \langle \mathsf{S}, p_s, v_s \rangle$ arrives, it is compared with the top order in the buy book, say $\sigma_b = \langle \mathsf{B}, p_b, v_b \rangle$. If $p_s > p_b$, then no transaction can occur and σ_s will be placed in the sell book according to the rules specified in the previous paragraph. If $p_b \geq p_s$, then $\min\{v_b, v_s\}$ shares are sold at price p_b per share. If $v_s \geq v_b$, then σ_b has been filled and is removed from the buy book, while the volume of σ_s is adjusted accordingly and a new matching buy order is sought. If $v_s < v_b$, then the order σ_s is filled, and the volume of σ_b is adjusted accordingly. An arriving buy order would be processed in an analogous fashion. The state of the book is public knowledge in many ECNs and can therefore be exploited by sophisticated traders.

Table 1 is an example of buy and sell order books. If a new sell order arrived at price 102.00 and volume 50, then it would transact with the buy order at price 102.20. The volume of this buy order would be reduced by 50.

Buy orders		Sell orders	
Price	Volume	Price	Volume
102.20	100	102.55	200
102.00	500	102.93	300

Table 1.

1.2 The Trading Problem

The trading over the course of the time horizon is represented by time steps $t = 1, \ldots n$. At time step t, order σ_t is placed by some market participant (not the algorithm). As each order arrives, executions occur and the book is updated as described above.

The problem that we consider is to design an algorithm that inserts sell orders into this stream in order to maximize its revenue. At each time step t, the algorithm may place orders before the arrival of σ_t. It cannot sell a total volume of more than N, nor can it submit buy orders.

We will consider both online and offline algorithms for this problem. In the online problem, the algorithm observes the market orders over time and, as they come in, must insert its own orders. The standard measure of quality of an online algorithm is defined by its *competitive ratio*. A randomized profit-maximizing algorithm has a competitive ratio at most $c > 0$ if, for all input sequences Σ, $\mathrm{E}[Rev] \geq \frac{1}{c} \cdot \mathrm{OPT}$, where OPT is the revenue that results from an optimum, offline placement of orders on input Σ and $\mathrm{E}[Rev]$ is the expected revenue of the algorithm. We refer the reader to the book by Borodin and El-Yaniv [1] for an introduction to online algorithms.

1.3 Our Contributions

We show that the standard online *reservation price* algorithm yields a competitive ratio of $e \log R \leq 2.72 \log R$ in the order book model, where $R = p_{\max}/p_{\min}$ is the *price ratio* between the highest and lowest possible prices in the order stream. This competitive ratio is optimal, in the sense that any *randomized* algorithm must have competitive ratio at least $\frac{1}{2}\log R$. All logarithms are base e. Our bound on the competitive ratio is an improvement over the $O(\log R \log N)$ bound for the reservation price algorithm by Kakade *et al.* [4], because our bound has no dependence on N.

Since no online selling algorithm can achieve a competitive ratio that is $o(\log R)$, we examine maximizing revenue in the offline case. We prove that this optimization problem is **NP**-hard by reducing from KNAPSACK, even for the restricted case where market participants only submit orders with three distinct prices. We also show that a simple dynamic programming algorithm will find the optimum solution. Like KNAPSACK, the running time is *pseudopolynomial*: polynomial in N and the largest volume of any order (which in general can be exponential in the size of the input) when the number of distinct prices that market participants may submit is a *fixed constant* k. We then prove that the input volumes can be adjusted so that running the dynamic programming algorithm on the adjusted input will yield a polynomial-time approximation scheme when there is a fixed constant k of distinct prices. The running time with approximation ration $1-\epsilon$ is polynomial in $n, 1/\epsilon$ and R . We note that all securities indeed only have a constant number of possible prices (*e.g.* US equities are traded in multiples of \$.01), although this constant is admittedly large.

The offline optimization problem is a natural and theoretically interesting question to consider. Although it cannot immediately be applied to the real online problem faced by a trader, nonetheless these results could be of interest to a broader audience. The NP-hardness of computing the offline optimum *even with knowledge of the entire order stream* is a very strong statement about the intractability of optimization, for any conceivable context.

Offline algorithms have many potential applications. For instance, an approximation algorithm can be used for studying historical data, and the output of the approximation algorithm can be used to compare the realized performance of a trading algorithm to its theoretical optimum. Also, an offline algorithm could be coupled with an appropriate statistical model for generating sample paths of the future evolution of market microstructure in order to design realizable trading strategies.

Finally, we generalize some of our results to the buying case.

1.4 Related Work in Theoretical Computer Science

Online algorithms for selling in limit order markets were first introduced by Kakade, Kearns, Mansour and Ortiz in [4]. Kakade *et al.* considered selling algorithms that seek to optimize revenue, as well as selling algorithms that seek to sell shares at the *average price* of the market (the *Volume Weighted Average Price*, which is a popular benchmark for commercially available trading algorithms). Even-Dar, Kakade, Kearns, and Mansour also considered the stability of limit order dynamics in [3].

The limit order market is a generalization of simpler online trading models. El-Yaniv, Fiat, Karp, and Turpin in [2] considered the *one-way trading*, *time* *series search* and *two-way trading* problems in this framework. This work was later extended by Lorenz, Panagiotou, and Steger in [5].

1.5 Preliminaries

The input order stream will be denoted by $\Sigma = \sigma_1, \ldots, \sigma_n$, where order $\sigma_t = \langle \theta_t, p_t, v_t \rangle$ arrives at time t. The algorithm's sequence of sell orders is represented by a set $\{\sigma_i^A\}$, where σ_t^A denotes a sell order that is placed right before the arrival of $\sigma_t \in \Sigma$ at time t. If the algorithm places a set of sell orders Σ^A in the order sequence Σ, we will denote the new order stream by $\Sigma \cdot \Sigma^A$.

If the limit prices of all orders in the input stream fall into the interval $[p_{\min}, p_{\max}]$, which is known to the algorithm, then $R = p_{\max}/p_{\min}$ is the *price ratio.* In our paper, we will refer to Lemma 5.3 by Evan-Dar *et al.* [3]. The lemma is most easily stated for the case where all orders have unit volume:

Lemma 1 (Stability Lemma [3]). *Suppose all orders have unit volume, and the order stream Σ' is derived from Σ by inserting a single order σ. (1) If σ is not executed in Σ', then the sets of executed orders in Σ and in Σ' are identical. (2) If σ is executed in Σ', then at most 1 order (as specified by id) was executed in Σ but not in Σ'. Conversely, at most 1 order (other than σ) was executed in Σ' but not in Σ.*

From a high-level perspective, the stability lemma states that if we insert an extra order into the order sequence, it will not affect the set of executed orders by very much.

2 Optimal Online Algorithms

We show in this section that the *reservation price* algorithm for selling in the order book model has an optimal competitive ratio. This algorithm was originally considered for the *max-search problem* (also called the *time-series search problem*) by El-Yaniv *et al.* [2] and later by Lorenz, Panagiotou and Steger in [5] and by Kakade *et al.* in [4] for selling in the order book model.

The reservation price algorithm is: pick an integer i uniformly at random between 0 and $\lfloor \log R \rfloor$ and place an order to sell all N shares of stock at price $e^i p_{\min}$ at time $t = 0$, where R is the price ratio.

Theorem 2. *The reservation price algorithm for selling in the order book model has competitive ratio $e \log R$. Furthermore, any randomized algorithm must have competitive ratio at least $\frac{1}{2} \log R$.*

Proof. We first prove that the reservation price algorithm has competitive ratio at most $e \log R$. Let $p_{\text{res}} = e^i p_{\min}$ be the reservation price randomly chosen by the algorithm. Suppose $p_1, \ldots, p_N$ are the prices realized for the sale of the N shares by the optimal solution, Σ^A, and let $\text{OPT} = \sum_1^n p_i$ be the optimal revenue. Let $P_i = \{p_j : p_{\text{res}} \leq p_j \leq e \cdot p_{\text{res}}\}$ be the set of prices of executed orders that are within a factor of e of the price p_{res}.

Claim. The reservation price algorithm will sell at least $|P_i|$ shares at price p_{res}.

The main idea of the proof, which is omitted for space, is that if $\sigma_0^R = \langle \mathsf{S}, p_{\text{res}}, N\rangle$ is the order placed by the reservation price algorithm, then in the set of executions of $\Sigma \cdot (\{\sigma_0^R\} \cup \Sigma^A)$, σ_0^R will execute $|P_i|$ shares, since it will have a higher priority than the orders in P_i. It follows that if σ_0^R is placed by itself, it will realize at least as much revenue.

With probability $1/\log R$, the reservation price algorithm will choose reservation price $p_{\text{res}} \leftarrow e^i p_{\min}$. Then the expected revenue of the algorithm is

$$\sum_{i=0}^{\lfloor \log R \rfloor} \frac{1}{\log R} |P_i| \, e^i p_{\min} \geq \sum_{i=1}^{\lfloor \log R \rfloor} \sum_{p_j \in P_i} \frac{p_j}{e \log R} = \frac{\text{OPT}}{e \log R}.$$

Thus, the competitive ratio of the reservation price algorithm is at most $e \log R$.

In order to establish the lower bound, we observe that it is straightforward to reduce the online *max-search* problem to our problem of selling in a limit order market. In this problem, a player observes a sequence of prices and tries to select the highest one. The $\frac{1}{2}\log R$ lower bound on the competitive ratio for any randomized algorithm for max-search proved by Lorenz *et al.* in [5] establishes the lower bound for limit order markets. $\square$

3 NP-Hardness of the Offline Problem

We prove the **NP**-completeness of optimal selling by reduction from KNAPSACK. Our reduction is to the special case of instances in which only three different prices occur, p_h, p_m, p_l, where $p_h \geq p_m \geq p_l$.

We will use several facts about the structure of the optimum solution in the three-price case.

Lemma 3. *There exists an optimal solution that places all its high-price sell orders at time $t = 0$. The number of such orders can be arbitrarily large.*

Lemma 4. *There exists an optimal solution that places all its low-price orders immediately after the last execution it realizes at either price p_h or price p_m.*

Lemma 5. *There exists an optimal solution that only places a medium-price order immediately after an execution occurs at a high price.*

Lemma 6. *There exists an optimal solution, such that if we attempt to insert another medium-price order after an execution at a high price, then the algorithm will achieve one less execution at a high price, except when this occurs after the last execution at a high price.*

The proofs of these lemmas show that any optimal solution can easily be converted to satisfy these properties.

We will reduce the **NP**-complete KNAPSACK problem to offline selling. An input to KNAPSACK $\mathcal{I}_{\mathcal{K}}$ is a set of n pairs, (w_i, v_i), a capacity C, and a value V.

It is in the language if there exists a subset $S \subseteq [n]$ such that $\sum_{i \in S} v_i \geq V$ and $\sum_{i \in S} w_i \leq C$. Let $v^* = \max_i(v_i)$ and let $W = \sum_i w_i$. We can assume that all numbers in the instance are positive integers, and that $v^* > 1$.

The three prices in the selling instance we will create are $p_l = 1$, $p_m = (C+1)v^*$, and $p_h = (C+1)v^* + 1$. Let $a_i = (C+1)(w_i v^* - v_i)$ and $b_i = ((C+1)(w_i v^* - v_i) + w_i)(C+1)(v^*)$. For $1 \leq i \leq n$, let Σ_i be the following sequence of limit orders:

1. $\langle \mathsf{B}, p_m, a_i + w_i \rangle$ 2. $\langle \mathsf{S}, p_l, w_i \rangle$ 3. $\langle \mathsf{B}, p_l, w_i \rangle$ 4. $\langle \mathsf{S}, p_m, a_i \rangle$ 5. $\langle \mathsf{B}, p_h, a_i + b_i \rangle$

Let Ω be the following order sequence:

1. $\langle \mathsf{S}, p_l, W - C \rangle$ 2. $\langle \mathsf{B}, p_m, p_m W \rangle$

Let Σ be the concatenated order sequence $(\Sigma_1, \ldots, \Sigma_n, \Omega)$. The total volume of buy orders in Σ is $N = \sum_i (2a_i + 2w_i + w_i p_m + b_i)$. N will be the number of shares to sell. The revenue to raise will be $R = p_h \sum_i (a_i + b_i) + p_m^2 W + (C+1)V$. Then $\mathcal{I}_{\mathcal{S}} = (\Sigma, N, R)$ is an input to the offline selling problem. It is in the language if R revenue can be obtained by selling at most N shares.

Lemma 7. *If there exists a solution $S \subseteq [n]$ to $\mathcal{I}_{\mathcal{K}}$ with total value V, then there exists a solution to $\mathcal{I}_{\mathcal{S}}$ with revenue of at least R.*

Proof Sketch: Given S, first insert the order $\sigma_0^A = \langle \mathsf{S}, p_h, \sum_i (a_i + b_i) \rangle$ at the beginning of Σ. σ_0^A executes with every high-price buy order, which yields revenue $p_h \sum_i (a_i + b_i)$. If $i \in S$, insert the order $\sigma_i^A = \langle \mathsf{S}, p_m, a_i + w_i \rangle$ at the beginning of Σ_i. When σ_i^A is added, then $a_i + w_i$ sales are made by σ_i^A at the medium price, and a_i less sales are made from the high-price buy order at the end of Σ_i. Therefore the change in revenue from σ_i^A is $p_m(a_i + w_i) - p_h(a_i) = (C+1)v_i$. Finally, insert the order $\sigma_{n+1}^A = \langle \mathsf{S}, p_m, p_m W \rangle$ after subsequence Σ_n, and insert $\sigma_{n+2}^A = \langle \mathsf{S}, p_l, C \rangle$ at the end of Ω. Regardless of the previous insertions, the revenue from σ_{n+1}^A and σ_{n+2}^A is $p_m^2 W + p_l \left(C - \sum_{i \in S} w_i \right) \geq p_m^2 W$. Then the total revenue obtained is at least $p_h \sum_i (a_i + b_i) + p_m^2 W + (C+1)V = R$. □

Next we prove the converse of Lemma 7. By Lemma 3, we can assume that the optimal solution for $\mathcal{I}_{\mathcal{S}}$ places a large sell order at price p_h at the beginning of Σ. Observe that in the resulting execution, that sell order executes at the high price after every subsequence Σ_i. By Lemma 5, we can assume that all medium-price orders in the optimal solution are each inserted at the beginning either of some Σ_i or of Ω. Let $S \subseteq [n]$ be the set of Σ_i subsequences for which this happens. Finally, by Lemma 4, we can assume that any low-price order is inserted at some point after the last high-price or medium-price order is inserted.

Lemma 8. *For $i \in S$, there is an optimal solution in which the medium-price order inserted at the beginning of Σ_i has volume $a_i + w_i$.*

Proof Sketch: Let $\sigma_i^A = \langle \mathsf{S}, p_m, v \rangle$, $i \in S$, be the medium-price order inserted at the beginning of Σ_i. If $v > a_i + w_i$ or $v \leq a_i$, decreasing v increases high-price sells and decreases medium-price sells, for a net gain in revenue. Otherwise, v can be increased to $a_i + w_i$ without reducing high sells, contradicting Lemma 6. □

Lemma 9. *There is an optimal solution in which any low-price order is inserted after the Σ_n subsequence.*

Proof Sketch: Suppose instead that the low-price order σ_l^A is inserted before Σ_i. Consider moving σ_l^A to the beginning of Σ_i. If σ_l^A reduces the volume of high-prices transactions before Σ_i, this move would increase net revenue. Otherwise, the move maintains revenue.

□

Lemma 10. *There exists an optimal solution with the order $\langle \mathsf{S}, p_m, p_m W\rangle$ inserted at the beginning of Ω and the order $\langle \mathsf{S}, p_l, C - \sum_{i\in S} w_i\rangle$ inserted at the end.*

Proof Sketch: By Lemmas 4, 5 and 10, the solution places a low-price order in Ω that may be preceded by a medium-price order. Inspection shows that including the medium-price order is optimal. □

We can see by inspecting Ω that the payoff of the orders described in Lemma 10, expressed as a function of the volume of the low-price buy book, is:

$$\rho(l) = \begin{cases} p_m^2 W + p_l(C-l) & 0 \le l \le C \\ p_m^2 W - p_m(l-C) & C < l \le W. \end{cases} \tag{1}$$

Lemma 11. *We can assume that $\sum_{i\in S} w_i \le C$.*

Proof. Suppose instead that $\sum_{i\in S} w_i > C$. Consider removing the medium-price order at the beginning of Σ_i, $i \in S$. The sequence loses the $p_m(a_i + w_i) - p_h(a_i) = (C+1)v_i$ revenue from that order. The transactions in Σ_j subsequences are unaffected, but at the end of each there is an additional w_i volume in the buy book at the low price. Equation 1 shows that this volume increases the revenue obtained by at least $(C+1)v^*$. Therefore the total change in revenue is nonnegative, and we can convert this sequence into an optimum such that $\sum_{i\in S} w_i \le C$.

Finally, we can show that the optimal revenue of $\mathcal{I}_\mathcal{S}$ can give us a lower bound on the value of optimal subsets in $\mathcal{I}_\mathcal{K}$.

Lemma 12. *If there exists a solution to $\mathcal{I}_\mathcal{S}$ with revenue of at least R, then there exists a solution to $\mathcal{I}_\mathcal{K}$ with total value at least V.*

Proof. If $\sum_{i\in S} v_i \ge V$, then, by Lemma 11, S is a solution to $\mathcal{I}_\mathcal{K}$ of value at least V. Suppose otherwise, that $\sum_{i\in S} v_i < V$. The solution receives $p_h \sum_i (a_i + b_i)$ revenue from the initial high-price sell order. It receives $(C+1)\sum_{i\in S} v_i \le (C+1)(V-1)$ revenue from the medium-price orders in S. We can see from Equation 1 that it receives no more than $p_m^2 W + p_l(C)$ revenue from the two orders in Ω. This accounts for all of the revenue R' of the solution. However, $R' - R = (C+1)(V-1) + p_l(C) - (C+1)V < 0$, contradicting the assumption of the lemma. □

Theorem 13. *The decision version of optimal offline selling is **NP**-complete.*

Proof. The decision version of the offline selling problem is in **NP**, because we can calculate the revenue of a solution by running the limit order market algorithm on the total sequence. Also, the previous lemmas show that the input to KNAPSACK is in the language if and only if its reduction is a member of the offline selling problem language. □

In contrast to the three-price case, it can be shown that if there are only two prices in the order sequence, the problem can be solved exactly in $O(n)$ time. The algorithm simply places one high-price sell order at the beginning and then tries all positions for the low-price order.

4 Offline Algorithms and Approximation Schemes

In this section, we present approximation schemes for the offline selling problem. The general approach to our algorithm is similar to the FPTAS for KNAPSACK, but the technical details are more involved. We first give a pseudopolynomial dynamic programming algorithm. Then we show that this algorithm can be used in an approximation scheme by reducing and rounding the order volumes. The approximation scheme will have running time polynomial in $1/\epsilon$, the maximum price ratio R, and n, if the number of prices at which market participants can submit orders is a fixed constant.

4.1 Pseudopolynomial Time Dynamic Programming Algorithm

A simple dynamic programming algorithm can compute the optimal placement of sell orders in polynomial time, under the assumption that the volume of each limit order is 1, and that the number of distinct prices at which the market participants can place orders is at most a fixed constant k.

The input to the dynamic programming subproblem is given by: (1) Times t_1 and t_2, such that $t_1 \leq t_2$. (2) The initial buy and and sell order books at time t_1: B_{t_1}, S_{t_1}. (3) The final order books at time t_2: B_{t_2}, S_{t_2}. (4) m, the number of shares to be sold by the algorithm between times t_1 and t_2.

Each subproblem is then: Given buy and sell order books B_{t_1} and S_{t_1} at time t_1 (prior to the arrival of order σ_{t_1}), find the optimum placement of orders between times t_1 and t_2 (inclusive), such that the buy and sell order books at the end of time t_2 are B_{t_2} and S_{t_2} and that the number of shares sold by the algorithm between times t_1 and t_2 is at most m.

Theorem 14. *For the case where each order has unit volume, the dynamic programming algorithm will find an optimal solution in time $O(N^{3k+5}n^{3k+5})$, where k is the number of distinct prices.*

The details of the algorithm and its analysis follow standard dynamic programming techniques. This algorithm can be used with orders of arbitrary volume, which adds an additional factor in the runtime.

Corollary 15. *If the volume of each order is unrestricted, the dynamic programming algorithm will run in time $O(n^2 N^{3k+5}(nV)^{3k+3})$, where V is the maximum volume of any order.*

4.2 PTAS for the Arbitrary Volume Case

We now show how the input can be preprocessed in two steps so that the dynamic programming algorithm from the previous section can compute a solution with revenue at least $(1-\epsilon)$OPT in time $O\left(n^{12k+16}\left(R/\epsilon\right)^{6k+8}\right)$, for input sequences with arbitrary volumes in each order.

Step 1: Reduce to the Significant Volume Case.

Our first step will be to modify the instance to ensure that N is at least a fraction of the total volume of all orders, which we call the *significant volume* condition. Σ satisfies this condition if $(n+1) \cdot N \geq V$, where V is the maximum volume of any order in Σ.

Lemma 16. *Given an order stream Σ, we can construct an order stream Π such that*

1. *If V is the maximum volume of any order $\pi_i \in \Pi$, then $(n+1) \cdot N \geq V$.*
2. *If Σ^A is any set of sell orders with total volume at most N placed by the algorithm, it will realize the same revenue in $\Pi \cdot \Sigma^A$ as in the original input $\Sigma \cdot \Sigma^A$.*

We assume that there is at least one order in the original sequence Σ, $\sigma_i = \langle t_i, p_i, v_i \rangle \in \Sigma$ such that $v_i \geq N \cdot (n+1)$. Since a selling algorithm will only transact N shares, intuitively its action should have very little effect on order σ_i, which contains many more shares.

Let $\text{trans}_\Sigma(\sigma_i)$ denote the set of orders that are matched with σ_i in the evolution of the order sequence Σ. Let $\text{unex}_\Sigma(\sigma_i)$ denote the volume of σ_i that is unexecuted in the evolution of Σ. Let $\text{match}_\Sigma(\sigma_i, \sigma_j)$ denote the number of shares of σ_i that are matched with σ_j in the order sequence Σ.

Lemma 17. *Let Σ^A be any set of orders placed by the algorithm with an aggregate volume of at most N. Then for any $\sigma_i, \sigma_j \in \Sigma$,*

$$match_{\Sigma \cdot \Sigma^A}(\sigma_i, \sigma_j) \geq match_\Sigma(\sigma_i, \sigma_j) - N.$$

The lemma follows from applying the Stability Lemma to the transactions between σ_i and σ_j.

Lemma 17 implies that σ_i and σ_j have excess volume that is, in some sense, superfluous to the problem of selling at most N shares. We eliminate these in a new order sequence Π.

For each order $\sigma_i \in \Sigma$, where $\sigma_i = \langle \theta_i, p_i, v_i \rangle$, we create order π_i such that $\pi_i = \langle \theta_i, p_i, v'_i \rangle$, where $v'_i = v_i - \sum_{\sigma_j \in \text{trans}_\Sigma(\sigma_i)} \max(\text{match}_\Sigma(\sigma_i, \sigma_j) - N, 0) - \max(\text{unex}_\Sigma(\sigma_i) - N, 0)$. With these new volumes,

$$\text{vol}(\pi_i) \le \text{vol}(\sigma_i) - \sum_j \text{match}_\Sigma(\sigma_i, \sigma_j) - \text{unex}_\Sigma(\sigma_i) + (n+1)N = (n+1)N.$$

Thus, the modified input sequence $\Pi = \pi_1, \ldots, \pi_n$ satisfies Condition 1 of Lemma 16. Condition 2 follows from Lemma 17.

Step 2: Round Volumes.

Recall that V is the maximum volume of any order. We now assume that our input has been reduced to the significant volume case, where $(n+1) \cdot N \ge V$. Let $M = \epsilon \cdot N/(nR)$. In the second preprocessing step, we round the volume of every order in Σ to the nearest multiple of M. The volume of each order will be changed by at most $M/2$ and the number of possible values for the volume of an order will be $V/M = nVR/(N\epsilon) \le n(n+1)R/\epsilon$. Let Σ' be the input sequence of orders with rounded volumes.

Lemma 18. *Finding the optimum solution to Σ' will induce a solution with revenue at least $(1-\epsilon)OPT$ for the original input sequence Σ.*

Proof. We may assume that N is at most the aggregate volume of all buy orders in Σ. Then, $\text{OPT} \ge p_{\min} \cdot N$, because the algorithm could place the order $\langle \mathsf{S}, p_{\min}, N \rangle$ at the beginning and sell to the first N buy orders. We first prove that there exists a solution with revenue at least $(1 - \epsilon/2)\text{OPT}$ for Σ'. Let Σ^A be an optimum set of sell orders for the input sequence Σ. We may assume that the realized price for every order in Σ^A is the same as the price of the order.

Recall that our rounding scheme changed the volume of each order by at most $M/2$. The Stability Lemma therefore implies that the total volume of shares that are executed in $\Sigma \cdot \Sigma^A$ but not in $\Sigma' \cdot \Sigma^A$ is at most $n \cdot M/2$. Therefore, there are at most $n \cdot M/2$ shares that the algorithm sold in $\Sigma \cdot \Sigma^A$ but did not sell in $\Sigma' \cdot \Sigma^A$. The total revenue lost is at most

$$p_{\max} \cdot \frac{nM}{2} = \frac{\epsilon}{2} \cdot p_{\min} \cdot N \le \frac{\epsilon}{2} \cdot \text{OPT}.$$

An analogous argument will prove that the optimum solution on Σ' with revenue OPT' will induce a solution on Σ with revenue at least $(1 - \epsilon/2)\text{OPT}'$. It can then be inferred that the optimum solution for Σ' will induce a solution on Σ with revenue at least $(1-\epsilon)\text{OPT}$. □

We combine the two preprocessing steps with the dynamic programming algorithm to obtain an approximation scheme that runs in polynomial time when the number of price levels k is constant.

Theorem 19. *For any $\epsilon > 0$, dynamic programming with preprocessing will yield an algorithm with approximation ratio at least $1 - \epsilon$ that runs in time $O\left(n^{12k+16} (R/\epsilon)^{6k+8}\right)$.*

5 Extension to Buying

In the buying case, the algorithm's task is to insert buy orders into the order sequence in order to buy at least N shares, with the goal of minimizing the total cost of the trade.

We note that there is an asymmetry between the profit maximization (selling case) and the cost minimization (buying case) online trading problems. The results of Steger *et al.* [5] for the *min-search* problem imply that no algorithm can achieve a competitive ratio better than $O(\sqrt{R})$. For improved guarantees, we consider offline algorithms. The dynamic programming algorithm can be easily modified to the buying case. During the rounding step , however, we must set $M = \alpha N/n$, for any $\alpha > 0$.

Theorem 20. *Let OPT be the cost of the offline optimum solution that buys exactly N shares. For any $\alpha > 0$, the dynamic programming with preprocessing will yield an algorithm that buys at least $(1-\alpha)N$ shares with cost at most OPT. The algorithm runs in time $O\left(n^{12k+16}\left(1/\alpha\right)^{6k+8}\right)$.*

References

1. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
2. R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139, 2001.
3. E. Even-Dar, S. M. Kakade, M. S. Kearns, and Y. Mansour. (In)Stability properties of limit order dynamics. In *ACM Conference on Electronic Commerce*, pages 120–129, 2006.
4. S. Kakade, M. J. Kearns, Y. Mansour, and L. E. Ortiz. Competitive algorithms for VWAP and limit order trading. In *ACM Conference on Electronic Commerce*, pages 189–198, 2004.
5. J. Lorenz, K. Panagiotou, and A. Steger. Optimal algorithms for k-search with applications in option pricing. In *ESA 2007, Proceedings 15th Annual European Symposium*, pages 275–286, 2007.