Multi-Attribute Exchange Market: Representation and Indexing of Orders

Eugene Fink
Computer Science and Eng.
University of South Florida
Tampa, Florida 33620
eugene@csee.usf.edu

Josh Johnson
Electronic Arts, Tiburon
2301 Lucien Way, Suite 395
Maitland, Florida 32751
joshjohnson@cfl.rr.com

John Hershberger
Computer Science and Eng.
University of South Florida
Tampa, Florida 33620
jhershbe@csee.usf.edu

ABSTRACT

We present an automated exchange for trading complex goods, such as used cars, which allows traders to describe desirable purchases and sales by multiple attributes. The developed exchange system supports markets with up to 300,000 orders.

Categories and subject descriptors:
J.1 [Administrative data processing]: Financial

General terms: Design, experimentation, performance.

Keywords: E-commerce, electronic trading, exchange markets, multi-attribute commodities.

1. INTRODUCTION

The growth of the Internet has led to the development of on-line marketplaces, including bulletin boards, auctions, and exchanges. Electronic bulletin boards help buyers and sellers find each other; however, they often require customers to invest significant time into reading multiple ads, and many buyers prefer on-line auctions, such as eBay. Auctions have their own problems, including high computational costs, lack of liquidity, and asymmetry between buyers and sellers. Exchange-based markets support fast-paced trading and ensure symmetry between buyers and sellers; however, they require rigid standardization of tradable items. For example, the New York Stock Exchange allows trading of about 3,000 stocks, and a buyer or seller has to indicate a specific stock. For most goods, the description of a desirable trade is more complex; for instance, a car buyer needs to specify a model, options, color, and other features.

A recent project at the University of South Florida has been aimed at developing an exchange for multi-attribute goods [1, 2, 3]. We have defined trading semantics and implemented an exchange that supports large-scale markets.

2. ORDER REPRESENTATION

We give an example of a multi-attribute market and outline a general model for trading multi-attribute goods.

Example. We consider an exchange for trading new and used cars. To simplify this example, we assume that a trader can describe a car by four attributes: model, color, year, and mileage. A prospective buyer can place a *buy order,* which includes a description of a desired car and a maximal acceptable price; for instance, she may indicate that she wants a red Mustang, made after 2000, with at most 20,000 miles, and she is willing to pay $19,000. Similarly, a seller can place a *sell order;* for example, a dealer may offer a brand-new Mustang of any color for $18,000. An exchange system must generate transactions that satisfy buyers and sellers. In the previous example, it must determine that a brand-new red Mustang for $18,500 satisfies the buyer and dealer.

Attributes. We define goods in a specific market by a list of attributes; as a simplified example, we describe cars by model, color, year, and mileage. An attribute may be a set of explicitly listed values, such as the car model and color, or an interval of numbers, such as the year and mileage.

Item sets. When a trader places an order, she has to specify some set I_1 of acceptable values for the first attribute, some set I_2 for the second attribute, and so on. The resulting set I of acceptable items is the Cartesian product $I = I_1 \times I_2 \times \dots$. For example, suppose that a car buyer is looking for a Mustang or Camaro, the acceptable colors are red and white, the car should be made after 2000, and it should have at most 20,000 miles; then, the item set is $I = \{\texttt{Mustang}, \texttt{Camaro}\} \times \{\texttt{red}, \texttt{white}\} \times [2001..2003] \times [0..20{,}000]$.

Price functions. A trader should specify a limit on the acceptable price. For instance, a car buyer may be willing to pay $19,000 for a Mustang, but only $18,000 for a Camaro, and she may subtract $1 for every ten miles on the odometer. Formally, a price limit is a real-valued function defined on the set I; for each item $i \in I$, it gives a certain limit $\mathit{Price}(i)$. If the price function is a constant, it can be specified by a numeric value; else, it is encoded by a C++ procedure that inputs an item and outputs the corresponding price limit. For a buyer, $\mathit{Price}(i)$ is the maximal acceptable price; for a seller, it is the minimal acceptable price.

Order sizes. If a trader wants to buy or sell several identical items, she can include their number in the order specification. She can specify not only an overall order size but also a minimal acceptable size. For instance, suppose that a Toyota wholesale agent is selling one hundred cars, and she works only with dealerships that are buying at least ten cars. Then, the overall size of her order is one hundred, and the minimal size is ten.

Fills. An order specification includes an item set I, price function *Price*, overall order size *Max*, and minimal acceptable size *Min*. When a buy order matches a sell order, the corresponding parties can complete a trade; we use the term *fill* to refer to the traded items and their price. We define a fill by a specific item i, its price p, and the number of traded items, denoted *size.* If $(I_b, \mathit{Price}_b, \mathit{Max}_b, \mathit{Min}_b)$ is a buy order, and $(I_s, \mathit{Price}_s, \mathit{Max}_s, \mathit{Min}_s)$ is a matching sell order, then a fill (i, p, size) must satisfy three conditions:

- $i \in I_b \cap I_s$.
- $\mathit{Price}_s(i) \le p \le \mathit{Price}_b(i)$.
- $\max(\mathit{Min}_b, \mathit{Min}_s) \le \mathit{size} \le \min(\mathit{Max}_b, \mathit{Max}_s)$.

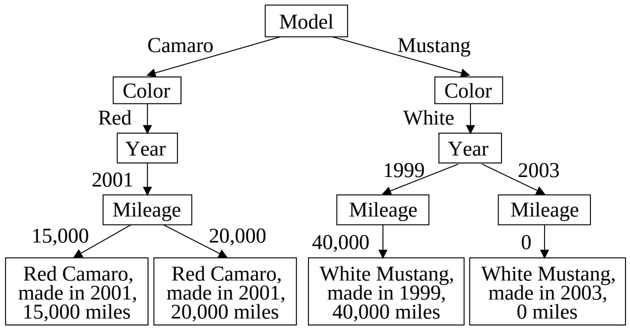
Model
Camaro
Mustang
Color
Red
Color
White
Year
2001
Year
1999
2003
Mileage
15,000
20,000
Mileage
40,000
Mileage
0
Red Camaro, made in 2001, 15,000 miles
Red Camaro, made in 2001, 20,000 miles
White Mustang, made in 1999, 40,000 miles
White Mustang, made in 2003, 0 miles

Figure 1: Indexing tree for a used-car market.

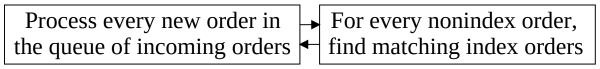
Process every new order in the queue of incoming orders
For every nonindex order, find matching index orders

Figure 2: Main loop of the matcher.

3. EXCHANGE SYSTEM

The developed exchange consists of a central matcher and user interfaces that run on separate machines. The traders enter their orders through interface machines, which send the orders to the matcher. The matcher includes a central structure for indexing of orders with fully specified items. If we can put an order into this structure, we call it an *index order.* If an order includes a set of items, rather than a fully specified item, it is added to an unordered list of *nonindex orders.* This indexing scheme allows fast retrieval of index orders that match a given order; however, the system does not identify matches between two nonindex orders.

The indexing structure consists of two identical trees: one is for buy orders, and the other is for sell orders. The height of an indexing tree equals the number of attributes, and each level corresponds to one of the attributes (Figure 1). A node at level i divides orders by the values of the ith attribute, and each node at level $(i+1)$ corresponds to all orders with specific values of the first i attributes. To find matches for a given order, the system identifies all children of the root that match the first attribute of the order's item set, and then recursively processes the respective subtrees.

In Figure 2, we show the main loop of the matcher, which alternates between processing new orders and identifying matches for old orders. When it receives a new order, it immediately identifies matching index orders. If there are no matches, and the new order is an index order, then the system adds it to the indexing structure. Similarly, if the system fills only part of a new index order, it stores the remaining part in the indexing structure. If it gets a nonindex order and does not find a complete fill, it adds the unfilled part to the list of nonindex orders. After processing all new orders, the system tries to fill old nonindex orders; for each nonindex order, it identifies matching index orders.

4. PERFORMANCE

We have applied the system to an extended used-car market and to a commercial-paper market. We have run the matcher on a 2-GHz Pentium computer with one-gigabyte memory. A more detailed report of the experimental results is available in Johnson's masters thesis [3].

Used cars. We have considered a used-car market that includes all models offered by AutoNation, described by eight attributes: transmission (2 values), number of doors (3 values), interior color (7 values), exterior color (52 values), year (103 values), model (257 values), option package (1,024 values), and mileage (500,000 values). We have varied the number of orders from one to 300,000, and measured the time of one pass through the system's main loop (Figure 2). We have also measured the maximal acceptable rate of placing new orders, called *throughput;* if the system gets more orders per second, the number of unprocessed orders keeps growing and eventually leads to an overflow. We show the dependency of the performance on the number of orders in Figure 3 (solid lines). The system scales to markets with 300,000 orders, and it processes 500 to 5,000 new orders per second. The main-loop time is approximately linear in the number of orders. The throughput in small markets grows with the number of orders; it reaches a maximum when the market grows to about four hundred orders, and slightly decreases with further increase in the market size.

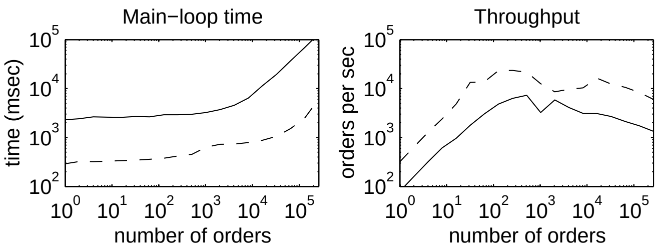

Figure 3: Dependency of the performance on the number of orders for the used-car market (solid lines) and commercial-paper market (dashed lines).

Commercial paper. When a large company needs a short-term loan, it may issue *commercial paper,* which is a fixed-interest "promissory note" similar to a bond. The company sells commercial paper to investors, and later returns their money with interest; the payment day is called the *maturity date.* The appropriate interest depends on the current rate of US Treasury bonds, company's reputation, and paper's time until maturity. We have described commercial paper by two attributes: maturity date (2,550 values) and company (5,000 values). We plot the dependency of the performance on the number of orders in Figure 3 (dashed lines). The system scales to markets with 300,000 orders, and it processes 2,000 to 20,000 new orders per second. The main-loop time is linear in the number of orders, whereas the throughput first increases and then slightly decreases with the number of orders.

5. ACKNOWLEDGMENTS

This work has been partially sponsored by the DYNAMiX Technologies Corporation and by the National Science Foundation grant No. EIA-0130768.

6. REFERENCES

[1] Jianli Gong. Exchanges for complex commodities: Search for optimal matches. Master's thesis, Computer Science and Engineering, University of South Florida, 2002.

[2] Jenny Hu. Exchanges for complex commodities: Representation and indexing of orders. Master's thesis, Computer Science and Engineering, University of South Florida, 2002.

[3] Josh Johnson. Exchanges for complex commodities: Theory and experiments. Master's thesis, Computer Science and Engineering, University of South Florida, 2001.