On Bid Selection Heuristics for Real-Time Auctioning for Wide-Area Network Resource Management*

Chunming Chen, Muthucumaru Maheswaran, and Michel Toulouse

Advanced Networking Research Laboratory
Department of Computer Science
University of Manitoba
Winnipeg, MB R3T 2N2
Canada

1. Introduction

The rapid advancements in computer and networking technologies coupled with the emergence of new classes of applications with significantly increased resource requirements tend to rapidly depreciate the value of state-of-the-art computer and networking equipments. Therefore, it has become increasingly important to maximize the utilization of the resources so that users can recoup their investment. This paper describes the design of an auction system that can be used for trading computational resources in a wide-area network. More specifically this paper proposes and analyzes several heuristics that can be used to select the "winning" bids in such an auction system.

The real-time auctioning system for resource trading in wide-area networks that is developed as part of this study is called the *Computation Market (CM)* [ChM00]. The universal connectivity made available by the Internet provides an ideal setting for the deployment of a system such as the CM. The model provided by CM supports on-demand acquisition of resources for temporary use. A facility such as this is crucial for the deployment of next generation Internet services [GrW00].

The CM architecture divides the wide-area network into regions called *local markets.* A local market has an auction server and manages the resource transactions in the given network vicinity. The first level of CM focuses on intra-market resource flows and the second level of CM handles the inter-market resource flows. The core of the CM system is a bidding protocol. In the CM system, the auction server receives bids for the available resources and uses some "profit" maximization strategy to select the set of bids that should receive the resources. Because the CM allows the users to bid on different combinations of resources, determining the winning bids becomes NP-complete [San99]. To address this problem, we propose several heuristics. We evaluate the performance of the proposed heuristics through simulations. The heuristics are also compared with an upper bound of the optimal scheme.

Section 2 reviews the related work in bid selection algorithms. Section 3 presents the system architecture and explains why a fast algorithm is needed for bid selection. Section 4 describes the proposed heuristics for selecting the winning set of bids in the combinatorial auction. The simulation results are examined in Section 5.

2. Related Work

Due to space limitations, we provide a very brief survey of the existing literature here. An extensive survey of the literature can be found in [ChM00]. Here we discuss auction-based resource management systems from wide-area computing and combinatorial auctioning systems.

Spawn [WaH92] is an implementation of distributed computational economy that uses money, price, and auction as mechanisms for exchange of resources. Our approach is distinct from Spawn in four aspects. First, Spawn is not an Internet-based system. Second, Spawn uses Vickery

* This research is supported by a TR*Labs* Scholarship.

(second-price sealed-price) auction, while CM uses English (first-price open-cry) auction. Third, in Spawn, each machine maintains an auction mechanism, while 0.CM uses a domain-based auction mechanism with each local market having its own auction server. In our approach, a user can bid for resources within a local market and resources flow from surplus to scarce markets.

Mariposa [StA95] implements an economic paradigm for managing query execution and storage management in a wide-area distributed database system. To decide where to run a query, a distributed advertising service is designed to provide information for finding sites that might want to bid on a query or portions of a query. During the bidding process, two protocols could be involved. One is a two-phase based expensive bid protocol, and the other is a purchase order protocol. Our two-level bidding architecture is similar to the architecture in Mariposa in several aspects. Our scheme allows bidding on various combinations of resources, which is not present in Mariposa.

Mark [WeM98, WeW98] is an ongoing project that uses market-based adaptive architectures for information survivability. Mark also implements an on-line market architecture, which is built on the top of a general purpose Internet auction server. The Java Market project [AmB98] is another working system that aims to transform the Internet into a metacomputing system. The Java Market allows the producers and consumers to access the Java Market web pages anywhere on the Internet by simply running secure Java Applets on the web browser. The major difference between their model and ours is that their model lacks a hierarchical structure, which limits the scalability. Also, their model is not developed to provide efficient aggregation of resources that are located in the same network neighborhood into clusters.

A search algorithm for optimally selecting a subset of winning bids is presented in [San99]. The search algorithm consists of four preprocessing steps and the main search. A special bid tree is used in the main search. The bid tree is a binary tree with the bids inserted at the leaves. The algorithm also uses an iterative deepening A* search strategy to speedup the main search. Similar to their algorithm, CM also uses heuristics to solve the winner determination problem in combinatorial auctions. However, our problem is more general because we can have multiple units for a particular item.

Combinatorial auction multi-unit search (CAMUS) [LeS00] introduces a branch-and-bound technique for selecting the bids. Their search procedure also uses some heuristics ideas similar to the ones used in this paper. Another work that also uses a branch-and-bound technique to solve the multi-unit combinatorial auction problem is presented in [GoL00]. Their experiments show that the branch-and-bound techniques require both an upper bound for the value of best allocation and a good criterion to decide which bids are to be tried first. They suggest making use of average price per unit or an average price per unit related criteria in a branch-and-bound algorithm, which is quite similar to the use of rate to rank the bids in our bidding selection algorithm. A future study will compare our heuristics that are optimized for speed against the above algorithms.

3. System Architecture

The CM is an interconnection of several local markets. A local market is made up of an *auction server* (AS), *local brokers* (LBs), *supplier agents* (SAs), and *consumer agents* (CAs). The SAs represent machines that provide resources such as CPU, memory, disk while the CAs represent clients that are willing to pay in some currency for those resources.

The resources are described using key performance attributes such as CPU speed, memory size, disk capacity, and I/O bandwidth and platform attributes such as operating system and compiler support. Users may specify the attributes of the desired resource using attribute-value pairs [RaL98]. To make the bid selection process manageable, we group the resources into a predefined set of hierarchical classes based on their attributes [ChM00].

Three major functions are offered by the AS: (a) a virtual market for local clients; (b) trading services in the global market; and (c) selection of the winning bids. The AS earns its revenue from three sources: (a) transaction fees charged on SAs and CAs; (b) profit made by selling cheap remote resources in the local market; and (c) profit made by selling surplus local resources in the global market. The goal of the AS is to maximize the profits from the services it offers. The LBs examine the bid post requests from the suppliers and determine whether different bid posts for independently managed but co-located resources could be grouped together to form clusters. The

aggregation of resources in this manner provides an efficient solution to the co-allocation problem [FoK99]. The combining performed by the local brokers does not enforce the resources to be allocated as a block, i.e., the AS can "unbundle" the resources. The SAs and CAs are responsible for handling the submission and consumption of resources, respectively. The SA sets the *reserved price* (the lowest acceptable price) for the resource and the lowest acceptable increment between two bid prices for the resource. The CA is responsible for generating the bids taking the performance and desirability of the resources into consideration.

In the CM, we use the English (first price open out-cry) auction. For a resource, the AS only accepts bids from a CA that are higher than the reserved price. Each CA bids iteratively for an item, it's required that each subsequent bid should be higher than the current highest bid. The auction closes at a predefined time and the winning bids are determined by the AS. In CM, we may have n_k resources of class k, where $n_k >> 1$. In this situation, multiple bids will be accepted for such resource or resource combinations. The per-unit price for such resource classes are set at the lowest price out of the accepted bids.

A CA can bid up to the maximum number of available items. We assume "universal" connectivity within a local market (i.e., communication cost is uniform irrespective of the source and destinations). An extension of this model is to include non-uniform intra-market connectivity and add "anonymous" remote market resources. The bidding process within a local market has is a two-step process.

i) The CAs bid on combinations of items that are posted at the AS with a price higher than the current price.

ii) The highest bidding price is reflected at the AS. The CAs can query the current highest price at the AS and review their bid if necessary.

Steps i) and ii) may be iterated until bid closing. A CA may use some "utility" function to determine whether they should out bid other CAs. The two-phase iterative bidding protocol can be communication intensive depending on the number of iteration performed until closing the bids.

In CM, the AS closes the auctions periodically. The period in between two auction closings is called an *auction session.* Once the AS closes an auction session, it needs to decide the winning CAs. The time available for selecting the winners is constrained by the start time of the next auction session and the "available" times of the resources. When SAs advertise their resources in advance, the AS might be able to use computationally expensive bid selection algorithms that are closer to optimal. However, if the auctions are proceeding continuously with periodic closings, the bid selections should be done fast or the auctions should continue without the knowledge of the previous assignments. This may affect the trading behavior of the CAs and SAs.

During an auction session, a local market may have higher demand than supply resulting in higher prices whereas another local market may have higher supply than demand resulting in lower prices. If remote resources can be consumed by at least some CAs, then inter-market resource flows should be carried out to even the supply and demand disparities. This paper, however, does not focus on inter-market resource flows [ChM00].

4. Bid Selection Heuristics

The bid selection algorithm uses a bid tree [San99] that is essentially a binary tree with its depth equals to the number of resource classes. Our bid tree (simple example shown in Figure 1) is different from that of [San99] because in our system each resource class can have multiple units. Assume three resource classes with {20, 50, 30} number of resources in each class and with a five history average selling price of {3.5, 2.3, 0.6}, i.e., average selling price for the items over the previous five auction closings. These prices are used to compute the *rate*. The *rate* is defined value of bid based on the offered price divided by the value of the bid based on the five history average price. The bids submitted to the auction are shown in Table 1

. The bids are inserted in the leaves of the bid tree. The path from the root to a leaf represents the classes of resources in a bid at the leaf. From any node, the left branch leads to a bid that includes the corresponding class of resource and the right branch otherwise.

Table 1: An example list of bids submitted to the AS.

Bid no.	Bidder	No. of items	Prices	Rate
1	3	{5, 10, 0}	{3.2, 2.3, 0}	0.963
2	5	{3, 5, 0}	{3.5, 2.3, 0}	1
3	1	{8, 0, 12}	{3.3, 0, 0.6}	1.012
4	6	{2, 0, 0}	{4, 0, 0}	1.143
5	12	{6, 0, 0}	{3.5, 0, 0}	1
6	4	{0, 10, 0}	{0, 2.3, 0}	1
7	8	{0, 5, 0}	{0, 2.4, 0}	1.043
8	2	{0, 8, 0}	{0, 2.5, 0}	1.087
9	10	{0, 30, 20}	{0, 2.3, 0.6}	1
10	7	{0, 5, 10}	{0, 2.4, 0.6}	1.087
11	9	{0, 0, 5}	{0, 0, 0.6}	1
12	11	{0, 0, 30}	{0, 0, 0.65}	1.083

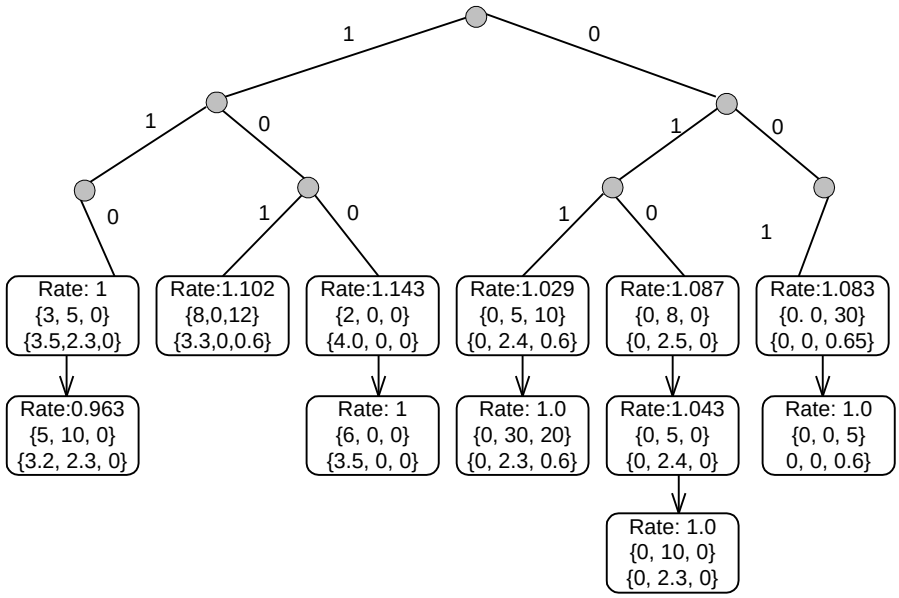

Figure 1: The bidtree data structure for the bid in Table 1.

The algorithm uses two stages in selecting the winning bids: the primary and secondary phases. We assume that the resource classes are arranged in descending order by their expected price. The primary phase starts with class 1 resources and adds bids until it can't find any such bids or they cannot be included due to resource exhaustion. Then it adds class 2 resource and so on.

A "selection mask" of size *N* is used to implement the primary phase. The *n*-th location of the mask corresponds to class n. A location in the mask has three states: "enabled," "disabled," and "any." If a class's corresponding mask value is "enabled," the search space includes those bids that consist of this resource class. If a class's corresponding variable is "disabled," the search space excludes those bids that consist of this resource class. If a class's corresponding variable is "any," this resource class can be in or out of the search space. The primary phase algorithm is given below.

1) The search starts with the first value "enabled" in the selection mask and others with "any." For example, the selection mask values {"enabled," "any," "any"} search the bids that can be reached by traversing the paths {111, 110, 101, 100}.

2) The search determines the next bid to be added by considering the bid with the highest "fitness" in the search space. By choosing different parameters such as price, value, or rate, we can create a family of heuristics. If the available

resources are enough to be allocated to the current highest rate bid, then move the bid to the *solution list.*

3) Repeat step 2 until no bids can be added to the solution list.

4) Change the "enabled" selection mask variable to "disabled" and set the next selection mask location to "enabled." The search space changes due to the changes in the selection mask.

5) Repeat step 2, 3, and 4 until all the selection mask locations are "disabled." The primary phase terminates with a "disabled" selection mask.

The solution obtained by the primary phase is improved by a secondary (refinement) phase. The algorithm records the minimum bid rate (b) in the solution list after the primary phase. During the refinement phase, we only consider those bids that remain in the bidtree with rates that are higher than b. Those bids are moved from the bidtree to a back-list. The back-list can be sorted by rate, value, or price, depending on which parameter is used in the refinement phase. Assume after the primary phase, the revenue is *m*. The following is a brief description of the refinement phase.

The refinement phase examines the bids in the back-list starting at the top of the list. For each bid, it determines whether sufficient resources are available to accommodate the bid. If resources are found, the bid is inserted into a new solution list. Otherwise, bids are removed from the previous solution list to make room for the new bid. Once enough space is found, the bid from the back-list is inserted into the solution list. There may be some space for reinserting some of the removed bids or some more bids from the back-list into the solution list due to the removal of bids from the solution-list. A series of adds and removes may have been performed on the solution-list at this time. To determine the impact of these changes, the total revenue provided by the current version of the solution-list is evaluated. Let this revenue be *p*. This revenue is noted down in a history-list and if $p > 0.8*m$, the search continues examining the rest of the bids on the back-list.

The modifications performed to the solution list are tagged by the last known revenue and recorded in a history-list. Once the refinement phase terminates due to the revenue falling below $0.8m$ or empty back-list, the history-list is used to roll-back to the best solution that was found during the refinement phase.

5. Simulation Results and Discussion

In this section, we present the results from the simulations that evaluate the performance of the bid selection heuristics under different scenarios. All heuristics are based on the primary and refinement phases explained in Section 4. In the first heuristic called the VR, we use the bid value (V) to order the entries at the leaves of the bidtree and then refine by the rate (R). The second heuristic (RV) uses rate for the primary phase and value for the refinement phase. Similarly, the third heuristic RP uses rate for the primary phase and price for the refinement phase.

Table 2 shows the relative performance of the heuristics for test case A: 1000 bids, 500 mean items per bid, and 4 mean classes per bid. Table 3 shows the relative performance of heuristics for test case B: 1000 bids, 100 mean items per bid, and 4 mean classes per bid. Table 4 shows the relative performance of the heuristics for test case C: 1000 bids, 100 mean items per bid, and 2 mean classes per bid. Each data point in the above tables is the average of 100 simulation runs. There were fifteen classes of resources in the simulations. The number of resources per class (i.e., number of machines per class) was set arbitrarily from the range (2000-2250) and the reserve price for the resources was set from the range (0.5-4.5).

Table 2: Performance of the bid selection heuristics for test case A.

Heuristics	Upper bound	Primary phase	P-P as percent of upper bound	Refinement phase	Improvement through refinement	Final result as percent of upper bound
VR	98226	79557	80.99 %	84411	6.10 %	85.94 %
RV	98226	84886	86.42 %	85388	0.60 %	86.93 %
RP	98226	84886	86.42 %	87265	2.80 %	88.84 %

Table 3: Performance of the bid selection heuristics for test case B.

Heuristics	Upper bound	Primary phase	P-P as percent of upper bound	Refinement phase	Improvement through refinement	Final result as percent of upper bound
VR	92001	79001	85.87 %	81847	3.60 %	88.96 %
RV	92001	83795	91.08 %	84005	0.25 %	91.31 %
RP	92001	93795	91.08 %	85231	1.71 %	92.64 %

Table 4: Performance of the bid selection heuristics for test case C.

Heuristics	Upper bound	Primary phase	P-P as percent of upper bound	Refinement phase	Improvement through refinement	Final result as percent of upper bound
VR	102516	80870	78.88 %	87924	8.72 %	85.76 %
RV	102516	88851	86.67 %	89532	0.77 %	87.33 %
RP	102516	88851	86.67 %	92428	4.02 %	90.16 %

The results show that bid value based primary phase has the worst results. For test cases A, B, and C, bid value based primary phase results are (80.99%, 85.87%, 78.88%) as comparing to (86.42%, 91.08%, 86.67%) for rate based primary phase. Particularly interesting is that observation that when average bid size is small (test case B), the primary phase yields better results than with bigger average bid sizes (test cases A and C). Further, the rate based primary phase combined with price based refinement performs the best in all test cases. This suggests that price (or price related factor such as rate) has the most impact on the revenue. The VR approach has the biggest refinement (6.10%, 3.60%, 8.72%) in any of the test cases. It can be observed that bigger the bid size is the refinement phase benefits more. However, irrespective of the refinement improvement, the final results of VR are always worse than RV and RP. Intuitively, this suggests that considering the most promising bids in terms of producing the revenue first has a significant impact on the final result.

6. Conclusion and Future Work

This study focuses on the development of an auctioning based “on demand” resource acquisition and/or trading system for wide-area network computing systems called the *Computation Market* (CM). This paper, in particular, focuses on designing bid selection heuristics that can be used by the auctioning system that is part of CM. We consider a real-time auctioning system that periodically closes the bidding process and chooses the winners. To address the scalability, site autonomy, heterogeneity, and extensibility we organize the CM as an interconnection of local markets. A local market spans a limited network vicinity.

Although auctioning has been studied in several contexts, only recently researchers have started examining the multi-unit combinatorial auctioning problem [GoL00, LeS00]. The combinatorial auctioning problem is NP-complete even for the single unit case [San99] and it is even harder to solve for the multi-unit case. Because we are examining the real-time variant of the problem, we need to solve the winner determination problem as soon as possible. Motivated by this need for a “speedy” solution, we present a class of fast heuristics in this paper.

The heuristics presented are within 10-15% of an upper bound for the optimal solution. The heuristics have two phases. Depending on the time constraints, we could use the primary phase or both phases. From the simulation studies, we note that the refinement phase improves the solution obtained by the primary phase in all cases. However, the margin of improvement depends on what parameter was used for ordering the bids.

Several aspects this problem needs further examination. Because there aren’t any existing systems like CM, our evaluation studies (for the heuristics) were based on synthetic data. Recently, a proposal has been made to develop a unified test suite for combinatorial auctions [LeP00]. Suitability of this test suite for the wide-area networking problem domain will be studied in the future. Further, our heuristics will be compared with

branch-and-bound based solutions [GoL00, LeS00] for speed and quality.

References

[AmB98] Y. Amir, B. Awerbuch, and R. S. Borgstrom, *The Java Market: Transforming the Internet into a Metacomputer*, Technical Report, CNDS-98-1, Department of Computer Science, John Hopkins University, 1998.

[ChM00] C. Chen, M. Maheswaran, and M. Toulouse, *Computation Market: An Online Auction for Wide-Area Network Computing Systems*, Technical Report TR-CS-01-10, Department of Computer Science, University of Manitoba, 2000.

[FoK99] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Fransisco, CA, 1999.

[GoL00] R. Gonen and D. J. Lehmann, "Optimal olutions for multi-unit combinatorial auctions: Branch and bound heuristics," *ACM Conference on Electronic Commerce*, 2000, pp. 13-20.

[GrW00] S. D. Gribble, M. Welsh, R. von Behren, E. A. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R.H. Katz, Z.M. Mao, S. Ross, and B. Zhao, "The Ninja Architecture for Robust Internet-Scale Systems and Services," *Computer Networks (Special Issue on Pervasive Computing)*, 2000.

[LeP00] K. Leyton-Brown, M. Pearson, and Y. Shoham, "Towards a Universal Test Suite for Combinatorial Auctions," *2000 ACM Conference on Electronic Commerce (EC'00)*, 2000.

[LeS00] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz, "An Algorithm for Multi-Unit Combinatorial Auctions," *17th National Conference on Artificial Intelligence*, 2000.

[RaL98] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed resource management for high throughput computing," *7th IEEE International Symposium on High Performance Distributed Computing*, 1998, pp. 28-31.

[San99] T. Sandholm, "An Algorithm for Optimal Winner Determination in Combinatorial Auctions," *International Joint Conference on Artificial Intelligence (IJCAI)*, 1999, pp. 542-547.

[StA95] M. Stonebraker, P. M. Aoki, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu, "Mariposa: A wide-area distributed database system," *VLDB Journal*, Vol. 5, No. 1, Jan. 1996, pp. 48-63.

[WaH92] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta, "Spawn: A Distributed Computational Economy," *IEEE Transaction on Software Engineering*, Vol. 18, No. 2, Feb. 1992.

[WeM98] M. P. Wellman, J. K. Mackie-Mason, and S. Jamin, "Market-based adaptive architectures for information survivability," `http://www.darpa.mil/ito/psum1998`, 1998.

[WeW98] M. P. Wellman and P. R. Wurman, "Real time issues for Internet auctions," *First IEEE Workshop on Dependable and Real-Time E-Commerce Systems (DARE-98)*, June 1998.