Web Intelligence and Agent Systems: An International Journal 5 (2008) 1–5
IOS Press

Fine-Grained Efficient Resource Allocation Using Approximated Combinatorial Auctions

A Parallel Greedy Winner Approximation for Large-Scale Problems

Naoki Fukuta [a,*] and Takayuki Ito [b,**]
[a] *Faculty of Informatics, Shizuoka University, 3 5 1 Johoku Hamamatsu, Shizuoka, Japan*
E-mail: fukuta@inf.shizuoka.ac.jp
[b] *Center for Collective Intelligence, Sloan School of Management, Massachusetts Institute of Technology, 5 Cambridge Center, Cambridge MA 02142,USA*
E-mail: takayuki@mit.edu

Abstract. Combinatorial auctions, one of the most popular market mechanisms, have a huge effect on electronic markets and political strategies. Combinatorial auctions provide suitable mechanisms for efficient allocation of resources to self-interested attendees. On the other hand, efficient resource allocation is also becoming crucial in many computer systems that should manage resources efficiently. Considering ubiquitous computing scenarios, the ability to complete an auction within a fine-grained time period without loss of allocation efficiency is in strong demand. Furthermore, to achieve such scenarios, it is very important to handle a large number of bids in an auction. In general, the optimal winner determination problem of a combinatorial auction is NP-hard. Thus, much work focuses on tackling the computational costs for winner determination. In this paper, we show that our approximation algorithms provide sufficient quality of winners for auctions that have a large number of bids on hard time constraints. Furthermore, we compare and discuss desirable properties of such approximation algorithms to be embedded in application systems.

Keywords: multi-agent system, combinatorial auction, optimization, approximation, resource allocation

1. Introduction

Combinatorial auctions [8], one of the most popular market mechanisms, have a huge effect on electronic markets and political strategies. For example, Sandholm et al. [36] proposed a market using their innovative combinatorial auction algorithms. Combinatorial auctions provide suitable mechanisms for efficient allocation of resources to self-interested attendees [8]. Therefore, many works have been done to utilize combinatorial auction mechanisms for efficient resource allocation. For example, the FCC tried to employ combinatorial auction mechanisms for assigning spectrums to companies [28].

On the other hand, efficient resource allocation is also becoming crucial in many computer systems that should manage resources efficiently, and combinatorial auction mechanisms are suitable for this situation. For example, considering a ubiquitous computing scenario, there is typically a limited amount of resources (sensors, devices, etc.) that may not cover all needs for all agents. Due to certain reasons (physical limitations, privacy, etc.), most of the resources cannot be shared with other agents. Furthermore, agents will use two or more resources at a time to achieve desirable services for users. Of course, each agent provides services to its own user, and the agent may be self-interested.

However, in such ubiquitous computing scenarios, there is strong demand for completing an auction within a fine-grained time period without loss of allocation efficiency. In a ubiquitous computing scenario,

*Corresponding author. E-mail: fukuta@inf.shizuoka.ac.jp.
**Visiting from Nagoya Institute of Technology, Japan

the physical location of users may always be changing and that could be handled by the system. Also, each user may have multiple goals with different contexts, and those contexts are also dynamically changing. Therefore, resources should be re-allocated in a certain fine-grained period to keep up with those changes in a timely manner. For better usability, the time period of resource reallocation will be 0.1 to several seconds depending on services provided there. Otherwise, resources will remain assigned to users who no longer need them while other users are waiting for allocation.

Also, in the above scenarios, it is very important to handle a large number of bids in an auction. Consider that if there are 256 resources and 100 agents, and each agent has 100 to 200 bids, then there will be 10,000 to 20,000 bids for 256 items in an auction. However, it has been difficult to complete such a large-scale combinatorial auction within a fine-grained time period.

In general, the optimal winner determination problem of a combinatorial auction is NP-hard [8] for the number of bids. Thus, much work focuses on tackling the computational costs for winner determination [12] [8] [36]. These works basically try to achieve the optimal solution in winner determination. Although these works can be used for attaining approximated solutions and they perform well in certain conditions, there remains a possibility for further improvements focusing on approximation.

Some works [26] [43] [18] try to achieve *approximate* solutions in winner determination. Lehmann et al. [26] proposed a greedy approximation algorithm that relaxes economic efficiency and proves the truthfulness in a certain condition. Lehmann et al. rather focused on elegant theoretical aspects, i.e., giving truthfulness for an auction mechanism with approximated allocations of items [26]. However, the performance of [26] is excellent in that it can successfully handle a very large number of bids and items in an auction.

Also, the approximation algorithms presented in [43] and [18] generally perform very well even if the number of bids is relatively large.

Recently, we proposed a set of extended approximation algorithms that are based on Lehmann's approach. In [14], we presented our preliminary idea and we have shown our algorithm performs well when the number of bids is very large. However, there is no detailed evaluation of the performance of those algorithms in the setting of a short time approximation that is needed in an actual usage scenario of the combinatorial auctions described above.

In this paper, we show that our approximation algorithms provide a sufficient quality of winners for auctions that have a large number of bids on hard time constraints. Furthermore, we compare and discuss desirable properties of such approximation algorithms to be embedded in application systems.

2. Preliminaries

2.1. Combinatorial Auction and Resource Allocation Problem

An auction mechanism is an economic mechanism for efficient allocations of items to self-interested buyers with agreeable prices. When the auction mechanism is truthful, i.e., it guarantees incentive compatibility, the mechanism enforces the bidders to locate their bids with true valuations. In such auctions, since we have an expectation of obtaining bids with true valuations, we can allocate items to buyers efficiently even though some buyers may try to cheat the mechanisms out of gaining sufficient incomes from them. For example, Vickrey proposed an auction mechanism that has incentive compatibility [38]. That is a basic difference from ordinary resource allocation mechanisms that have implicit assumptions of truth-telling buyers.

Combinatorial auction is an auction mechanism that allows bidders to locate bids for a bundle of items rather than single item [8]. Combinatorial auction has been applied for various resource allocation problems. For example, McMillan et al. reported a trial on an FCC spectrum auction [28]. Rassenti et al. reported a mechanism for an airport time slot allocation problem [32]. Ball et al. discussed applicability of combinatorial auctions to airspace system resource allocations [3]. Caplice et al. proposed a bidding language for optimization of procurement on freight transportation services [7]. Estelle et al. proposed a formalization on auctioning London Bus Routes [6]. Hohner et al. presented an experience on procurement auctions at a software company [17].

However, on emerging applications with such resource allocation problems, their problem spaces are larger, more complex, and much harder to solve compared to previously proposed applications. For example, Orthogonal Frequency Division Multiple Access (OFDMA) technology enables us to use a physically identical frequency bandwidth as virtually multiplied channels at the same time, and this causes the channel allocation problem to become more difficult [42]. Also

some recent wireless technologies allow us to use multiple channels on the same, or different physical layers (i.e., WiFi, WiMax, and Bluetooth at the same time) for attaining both peak speed and robust connectivity [34][29]. Furthermore, such resource allocation should be done for many ordinary users rather than a fixed limited number of flights or companies. Also the allocation should consider the contexts of users, which are dynamically changing through the time.

In this paper, to maintain simplicity of discussion, we only focus on utility-based resource allocation problems such as [37], rather than generic resource allocation problems with numerous complex constraints. The utility-based resource allocation problem is a problem that aims to maximize the sum of utilities of users for each allocation period, but does not consider other factors and constraints (i.e., fair allocation [33] [1], security and privacy concerns [41], uncertainty[40][39], etc). Combinatorial auction is not only the approach for resource allocation problems. For example, Crawford et al. proposed a mechanism for multi-agent meeting scheduling problem[9] based on Clake Tax mechanism, and Camorlinga et al. proposed a swarm intelligence approach for distributed resource allocation. In this paper, to clarify the focus of discussion, we focus on solving the problem by using combinatorial auctions. We also assume private value auction model, i.e., each utility is private value for each user. There are studies that does not use private value auctions[19]. The consideration of these complex problem settings is a future work.

Also, throughout this paper, we only consider auctions that are single-sided, with a single seller and multiple buyers to maintain simplicity of discussion. It can be extended to the reverse situation with a single buyer and multiple sellers, and the two-sided case. The two-sided case is known as the combinatorial exchange. In the combinatorial exchange mechanisms, multiple sellers and multiple buyers are trading on a single trading mechanism. About this mechanism, the process of determining winners is almost the same as single-sided combinatorial auctions. However, it is reported that the revenue division among sellers can be a problem. There are a lot of interesting studies on combinatorial exchange [30]. The consideration and enhancement of support for combinatorial exchange is one of our future works.

2.2. Winner Determination Problem

The winner determination problem on combinatorial auctions is defined as follows [8]: The set of bidders is denoted by $N = 1, \ldots, n$, and the set of items by $M = \{m_1, \ldots, m_k\}$. $|M| = k$. Bundle S is a set of items: $S \subseteq M$. We denote by $v_i(S)$, bidder i's valuation of the combinatorial bid for bundle S. An allocation of the items is described by variables $x_i(S) \in \{0, 1\}$, where $x_i(S) = 1$ if and only if bidder i wins bundle S. An allocation, $x_i(S)$, is feasible if it allocates no item more than once,

$$\sum_{i \in N} \sum_{S \ni j} x_i(S) \leq 1$$

for all $j \in M$. The winner determination problem is the problem to maximize total revenue

$$\max_X \sum_{i \in N, S \subseteq M} v_i(S) x_i(S)$$

for feasible allocations $X \ni x_i(S)$. [1]

Even when we only focus on utility-based resource allocation problems, they force us to solve the winner determination problem with a very hard time constraint for achieving fine-grained resource allocation. Here, we have to consider that, in such resource allocation procedures, we need to spend much time for pricing and communications for actual resource allocation protocols. Therefore, we need a fast winner determination algorithm for auctions with a large number of bids. In this paper, primarily we focus on solving this problem, and then discuss other parts of the problem such as pricing, communication overheads, etc.

2.3. Lehmann's Greedy Winner Determination

Lehmann's greedy algorithm [26] is a very simple but powerful linear algorithm for winner determination in combinatorial auctions. Here, a bidder declaring $< s, a >$, with $s \subseteq M$ and $a \in \mathcal{R}_+$ will be said to put out a bid $b =< s, a >$. Two bids $b =< s, a >$ and $b' =< s', a' >$ conflict if $s \cap s' \neq \emptyset$. The greedy

[1] Note that in ordinary auction mechanisms, the actual winner's payment will not be the same as the price of the placed bid. Therefore, the actual income of the auctioneer is not the same as the total revenue calculated here. Since our goal is to compare optimality of winner determination, we simply use total revenue as the sum of prices of winner bids.

algorithm can be described as follows. (1) The bids are sorted by some criterion. The researchers [26] proposed sorting list L by descending average amount per item. More generally, they proposed sorting L by a criterion of the form $a/|s|^c$ for some number c, $c \geq 0$, possibly depending on the number of items, k. (2) A greedy algorithm generates an allocation. L is the sorted list in the first phase. Walk down the list L, accepting bids if the items demanded are still unallocated and not conflicted.

In [26], Lehmann et al. argued that $c = 1/2$ is the best parameter for approximation when the norm of worst case performance is considered. Also they have shown that the mechanism is truthful when single-minded bidders are assumed and their proposed pricing scheme is used.

Example: Assume there are three items a, b, and c, and three bidders $Alice$, Bob, and $Charles$. $Alice$ bids 10 for a. Bob bids 20 for $\{b, c\}$. $Charles$ bids 18 for $\{a, b\}$. We sort the bids by the criterion of the form $a/\sqrt{|s|}$. $Alice$'s bid is calculated as $10/\sqrt{1} = 10$. Bob's bid is calculated as $20/\sqrt{2} = 14$ (approximately). $Charles$'s bid is calculated as $18/\sqrt{2} = 13$ (approximately). The sorted list is now Bob's bid $< \{b, c\}, 20 >$, $Charles$'s bid $< \{a, b\}, 18 >$, and $Alice$'s bid $< \{a\}, 10 >$. The algorithm walks down the list. At first, Bob wins $\{b, c\}$ for 20. Then, $Charles$ cannot get the item because his bid conflicts with Bob's bid. Finally, $Alice$ gets $\{a\}$ for 10.

2.4. Zurel's Approximation

Zurel and Nisan [43] proposed a very competitive approximate winner determination algorithm for combinatorial auctions. The main idea is a combination of approximated positive linear program algorithms for determining initial allocation and stepwise updates of allocations.

In the approximated linear program phase, a variant of a primal-dual approximation algorithm is used. Unlike ordinary methods, a scaled price p_i for each item i is maintained. Finally, a fractional allocation $0 \leq A_i \leq 1$ for every bid and an item price p_i for every item are assigned. And then, the bids are sorted by the descending order of the value $v_j / \sum_{i \in S_j} P_i$ for bid price v_j of the bundle S_j, and then greedy allocation of bids is made for attaining initial allocation.

After initial allocation is attained, a kind of stepwise update is made. Here, a non-winner bid is selected and put at the top of the sorted bid list attained in the linear program phase. And then new greedy allocation is calculated. When the total revenue of new allocation is larger than before, the new sorted bid list is used instead of the last one. This process is repeated until no more improvements are possible. The details of the algorithm are shown in [43].

2.5. Casanova

Hoos [18] proposed that a generic random walk SAT solver may perform well for approximation of combinatorial auctions. In [18], the Casanova algorithm is proposed for the purpose. It is based on scoring each search state using the revenue-per-item of the corresponding allocation. Casanova starts with an empty allocation. At each step, the highest bid is selected in probability $1 - wp$ or a bid is randomly selected in probability wp. Note that the *age* of the bid is considered in probability np to avoid doing a short loop. Also a soft restart strategy is used. The algorithm stops when the search process exceeds *maxSteps* or the soft restart occurs and the restart count exceeds *maxTrials*.

3. Enhanced Approximation Algorithms

3.1. Hill-climbing Search

In [14], we have shown that the hill-climbing approach performs well when an auction has a massively large number of bids. In this section, we summarize our proposed algorithms.

Lehmann's greedy winner determination could succeed in specifying the lower bound of the optimality in its allocation [26]. The straightforward extension of the greedy algorithm is to construct a local search algorithm that continuously updates the allocation so that the optimality is increased. Intuitively, one allocation corresponds to one state of a local search.

The inputs are $Alloc$ and L. L is the bid list of an auction. $Alloc$ is the initial greedy allocation of items for the bid list.

```
1: function LocalSearch(Alloc, L)
2:   RemainBids:= L - Alloc;
3:   for each b ∈ RemainBids as sorted order
4:     if b conflicts Alloc then
5:     Conflicted:=Alloc - consistentBids({b}, Alloc);
6:       NewAlloc:= Alloc - Conflicted + {b};
7:       ConsBids:=
8:         consistentBids(NewAlloc, RemainBids);
9:       NewAlloc:=NewAlloc+ConsBids;
```

10: **if** $price(Alloc) < price(NewAlloc)$ **then**
11: **return** LocalSearch($NewAlloc$,L);
12: **end for each**
13: **return** $Alloc$

The function $consistentBids$ finds consistent bids for the set $NewAllocation$ by walking down the list $RemainBids$. Here, a new inserted bid will wipe out some bids that conflict with the inserted bid. So there will be free items to allocate after the insertion. The function $consistentBids$ tries to insert the other bids for selling as many of the higher value items as possible.

Example: Assume there are five items a, b, c, d, and e, and there are six bids, $< \{a,b,c\}, 30 >$ $, < \{a\}, 15 >, < \{c\}, 13 >, < \{d,e\}, 15 >, < \{a,c\}, 14 >, < \{b\}, 8 >$. We can calculate the values of Lehmann's criterion $a/\sqrt{|s|}$ as 17.6, 15, 13, 10.7, 10, and 8, respectively. In this case, the initial allocation is Lehmann's greedy allocation $< \{a,b,c\}, 30 >$ $, < \{d,e\}, 15 >$ and the total revenue is 45. Here, the remaining list is $< \{a\}, 15 >, < \{c\}, 13 >, < \{a,c\}, 14 >, < \{b\}, 8 >$. In this algorithm, we pick $< \{a\}, 15 >$ since it is the top of the remaining list. Then we insert $< \{a\}, 15 >$ into the allocation and remove $< \{a,b,c\}, 30 >$. The allocation is now $< \{a\}, 15 >, < \{d,e\}, 15 >$. We then try to insert the other bids that do not conflict with the allocation. Then, the allocation becomes $< \{a\}, 15 >$ $, < \{b\}, 8 >, < \{c\}, 13 >, < \{d,e\}, 15 >$. The total revenue is 51, and is increased. Thus, the allocation is updated to it. Our local algorithm continues to update the allocation until there is no allocation that has greater revenue. This could improve the revenue that Lehmann's greedy allocation can achieve.

3.2. Local Search for Multiple Values of the Sorting Criterion c

The optimality of allocations got by Lehmann's algorithm (and the following hill-climbing) deeply depends on which value was set to the bid sorting criterion c. Again, in [26], Lehmann et al. argued that $c = 1/2$ is the best parameter for approximation when the norm of the worst case performance is considered. However, the optimal values for each auction are varied from 0 to 1 even if the number of items is constant. Here, we use an enhancement for our local search algorithm with parallel search for the sorting criterion c. In the algorithm, the value of c for Lehmann's algorithm is selected from a pre-defined list. It is reasonable to select c from neighbors of $1/2$, namely, $C = \{0.0, 0.1, \ldots, 1.0\}$. The results are aggregated and the best one (that has the highest revenue) is selected as the final result.

3.3. Simulated Annealing Search

We also prepared a small extension of the proposed algorithm to the simulated annealing local search. The algorithm is a combination of the presented hill-climbing approach and a random search based on the standard simulated annealing algorithm. We use a parameter that represents the temperature. The temperature is set at a high value at the beginning and continuously decreased until it reaches 0. For each cycle, a neighbor is randomly selected and its value may be less than the current value in some cases. Even in such a case, if a probability value based on the temperature is larger than 0, the state is moved to the new allocation that has less value. This could make us get off the local minimum. Also, the algorithm automatically restarts when it reaches the local minimum. It repeats until the highest result is not updated in the last k restarts. Here, we use $k = 5$ for our experiments. We prepared this algorithm only for investigating how random search capability will improve the performance. Note that the proposed SA search may not satisfy our proposed features discussed later.

4. Evaluation

4.1. Experiment Settings

We implemented our algorithms in a C program for the following experiments. We also implemented the Casanova algorithm in a C program. However, for the following experiments, we used Zurel's C++ based implementation that is shown in [43].

The experiments were done with the above implementations to examine the performance differences among algorithms. The programs were employed on a Mac with Mac OS X 10.4, CoreDuo 2.0GHz CPU, and 2GBytes of memory. Thus, actual computation time will be much smaller when we employ parallel processor systems in a distributed execution environment. We leave this for future work.

We conducted several experiments. In each experiment, we compared the following search algorithms. greedy(C=0.5) uses Lehmann's greedy allocation algorithm with parameter ($c = 0.5$). greedy-all uses

the best results of Lehmann's greedy allocation algorithm with parameter ($0 \leq c \leq 1$ in 0.1 steps). This is a simple algorithm but Lehmann et al. did not mention it. HC(c=0.5) uses a local search in which the initial allocation is Lehmann's allocation with $c = 0.5$ and conducts the hill-climbing search shown in the previous section. HC-all uses the best results of the hill-climbing search with parameter ($0 \leq c \leq 1$ in 0.1 steps). SA uses the simulated annealing algorithm. Also, we denote the Casanova algorithm as Casanova and Zurel's algorithm as Zurel.

In the following experiments, we used 0.2 for the epsilon value of Zurel's algorithm in our experiments. This value appears in [43]. Also, we used 0.5 for np and 0.15 for wp on Casanova, which appear in [18]. Note that we set $maxTrial$ to 1 but $maxSteps$ to ten times the number of bids in the auction.

4.2. Evaluation on Basic Auction Dataset

In [43], the researchers evaluated the performance of their presented algorithm with the data set presented in [10], compared with CPLEX and other existing implementations. Figure 1 shows the comparison of our algorithms, Casanova, and Zurel's algorithm with the dataset provided in [10]. This dataset contains 2240 auctions with optimal values, ranging from 25 to 40 items and from 50 to 2000 bids. Though the dataset has more than 100 different auction settings, we made a plot of average optimality on all those settings. The average optimality of each algorithm is also shown.

Since problems in the dataset have relatively small size of bids and items, we omitted the execution time since all algorithms run in very short time. This comparison shows that the performance of Casanova is better than Zurel, our SA, and our HC-all follows them. This is because we can spend enough time for each auction problem so that generic random search algorithm performs well. The performance of our algorithms is competitive but there is little advantage on this dataset.

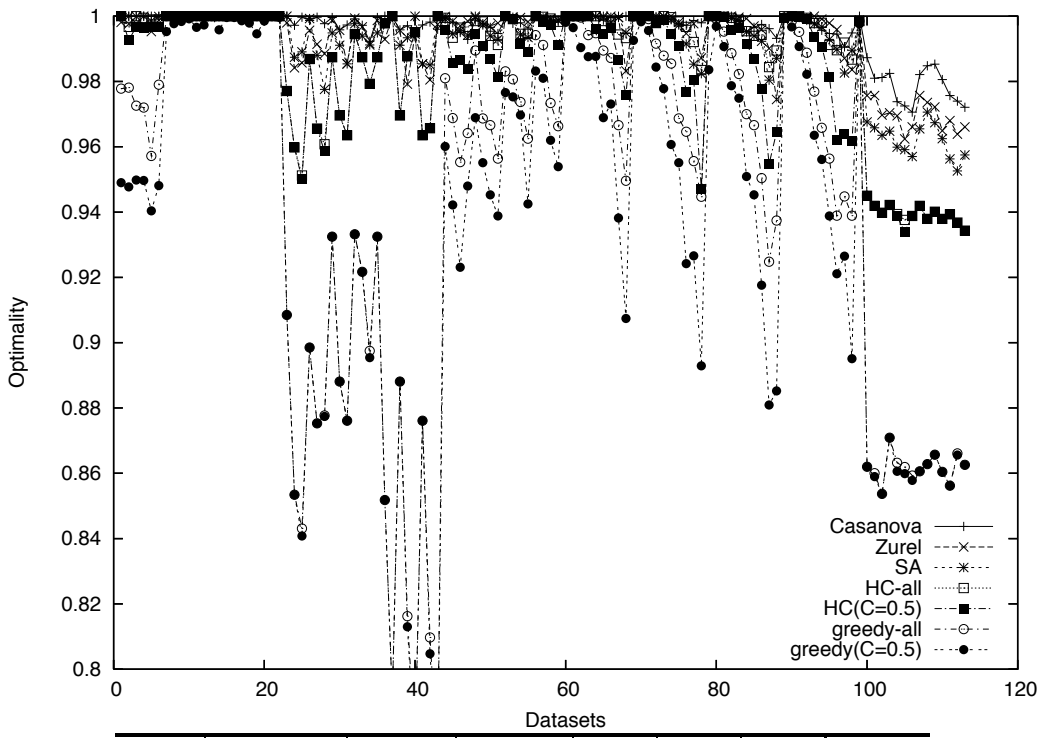
We conducted detailed comparisons with common datasets from CATS benchmark[27]. Compared to deVries' dataset shown in [10], the CATS benchmark is very common and it contains more complex and larger datasets.

Table 1 shows the comparison of our algorithms, Casanova, and Zurel's algorithm with a dataset provided in the CATS benchmark [27]. The dataset has numerous auctions with optimal values in several distributions. Here we used 'varsize' which contains a total of 7452 auctions with reliable optimal values in 9 different distributions.[2] Numbers of items range from 40 to 400 and numbers of bids range from 50 to 2000. The name of each distribution is referred from [27].

Following is a summary of bid distributions used in this paper. For legacy distributions (i.e., L2, L3, L4, L6, and L7), the bid generation algorithm is defined by three parts: number of items in a bid, which items to be chosen, and price offer for the bundle. For 'number of items,' one of the following approaches is used: Uniform (Uniformly distributed on $[1, num\_items]$), Normal (Normally distributed with $\mu = \mu\_items$ and $\sigma = \sigma\_items$), Constant (Fixed at $constant\_items$), Decay (Starting with 1, repeatedly increment the size of the bundle until rand (0,1) exceeds α), Binomial (Request n items with probability $p^n(1-p)^{num\_items-n}$), and Exponential (Request n items with probability $C \cdot e^{-n/q}$). For 'which items', only Random (choose n items randomly) is used. For 'price offer', one of the following approaches is used: Fixed Random (Uniform on $[low\_fixed$,$hi\_fixed]$), and Linear Random (Uniform on $[low\_linearly \cdot n$,$hi\_linearly \cdot n]$ for number of items n). For details about datasets, see [27].

- **L2** Number of items in a bid: **uniform**, price offer: **linearly random** with $low\_linearly = 0$, $hi\_linearly = 1$ (when bid price is defined as a float value), or $low\_linearly = 500$, $hi\_linearly = 1500$ (when bid price is defined as an integer value). This is also called a 'Sandholm uniform linearly random' or an 'Andersson uniform linearly random' dataset.
- **L3** Number of items in a bid: **constant** with $constant\_items = 3$, price offer: **fixed random** with $low\_fixed = 0$, $hi\_fixed = 1$. This is also called a 'Sandholm constant fixed random' dataset.
- **L4** Number of items in a bid: **decay** with $\alpha = 0.55$, price offer: **linearly random** with $low\_linearly = 0$, $hi\_linearly = 1$ (when bid price is defined as a float value), or $low\_linearly = 1$,$hi\_linearly = 1000$ (when bid price is defined as an integer value). This is also called a 'Sand-

[2] Since some of the original data seems corrupted or failed to obtain optimal values, we excluded such auction problems from our dataset. Also, we excluded a whole dataset of a specific bid distribution when the number of valid optimal values is smaller than the other half of the data. The original dataset provides optimal values of auction problems by two independent methods, CASS and CPLEX. Therefore, it is easy to find out such corrupted data from the dataset.



	greedy(C=0.5)	greedy-all	HC(C=0.5)	HC-all	SA	Zurel	Casanova
average	0.9351	0.9450	0.9832	0.9868	0.9921	0.9934	0.9960

Fig. 1. Optimality on deVries' dataset

Table 1
Optimality on CATS-VARSIZE dataset

	arbitrary	L2	L3	L4	L6	L7	matching	regions	scheduling	average
greedy(C=0.5)	0.8641	0.9968	0.8037	0.9076	0.9403	0.8652	0.9721	0.8734	0.9143	0.9042
greedy-all	0.8973	1.0000	0.8359	0.9355	0.9683	0.9010	0.9749	0.9028	0.9664	0.9314
HC(C=0.5)	0.9485	1.0000	0.9433	0.9611	0.9902	0.9822	0.9957	0.9703	0.9826	0.9749
HC-all	0.9750	1.0000	0.9663	0.9807	0.9957	0.9920	0.9967	0.9857	0.9975	0.9877
SA	0.9768	1.0000	0.9756	0.9813	0.9950	0.9921	0.9975	0.9872	0.9969	0.9892
Zurel	0.9671	0.9998	0.9571	0.9811	0.9977	0.9838	0.9994	0.9836	0.9909	0.9845
Casanova	0.9567	1.0000	0.9741	0.96457	0.9753	0.9905	0.9946	0.9711	0.9979	0.9805

holm decay linearly random', or an 'Andersson decay linearly random' dataset.

L6 Number of items in a bid: **exponential** with $q = 5$, price offer: **linearly random** with $low\_linearly = 0.5$, $hi\_linearly = 1.5$ (when bid price is defined as a float value), or $low\_linearly = 500$, $hi\_linearly = 1500$ (when bid price is defined as an integer value).

L7 Number of items in a bid: **binomial** with $p = 0.2$, price offer: **linearly random** with $low\_linearly = 0.5$, $hi\_linearly = 1.5$ (when bid price is defined as a float value), or $low\_linearly = 500$, $hi\_linearly = 1500$ (when bid price is defined as an integer value).

arbitrary This bid distribution considers complementary (arbitrary) relationships between items. The default parameters are as follows: $max\_item\_value = 100$, $additional\_item = 0.9$, $max\_substitutable\_bids = 5$, $additivity = 0.2$, $deviation = 0.5$, $budget\_factor = 1.5$, $resale\_factor = 0.5$, and $S(n) = n^{1+additivity}$. For details, see [27].

regions This bid distribution considers the adjacency problem. In this distribution, a real estate graph

in 2 dimensions is generated, and then bids are generated by combinations of regions considering proximity. The default parameters are as follows: $three\_prog = 1.0, additional\_neighbor = 0.2$, $max\_item\_value = 100$, $max\_substitutable\_bids = 5$,$additional\_location = 0.9$, $jump\_prob = 0.05$, $additivity = 0.2$, $deviation = 0.5$, $budget\_factor = 1.5$, $resale\_factor = 0.5$, and $S(n) = n^{1+additivity}$. For details, see [27].

scheduling This bid distribution is based on a distributed job-shop scheduling with one resource. The default parameters are as follows: $deviation = 0.5, prob\_additional\_deadline = 0.9, additivity = 0.2$, and $max\_length = 10$. For details, see [27].

Here, we can see that the performances of HC-all and SA are better than Zurel on arbitrary, L2, L3, L7, regions, and scheduling. Others are nearly equal to Zurel's. The performance of Casanova is nearly equal to or less than HC (C=0.5) excluding L3 and scheduling.

Note that those differences come from the differences of the termination condition on each algorithm. In particular, Casanova spent much more time compared with the other two algorithms. However, we do not show the time performance since the total execution time is relatively too small to be compared.

4.3. Evaluation on Large Auction Dataset

The CATS common datasets we used in Section 4.2 have a relatively smaller number of bids than we expected. We conducted additional experiments with much greater numbers of bids. We prepared additional datasets having 20,000 non-dominated bids in an auction. The datasets were produced by CATS [27] with default parameters in 5 different distributions. In the datasets, we prepared 100 trials for each distribution. Each trial is an auction problem with 256 items and 20,000 bids.[3]

[3]Due to the difficulty of preparing the dataset, we only prepared 5 distributions. Producing a dataset with other distributions is difficult in a feasible timeframe when we need non-dominated bids in each auction problem. Also, we omit L8 since this distribution needs a set of parameters for preparing the valuation of bids, but there is no common method to set such parameters. The distributions do not always reflect a certain situation for a specific resource allocation application scenario. For more details about the bid generation problem, see [27]. A preliminary result of this experiment was shown in [13].

Table 2 shows the experimental result on the datasets with 20,000 bids in an auction focused on execution time of approximation. Due to the difficulty of attaining optimal values, we normalized all values as Zurel's results equaling 1 as follows.

Let A be a set of algorithms, $z \in A$ be the Zurel's approximation algorithm, L be a dataset generated for this experiment, and $revenue_a(p)$ such that $a \in A$ be the revenue obtained by algorithm a for a problem p such that $p \in L$, the average revenue ratio $ratioA_a(L)$ for algorithm $a \in A$ for dataset L is defined as follows:

$$ratioA_a(L) = \frac{\sum_{p \in L} revenue_a(p)}{\sum_{p \in L} revenue_z(p)}$$

Here, we use $ratioA_a(L)$ for our comparison of algorithms.

We prepared cut-off results for Casanova and HC. For example, casanova-10ms denotes the result of Casanova within 10 milliseconds. Here, for faster approximation, we used greedy-3 and HC-3 instead of greedy-all and HC-all. greedy-3 uses the best results of Lehmann's greedy allocation algorithm with parameter ($0 \leq c \leq 1$ in 0.5 steps). HC-3 uses the best results of the hill-climbing search with parameter ($0 \leq c \leq 1$ in 0.5 steps). Also, we prepared a variant of our algorithm that has a suffix of -seq or -para. The suffix -seq denotes the algorithm is completely executed in a sequence that is equal to one that can be executed on a single CPU computer. For example, greedy-3-seq denotes that the execution time is just the sum of execution times of three threads. The suffix -para denotes the algorithm is completely executed in a parallel manner, and the three independent threads are completely executed in parallel. Here, we used the ideal value for -para since our computer has only two cores in the CPU. The actual execution performance will be between -seq and -para. Also, we denote the initial performance of Zurel's algorithm as Zurel-1st. Here, Zurel-1st is the result at the end of its first phase and no winners will be approximately assigned before it. Note that we did not include results of SA, since it did not produce comparable results because of its highly random behavior on start-up.

On most distributions in Table 2, Zurel-1st takes more than 1 second but the obtained $ratioA$ is lower than greedy-3-seq. Furthermore, the average $ratioA$ of HC-3-para-1000ms is higher than Zurel while its computation time is less than both Zurel and Zurel-1st.

Table 2
Time Performance on 20,000 bids-256 items

	L2		L3		L4		L6		L7		average	
greedy(c=0.5)	1.0002	(23.0)	0.9639	(19.0)	0.9417	(23.0)	0.9389	(23.4)	0.7403	(22.1)	0.9170	(22.1)
greedy-3-seq	1.0003	(69.1)	0.9639	(59.2)	0.9999	(72.9)	0.9965	(67.8)	0.7541	(66.8)	0.9429	(67.2)
greedy-3-para	1.0003	(26.4)	0.9639	(20.9)	0.9999	(28.4)	0.9965	(26.0)	0.7541	(25.5)	0.9429	(25.4)
HC(c=0.5)-100ms	1.0004	(100)	0.9741	(100)	0.9576	(100)	0.9533	(100)	0.8260	(100)	0.9423	(100)
HC-3-seq-100ms	1.0004	(100)	0.9692	(100)	1.0000	(100)	0.9966	(100)	0.8287	(100)	0.9590	(100)
HC-3-para-100ms	1.0004	(100)	0.9743	(100)	1.0001	(100)	0.9969	(100)	0.9423	(100)	0.9828	(100)
HC(c=0.5)-1000ms	1.0004	(1000)	0.9856	(1000)	0.9771	(1000)	0.9646	(1000)	1.0157	(1000)	0.9887	(1000)
HC-3-seq-1000ms	1.0004	(1000)	0.9804	(1000)	1.0003	(1000)	0.9976	(1000)	1.0086	(1000)	0.9975	(1000)
HC-3-para-1000ms	1.0004	(1000)	0.9856	(1000)	1.0006	(1000)	0.9987	(1000)	1.0240	(1000)	1.0019	(1000)
Zurel-1st	0.5710	(11040)	0.9690	(537)	0.9983	(2075)	0.9928	(1715)	0.6015	(1796)	0.8265	(3433)
Zurel	1.0000	(13837)	1.0000	(890)	1.0000	(4581)	1.0000	(4324)	1.0000	(3720)	1.0000	(5470)
casanova-10ms	0.2583	(10)	0.0069	(10)	0.0105	(10)	0.0202	(10)	0.2577	(10)	0.0632	(10)
casanova-100ms	0.2583	(100)	0.0069	(100)	0.0105	(100)	0.0202	(100)	0.2577	(100)	0.1107	(100)
casanova-1000ms	0.5357	(1000)	0.1208	(1000)	0.0861	(1000)	0.1486	(1000)	0.7614	(1000)	0.3305	(1000)

(each value in () is time in milliseconds)

Table 3
Time Performance on 100,000 bids-256 items

	L2		L3		L4		L6		L7		average	
greedy-3	1.1098	(129.9)	0.9836	(132.4)	1.0003	(131.9)	1.0009	(130.7)	0.8688	(130.9)	0.9927	(131.2)
HC-3-para-333ms	1.1098	(333)	0.9859	(333)	1.0003	(333)	1.0009	(333)	0.9395	(333)	1.0073	(333)
HC-3-para-1000ms	1.1098	(1000)	0.9880	(1000)	1.0003	(1000)	1.0010	(1000)	0.9814	(1000)	1.0161	(1000)
zurel-1st	0.8971	(74943)	0.9827	(2257)	0.9998	(5345)	0.9987	(4707)	0.7086	(8688)	0.9174	(19188)
Zurel	1.0000	(91100)	1.0000	(6036)	1.0000	(30568)	1.0000	(44255)	1.0000	(17691)	1.0000	(37930)
casanova-130ms	0.3031	(130)	0.0061	(130)	0.0117	(130)	0.0182	(130)	0.2246	(130)	0.1127	(130)
casanova-333ms	0.3506	(333)	0.0379	(333)	0.0328	(333)	0.0673	(333)	0.7536	(333)	0.2484	(333)
casanova-1000ms	0.4954	(1000)	0.1176	(1000)	0.0946	(1000)	0.1605	(1000)	0.7832	(1000)	0.3303	(1000)

(each value in () is time in milliseconds)

Table 3 shows the experimental result on the dataset with 100,000 bids in an auction focused on the early anytime performance. While HC-3 and Zurel's algorithm are competitive in Table 2, it is clear that our proposed HC-3 outperforms Zurel's algorithm in any time performance. Note that the time needed to attain initial allocations increased dramatically (approx. 2 times in L3 to over 7 times in L7) when the number of bids becomes five times larger than that of Table 2. However, our HC-3-para-1000ms only takes the same execution time (i.e., 1000 msec) but its average $ratioA$ is higher than Zurel. Note that the HC-3-para-333ms has still higher $ratioA$ value than Zurel while its average computation time is 100 times less. We argue that our algorithm has an advantage when the number of bids increases.

The total computation time of Zurel's algorithm is shorter in a typical case when the numbers of bids are ranging within the size of data we used in this paper. This is an excellent advantage of Zurel's algorithm and we do not argue our algorithm outperforms Zurel's when the finally attained approximated allocations are important. However, our algorithms have important characteristics that Zurel's algorithm does not have. We will discuss the importance of these characteristics in section 5.

4.4. Detailed Analysis on 20k-256 dataset

In this section, we provide detailed analysis for the results in each bid distribution on the dataset with 20,000 bids and 256 items.

Figure 2 shows the average approximation curve for the HC-3-para algorithm. Values of the vertical axis denote $ratioA$ we defined in section 4.3. We plotted average anytime approximation performance in each distribution from the start to 1000 msec. Here, we can see that the approximation for the dataset based on L7 distribution is very hard to solve in a short time. Actually, L7 is dominating other distributions to obtain higher average revenues for all distributions. The dataset on L3 distribution is a bit hard for the HC-3-

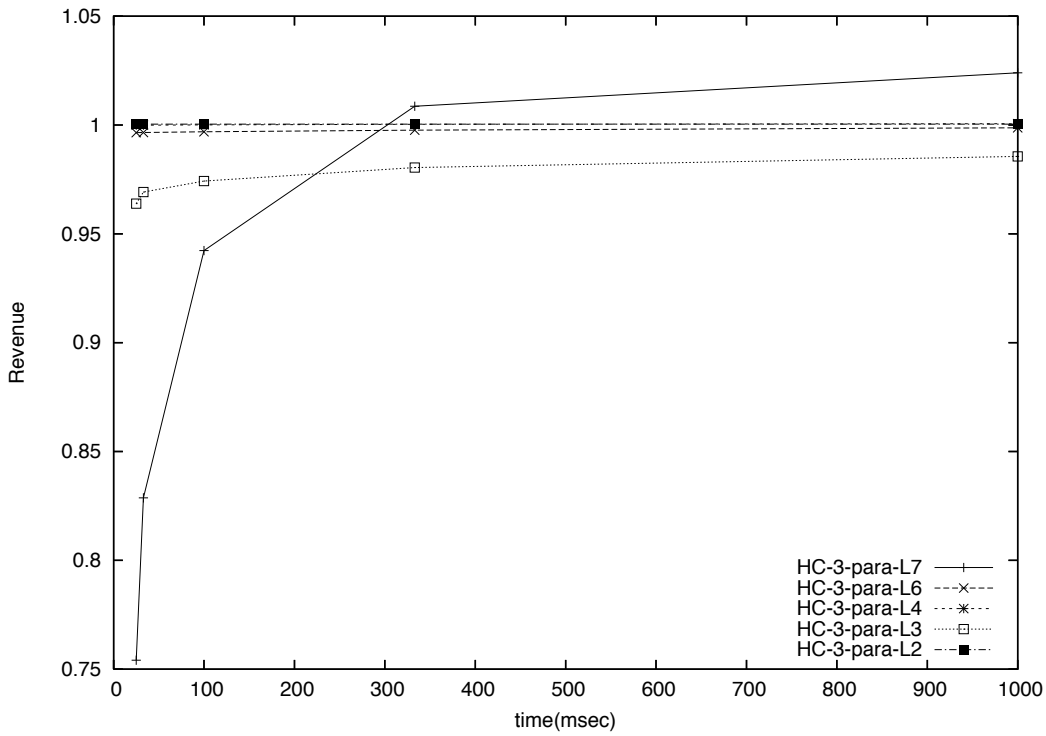


Fig. 2. Average Approximation Curve of HC-3-para on 20,000 bids-256 items

para algorithm since it increases slowly but does not reach 1 at 1000 msec. The other three distributions are rather easy for simple greedy approximation.

Here, we provide more detailed analysis for each distribution in the following.

4.4.1. Detailed Analysis on L2

Auction problems based on this distribution are relatively easier to approximate than others. Typically, one to a few winners will be found in a problem.

Figure 3 shows a plot of approximation results for all auction problems in the dataset based on L2 distribution with 20,000 bids and 256 items. In Figure 3, we plotted the total revenue for 5 algorithm settings: greedy-3, HC(C=0.5)-1000ms, HC-3-para-1000ms, Zurel-1st, and Zurel.

In most problems, approximated total revenues are almost 256,000, and the values are stable and not distributed very much among problems. Here, only Zurel-1st obtained lower revenues and Zurel obtained slightly lower revenues than others. This distribution is hard for Zurel's algorithm, since the generated auction problems may contain many *errors* that cause worse performance and slower convergence speed in its approximated linear programming phase.

4.4.2. Detailed Analysis on L3

Auction problems based on this distribution are a bit tricky since a bid only has exactly three items in the bundle. Typically, about 85 final winners will be found in a problem and the optimal revenue could be varied.

Figure 4 shows a plot of approximation results for all auction problems in the dataset based on L3 distribution with 20,000 bids and 256 items. In Figure 4, we plotted the total revenue for 5 algorithm settings that are the same as Figure 3.

Approximated total revenues are distributed from 79,000 to 82,000. Here, Zurel obtains the best results in many cases and HC-3-para-1000ms obtains slightly lower ones than Zurel. In this distribution, there is no advantage to searching for different bid weightings since there are only bids that have exactly 3 items. Therefore, in all cases, revenues obtained by HC-3-para-1000ms and HC(C=0.5)-1000ms are exactly the same. However, the obtained results of HC-3-para-1000ms are still competitive with Zurel. For Zurel's algorithm, convergence speed is typically faster than other distributions on its approximated linear programming phase.

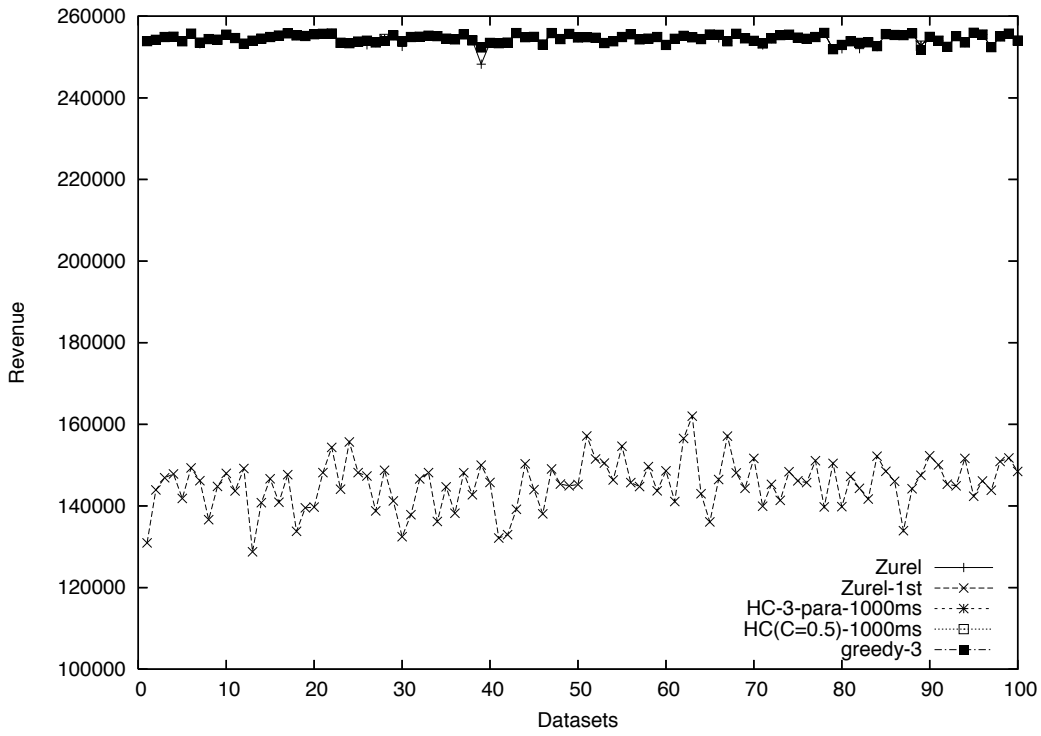


Fig. 3. All Revenue Plot on 20,000 bids-256 items (L2)

4.4.3. Detailed Analysis on L4

Auction problems based on this distribution are used so often since the bid generation algorithm is simple but the generated problems look very realistic. Typically, a few dozen winners will be found in each problem.

Figure 5 shows a plot of approximation results for all auction problems in the dataset based on L4 distribution with 20,000 bids and 256 items. In Figure 5, we plotted the total revenue for 5 algorithm settings that are the same as Figure 3.

Approximated total revenues are stable and number almost 255,000. Here, greedy-3 obtains good revenues, and the results of Zurel and HC-3-para-1000ms are slightly higher than greedy-3. However, HC(C=0.5)-1000ms obtains much worse results in all cases. This shows our *parallel search for different bid weighting* strategy has a certain advantage. Furthermore, we think this is also a big reason that some researchers have underestimated the performance of this simple greedy-with-hill-climbing approach in past research.

4.4.4. Detailed Analysis on L6

Auction problems based on this distribution are similar to L4 but a little difficult to approximate. Typically, several tens of winners will be found at once in a problem.

Figure 6 shows a plot of approximation results for all auction problems in the dataset based on L6 distribution with 20,000 bids and 256 items. In Figure 6, we plotted the total revenue for 5 algorithm settings that are the same as in Figure 3.

Approximated total revenues are around 252,000. Here, greedy-3 obtains good revenues, and the results of HC-3-para-1000ms are slightly higher than greedy-3. The results of Zurel are sometimes slightly better than HC-3-para-1000ms, but they are not largely different. Similar to L4, HC(C=0.5)-1000ms obtains much worse results in all cases. This shows our *parallel search for different bid weighting* strategy has a certain advantage.

Also notice that, although the average approximation result is slightly below 1.0 for HC-3 at 1000msec, it took less computation time than Zurel-1st. Therefore the approximation speed is still faster than Zurel-

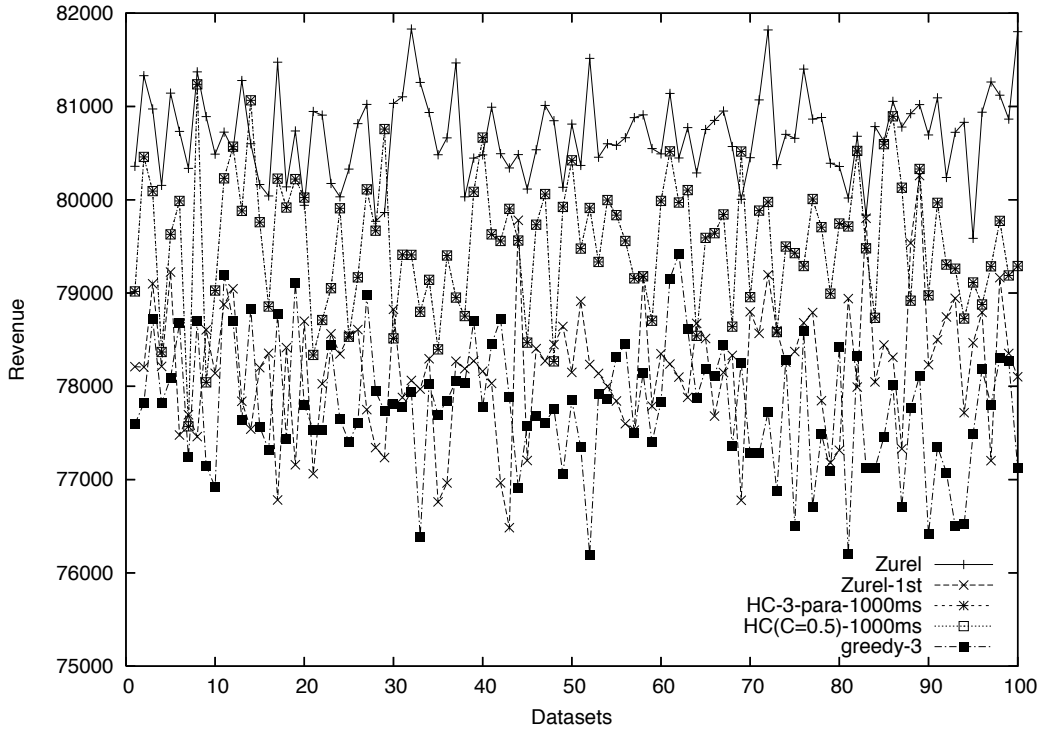


Fig. 4. All Revenue Plot on 20,000 bids-256 items (L3)

1st when we consider a condition with a hard time constraint.

4.4.5. Detailed Analysis on L7

Auction problems based on this distribution are very hard to approximate in what we used here. Typically, a few winners will be found in a problem but none of them can be found in a quick greedy approximation. Rather, it is very important to find out *good combinations* of bids to get higher revenues.

Figure 7 shows a plot of approximation results for all auction problems in the dataset based on L7 distribution with 20,000 bids and 256 items. In Figure 7, we plotted the total revenue for 5 algorithm settings that are the same as in Figure 3.

Approximated total revenues are around 100,000, but they are varied. Here, neither greedy-3 nor Zurel-1st could obtain good revenues. The results of HC-3-para-1000ms are constantly higher than greedy-3. Those problems are hard for simple greedy approximation, and therefore we employed our HC-3 to overcome this issue. The results of Zurel are sometimes worse than HC-3-para-1000ms. Although Zurel and HC-3-para-1000ms are not largely different in many cases, in some cases the results of Zurel are about 10 percent worse in total revenue, and this makes a clear difference from HC-3-para-1000ms on averaged performance.

5. Discussion

5.1. Winner Price Monotonicity

In real world auctions, often we open the winners and their bidding prices after the auction is finished. When we employ an approximated algorithm for winner determination, a loser who might be a winner in the optimal allocation could know the winner's bidding price in an approximate allocation after the auction finishes. In some cases, this loser had placed a higher price than the winner's for the same or a subset of the bundle. This would result in *unacceptable* allocations for bidders.

We believe that the above issue should be considered to make our mechanism acceptable by participants in the real world. Therefore, we propose two desirable properties, Winner-Price-Monotonicity and Weak-Price-Monotonicity to avoid *unacceptable* allocations.

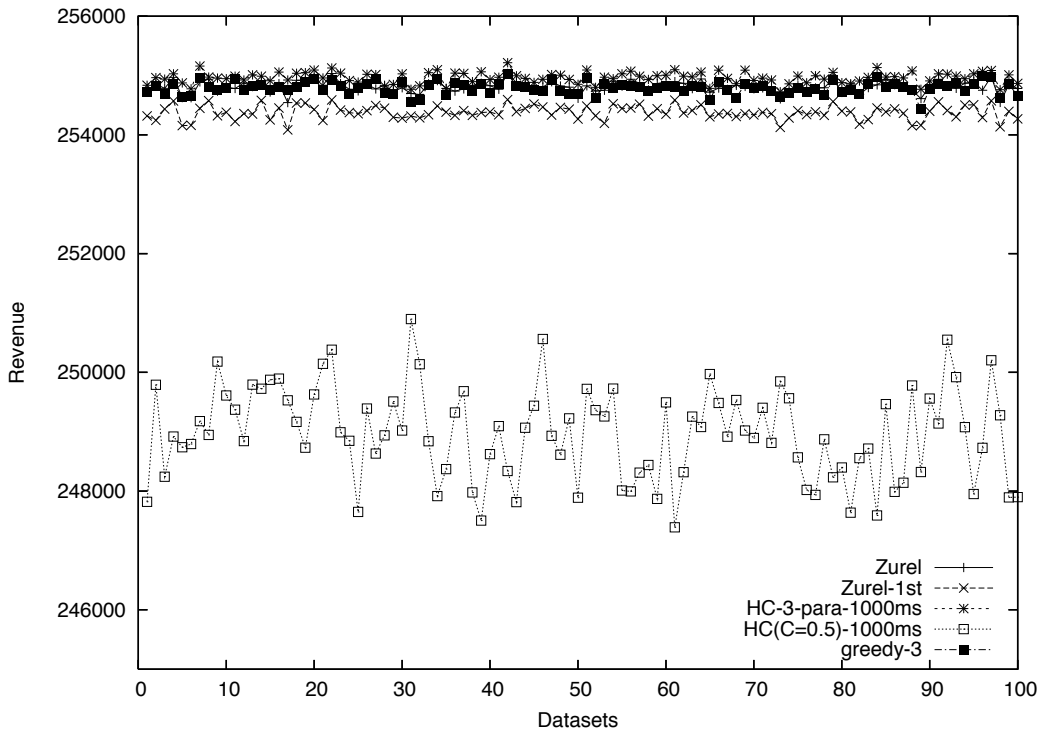


Fig. 5. All Revenue Plot on 20,000 bids-256 items (L4)

Definition 1 (Winner-Price-Monotonicity: WPM) *For two non-empty bundles B and B', if $B \subseteq B'$and $v_i(B) > v_j(B')$, then j must not win bundle B'.*

Definition 2 (Weak-Winner-Price-Monotonicity: Weak-WPM) *For non-empty bundle B, if $v_i(B) > v_j(B)$, then j must not win bundle B.*

We have the following propositions.

Proposition 1 *Our proposed winner determination algorithms, except for the simulated annealing-based algorithm, produce allocation W_{fin} that satisfies WPM when the algorithm reaches an end.*

Proposition 2 *In terms of any allocations that are achieved during computation (as an anytime algorithm), our proposed winner determination algorithms, except for the simulated annealing-based algorithm, satisfy Weak-WPM.*

It is a big merit to guarantee WPM and/or Weak-WPM at the algorithm level when we use it where slightly different combinatorial auctions are conducted iteratively. It seems easy to satisfy WPM and/or Weak-WPM by using any approximated winner determination algorithms by adding a pre-processing that removes all dominated bids from the bidset before starting the approximation. However, we should consider its computational overhead. For simplicity, consider a case $B = B'$ instead of $B \subseteq B'$. Let n be the number of items and m be the number of items in an auction. When m is very small, it is easy to look up the highest bids of each bundle by using a hash algorithm. In this case, the computational order is $O(n)$. However, it consumes a great deal of memory (of course it can be smaller than 2^m but at least additional $O(n)$ of working space), and it is actually very difficult to determine good hash functions for a smaller hash table size without loss of computational speed. It is a serious problem when the memory is almost completely used up for storing the data of a large number of bids. Sometimes its computational order might reach $O(n^2)$, which is greater than that of typical good approximation algorithms. For example, the computational order of Lehmann's greedy algorithm is $O(n\ log\ n)$ when we use one of the $O(n\ log\ n)$ sorting algorithms on it. Furthermore, when we consider the deletion of a bid, we have to determine the highest price bid that has been made obsolete by the deleted bid, or recalculate such

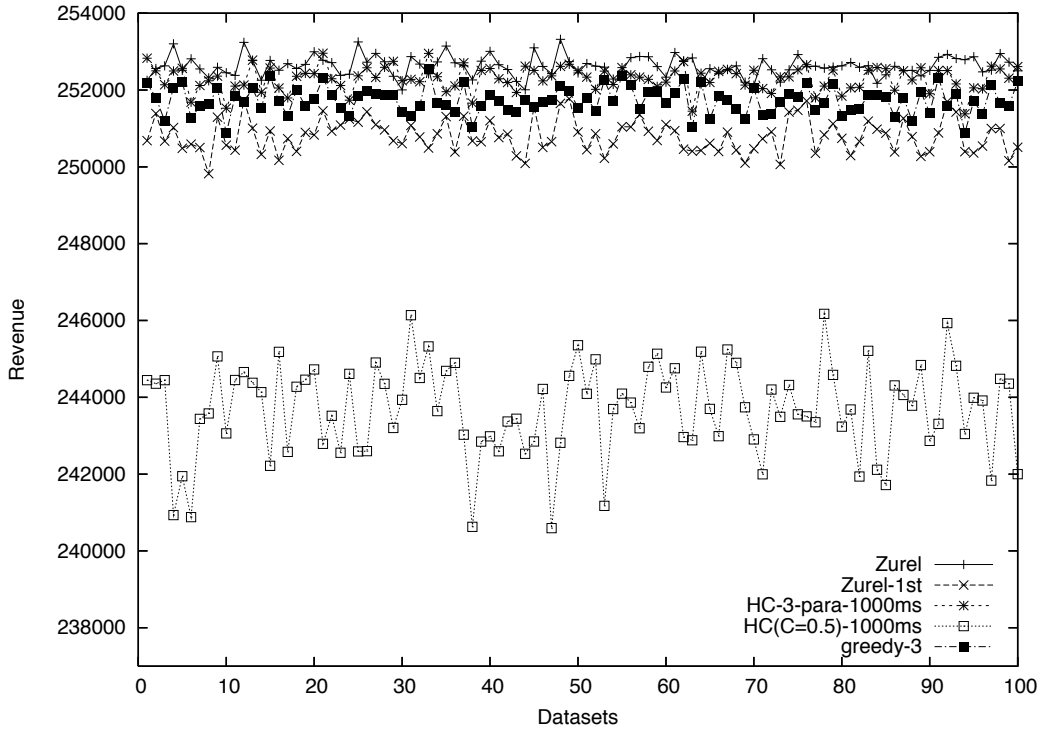


Fig. 6. All Revenue Plot on 20,000 bids-256 items (L6)

pre-processing for all bids again. Considering a case $B \subseteq B'$ will make the problem more difficult.

5.2. *Requirements for Acceptable Approximation*

Concerning the actual application system implementation using approximated winner determination, important desirable properties should be retained but are not pointed out by other papers, including the time required to attain initial approximated allocations. Table 4 summarizes characteristics of five algorithms: our HC-3, our SA extension, Zurel, Casanova, and Lehmann's greedy algorithm.

Peak optimality means how the optimality grows when we have enough time for computation. Typically, this property has much weight for ordinary evaluation. Here, Zurel, Casanova, and our SA give excellent optimality compared with the others. Of course, using other algorithms that can obtain optimal solutions is best for this purpose. However, the gaps among them are very small (less than 5 percent), excluding Lehmann's approach. Also, the result is varied for distributions of bids. There is no one perfect approximation algorithm for all distributions of bids. In this paper, we did not focus much on this point. Rather, we focused on providing acceptable solutions within a realistic time period.

Response time for initial allocation is an important issue, especially for real-time systems. A case exists where we spent an unexpectedly long time attaining an initial allocation even if the approximation algorithm was an anytime algorithm. In particular, Zurel's algorithm took much more time to attain the initial allocation when the number of bids got larger for certain bid distributions. Thus we demonstrated that our algorithm outperforms the other approaches in attaining an initial allocation for these bid distributions.

Optimality in short-time approximation is also an essential issue when we handle a large number of bids in an auction. On the aspect of initial response time, Casanova is also sufficient. However, the quality of the initial approximation is very low compared with HC-3 and Lehmann's greedy approach. We demonstrated that our algorithm performs very well in short-time (e,g. less than 1000 msec) approximation, within the time that is shorter than attaining the initial allocation on Zurel's algorithm. Possibility of extremely short time winner approximation is useful since it en-

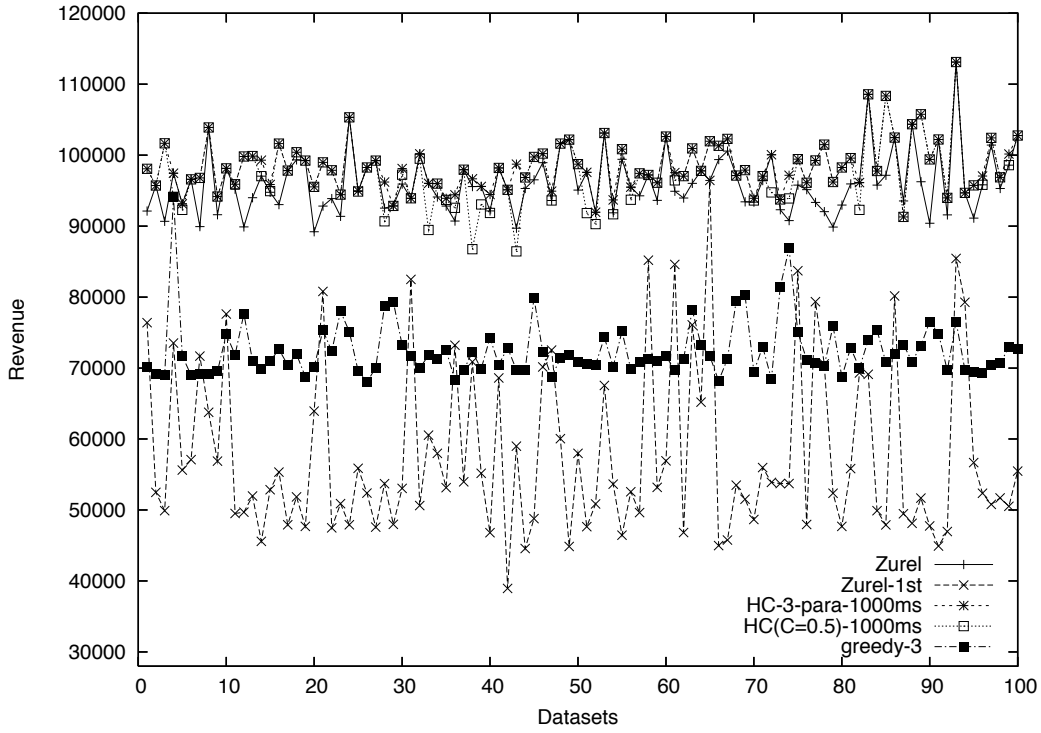


Fig. 7. All Revenue Plot on 20,000 bids-256 items (L7)

Table 4
Comparison of Algorithms

	HC-3	Zurel's	Casanova	Lehmann's	SA
Peak Optimality	High	Very High	Very High	Good	Very High
Initial Response Time	Fast	Slow	Fast	Fast	Fast
Short Time Approximation	High	N/A	Low	Good	High
N. of Parameters	0	1	5	0	2
Monotonicity	No	No	No	Yes	No
Winner Price Monotonicity	Yes	(not proven)	No	Yes	No

ables the system to utilize much time for pricing and to have a margin for communication overheads in allocation protocols. Therefore we used one second for the time limit of computation.

Concerning the above issue, application developers must know which values should be set for several parameters used in the algorithm. In general, predicting appropriate parameters for an algorithm is very difficult before running and testing an actual system that uses it. Our algorithm does not force developers to tune a number of parameters without a loss of performance.

Of course, monotonicity is an essential property for achieving a truthful auction mechanism. Here, only Lehmann's algorithm satisfies monotonicity. None of the other approximation algorithms satisfies monotonicity [4]. Rather, we argue that it is important to analyze how the approximated allocations are acceptable for the actual users of the system. As mentioned in the last section, our algorithms (excluding SA) satisfy WPM and Weak-WPM to avoid *unacceptable* allocations that can be noticed by other bidders. A discussion about various *monotonicity* and truthfulness is shown in [24]. However, our proposed WPM and W-WPM are prepared to show acceptability of allocations in an auction.

[4] A counter-example of monotonicity on our algorithms appears in [15]

For incentive compatible auction mechanisms, competitive analysis has been done[22][4]. Competitive analysis is a theoretical analysis about relationships between actual social efficiency of allocations (e.g., the sum of winners' valuations) and optimal allocations. Let B be a set of all bids, $E_A(B)$ be the sum of valuations for the actual allocations for B, and $E_{opt}(B)$ be the sum of valuations for the optimal allocations for B, respectively. Here, c-competitive means that it is guaranteed for every auction B, $E_A(B) \leq E_{opt}(B)$[22].

Since our proposed approximation might not make the auction incentive compatible, those analyses cannot be directly applied to our approach. Furthermore, in our approach, we allow the algorithm to have time limit for computation. There is a possibility that the hill-climbing search does not improve allocations that are obtained by Lehmann's greedy approach. When we could assume single-minded bidders and the all valuations are true values, the sum of valuations for the approximated allocations should be $\sqrt{k}$-competitive for the number of items k, that is guaranteed by Lehmann's mechanism[26]. Competitive analysis can be applied for revenues instead of social efficiency. However, in this paper, we do not present a pricing mechanism so it cannot be applied for revenues in our approximation. Therefore, the impact of our proposed approach for these theoretical aspects is limited.

5.3. Difficulty of Preparing Test Sets

It is very important to prepare good datasets to evaluate algorithms. In this paper, we mainly used auction problem datasets that do not contain *dominated bids*. *Dominated bids* are bids that can be easily identified as bids that must not be winners. A dominated bid has a bid that has the same or a subset of items but higher valuation in the same auction problem. When an auction problem has dominated bids, the problem space can be easily shrunk down and therefore it is far easier to solve. For fair and meaningful evaluation of winner determination algorithms, datasets that have no dominated bids have been used.

However, in some bid distributions, it is very hard to generate a large number of non-dominated bids. For example, when we have two bids $< \{a, b, c\}, 30 >$ and $< \{d, e\}, 20 >$, and a new bid $< \{a, b\}, 35 >$ is generated for an additional bid, a bid $< \{a, b, c\}, 30 >$ should be deleted from the auction problem since it is dominated by $< \{a, b\}, 35 >$. Therefore, the total number of bids in the auction problem does not increase while we generate a new bid. Furthermore, when we generate a problem with distribution L3 with $constant\_items$=2 and the total number of items is n, there should be only n^2 or less of non-dominated bids. For this reason, we only used limited datasets based on distributions L2, L3, L4, L6, and L7 but not L1, L5 and others. This restricts possible bid distributions for data preparation, and therefore we can only evaluate algorithms with limited and artificial datasets.

Figure 5 shows a result for a dataset with 20,000 bids and 256 items that contain dominated bids in their auction problems. Here, we can see that the advantage of our algorithm is less than the result shown in Figure 2. This is because the actual problem space could be shrunk down to easier problems. However, notice that our algorithms do not have serious disadvantages in average performance even in this case. Furthermore, our algorithm tries to keep dominated bids as losers to satisfy Weak-WPM. This property is missing or not proven for other algorithms, excluding Lehmann's greedy. We observed that Casanova and SA produced many winners that violate Weak-WPM.

5.4. Communication Overhead

The communication overhead problem is that of the communication overhead incurred between an auctioneer and bidders in exchanging bid information (e.g., [43]). In [35], Sandholm pointed out that it is relatively easy to solve a winner determination problem when it has a huge number of bids but a small number of items.[5] It can be interpreted that the problem is rather the communication overhead for gathering too many bids [25]. In particular, it takes a certain overhead when we use a kind of agent communication protocol via Internet to gather bid information for an auction.

In our experiments, we used a CATS format file (a simple text file) to store information about bids in an auction. Typically, the program spent 200msec of CPU time to load 20,000 bids, and 1000msec for 100,000 bids. This is relatively large and hard since our cutoff time was set around 100msec to 1000msec for each approximation. Furthermore, it takes much more time when the data are stored on slower storage devices. When we gather such a number of bids via network rather than from a local storage, it will take much time.

[5] Note that this "small number" is typically less than 30, since the algorithm could use $O(2^n)$ of memory spaces. Therefore, it cannot be applicable to our experimental cases.

Table 5
Time Performance on 20,000 bids-256 items with dominated bids

	L2		L3		L4		L6		L7		average	
greedy(c=0.5)	0.9997	(23.6)	0.9632	(24.5)	0.9350	(24.9)	0.9361	(23.7)	0.7401	(23.7)	0.9148	(24.08)
greedy-3-seq	1.0000	(71.4)	0.9632	(69.2)	0.9966	(74.0)	0.9923	(77.6)	0.7574	(76.4)	0.9419	(73.7)
greedy-3-para	1.0000	(27.7)	0.9632	(27.4)	0.9966	(28.7)	0.9923	(29.5)	0.7574	(30.1)	0.9419	(28.68)
HC(C=0.5)-100ms	1.0001	(100)	0.9750	(100)	0.9544	(100)	0.9493	(100)	0.8529	(100)	0.9463	(100)
HC-3-seq-100ms	1.0001	(100)	0.9688	(100)	0.9967	(100)	0.9925	(100)	0.8407	(100)	0.9598	(100)
HC-3-para-100ms	1.0001	(100)	0.9753	(100)	0.9972	(100)	0.9931	(100)	0.9544	(100)	0.9840	(100)
HC(C=0.5)-1000ms	1.0001	(1000)	0.9856	(1000)	0.9771	(1000)	0.9601	(1000)	1.0111	(1000)	0.9868	(1000)
HC-3-seq-1000ms	1.0001	(1000)	0.9813	(1000)	0.9979	(1000)	0.9945	(1000)	1.0051	(1000)	0.9958	(1000)
HC-3-para-1000ms	1.0001	(1000)	0.9856	(1000)	0.9987	(1000)	0.9962	(1000)	1.0297	(1000)	1.0021	(1000)
Zurel-1st	0.5495	(9511)	0.9713	(211)	0.9959	(862)	0.9888	(776)	0.6018	(1309)	0.8215	(2534)
Zurel	1.0000	(11332)	1.0000	(402)	1.0000	(1878)	1.0000	(1628)	1.0000	(2685)	1.0000	(3585)
casanova-10ms	0.1950	(10)	0.0063	(10)	0.0047	(10)	0.0110	(10)	0.0123	(10)	0.0458	(10)
casanova-100ms	0.3894	(100)	0.0558	(100)	0.0377	(100)	0.0855	(100)	0.0993	(100)	0.1335	(100)
casanova-1000ms	0.9410	(1000)	0.4223	(1000)	0.2601	(1000)	0.3534	(1000)	0.6845	(1000)	0.5323	(1000)

(each value in () is time in milliseconds)

However, considering a differential bid updating approach for iteratively conducted combinatorial auctions, the overhead will be much smaller than loading all bid information for each time of iteration when there are small numbers of updates for bids. Therefore, it is meaningful to achieve such fast approximation algorithms for an auction with a large number of bids.

Furthermore, it is possible to make a concurrent and *pipelined* mechanism that has a thread for gathering bid information for the next auction and another thread for approximating winners in the current auction. In this case, there will be negligible overhead for loading bid data since it will be processed simultaneously. For the above reason, we excluded communication overhead (e.g., overhead of loading data from a file) from the recorded computation time in our experiments.

To solve this communication overhead problem, using complex *Bidding Language* is one approach. In [25], basic definitions and discussions about *Bidding Language* are shown. This approach describes bids not only by simple *OR* representation but rather by their underlying semantics and algorithms to generate those bids.

For example, describe bids as "*generate bids that contain items a, b, and c and assign their base prices as follows: a=10, b = 15, and c = 5. And then assign valuation for bids by a quadratic function*". This complex description will produce $2^3 - 1$ bids at once. However, there is no common bidding language that is widely used excluding *OR* bids. Furthermore, it is easy to represent the same results of bids that are described in other bidding languages. In this paper, we focused on the case where we use *OR* bid representation. Further discussions about *Bidding Language* exceed the scope of our paper and are therefore omitted.

6. Related Work

6.1. Approaches for Optimization Problems

There are really many approaches to optimization problems. Linear programming is one of the well-known approaches in this area. The winner determination problem on combinatorial auctions can be transformed into a linear programming problem. Therefore, it is possible to use a linear programming solver for the winner determination problem.

CPLEX is a well-known, very fast linear programming solver system. In [43], Zurel et al. evaluated the performance of their presented algorithm with many data sets, compared with CPLEX and other existing implementations. While the version of CPLEX used in [43] is not up-to-date, the shown performance of Zurel's algorithm is approximately 10 to 100 times faster than CPLEX. In this paper, we showed that our approach is at least several times faster than Zurel's approach with the same optimality on average for large-scale winner determination problems. Therefore, the performance of our approach is competitive enough with CPLEX or similar solver systems. This is natural since Zurel's and our approaches are specialized for combinatorial auctions, and also focus only on faster approximation but do not seek optimal solutions.

Random-walk search is also a strong approach for approximating combinatorial optimization problems. There have been many algorithms proposed based on random-walk search mechanisms.

In [18], Casanova was proposed, which applies a random walk SAT approach for approximating the winner determination problem in combinatorial auctions. In this paper, we showed that our approach outperforms Casanova when the time constraint is very hard but the problem space is really large.

Simulated Annealing (SA) is another similar approach. We prepared an SA-based extension for our approach and we confirmed it increases the performance when the problem size is relatively small. However, SA needs random-walk in the early stage of its search and it decreases performance on short-time approximation.

Genetic Algorithm is another similar approach. In [2], Avasarala et al. proposed an approach for the winner determination problem on combinatorial auctions. However, in [2], they noticed that their algorithm is not effective for approximation in short time but is effective for obtaining higher optimal solutions with enough computation time. Random-walk searching is really effective approximation approach for combinatorial optimization problems. However, it is not effective when there are such hard time constraints. We focused on solving problems that are hard for such random-walk search approaches.

6.2. Approaches to Obtain Optimal Solutions

There have been a lot of works on obtaining optimal solutions for winner determination in combinatorial auctions [10]. For example, CABOB [36] and CASS [12] have been proposed by aiming to get the optimal allocations.

In [18], it is shown that the Casanova algorithm outperforms approximation performance of CASS on winner determination. In this paper, we showed that our approach outperforms Casanova in settings of a very large number of bids in an auction. Therefore, our approach should also outperform CASS in the same settings.

In [36], Sandholm et al. showed that CABOB outperforms CPLEX in several settings. However, by indirect comparison of CPLEX and Zurel's approach, our algorithm should outperform CABOB in our shown settings. We argue that our approach is rather complementary to those algorithms that are seeking exact optimal solutions. It is not fair to compare their approximation performances when one guarantees obtaining optimal solutions but the other does not. Our approximation approach only covers large size problem settings that can only be handled by specialized approximation algorithms. Our approach does not contribute to advances in developing algorithms to obtain optimal solutions directly.

6.3. Greedy Approaches

Some researchers have noticed the better performance of simple greedy and incremental approaches for very large-scale problems. For example, [35] noticed the ease of approximation on very large auction problems. In [26], Lehmann et al. mentioned that a simple greedy approach obtains very high results when the auction problem is rather huge.

Also in [20], Kastner et al. mentioned a potential capability of a simple incremental search approach to apply to very large auction problems and discussed the sensitivity for the number of bids in an auction. However, there is little mentioned about a detailed comparison of actual performances for several different types of datasets. In [20], they only presented their preliminary experimental results on a dataset that is based on a single bid distribution.

Guo et al. [16] proposed similar local-search based algorithms and they argued that their approach is good for the settings of a large number of bids in a combinatorial auction problem. However, in [16], they presented very limited experimental results and little analysis or comparison to other high performance algorithms. Also in [16], they did not propose an idea that is similar to our multiple bid-weighting search. We argue that this multiple weighting search approach is very effective and that it distinguishes our approach from others. Also, we showed a detailed analysis of our experiments based on datasets generated by possible different bid distributions. We also showed direct comparisons to Zurel's approach presented in [43].

6.4. Other Approaches

When we have some assumptions about models for valuation of bids, we can utilize those assumptions for better approximation. Dobzinski et al. proposed improved approximation algorithms for auctions with submodular bidders [11]. Penya et al. proposed a reverse combinatorial auction mechanism for electricity markets[31]. The mechanism uses a second-price sealed-bid multi-item auction with supply function bidding rather than fixed valuation bidding. They reported that the complexity of winner determination could be far less than other reported problems. In our approach, we do not assume any problem-specific as-

sumptions about models for valuation of bids to keep generality of proposed algorithms. Lavi et al, reported an LP-based algorithm that can be extended to support the classic VCG [23]. Those studies mainly focused on theoretical aspects. In contrast to those papers, we rather focus on experimental analysis and implementation issues. Those papers did not present experimental analysis of the settings with a large number of bids as we presented in this paper.

Using sequential auctions [5] is another approach to overcoming the communication cost problem. Koenig et al. proposed a multiple-round auction mechanism that guarantees the upper bound of communication cost as fixed size k, that is independent from the number of agents or items in the auction [21]. In our approach, we only assume that there is a small number of updated bids from the last round of the auction. Although our algorithm itself can approximate winners within a very short time with a huge number of updated bids, the communication cost problem remains. This is a limitation of our approach.

7. Conclusions

In this paper, we showed that our approximation algorithms for combinatorial auctions could produce excellent optimality of approximated winners from a large number of bids within a very short computation time that can be applied to fine-grained resource allocation in a ubiquitous computing environment. In particular, our proposed algorithms outperformed other algorithms in optimality when a hard time constraint existed. Furthermore, we compared desirable properties for those algorithms that will be considered when they are actually used within application systems. Through the comparison, we showed that our approach has certain advantages on acceptability of approximated allocations, ease of use, and scalability for larger problems.

In this paper, we focused on the winner determination problem. Therefore, we eliminated other important issues to be solved. The pricing mechanism is an important part of an auction mechanism. A number of pricing mechanisms are proposed for different goals and situations (e.g., [26], [30], etc.). Although it is possible to treat our mechanism as a simple *ascending* combinatorial auction, many issues will be left unsolved. Furthermore, when we consider two-sided cases (i.e., combinatorial exchange [30]), we need to solve the pricing issue for sellers. In this paper, we left the pricing issue for future work.

References

[1] Lachlan L.H. Andrew, Stephen V. Hanly, and Rami G. Mukhtar. Active queue management for fair resource allocation in wireless networks. *IEEE Transactions on Mobile Computing*, pages 231–246, Feb. 2008.

[2] Viswanath Avasarala, Himonshu Polavarapu, and Tracy Mullen. An approximate algorithm for resource allocation using combinatorial auctions. In *Proc. of The 2006 WIC/IEEE/ACM International Conference on Intelligent Agent Technology(IAT2006)*, pages 571–578, 2006.

[3] Michael O. Ball, George L. Donohue, and Karla Hoffman. Auctions for allocation of airspace system resources. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, chapter 20, pages 507–538. The MIT Press, 2006.

[4] Laid Blumrosen and Noam Nisan. Combinatorial auctions. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, chapter 11, pages 267–299. Cambridge University Press, 2007.

[5] C. Boutiler, M. Goldszmidt, and B. Sabata. Sequential auctions for the allocation of resources with complementarities. In *Proc. of International Joint Conference on Artificial Intelligence(IJCAI1999)*, pages 527–534, 1999.

[6] Estelle Cantillon and Martin Pesendorfer. Combination bidding in multi-unit auctions. *Working Paper of Harvard Business School and London School of Economics*, 2004.

[7] Chris Caplice, Clint Plummer, and Yosef Sheffi. Bidder behavior in combinatorial auctions for transportation services. *Working Paper of Massachusetts Institute of Technology Center for Transportation and Logistics*, 2004.

[8] Peter Cramton, Yoav Shoham, and Richard Steinberg. *Combinatorial Auctions*. The MIT Press, 2006.

[9] Elisabeth Crawford and Manuela Veloso. Mechanism design for multi-agent meeting scheduling. *Web Intelligence and Agent Systems*, 4(2):209–220, 2006.

[10] Sven de Vries and Rakesh V. Vohra. Combinatorial auctions: A survey. *International Transactions in Operational Research*, 15(3):284–309, 2003.

[11] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proc. of the seventeenth annual ACM-SIAM symposium on Discrete algorithm(SODA2006)*, pages 1064–1073. ACM Press, 2006.

[12] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approarches. In *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI1999)*, pages 548–553, 1999.

[13] N. Fukuta and T. Ito. Short-time approximation on combinatorial auctions – a comparison on approximated winner determination algorithms. In *Proc. of The 3rd International Workshop on Data Engineering Issues in E-Commerce and Services(DEECS2007)*, pages 42–55, June 2007.

[14] Naoki Fukuta and Takayuki Ito. Towards better approximation of winner determination for combinatorial auctions with large number of bids. In *Proc. of The 2006 WIC/IEEE/ACM International Conference on Intelligent Agent Technology(IAT2006)*, pages 618–621, 2006.

[15] Naoki Fukuta and Takayuki Ito. Toward a large scale e-market : A greedy and local search based winner determination. In

Proc. of The 20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems(IEA/AIE2007), pages 354–363, 2007. (poster presentation).

[16] Y. Guo, A. Lim, B. Rodrigues, and Y. Zhu. A non-exact approach and experiment studies on the combinatorial auction problem. In *Proc. of the 38th Hawaii International Conference on System Sciences(HICSS2005)*, page 82.1, 2005.

[17] Gail Hohner, John Rich, Ed Ng, Grant Reid, Andrew Davenport, Jayant Kalagnanam, Ho Soo Lee, and Chae An. Combinatorial and quantity discount procurement auctions with mutual benefits at mars, incorporated. *Interfaces*, 33:23–35, 2003.

[18] Holger H. Hoos and Craig Boutilier. Solving combinatorial auctions using stochastic local search. In *Proc. of the Proc. of 17th National Conference on Artificial Intelligence (AAAI2000)*, pages 22–29, 2000.

[19] Takayuki Ito and David C. Parkes. Instantiating the contingent bids model of truthful interdependent value auctions. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS2006)*, pages 1151–1158, 2006.

[20] R. Kastner, C. Hsieh, M. Potkonjak, and M. Sarrafzadeh. On the sensitivity of incremental algorithms for combinatorial auctions. In *Proc. International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems(WECWIS2002)*, pages 81–88, 2002.

[21] Sven Koenig, Craig Tovey, Xiaoming Zheng, and Ilgaz Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proc. of International Joint Conference on Artificial Intelligence(IJCAI2007)*, pages 1359–1365, 2007.

[22] Ron Lavi and Noam Nisan. Competitive analysis of incentive compatible on-line auctions. In *Proc. of the 2nd ACM conference on Electronic commerce(EC2000)*, pages 233–241. ACM, 2000.

[23] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS2005)*, pages 595–604, 2005.

[24] Ron Lavi and Chaitanya Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In *Proc. of ACM Conference on Electronic Commerce(EC2007)*, pages 252–261, June 2007.

[25] Daniel Lehmann, Rudolf Müller, and Tuomas Sandholm. The winner determination problem. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, chapter 20, pages 507–538. The MIT Press, 2006.

[26] Daniel Lehmann, Liadian Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49:577–602, 2002.

[27] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proc. of ACM Conference on Electronic Commerce (EC2000)*, pages 66–76, 2000.

[28] John McMillan. Selling spectrum rights. *The Journal of Economic Perspectives*, 1994.

[29] Dusit Niyato and Ekram Hossain. A noncooperative game-theoretic framework for radio resource management in 4g heterogeneous wireless access networks. *IEEE Transactions on Mobile Computing*, pages 332–345, March 2008.

[30] David C. Parkes, Ruggiero Cavallo, Nick Elprin, Adam Juda, Sebastien Lahaie, Benjamin Lubin, Loizos Michael, Jeffrey Shneidman, and Hassan Sultan. Ice: An iterative combinatorial exchange. In *The Proc. 6th ACM Conf. on Electronic Commerce (EC2005)*, 2005.

[31] Yoseba K. Penya and Nicholas R. Jennings. Optimal combinatorial electricity markets. *Web Intelligence and Agent Systems*, 6(2):123–135, 2008.

[32] Stephen J. Rassenti, Vernon L. Smith, and Robert L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[33] Fariza Sabrina, Salil S. Kanhere, and Sanjay K. Jha. Design, analysis, and implementation of a novel low complexity scheduler for joint resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, pages 749–762, June 2007.

[34] Naouel Ben Salem, Levente Buttyan, Jean-Pierre Hubaux, and Markus Jakobsson. Node cooperation in hybrid ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(4):365–376, April 2006.

[35] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[36] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Cabob: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, March 2005.

[37] M. E. Thomadakis and J.-C. Liu. On the efficient scheduling of non-periodic tasks in hard real-time systems. In *Proc. of IEEE Real-Time Systems Symp.*, pages 148–151, 1999.

[38] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, XVI:8–37, 1961.

[39] James Wang and Matti Vanninen. Self-configuration protocols for P2P networks. *Web Intelligence and Agent Systems*, 4(1):61–76, 2006.

[40] Li Xiao, Songqing Chen, and Xiaodong Zhang. Adaptive memory allocations in clusters to handle unexpectedly large data-intensive jobs. *IEEE Transactions on Parallel and Distributed Systems*, 15(7):577–592, July 2004.

[41] Tao Xie and Xiao Qin. Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters. *IEEE Transactions on Parallel and Distributed Systems*, 19(5):682–697, May 2008.

[42] Jianchang Yang and D. Manivannan. An efficient fault-tolerant distributed channel allocation algorithm for cellular networks. *IEEE Transactions on Mobile Computing*, 4(6):578–587, Nov. 2005.

[43] Edo Zurel and Noam Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proc. of the Third ACM Conference on Electronic Commerce (EC2001)*, pages 125–136, 2001.

APPENDIX

Proof of Proposition 1.

Proposition 1. *Our proposed winner determination algorithms, except for the simulated annealing-based algorithm, produce allocation W_{fin} that satisfies WPM when the algorithm reaches an end.*

Proof: Suppose there are arbitrary bundles $X, Y \subseteq M$ such that $X \subseteq Y$ and two bids $b_i(X)$ and $b_j(Y)$ with prices $v_i(X)$ and $v_j(X)$ such that $v_i(X) > v_j(Y)$. We will show $b_j(Y) \notin W_{fin}$.

Lemma 1.1 *Initial winner set W_0 always satisfies WPM.*

Proof of Lemma 1.1 In our algorithm, initial allocation W_0 before updating by hill-climbing always satisfies WPM. This is because the initial allocation is generated by Lehmann's algorithm. In Lehmann's algorithm, $b_i(X)$ is always ordered before $b_j(X)$ with any $c(c \geq 0)$. Therefore, there is no chance for $b_j(X)$ to become a winner. □

Lemma 1.2 *A winner set W_t such that $b_j(X) \in W_t$, will not be a final winner set.*

Proof of Lemma 1.2 Summary: After attaining an initial winner set, our algorithms repeatedly update the winner set by using our proposed hill-climbing algorithm. Here, we show that final winner W_{fin} satisfies WPM by using the proof by contradiction.

Suppose W_f is the final solution such that $b_j(Y) \in W_f$.

When our algorithm tries to insert $b_i(X)$ into W_f, $b_i(X)$ only conflicts with $b_j(Y)$ because $X \subset Y$.

Thus, this insertion procedure will not remove other bids from W_f. And then $b_i(X)$ (and possibly some other bids $b_k, b_l, ...$ for an item set $Z = Y - X$) will be inserted instead of $b_j(Y)$, and W_{f+1} will be produced as a candidate of the next winner set.

Hence $v_i(X) > v_j(Y)$ and $v_k, v_l, ... > 0$, the total revenue for W_{f+1} must be greater than W_f. Therefore, W_f has at least one possible updated winner set W_{f+1}. This contradicts with the assumption that W_f is W_{fin}. So W_{fin} will not include $b_j(Y)$. □

Lemma 1.3 *It is guaranteed that our hill-climbing algorithm stops.*

Proof of Lemma 1.3 It is trivial to show that our hill-climbing algorithm will stop since every winner set update will occur when the total revenue is higher than before, and it should be equal to or smaller than optimal revenue. Also note that the number of bids is bounded. □

Therefore, it is guaranteed that our algorithm produces W_{fin}. □

Proof of Proposition 2.

Proposition 2. *In terms of any allocations that are achieved during computation (as an anytime algorithm), our proposed winner determination algorithms, except for the simulated annealing-based algorithm, satisfy Weak-WPM.*

Proof: We show $b_j(X) \in W_{fin}$ is always false for two arbitrary bids $b_i(X)$ and $b_j(X)$ for the same bundle $X \subseteq M$, and $v_i(X) > v_j(X)$.

Lemma 2.1 *Initial winner set W_0 always satisfies Weak-WPM.*

Proof of Lemma 2.1 Recall that all algorithms that satisfy WPM will satisfy Weak-WPM and since W_0 satisfies WPM, W_0 clearly satisfies Weak-WPM. □

Lemma 2.2 *Arbitrary updated winner set W_t at time t satisfies Weak-WPM if the direct ascendant winner set W_{t-1} satisfied Weak-WPM.*

Proof of Lemma 2.2 Summary: We will provide separate proofs whether W_{t-1} includes $b_i(X)$ or not.

Lemma 2.3 *Arbitrary updated winner set W_t will not include $b_j(X)$ if the direct ascendant winner set W_{t-1} satisfied Weak-WPM and $b_i(X) \in W_{t-1}$.*

Proof of Lemma 2.3 Since W_{t-1} satisfies Weak-WPM, $b_j(X)$ is not included in W_{t-1}. When our algorithm tries to insert $b_j(X)$, only $b_i(X)$ conflicts with $b_j(X)$. Recall that $v_i(X) > v_j(X)$, the produced updated winner set candidate $W_{t-1'}$ has smaller revenue than that of W_{t-1}. Therefore, $W_{t-1'}$ will not be W_t. □

Lemma 2.4 *Arbitrary updated winner set W_t will not include $b_j(X)$ if the direct ascendant winner set W_{t-1} satisfied Weak-WPM and $b_i(X) \notin W_{t-1}$.*

Proof of Lemma 2.4 Since W_{t-1} satisfies Weak-WPM, $b_j(X)$ is not included in W_{t-1}. Suppose we have a list U that includes all bids that are not included in W_{t-1}. Here, recall that U is an ordered list, and in U, $b_j(X)$ is ordered after $b_i(X)$. So $b_i(X)$ has a chance to update W_{t-1} before $b_j(X)$ has.

Suppose $b_i(X)$ successfully updates W_{t-1}. No chance to update will be given for $b_j(X)$ and the updated result will become W_t immediately. Here, since $b_j(X)$ conflicts with $b_i(X)$, W_t will not include $b_j(X)$.

Suppose $b_i(X)$ failed to update W_{t-1}. Let $b_k, b_l, ...$ be bids that are inserted with the bid $b_i(X)$ in the trial, and the sum of revenue for $b_k, b_l, ...$ be v. Recall that because $v_i(X) + v > v_j(X) + v$, the trial with $b_j(X)$ will also fail to update W_{t-1}. Therefore, W_t will not include $b_j(X)$. □

As shown above, W_t satisfies Weak-WPM whether W_{t-1} includes $b_i(X)$ or not. □

Recall that W_0 satisfies Weak-WPM, so all updated winner sets of W_0 satisfy Weak-WPM. Thus, for any winner set W_t, including W_0 and W_{fin}, Weak-WPM is satisfied. □