A Pre-processing Approach for Fast and Stable Allocations on Approximation-based Pricing for Multi-unit Combinatorial Auctions

Naoki Fukuta *

Abstract

In this paper, some discussions about a pre-processing approach of fast approximation on stable pricing and allocation of resources in a combinatorial auction are presented. On the discussions, an approximate auction which has VCG-like pricing mechanism is used which considers the situation when a cancellation of winner bid(s) could be occurred after its completion of winner determination. An analysis about stable approximate pricing mechanisms against cancellation of a winner after its winner determination is also presented, where a single-unit non-combinatorial reserve price bidding on a combinatorial auction is employed on it. The pricing algorithms employ a kind of approximate allocation and pricing algorithms that are capable of handling multi-unit auctions with reserve price biddings. We consider a scenario on the allocation of electricity power usage rights while considering electricity generation costs on the power suppliers as well as external conditions such as violations of regulations done by some bidders outside the auction mechanism. An experimental analysis is presented on the scenario and the presented algorithms efficiently produced approximation allocations that are necessary in the pricing phase, while keeping the same level of stability in the case of single-winner cancellation scenario.

Keywords: combinatorial auction, approximation, algorithm

1 Introduction

Since it is crucial to realize dynamic allocations of limited resources with rational and self-interested attendees[17][1], numerous efforts have been done to provide better criterion and associated mechanisms to stably assign those limited resources, e.g., by fair allocations without money[2], or by efficient allocations using money[4].

In the context of realizing stable allocations using money, there have been proposed a number of variants of auctions to capture various conditions in the resource allocation problems[4]. For example, multi-unit auctions can handle amount

* Shizuoka University, Shizuoka, Japan
of resources to be allocated for each bidder in an auction[22], and combinatorial
auctions can handle complex allocations among different types of resources[23][4].
There exist numerous works on giving important theoretical properties, i.e., incentive compatibility[19][20], strategy proofness and other related properties[28], or considering locally envy free equilibrium[6] or similar properties[7] for more realistic
scenarios. In these cases, stability of allocations is rather discussed in the context of
how the allocations give the players incentives to follow the decisions made by the
mechanisms while there could be some exceptions as discussed below.

There might be cases that one or more winners have been cancelled for allo-
crating items that were auctioned[3], because of keeping revenue of the auctioneer,
or just some winners did not have the rights to have them on a regulation. Such
cancellations could be done by both buyers, sellers, and even by the auctioneers,
due to some misconduct of the buyers or sellers. Although there are some ideas
to prevent buyers from doing misconduct such as false name biddings, these mech-
anisms could not easily be realized when an auction would have other important
characteristics[26]. Furthermore, such cancellations could be caused from the rea-
sons that were completely outside of the mechanisms, e.g., a violation of law or
regulations by the bidders’ activities on the outside of the auction and its associated
punishment has been applied to the bidders’ rights that were also affected to the
auction, even after the decision about the allocation has been made. Therefore,
having considerations of such unexpected cancellations on an auction mechanism
and its underlying algorithms is still an important issue to be discussed[10].

Because of the nature of combinatorial optimization problem, only a small change
of its condition may affect to large parts of the allocation when we re-calculate its
(sub)optimal allocations from scratch. In the above-mentioned context, we would
say the allocation may not be stable for cancellations of bids. Although there is
an approach to realize such a stability for a cancellations of bids[12], there are still
some issues on its computation speed when it is applied to large-scale problems or
massively repeated simulations.

In this paper, a preprocessing-based fast and stable approximate pricing algo-
rithm against cancellation of a winner after its winner determination is presented1.
In there, a single-unit non-combinatorial reserve price biddings on multi-unit com-
binatorial auction can also be employed to consider the costs of producing resources
to be allocated. The approach should be helpful for numerous types of problems
on making consensus among entities in difficult situations, e.g., some conflicts ex-
isted among those entities[18], as well as assigning limited amount of resources to
self-interested entities (e.g., agents)[25].

2 Preliminary

2.1 Multi-unit Combinatorial Auctions

Combinatorial auction is an auction that allows bidders to place bids for a combi-
nation of items rather than a single item[4].

The winner determination problem on single unit combinatorial auctions is de-
defined as follows[4]: The set of bidders is denoted by \( N = 1, \ldots, n \), and the set of

---

1 Initial idea has been presented in [10], and further discussion has been done in [12].
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, for all $j \in M$.

\[ \forall j \in M, \sum_{i \in N} \sum_{S \ni j} x_i(S) \leq 1 \]

The winner determination problem is the problem to maximize total revenue for feasible allocations $X \ni x_i(S)$.

\[ \max_X \sum_{i \in N} \sum_{S \subseteq M} v_i(S)x_i(S) \]

When some items in auction can be replaceable each other, i.e., they are indistinguishable, the auction is called multi-unit auction. Multi-unit combinatorial auction is the case when some items are indistinguishable in a combinatorial auction[4]. Multi-unit combinatorial auction can be applied to electricity allocation problems, and other problems that considers quantitative or countable items in allocation problem[22].

In [13], [14], and [15], it has been shown that the presented hill-climbing approach outperforms SA[13], SAT-based algorithms[16], LP-based heuristic approximation approach[29], and a recent LP solver product in the setting when an auction has a massively large number of bids but the given time constraint is very hard.

### 2.2 Winner Approximation and Pricing

It is crucial for an auction mechanism to have a proper pricing mechanism to incentivize bidders to reveal their true valuations of items appropriately[4]. In VCG(Vickery-Clarke-Groves) mechanism[27][5], prices that winners will pay will be given as follows[24]. A payment $p_n$ for a winner $n$ is calculated by

\[ p_n = \alpha_n - \sum_{i \neq n, S \subseteq M} v_i(S)x_i(S) \]

Here, the right part of the right side of the equation denotes the sum of all bidding prices of won bids, excluding the bids that are placed by the bidder $n$. The left part of the right side of the equation, $\alpha_n$ is defined by

\[ \alpha_n = \max_{i \neq n, S \subseteq M} \sum v_i(S)x_i(S) \]

for a feasible allocation $X \ni x_i(S)$. This means that the $\alpha_n$ is the sum of all bidding prices of won bids when the allocation is determined as if a bidder $n$ does not place any bids for the auction.

In [24], Nisan et al. showed that optimal allocations should be used for VCG-based pricing to make the auction incentive compatible (i.e., revealing true valuations is the best strategy for each bidders). Also, Lehmann et al. showed that VCG-based pricing with approximate winner determination will not make the auction
incentive compatible even when it is assumed that all bidders are single-minded (i.e., each bidder can only place single bid at each auction) [21].

To overcome this issue, Lehmann et al. prepared a special pricing mechanism that can only be applied for their approximate greedy winner determination [21]. However, this pricing mechanism can only be applied to their allocation algorithm but it cannot be applied to other approximation allocation algorithms. Also the mechanism is incentive compatible only when single-minded bidders are assumed [21].

The main problem in which VCG-based pricing is applied to approximation allocation of items is that there are the cases that: (1) the price for a won bid is rather higher than the bid price, and (2) the price for a won bid is less than zero, it means the bidder will win the items and also will obtain some money rather than paying for it [24]. In the situation of (1), it breaks individual rationality (i.e., the one will not pay a higher price than the placed bid when the one won the bundle of items). Also the situation of (2) is not preferable for both auctioneers and sellers.

To overcome the inability results shown in [24] and [21], a more relaxed condition called *Strong Winner Price Monotonicity (SWPM)* and an associated pricing algorithm *transformToSWPM* have been introduced [8], rather than applying the condition of strict incentive compatibility to a VCG-like pricing with approximation allocations.

As mentioned in [8], the algorithm *transformToSWPM* is normally used with a good winner determination algorithm for the preparation of initial allocations to shorten the calculations for pricing. In the latter part of this paper, we assume that the MHC algorithm is used for providing initial allocations of items to those pricing algorithms, as described in [8].

### 2.3 Approximate Pricing with Reserve Prices

When approximately solving a combinatorial auction problem with reserve price biddings, a naive approach may produce a winner bid which includes a set of items whose prices are higher than the sum of reserve prices. For example, when we naively apply the pricing mechanism *transformToSWPM* introduced in [8], the mechanism does not satisfy reserve price conditions (i.e., each winner places the price which is higher than the best combination of reserve price bids for the items in the bundle), since the mechanism used Lehmann’s approximation allocation [21].

To overcome the issue shown in the previous part of this section, the algorithm *transformToSWPMRP* that addresses the issue mentioned was presented [9] [11].

### 2.4 Approximation on Locally-strong Winner Price Monotonicity

Here, we would consider a way to have a more relaxed condition than *Strong Winner Price Monotonicity* for better stabilization of pricing and re-allocation on bid cancellations. A weaker condition, *Locally-strong Winner Price Monotonicity*\(^2\), can be defined as follows:

**Definition 1. (Locally-strong Winner Price Monotonicity: LWPM)** For any non-empty bundle \(s' \subseteq S\) for a bidder \(j \in N\), \(X_j(s') = 1\) (i.e., a winner) and whose bid price is \(v_j(s')\), the Lehmann’s greedy allocation of bids for the bundle of

\(^2\)The idea has been initially presented in [10]
items $s'$ by bids from bidders $N' = \{i| i \in N, i \neq j\}$ that satisfies $X_{i\neq j}(s'') = 0$ (i.e., not winners) for any $s'' \supseteq s'$, always produces non-efficient allocation for the bundle $s'$ (i.e., the total price of greedy allocation is less than $v_j(s')$).

Here, to satisfy this Locally-strong Winner Price Monotonicity, a modified version of the algorithm transformToSWPMRP which is called transformToLWPMRP has been introduced [10], as follows:

```
1: function transformToLWPMRP(Alloc, L, Stocks)
2:   RemainedBids:= L - Alloc;
3:   sortByLehmannC(RemainedBids);
4:   clear(payment);
5:   for each $b \in Alloc$
6:     RestStocks:=getPlacedStocksInBid(b);
7:     AllocForB:=greedyAlloc(RestStocks, RemainedBids);
8:     NewAlloc:= Alloc - \{b\} + AllocForB;
9:     if price(Alloc) < price(NewAlloc) then
10:       return transformToLWPMRP(NewAlloc, L, Stocks);
11:     else
12:       RemainedReserveBids:=getReservedBids(RemainedBids)
13:       AllocForR:=greedyAlloc(RestStocks, RemainedReserveBids);
14:       NewAllocR:= Alloc - \{b\} + AllocForR;
15:       if price(Alloc) < price(NewAllocR) then
16:         return transformToLWPMRP(NewAllocR, L, Stocks);
17:       else
18:         payment_{b} = price(NewAlloc) − price(Alloc - \{b\})
19:       end for each
20:   return (Alloc,payment)
```

The algorithm transformToLWPMRP applies its pricing and respective re-allocations of winners in a similar way that the algorithm transformToSWPMRP does. The major difference between them is which property is applied to justify the allocations and calculate the price to pay for each winner. On the algorithm transformToSWPMRP, it applied Strong Winner Price Monotonicity, while the algorithm transformToLWPMRP applied Locally Strong Winner Price Monotonicity, instead, as both algorithms also applied Reserve Price Condition. For instance, the line 6 in the above algorithm is different. Here, a function getPlacedStocksInBid(b) returns the bundle of items that the bid $b$ placed to, as a set of pairs of the items and their unites to be won.

The two algorithms, transformToLWPMRP and transformToSWPMRP were compared to see the differences on its allocation stability over pricing in the condition that any single winner has been cancelled on each auction. Here, the comparison has been done on how many winners have been removed from winners due to such single winner cancellation had been made and re-pricing had been applied. The auction problems that were used are the same datasets used in [11], one of which includes 1000 bidders and 100 percent of electricity production ratio, with 515024 bids and 67392 reserve-price bids, and another of which includes 3566 bidders and 100 percent of electricity production ratio, with 1710967 bids and 238584 reserve-price bids.

It is observed that the both algorithms produced the same number of winners for those auction problems. In [10], the comparison result in the case of 1000 bidders for
the above scenario has been presented to analyze how those winners will be shifted when single winner has been cancelled and the respective pricing and allocation adjustment algorithms were applied. In this case, it was observed that the algorithm \texttt{transformToSWPMRP} produced more losers that had been shifted from winners than that in the algorithm \texttt{transformToLWPMRP}. On the algorithm \texttt{transformToLWPMRP}, it produced approximately 87.5 percent\footnote{This means it has been decreased from 857 to 750.} of such winner-shifts compared with \texttt{transformToSWPMRP}[10]. Also a comparison in the case of 3566 bidders for the same scenario has also been done in the extended version of [10].

Although the algorithm \texttt{transformToSWPMRP} is tractable enough for a single problem and it produces its result much faster than that on ordinary VCG-based pricing mechanism, it still takes a long time when the scale of auction is huge. Furthermore, this makes it difficult to analyze how these kind of mechanisms make impacts to the behaviors of autonomous attendees in such auctions with a set of repeated simulations.

### 3 Improving LWPM-checking Process of Stable Allocation with Winner Cancellations

#### 3.1 Faster Approximation on \texttt{transformToSWPMRP}

To calculate \textit{Locally-strong Winner Price Monotonicity} solutions faster, a modified version of the algorithm \texttt{transformToLWPMRP} is proposed, which includes a mechanism to skip a part of its stability checking process in some conditions\footnote{In [12], line 6 of the algorithm was accidentally wrongly printed without “\texttt{Checked}” and it has been fixed in this version.}.

```plaintext
1: Checked = φ
2: function transformToLWPMRPFast(Alloc, L, Stocks)
3:   RemainedBids := L - Alloc;
4:   sortByLehmannC(RemainedBids);
5:   clear(payment);
6:   for each b ∈ (Alloc − Checked)
7:     RestStocks := getPlacedStocksInBid(b);
8:     AllocForB := greedyAlloc(RestStocks, RemainedBids);
9:     NewAlloc := Alloc − {b} + AllocForB;
10:    if price(Alloc) < price(NewAlloc) then
11:       return transformToLWPMRP(NewAlloc, L, Stocks);
12:    else
13:      RemainedReserveBids := getReservedBids(RemainedBids)
14:      AllocForR := greedyAlloc(RestStocks, RemainedReserveBids);
15:      NewAllocR := Alloc − {b} + AllocForR;
16:      if price(Alloc) < price(NewAllocR) then
17:        return transformToLWPMRP(NewAllocR, L, Stocks);
18:      else payment_b = price(NewAlloc) − price(Alloc − {b})
19:        Checked := Checked + {b}
20:    end for each
21:  return (Alloc, payment)
```

Copyright © by IIAI. Unauthorized reproduction of this article is prohibited.
The algorithm `transformToLWPMRPFast` efficiently omits the checking process of *Locally-strong Winner Price Monotonicity* once a winner has been checked in the algorithm. If the algorithm `transformToLWPMRPFast` is applied to a single-unit auction scenario, it guarantees *Locally-strong Winner Price Monotonicity* in case the order of getting \( b \) in the line 6 is preserved as the order of allocation that has been done by `greedyAlloc`. A sketch of proof is that, in case a winner bid \( b \) is said to be *Locally-strong Winner Price Monotonicity* satisfiable, there are no set of bids that can be replaced by the `greedyAlloc` function, since such bids should not remain in the remained bids nor they should not be in the `Alloc` since they should conflict to the winner \( b \) when they are in the `Alloc`. However, in the case of multi-unit scenarios, the latter condition may not be guaranteed. Therefore, the algorithm `transformToLWPMRPFast` does not guarantee *Locally-strong Winner Price Monotonicity* by itself.

### 3.2 Preprocessing Approach to Satisfy LWPM on `transformToLWPMRPFast`

Even when the algorithm `transformToLWPMRPFast` does not guarantee *Locally-strong Winner Price Monotonicity*, it can be used as a pre-processing to generate initial allocations for `transformToLWPMRP`. This will greatly improve the performance of `transformToLWPMRP`. The improved algorithm `transformToLWPMRPFastX` can be prepared as follows:

```plaintext
1: function transformToLWPMRPFastX(Alloc, L, Stocks)
2:   (NewAlloc, X) := transformToLWPMRPFast(Alloc, L, Stocks)
3:   return transformToLWPMRP(NewAlloc, L, Stocks)
```

When this approach is applied to the case used in [10], the execution performance to complete its pricing process is improved. Table 1 shows how this `transformToLWPMRPFastX` can improve the computation performance while it also guarantees LWPM as the `transformToLWPMRP` does. In the case of 3566 bidders, the whole computation with `transformToLWPMRPFastX` to produce stable allocations has been done on approximately 523 seconds, while the computation with the pure `transformToLWPMRP` used in [10] could not complete its computation for this scale.

The proof of guaranteeing LWPM on `transformToLWPMRPFastX` is simply given by the fact that the `transformToLWPMRPFast` in `transformToLWPMRPFastX` is used as a pre-processing to generate an initial allocation for the computation in `transformToLWPMRP` and thus the all properties kept on it is also effective on `transformToLWPMRPFastX`.

Note that, in case the given initial solution (i.e., `Alloc`) is good enough and there is no chance to reallocate any winners in both `transformToLWPMRPFast` and `transformToLWPMRP`, the computation time for `transformToLWPMRPFastX` will be roughly twice than that on just simply applying `transformToLWPMRP`. Since in that case the behaviors of `transformToLWPMRPFast` and `transformToLWPMRP` are identical, we can eliminate this overhead by just skipping `transformToLWPMRP` in `transformToLWPMRPFastX` and this can easily be implemented by adding a return value that indicates the number of reallocations done in the `transformToLWPMRPFast`.

---

5Note that the value in the variable X will not be used since the same value will be computed in `transformToLWPMRP` and returned from it.
4 Conclusions

In this paper, discussions about the preprocessing-based approach to realize faster computation of stable pricing and allocation of resources using an approximate auction which has VCG-like pricing mechanism when cancellation of winner bid(s) after its winner determination is considered.

As shown in the experimental result, the algorithm efficiently produced approximation allocations that are necessary in the pricing phase, while it keeps the same stability in the case of single-winner cancellation. As discussed in [11], it could behave as an approximation of VCG(Vickrey-Clarke-Groves) mechanism satisfying budget balance condition and bidders’ individual rationality without enforcing the single-minded bidders assumption, while it does not guarantee strict strategy-proofness.

In this paper, the presented analysis has been done only for a limited number of allocation scenarios for electricity usage in industries, although the mechanisms and the algorithms themselves could be applied to other problems such as bandwidth and channel allocation problem in wireless networks[23]. Applying to other possible scenarios and analyze how these pricing mechanisms affect the behaviors of attendees in an auction is one of our future work. Also further analysis on theoretical characteristics of the mechanism is future work.

Acknowledgments

The work was partly supported by Grants-in-Aid for Challenging Exploratory Research 26540162, Granst-in-Aid for Scientific Research(B) 15H02972, and JST CREST.

References


A Pre-processing Approach for Fast and Stable Allocations on Approximation-based Pricing


