An Introduction to Iterative Combinatorial Auctions

Baharak Rastegari
Department of Computer Science
University of British Columbia
Vancouver, B.C, Canada V6T 1Z4
baharak@cs.ubc.ca

Abstract

Combinatorial auctions let bidders to express their preferences on bundles of items and are necessary for optimal allocation of goods. Direct revelation of bidders's valuation on all combination of goods is exponential in the number of goods, so its expensive and could be impossible. Iterative combinatorial auctions (iterative CAs) are designed to resolve the problem of costly preference elicitation by repeatedly interacting with bidders and collecting information from them, which might guide them through preference elicitation process. This paper is an introduction on iterative CAs. We describe the computational problems regarding the design space of iterative CAs that are present in literature and also describe different design models and approaches and their limitations and advantages.

Keywords: Iterative combinatorial auctions, Ascending auctions

1 Introduction

Combinatorial auctions (CAs) are those auctions in which bidders can place bids on combinations of items, called packages (bundles), rather than just individual items. CAs are necessary for optimal auction-bases solutions to resource allocation problems with agents that have non-additive values for resources. There might be economic relations between goods in an auction: the goods may be (i) *complements*: value of a bundle is greater than the sum of the values of its parts, or the opposite case (ii) *substitutes*. These relations could be ignored by selling each item in a separate auction. CAs take the relations between goods into consideration while allocating them, by letting bidders to more fully express their preferences (e.g. by valuating bundles instead of only single items).

The most famous combinatorial auction is the combinatorial generalization of the Vickery auction (GVA). Assuming the goal is to maximize the social welfare, GVA guarantees truthful bidding and efficiency. However, despite these attractive characteristics, GVA is abandoned in practice since it has several potential weaknesses, (set aside the fact that determining the optimal allocation is NP-complete): (i) GVA needs each bidder to valuate all different subset of goods; this could be costly and complex or even impossible, (ii) it uses explicit price discrimination in the sense that two bidders may pay different prices for identical allocations, (iii) bidders' true valuation will be revealed to everyone, so bidders may fear that this information will be used by auctioneer to cheat them or by other agents to disadvantage them in future transactions. There are also several weaknesses from the point of view of seller, such as: (iv) possibility of low (or zero) seller revenues, (vi) vulnerability to shill bidding and collusion.

These various defects of the Vickery auction have led to increased interest in exploring alternative designs. Iterative combinatorial auctions (iterative CAs) are designed to address the problem of costly preference elicitation that arises due to hard valuation problem (weakness (i) in above) and the communication cost (between bidders and auctioneer). An iterative CA repeatedly interacts with bidders and collect information from them. Information feedback could support adaptive and focused elicitation. For example, an ascending price auction maintains ask prices and allows bidders to revise bids as prices are discovered. Significantly, it is often possible to determine an efficient allocation without bidders reporting (or even determining) exact values for all bundles.

Although the central concern of iterative CA design is preference elicitation problem, it has also other important benefits: (i) iterative CAs can help to distribute the computation in an auction across bidders through the interactive involvement of bidders in guiding the dynamics of the auction, and (ii) bidders can verify and validate the outcome of an auction (which could be difficult in case of VGA).

However, iterative CAs offer new opportunities to bidders for manipulation; Therefore, they need to be carefully designed. Different works in literature tried to design an iterative CA which is efficient (i.e. can find an optimal allocation) or approximates the efficiency for a general or restricted class of CAs. Ascending CAs are one the main families of iterative CAs. In an *ascending auction*, the prices in the queries to the same bidder can only increase in time. iBundle by Parkes et al.[5] and "Proxy Auction" by Ausubel et al.[6] are both examples of Ascending CAs that are proved to find the optimal allocation for unrestricted valuation.

The design space for iterative CAs is larger than for one-shot auctions, and as we will discuss later in the paper, many problems should be solved and taken care of during the design process, such as: "How to represent the information from bidders' valuations?", "How to query bidders about their valuation in each iteration?", and "How to decide the termination of the process?". Many works in literature tried to characterize different aspects of iterative CAs, compare the different potential answers to the above (and similar) questions, and find the limitations and advantages of different models by analyzing and comparing them to each other; Works by Nisan et al.[1, 2] and more generally the forthcoming book "Combinatorial Auctions" [4] are among those.

2 Preliminaries

A combinatorial auction problem is defined as follows. Let $G = \{1, ..., m\}$ denote the set of items, $I = \{1, ..., n\}$ denote the set of bidders (agents) and $S \subseteq G$ denote a bundle of items. Each bidder i has a valuation function $v^i : 2^G \rightarrow \Re^+$. Usually, the goal is to select an allocation of goods to agents that maximizes the social welfare function.

Valuation functions could be categorized into different classes: (i) sub-additive, where the goods could be substitutes, so there exists two sets of goods $S, T \subseteq G$ s.t. $v(S \bigcup T) \leq v(S) + v(T)$, and (ii) super-additive, where the goods could be complement, so there exists two sets of goods $S, T \subseteq G$ s.t. $v(S \bigcup T) \geq v(S) + v(T)$.

Most of the models assume the following assumption to simplify the design and analysis process of CAs: (i) *private value:* each bidder i knows its own value vector, (ii) *quasilinear utility without externalities*: a bidder i has utility $u_i(S, p) = v_i(S) - p$ for bundle S at price $p \geq 0$ and $v_i(\emptyset) = 0$, where valuation function domain is just the set of goods that she wins, and (iii) *monotonicity (free disposal):* $v_i(T) \geq v_i(S)$ for all $T \supseteq S$, which means that goods have non-negative value.

As we said, the goal is to maximize social welfare function, or more formally to solve the following efficient combinatorial allocation problem (CAP):

$$
\begin{aligned}
max_{S=(S_1,\dots,S_n)} & \sum_{i \in I} v_i(S_i) \\
s.t. \quad & S_i \bigcap S_j = \emptyset, \forall i, j
\end{aligned}
$$

Winner determination and Integer programming: the winner determination problem (WDP) is to find an efficient allocation, which is equivalent to solving the above equation. WDP may be formulated as an integer program:

$$
\begin{aligned}
max_{x_i(S)} & \sum_{S \subseteq G} \sum_{i \in I} x_i(S) v_i(S) \\
s.t. \quad & \sum_{S \subseteq G} x_i(S) \leq 1, \qquad \forall i \\
\sum_{S \subseteq G, j \in S} & \sum_{i \in I} x_i(S) \leq 1, \qquad \forall j \\
& x_i(S) \in \{0, 1\}, \qquad \forall i, S
\end{aligned}
$$

The concept of equilibrium can be defined in the context of CAs as follows:

Definition 2.1 (Competitive Equilibrium): *Prices, p, and allocation S^* are in competitive equilibrium (CE) if allocation S^* maximizes the payoff of every bidder and the seller revenue given prices p. Allocation S^* is said to be supported by prices p in CE.*

Theorem 2.1 (First welfare theorem): *Allocation S^* is supported in competitive equilibrium if and only if S^* is an efficient allocation.*

Following the theorem, any auction that implements an efficient allocation must determine CE prices, so many iterative CAs are designed to converge to CE prices.

3 The Design Space for Iterative Combinatorial Auctions

The design space for iterative CAs is larger than for one-shot auctions and, as it mentioned in introduction, different features should be considered in designing an iterative CA:

Timing issues: Iterative auctions may be *continuous* or *discrete* (round-based); in the former, bids are allowed to be submitted at any time, while in the latter, agents have an opportunity to revise their bids at any round. Continuous auctions can promote faster propagation of feedback information to bidders and help to quickly focus elicitation, but the winner-determination problem must be resolved whenever a new bid is submitted and that could make them infeasible.

Information feedback: Information feedback about the state of an auction can include information about the bids submitted and also aggregate information, such as price feedback and the current provisional allocation to guide biding. Information hiding is also possible to avoid (or minimize) collusion or other kinds of manipulation.

Bidding rules: Iterative auctions need to elicit information about bidders' preferences. Most of the suggested iterative auctions proceed by maintaining temporary prices for the bundles of items and repeatedly querying the bidders as to their preferences between the bundles under the current set of prices. It might then update the set of bundle prices according to the replies received. More details on different types of queries can be found in Sec. 3.1

Termination conditions: Auctions may close at a *fixed deadline*, or alternatively can have a *rolling closure* with the auction kept open while one or more losing bidders continue to submit competitive bids.

Bidding languages: Auctions need to decide on a language for expressing bids. One popular bidding language is exclusive-or (XOR), e.g. S_1 XOR S_2 indicates that an agent wants to buy at most one of S_1 and S_2 and not both of them.

3.1 Queries

Most of the current iterative CAs access the bidders's preferences by repeatedly making some queries and ask them to reply. Auctioneer could even exactly define the reply function and put constraints on how a reply should look like.

One of the common form of queries is the one in which the auctioneer provides ask prices to coordinate the bidding process. iBundle [5] is an example of iterative CAs which uses ask price as a bid improvement rule (more details about iBundle are in Sec. 4.1).

There are different types of queries that can be used by iterative auctions, such as: Value query ("what is bundle S worth?"), marginal-value query, indirect-utility query, and demand query [1].

A **demand query** specifies a price $p(S) \in \Re^+$ for each bundle S in the query. The reply of bidder i to the query is simply a bundle of items $S \subseteq M$ that maximizes $v_i(S) - p(s)$.

Nisan et al. [1] proved that if $v_i(S)$ (for all agent i and bundles S) can be represented by t bits, then each of the above queries can be efficiently (i.e. in time polynomial in n, m, and t) simulated by a sequence of demand queries with item prices. In **demand queries with item prices** the auctioneer presents a vector of item prices $p_1 ... p_m$; bidder i replies by some set $S \in M$ that maximizes $v_i(s) - \sum_{i \in S} p_i$.

3.1.1 Demand Queries

Usually two main restrictions on the types of allowed demand queries are considered:

Linear (Item-price) vs. Nonlinear (Bundle-price): In item-price auction, each item $j \in G$ has a price p_j and the price of a set S is defined as $P(S) = \sum_{j \in S} p_j$. An auction uses bundle-prices if the price of a bundle S is not necessarily the sum of the prices of the items in S.

Anonymous vs. Non-anonymous: In anonymous auctions, the prices seen by the bidders at any stage in the auction are the same. In non-anonymous auctions, each bidder i has personalized prices denoted by $p_i(S)$ where it is possible that $p_i(S) \neq p_j(S)$ for bidder $i \neq j$.

3.1.2 Demand Queries and Linear Programming

Unfortunately, winner determination problem is NP-complete so it can't be solved in polynomial time. However, we might be able to approximate it or solve it polynomially for some special classes of problems. In both cases he following *linear programming* relaxation of the above formulation can be useful.
Primal:

$$\begin{aligned} max_{x_i(S), y(k)} \sum_{S \subseteq G} \sum_{i \in I} x_i(S) v_i(S) \\ s.t. \quad \sum_{S \subseteq G} x_i(S) \leq 1, \quad \forall i \\ \sum_{i \in I} x_i(S) \leq \sum_{k \in K, S \in k} y(k), \quad \forall S \\ \sum k \in K y(k) \leq 1 \\ x_i(S), y(k) \geq 0, \quad \forall i, S, k \end{aligned}$$

Dual:

$$\begin{aligned}
& & min_{p(i),p(S),\pi} \sum_{i\in I} p(i) + \pi & \\
s.t. & \quad & p(i) + p(S) \geq v_i(S), & \quad \forall i, S \\
& & \pi - \sum_{S\in k} p(S) \geq 0, & \quad \forall k \\
& & p(i), p(S), \pi \geq 0, & \quad \forall i, S
\end{aligned}$$

Nisan et al.[1] showed that despite the exponential number of variables, this linear program may be solved in polynomial time (in n, m and the number of bits of precision t) using only demand queries with item prices.

It should be note that in general, an optimal solution to the linear program can allocate fractional items to agents and may not be a feasible solution to the Linear program (where bids are indivisible). But it could help approximating the optimum, and might even find the integral (feasible) solutions for some restricted classes of valuations.

4 Ascending Auctions

In an *ascending auction*, the prices in the queries to the same bidder can only increase in time. One the main families of iterative CAs is ascending CAs.

Nisan et al.[2] characterized the power of four families of ascending auctions with demand queries: (1) Anonymous item-price, (2) Non-anonymous item-price, (3) Anonymous bundle-price, and (4) Non-anonymous bundle-price ascending auctions.

They first proved that none of item-price ascending auctions (anonymous or non-anonymous) are powerful enough to determine the optimal allocation for all instances of different classes of valuations. They also proved that anonymous bundle-price ascending auctions achieve poor results in the worst-case (i.e. there exists an instance of agents' valuations such that this family of auctions can not determine the optimal allocation for it). They showed that the power of non-anonymous item-price and anonymous bundle-price ascending auctions is incomparable. Based on their analyses, they concluded that the only family from these four which has the potential to find the optimal allocation is non-anonymous bundle-price ascending auctions. In fact, all the ascending auctions in the literature that are proved to find the optimal allocation for unrestricted valuations are in this family (e.g. iBundle by parkes et al.[5] and "Proxy Auction" by Ausubel et al.[6]).

4.1 iBundle: an Ascending Bundle-Price Auction

iBundle is the first iterative CA that is optimal for myopic utility-maximizing agents that place best-response bids to prices (myopic in the sense that they only consider the current round of the auction and play a best response to the current ask prices and allocation in the auction).

iBundle uses XOR bidding language. An auctioneer set initial ask prices on all bundles to zero. Then in each round, agents are asked to associate a bid price $p_i^t(S)$ (agent i, round t) on bundle S which must either be within ϵ, or greater than, the ask price announced by the auctioneer. Agents may repeat bids for bundles in the current allocation, but can bid at the same price if the ask price has increased since the previous round. An agent can also bid ϵ below the ask price for any bundle in any round but then it cannot place a higher bid price for that bundle in the future.

The auctioneer solves a WDP in each round to compute an allocation of bundles to agents that maximizes revenue, given prices (with respect to agents' XOR bid constraints). The price on a bundle is increased when one or more agents that do not receive a bundle in the current allocation bid at (or above) the current ask price for a bundle. The price is increased to ϵ above the greatest failed bid price and the auctioneer announces a new ask price in the new round for all bundles that increase in price. Other bundles are implicity priced at least as high as the greatest price of any bundle they contain.

Auctioneer introduces price discrimination based on agents' bids, with different ask prices to different agents, when an agent submits bids that are not safe. An agent's bids are **safe** if the agent is allocated a bundle in the current allocation or it doesn't bid at or above the ask price for any pair of compatible bundles S_1, S_2, such that $S_1 \bigcap S_2 = \emptyset$. Auctioneer starts the price discrimination for agent i, whenever bids from agent i are not safe, by introducing a new dummy item that is specific to that agent, X_i and concatenate it to all bids from agent i from this round on.

The auction terminates when: all agents submit the same bids in two consecutive rounds, or all agents that bid receive a bundle.

Parkes et al.[5] proved that iBundle terminates with an allocation that is within $3min\{|G|, |I|\}\epsilon$ of the optimal solution, for myopic best-response agent bidding strategies.

The auction is optimal as the bid increment, ϵ approaches zero. They proved the above claim (optimality) with a novel connection to primal-dual optimization theory. They showed that iBundle implements a primal-dual algorithm for WDP and computes integral solutions to the primal when agents follow myopic best-response bidding strategies.

From the computational complexity point of view, in iBundle the auctioneer must solve a WD problem (which is NP-hard) in each round to maintain a provisional allocation as agents bid. However, each WD problem in iBundle is smaller than in the GVA, because agents bid for less bundles. A naive worst case analysis gives $O(BV_{max}/\epsilon)$ rounds to terminate, where B is the total of bundles with positive value over all agents and V_{max} the maximum value for any bundle. One might note that B could be huge or even of an exponential order of $|G|$ which adds another computation complexity to the problem.

The auctioneer can make a tradeoff between the efficiency and computational cost, and reduce the number or rounds to termination by increasing the minimal bid increment, ϵ. Further speed-ups are achieved through some tricks that are explained in [5].

5 Other Design Approaches

In above we mostly focused on price-based approaches where an auctioneer provides ask prices to coordinate the bidding process. There are also non price-based approaches to iterative CA design which we briefly describe: (i) **Decentralized approaches**: WDP is moved to the bidders, who are responsible for submitting bids and also computing efficient allocations of items given prices, (ii) **Proxy auctions**: proxy agents automatically submit bids through a predetermined bidding procedure [6], and (iii) **Direct-elicitation approaches**: explicit queries are formulated by auctioneer and a bidder's strategy determines how to respond to these queries.

6 Summary and Future Works

Iterative combinatorial auctions address the problem of preference elicitation in CAs which might be one of the biggest issue in applying CAs in real world. Iterative CAs aims to adaptively elicit enough information about the bidders's preferences, often through price discovery, to find efficient allocations without bidders reporting or even computing their exact valuations.

However, the design space for iterative CAs is larger than for one-shot auctions and different features should be considered in the design process. This work was an introduction to iterative CAs which tried to describe different issues in iterative CAs design. It mainly focused on ascending auctions as one the main families of iterative CAs, and described iBundle [5] as an example of this family.

Many works have been done till now on designing iterative CAs, but there is still a number of outstanding design problems which can serve as future works:

Most of the current iterative CAs ignored the potential complexity of representing bundle prices. They assume that a query can be represented somehow (e.g. by XOR language) but side-stepped the fact that the representation of a query might have exponential length. They usually do not take into count the cost of representing queries when defining the cost of an iterative CA.

Current methods are mainly seek to find an efficient allocation with as little information as possible, but what happens if this minimal information remains too costly to provide? There is surely need for designing iterative CAs which make the right tradeoff between the cost of information and efficient allocation.

Also as you can see in iBundle, current auctions for general valuations mostly use XOR bidding languages which are not concise enough to be usable for many real-world applications. So we need iterative CAs that support richer bidding languages.

References

[1] Blumrosen, L., Nisan, N.: On the computational power of Iterative Auctions I: Demand Queries. *Part I of an extended abstract in EC 2005, http://www.cs.huji.ac.il/ noam/iter.pdf*

[2] Blumrosen, L., Nisan, N.: On the computational power of IA II: Ascending Auctions. *Part II of an extended abstract in EC 2005, http://www.cs.huji.ac.il/ noam/asc.pdf*

[3] Parkes, D.C.: Combinatorial Auctions: Chapter 2 - Iterative Combinatorial Auctions. *MIT Press* (2006)

[4] P. Cramton, Y. Shoham, and R. Steinberg (Editors). Combinatorial Auctions. *MIT Press* (2006)

[5] Parkes, D.C., Ungar, L.H.: Iterative combinatorial auctions: theory and practice. *In Proc. 17th National Conference on Artificial Intelligence (AAAI-00)* (2000) 74–81

[6] Ausubel, L.M., Milgrom, P.R.: Ascending auctions with package bidding. *Frontiers of Theoretical Economics* **1:1** (2002)