Combinatorial Auctions: A Survey *

Nikolaj Cankar, 46700V

7th March 2002

*Original paper from Sven de Vries & Rakesh Vohra

1 Introduction

The paper in question is a survey on different kinds of combinatorial auctions. Clearly these are terms that have to be defined in some way. First, an auction is a situation, real or fictional where a seller and possibly many buyers meet to make sales on a variety of distinct assets. Buyers, also referred to as bidders in an auction, have some preferences on items and on sets of items because of complementarities in these assets. This is a reason why we are talking about combinatorial auctions since both seller and bidders are interested in selling or buying bundles of different objects. The reason why the seller is interested in selling bundles of objects is that buyers might be interested in buying bundles of complementary objects and not just individual objects. It might for example be more interesting for a cook, of buying all the necessary ingredients for a soup at a time than to go around different shops buying individual items, even if this latter might become slightly cheaper in terms of money. In combinatorial auction the idea is then for the seller to find out what is the best combination of objects so as to get the best profit out of them.

While presenting these combinatorial auctions a special point is made on integer programming. Here the term programming should be understood in terms of planning more than in terms of computer programming. Integer programming is a type of linear programming with the constraint that all variables have to have integer values (e.g. 0-1). An example of a simple linear programming problem could be; Suppose we wish to invest \$14 000 into four investment opportunities. The first requires an investment of \$5 000 and rends \$8 000, the second needs \$7 000 now and rends \$11 000, the third needs now \$4 000 and rends \$6 000 while the last needs now \$3 000 and rends \$4 000. How should the money be invested to maximize the present value? This can be formulated as follows:

Maximize

$$8x_1 + 11x_2 + 6x_3 + 4x_4$$

Subject to

$$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$$
$$x_i \in \{0, 1\}, \quad j = 1, \dots 4.$$

The optimal linear programming solution to the above problem would be

$$x_1 = x_2 = 1, x_3 = 0.5 \text{ and } x_4 = 0.$$

However, in practice it might be quite difficult to make only half of an investment like here for the third variable. To make this problem an integer problem, we would add a restriction saying all answers have to be integer valued, thus 0 or 1 in this example. This would mean we either invest or do not in a specific investment opportunity. Other restrictions could also be used, if there was other constraints in how these investments could be made.

The combinatorial auction problem will be explained through an other problem called the set packing problem. Once the set packing problem is presented in the next chapter, several methods of solving it will be presented. Since this is a difficult problem to solve, methods telling us when the problem can be solved will be showed, followed by exact and approximate methods. Finally a scheme alleviating the burden of the auctioneer is presented, by activating the bidders.

2 The Set Packing Problem

In a general auction setup, where bidders (buyers) are allowed to bid on any combination of objects at sale, the auctioneer (seller) has to get bids for all of the combinations of goods, from all bidders. This could generate a great amount of traffic between the seller and the buyers. There are different ways to diminish the amount of necessary traffic. To do this, the seller might want to restrict the combinations bidders can bid on. Or still another way the amount of transfered information can be limited, is by using 'oracles' as bids. These oracles are small programs set up by the bidders, that the seller runs on the set of objects on auction. The oracle then tells what set and at what price is the buyer interested in buying. This could have other advantages too. The auction process could happen at any time once all bids (oracles) are collected.

The real mathematical problem follows once the seller has all the bids. How to choose the best one i.e. the one that is the most profitable? This is what is called the Combinatorial Auction Problem (CAP) and can be formulated as a linear problem. Taking N to be the set of bidders, M the set of m distinct objects at sale, then for every subset S of M, $b^j(S)$ is the bid that agent $j \in N$ is prepared to pay for the subset S. To make this problem integer lets take $x_S = 0, 1$, where x=0 means no bid is taken on the set S.

$$\max \sum_{S \subset M} b(S) x_S$$
$$\text{s.t.} \quad \sum_{S \ni i} x_S \leq 1 \quad \forall i \in M$$
$$x_S = 0, 1 \quad \forall S \subset M$$

Here $b(S) = max_{j \in N} b^j(S)$ is the best bid from buyer j. The constraint on the sum of x_S, makes sure no set is sold more than once. This formulation supposes that the goods are complementary, meaning

$$b^j(A) + b^j(B) \leq b^j(A \cup B) \quad \forall j \in N,$$
$$\text{and } A, B \subset M, \text{ such that } A \cap B = \emptyset.$$

This can be understood so that the buyer values more the sets A and B together, than he/she does the sets separately. This linear programming problem can be made more general by extending the formulation presented to accept many instances of a good. Now the right hand side of the constraint on the sum of x_S in CAP, can have values greater than one. The CAP problem presented is part of a more general class of problems known as the set packing problem (SPP) presented here below.

The CAP problem presented tried to find the best combination of subsets with the best bids. In the set packing problem one supposes there is a collection V of subsets of goods from the set M. Each subset can be given a non-negative weight c_j to announce its monetary value to a certain bidder j. Now the SPP is to find the largest weight collection of subsets that are pairwise disjoint. Let $x_j = 1$ be the j:th set in V with weight c_j and $x_j = 0$ otherwise. Lets still define a_{ij} to be 1 if the j:th set in V contains element $i \in M$. Now the SPP can be formulated as follows:

$$\max \sum_{j \in V} c_j x_j$$
$$\text{such that} \quad \sum_{j \in V} a_{ij} x_j \leq 1 \quad \forall i \in M$$
$$x_j = 1, 0 \quad \forall j \in V$$

It is fairly easy to see that the CAP is an instance of SPP. Here the problem is to pack all the elements in the set M into subsets that will maximize e.g. the revenue of the vendor. Each set element of M must appear in at most one subset. This class of problems has some other problems of similar type, where the interest is not on maximizing the revenue but on assigning objects to a set. This kind of problems are the set covering problem (SCP) and the set partitioning problem (SPA) where one tries to partition the set on sale to get such subsets that one is interested in. Both of these two (SPC and SPA) are of interest when we take the buyers point of view. Then we will wish to make such sets that are most advantageous to us.

The formulation of SPA:

$$\begin{aligned} min \sum_{j \in V} c_j x_j \\ \text{such that } \sum_{j \in V} a_{ij} x_j = 1 \quad \forall i \in M \\ x_j = 1, 0 \quad \forall j \in V, \end{aligned}$$

and of SCP:

$$\begin{aligned} min \sum_{j \in V} c_j x_j \\ \text{such that } \sum_{j \in V} a_{ij} x_j \geq 1 \quad \forall i \in M \\ x_j = 1, 0 \quad \forall j \in V \end{aligned}$$

Note how the only things that change, are minimizing the sum of weighted objects and the constraint allowing the number of separate variables into each subset. Minimization is done, because as said, these problem types are often interesting to the customer. For example the seller is making sure there is at least one object in each subset (SCP), which is an expensive constraint. In the SPA, the seller makes sure each good is in exactly one subset, which can also become expensive for the auctioneer.

As we saw in the beginning of this chapter, the CAP and more generally the SPP become very complex problems as the solutions consist of choosing the right subset from all the 0-1 combinations of a set V. For the auctioneer this means he/she will have to check $2^{|V|}$ solutions to make his/hers decision on the best bid, when $|V|$ is the number of subsets of set M. In any real-life situation this is too big a number of possibilities to go through, since the number of variables (i.e.

goods) in M easily grow large. There are strong reasons to believe there is no polynomial time algorithm for solving the SPP and that it belongs to the class of NP-hard problems. This does not mean there is no solvable instances of the SPP. It tells us, that the only way to make the CAP any easier to solve is by somehow restricting the problem.

3 Solving the SPP

We saw the SPP is generally a difficult problem to solve. To know which of the problems can be solved there exists exact and approximate methods. Often approximate methods are faster and give a good enough answer, where as exact methods are known to give, an exact and optimal solution. Let us start by defining when the SPP is solvable in polynomial time and then have a look at exact methods followed by a few approximate methods for solving the SPP.

The SPP can be solved in polynomial time, when the extreme points of the polyhedron defined by $P(A) = \{x : \sum_{j \in V} a_{ij} x_j \leq 1 \forall i \in M; x_j \geq 0 \forall j \in V\}$ are all integral i.e. 0-1. The polyhedron is then said to be integral. In these cases one does not need to worry about the integer requirement of the problem and solve it as a linear program. This does not yet tell when we have a solvable instance of SPP. Sufficient conditions for the polyhedron defined to be integral, involve restricting the constraint matrix A i.e. restrict the possible subsets for which bids can be submitted. There are other restriction possibilities too. We will not go into detail in them, but just list the most important ones and give pointers to papers or articles that talk about them in more detail (see Rothkopf *et. al.* (1998)).

- Total Unimodularity

  This is the most well known sufficient condition for the polyhedron to be integral. For a matrix to be total unimodular (TU), the determinants of every square sub-matrix has to be -1, 0 or 1. The matrix being considered here is $A = \{a_{ij}\}_{i \in M, j \in V}$. The most important class of TU matrices are called network matrices.

- Balanced Matrices

  For a 0-1 matrix B to be balanced it should have no square sub-matrix of odd order with exactly two 1's in each row and column. This does not guarantee the integrality of the polyhedron formulated above, but it does say that if B is balanced, then the polyhedron $P(B) = \{x : \sum_j b_{ij}x_j = 1 \forall i, x_j \geq 0 \quad \forall j\}$ is balanced. It should be noted that this polyhedron could be used to solve the set partitioning problem.

- Perfect Matrices

  If the constraint matrix A can be identified with the vertex clique adjacency matrix of what is known as a perfect matrix, then the SPP can be solved in polynomial time. For more about perfect matrices, see Chapter 9 of Grtschel *et. al.* (1988).

- Graph Theoretic Methods

  Using graph theoretic methods the SPP can be solved in polynomial time even if P(A) is not integral. The constraint matrix A can have an easy interpretation using graph theoretic methods. For example when both columns of A contains at most two 1's, this constraint can be thought of as a weight matching problem that is known to be a polynomial time problem. Each row (object) is represented by a vertex and each column (bid) represents an edge. There are downsides here too. If A has $k \geq 3$ non-zero entries in each column then the SP problem becomes NP-hard.

- Using Preferences

  All of the instances seen so far restrict the sets of objects over which preferences can be expressed. By using preferences the idea is to restrict the orderings of the bidders into two separate groups. Here too one cannot use more than two groups, without the problem become NP-hard again. For more information about how this works look in Bikhchandani and Mamer (1997).

Having seen when the SPP is solvable let us now have a look at different solving techniques.

3.1 Exact Methods

Solving the SPP by exact methods means the problem has to be relaxed in some way first. This is, the original problem is relaxed to an other problem that is known to have an answer. This new problem's solution should contain all feasible solutions of the original problem. There are two standard ways of how this can be done for the SPP: Lagrangean and linear programming relaxation. Using these there exists three different varieties of exact methods: branch and bound, cutting plane and branch and cut.

Branch and bound method can be seen as an intelligent enumeration of the solutions of the linear program. After having solved the linear program, at each stage of the branch and bound, a fractional variable is selected and given integer values, 0-1. Now both of these new subproblems are solved and the same is done again until all variables are integral. At each stage we get an upper bound for the final solution. This ends up in a binary tree including all possible solutions in the worst case.

Cutting plane methods try to find linear inequalities that are violated by a solution of a given relaxation, but are satisfied by all feasible 0-1 solutions. This is done by adding constraints to a linear program until the optimal feasible solution takes on integer values. One has to be careful when adding constraints, not to the change the problem by adding the constraints.

Branch and cut is a combination of the previous two. This method works like branch and bound, but tightens the bounds in every node of the tree by adding cuts.

3.2 Approximate and Other Methods

Since finding an answer to the integer programs using exact methods seems to be laborious, maybe one should give up on finding the optimal solution. Rather one settles for a feasible solution fast, an approximation, and hopes it be as close to optimal as possible. There exists roughly three ways to finding out the precision of approximate methods for solving the SPP. These methods are worst case analysis, probabilistic analysis and empirical method. We will not go through these methods here, but for more information on approximate methods one should try to look into Crescenzi *et. al.* (1998).

Generally approximate methods are used because they are easy to implement and they give fast and fairly good results. One has to remember that they give only approximate information though. For example the worst case analysis does not tell anything about finding solutions to a typical SPP. When using approximate methods, one cannot tell do they really model the problem at hand truthfully, realistically and as in a real environment.

A simple and straightforward way of reducing the computational burden in solving the SPP is using a Walrasian auction system. Here the auctioneer sets prices for objects and bidders announce what sets they are willing to buy and pay at the set prices. After this the Walrasian auctioneer adjusts the price vector. In such a system the bidders do not have to bid on all possible subsets of objects and the auctioneer does not need to process as many bids as in a normal system. This obviously reduces the computational burden of the auctioneer and is decentralized to all participants of the auction. More generally the computational task can be decentralized if information about the price is conveyed to the bidders. They can now participate in the calculation of the seller's revenue.

There are other ways of reducing the computational burden of SPP. Here three different methods are presented, Lagrangean relaxation has already been talked earlier while the other two are column generation and cuts using extended formulations and non linear prices.

Using relaxations one allows impossible solutions to the original problem, but these are penalized by the amount of their 'infeasibility'. Basically one relaxes the constraints of the original problem, such as the remaining set of constraints of the problem is easy to optimize. The Lagrangean relaxation method is not guaranteed to find the optimal solution to the original SPP, but it does guarantee finding an optimal solution to the relaxed problem.

Linear programs that have an exceeding large number of columns (variables in the auction setup) can use a the column generation technique. It works by first finding an optimal solution to the linear problem with only a subset of columns (variables) used. This solution is presented to the bidders that choose the columns which they value most and send this information back to the seller with their valuation of it. Now the auctioneer will solve a new linear problem using what he/she just got from the bidders. This goes on until an extreme point solution is identified by the seller. This system avoids long lists of subsets with bids. This could still be made smarter if the bidders are allowed to generate new alternate columns (subsets of variables) provided that this increases the revenue of the seller. This obviously decentralizes the computational tasks in the problem.

4 Activating the bidder

So far we have been only interested in solving CAP such as to choosing an allocation of objects to maximize the seller's revenue. This setup does not in any way take into account how the bidders actually value the various subsets that they are bidding on. This is easily seen by taking an example where three bidders {1,2,3} bid on two objects {x,y}. Bidders value these objects as follows:

$$v^1(x,y) = 100, \quad v^1(x) = v^1(y) = 0,$$
$$v^2(x,y) = 0, \quad v^2(x) = v^2(y) = 75 \text{ and}$$
$$v^3(x,y) = 0, \quad v^3(x) = v^3(y) = 40.$$

The auctioneer should for example award x to user 2 and y to bidder 3. If the bidding goes on and the buyers know what is going on, bidder two would be tempted to diminish his bid to say 65, winning the bid and having to pay less. At the same time, user three might want to valuate his bid for x downwards. Both bidders 2 and 3 cannot on the other hand keep bidding downwards because otherwise they would lose to user 1, who would value better the two subject together than user 2 and three together. One can find more about this combinatorial threshold problem, in Bykowsky *et. al.* (1995). More generally it might be difficult for several smaller bidders to coordinate their bidding such that they do not get over bid by a bigger buyer on particular subset of objects. For the auctioneer the ideal market design would be:

- Induce bidders to reveal their true valuations on objects and subsets.
- No bidder is put worst off by participating to the auction.
- Under the previous conditions the seller can maximize its revenue.

Finding and putting up such an auction turns out to be difficult. By using efficiency as a restriction to finding such an auction, then the optimal auction is known. 'Efficient' here means the auction has less rounds and is of shorter duration. It can be formulated as a linear program such that, let $y(S,j) = 1$ if the bundle of objects $S \subseteq M$ is allocated to user $j \in N$ and zero otherwise. Also let $v^j(S)$ be the valuation of the subset S by user j. Now the linear program is:

$$
\begin{aligned}
max \sum_{j\in N}\sum_{S\subseteq M} v^j(S)y(S,j) \\
\text{s. t. } \sum_{S\ni i}\sum_{j\in N} y(S,j) \le 1 \quad \forall i \in M \\
\sum_{S\subseteq M} y(S,j) \le 1 \quad \forall j \in N \\
y(S,j) = 0,1 \quad \forall S \subseteq M, \quad j \in N.
\end{aligned}
$$

This auction system does not guarantee the auction to be optimal for a single object. A known, efficient and optimal auction system is the VCG (Vickrey-Clarke-Groves) scheme. Sadly it becomes very difficult and impractical with real size problems, where the number of bidders gets large. An easy, but not very practical solution is restricting the number of allowed valuations of the bidders drastically. Obviously this isn't a real solution to the problem. For a greater discussion and proof one will want to read Krishna and Perry (1997) or Williams (1999).

Before ending this survey on combinatorial auctions, it should be told on what kind of problems have these methods been tested on. The test cases can be grouped into four different classes of distributions. In a *random* setup the bidder picks a random number of objects at auction, 1 ... m and picks the price [0,1] randomly. A *weighted random* set up is similar to the random method, but here the price is randomly picked between [0 ... m], where m is again the number of objects. In *uniform* method the same number of objects is chosen into each bid and the prices are picked from [0,1]. Lastly in the *decay* method each bid is given a random item and repeatedly given a new random item (at a given probability) until there are no more items left or it is chosen not to add them anymore. The price is picked as a real number between 0 and the number of item in the bid.

Here many different methods and problems for solving problems associated with auctions have been shown. None of them is perfect nor fits into all of the situations. Putting up an auction still seems to be a difficult task and needs a very experienced personnel to understand and analyze what is going on. Maybe there will be some new drastic improvements in the near future when 'virtual' auctions become an everyday task for all of us. Until then we will just have to do with what we have.