A constraint logic programming approach to the Winner Determination Problem

Tomasz Tatoń[1)]
[1)] *Silesian University of Technology, Faculty of Automatic Control, Electronics and Computer Science, 44-100 Gliwice, Poland, e-mail: tomasz.taton@polsl.pl*

The paper presents the winner determination problem (WDP) as a constraint programming optimization problem. The WDP is a problem usually discussed and solved in the framework of combinatorial auctions. A combinatorial auction consists of finding, from a given finite set of combinatorial bids B, a feasible subset B' of B with a maximum revenue. The paper presents a constraint logic programming. approach to this problem. A model has been defined and solved using the constraint logic programming language CHIP.

Keywords: combinatorial optimization, combinatorial auction, winner determination problem, constraint logic programming, CHIP.

1. INTRODUCTION

According to Encyclopædia Britannica, auction is defined as the buying and selling of real and personal property through open public bidding. The traditional auction process involves a succession of increasing bids or offers by potential purchasers until the highest (and final) bid is accepted by the auctioneer (who is usually an agent of the seller). *Combinatorial auctions* (CA) are auctions in which bidders place bids on combinations of items, called ‘‘packages,’’ rather than just individual items.
The field of combinatorial auction has grown rapidly in the past ten years. Auction theory is one of the basic topics in economics, because it asks and answers the most fundamental questions in market economy: who should get what and at what price?
The rational behind combinatorial auction is due to the fact that the value of an item to a bidder may depend significantly on what other items the bidder does or does not acquire. This dependence is of two main types: complements and substitutes:

- items A and B are complements if the value of A to a bidder is greater if the bidder also acquires B;
- items A and B are substitutes if the bidder is interested in either A or B.

The importance of combinatorial auction has resulted in a number of monographic publications, for the latest see e.g. Toczyłowski (2003) and Cramton-Shoham-Steinberg (2006). Perhaps the oldest kind of combinatorial auctions has been used in real estate trading, where individual items have been auctioned first and next, at the end, bids for packages of items have to be called for. If a package bid exceeded the sum of individual bids for the package items, then the items were sold as a package. The advantage of

such combinatorial bidding extension is that bidders can more fully express his preferences and the auctioneer can make a more profitable sale.
Recently, a number of new applications has emerged, e.g. for electrical energy trading (Toczyłowski 2003), for load transportation problems, for bus routes, for allocating radio spectrum for wireless communications services, see Cramton *et al*. (2006). Package bidding is necessarily a combinatorial decision problem, most often formulated as combinatorial optimization problem and solved using traditional techniques of operation research, especially techniques from combinatorial optimization and mathematical programming. The Author claims that Constraint Logic Programming is a particularly suitable tool for modeling and solving combinatorial auction problems. To the best of his knowledge, the present paper seems to be the first advocating this approach.
Any item or service offered by some agent may be the target of combinatorial auction. The participants can bid for particular items or optionally for sets of items. After reporting the purchase bid, particular item or set of items, which were the substance of reporting, have to be accepted as a whole or integrally rejected after the auction is finished.
The bids for such sets may differ from the sum of individual bids for their elements. The highest profit from the sale is the goal of the auction. The important feature is that not every item or set of items has to be sold after the auction finishes.

2. PROBLEM FORMULATION

The winner determination problem (WDP) is the problem of finding a feasible subset B' of B with a maximum revenue, given a finite set of combinatorial bids B.

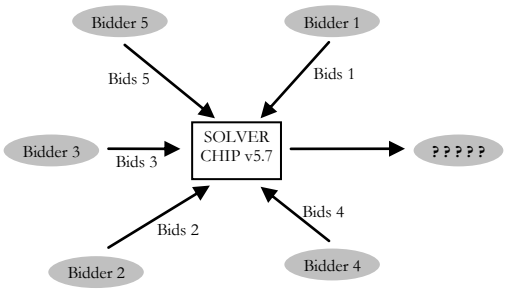

Fig. 1. The Winner Determination Problem

Of course each bidder can submit one or more bids at the auction. The bids may be complementary or substitutive.

From the WDP perspective combinatorial auctions are of different types.
1. *Single - unit combinatorial auctions* (each item is unique), defined as follows.
The auctioneer has a set of unique items, $M = \{1,2,\ldots,m\}$ to sell, and the buyers submit a set of bids, $B = \{B_1, B_2, ..., B_n\}$. A bid is a tuple $B_j = \{S_j, p_j\}$, where $S_j \subseteq M$ is a set of items and $p_j \geq 0$ is the price for the set. The WDP for the single-unit combinatorial auction aims at labelling the bids as winning or losing so as to maximize the auctioneer's revenue under the constraint that each item can be allocated to at most one bidder:

$$\max \sum_{j=1}^{n} p_j x_j \tag{1}$$

s.t.

$$\sum_{j=1}^{n} x_j \leq 1, \quad i = 1,2,\ldots,m \tag{2}$$

$$x_j \in \{0,1\}$$

2. *Multi – unit combinatorial auctions* (each item is available in many units):
The auctioneer has a set of items, $M = \{1,2,\ldots,m\}$ to sell. The auctioneer has some number of units of each item available: $U = \{u_1, u_2, ..., u_m\}, u_i \in \Re^+$. The buyers submit a set of bids, $B = \{b_1, b_2, ..., b_n\}$. A bid is a tuple $b_j = \langle a_{1j}, a_{2j}, \ldots, a_{mj}, p_j \rangle$, where $a_{jk} \geq 0$ is the number of units of item k that the bid requests, and $p_j \geq 0$ is the price. Obviously, for a single-unit combinatorial auction the number of units for any item is always 1.
The WDP for the multi-unit combinatorial auction aims at labelling the bids as winning or losing so as to maximize the auctioneer's revenue under the constraint that each unit of the items can be allocated to at most one bidder:

$$\max \sum_{j=1}^{n} p_j x_j \tag{3}$$

s.t.

$$\sum_{j=1}^{n} a_{ij} x_j \leq u_i \,, i = 1,2,\ldots,m \tag{4}$$

$$x_j \in \{0,1\}$$

3. COMPUTATIONAL COMPLEXITY

It is well known that the WDP become harder to solve as the number of items, units and bids increases. The size of the WDP is defined by the number of variables that have to be instantiated to solve the problem. Usually each combinatorial auction has 2 variables types, bids and items. The number of bids and items may be several hundreds or more.
Almost every paper (see e.g. Rothkopf *et al*. (1998), Lehmann *et al*. (2006)) on combinatorial auction mentions that the WDP is NP-hard. That means that the computation time increases exponentially on the size of the problem (n) as $n \to \infty$. This can be simply derived from the fact that the NP-hard weighted set packing problem is equivalent to the WDP see Rothkopf (1998) . Moreover Lehmann, Müller and Sandholm (2006), have proven NP-hardness by reducing the knapsack problem to this problem.

4. CONSTRAINTS

Constraint Logic Programming (CLP) is a programming paradigm in which a properly defined model of problem is at the same time a program for the solution of the problem. In accordance with the CLP methodology, the problem must be modelled using predicates and constraints provided by the CLP language used and by the program designer. The WDP in combinatorial auction can be modelled by the constraint logic programming language CHIP v5.7.
Constraints are the most important things in CLP. The ability to express complex constraints and precise description of models is crucial. CHIP does it in an efficient and declarative way. CLP is based on the interplay of search and constraint propagation. This interplay includes e.g. heuristic methods like Forward Checking (FC) and Looking Ahead (LA), see Barták (1999).

The Winner Determination Problem has to fulfill the following constraints:

- every unit of an item can be allocated to at most one bid;
- the sum of allocated units must not be larger then the total number of those units the auctioneer has for selling.

From those two constraints follows that bids are treated as indivisible whole i.e. they cannot be divided into primary elements, see Tatoń (2005).

5. MODELING THE WDP PROBLEM IN CHIP

Modeling and solving the winner determination problem in CHIP application is the main task because the proper description of problem in CLP, is the program of its solution. The model and program that solves the problem has the following typical structure:

- variable declaration;
- formulation of constraints;
- labeling procedure,

discussed below.

5.1. Variable declaration

Every bidder submits bids including subsets of items, number of units for every item and maximal prices willing to pay for bids. The first step of the procedure design is *variable declaration*. For CHIP programs the needed variable declaration is as follow:

- Predicate *price()* has a single list as its argument. The list contains bid values for every bidders:

  $price([p_1,p_2,\ldots,p_n]).$

- Predicate *data()* has a list of lists as its argument.. The consecutive lists contain bids for consecutive items:

  data([
  $[X_{1,1},X_{1,2},\ldots,X_{1,n}]$,
  $[X_{2,1},X_{2,2},\ldots,X_{2,n}]$,
  […,…,…,…],

$[X_{m,1},X_{m,2},\ldots,X_{m,n}]$
]).

I.e. $X_{2,1}$.denotes the bid for item 2 by bidder 1. The list of lists may be looked upon as an n×m matrix. The entries $X_{i,j}$ of an n×m matrix are indexed by a double index. The first index represents the row (items) of that entry and the second index represents the column (bids) of that entry.

- Predicate *number_of_items()* has a single list as its argument. The list contains numbers of consecutive item to be auctioned:

  number_of_items$[U_1,U_2,\ldots,U_m]$*).*

For the case of a single unit combinatorial auction any item is available for sale only in one unit, for example:

number_of_items ([1,1,1,1,1]).

For this case the matrix in predicate *data()* may contain only 1 or 0, where 1 if bidder bids item and 0 if bidder doesn't bid the item:

data([1,0,0,1,1],
[0,0,1,1,0],
[0,1,0,1,0],
[1,1,0,0,0],
[0,0,0,1,0]).

For the case of multi-unit combinatorial auction any item is available for sale in a number of copies e.g.:

number_of_items ([5,7,3,9,4]).

In this case the matrix in predicate data() may contain values from 0 to the maximal number of available units for consecutive items. If the number is different from 0, then the bidder would bid for the number of units of corresponding item. Else if the number is equal to 0, the bidder would not bid for this item:

data([5,7,0,9,4],
[0,4,3,1,2],
[2,1,0,3,0],
[3,0,1,0,4],
[2,1,3,4,2]).

If two or more bidders bid for the same combination of items, only the best offer is considered. The remaining are pruned as they would loose anyway.

5.2. Formulation of constraints - first approach

CLP makes possible the formulation of constraints in different ways. This paper presents two possibilities. One of them defines the constraints by simple logical statements in the program, having the form of relations among several variables (k-ary constraints). For a single unit combinatorial auction it may look as follow:

$D_2+D_5+D_7+D_8$ *#<=1,*

For a multi unit combinatorial auction it may look like this:

$$D_2*U_2+D_5*U_2+D_7*U_2+D_8*U_2 \ \#=MaxUnitsItems_2$$

This approach of modelling constraints in CLP results in a faster program than the second. However this approach needs a larger number of variables declaration making the program less readable and increasing the chances to violate CHIP inherent constraints on the number of variables. For example if the number of items and the complexity of bids increase, the number of model variables increases as well. In CHIP v5.7 the maximum number of variables is defined by the "Dictionary size" as equal to 262139. The dictionary uses a fixed size, which can not be changed by the user (this is a system limits of the number of variable). This limit allows only modelling auctions containing only about 100 items and 50 units for every item.
This is a serious disadvantage. Therefore attention will be concentrated in the sequel on the second approach

5.3. Formulation of constraints - second approach

The second approach is to build the constrains dynamically from the input data while solving the problem. In this case there is no need to define the constraints in the program, but instead we have to define special predicates who take care of the constraints. The constraints in CHIP are build by a predicate *build_constraints()*, defined as follow:

```
build_constraints([],Vars,[]).
build_constraints([],_,_) :-
      sprintf(Msg, "error: Lists with different_
      length",[]),
      writeln(Msg),
      fail.
build_constraints(_,_,[]) :-
      sprintf(Msg,"error: Lists with different_ length",[]),
      writeln(Msg),
      fail.
build_constraints([Li|List],Vars,[Ui|Units]) :-
      !,
      linear_exp_i(Li,Vars,Lin_exp),
      Lin_exp #<= Ui,
      build_constraints(List,Vars,Units).
```

For this case no definitions of constraints by logical statements in the program are needed, because the predicate *build_constraints()* builds these constraints dynamically.
Predicate *build_constraints()* is building constraints by another predicate *linear_exp_i()*,which builds linear expression of the following form:

$$([A_1,A_2,..,A_n],[X_1,X_2,..,X_n],Lin\_exp) \rightarrow Lin\_exp \ Lin\_exp = A_1*X_1+A_2*X_2+..+A_n*X_n,$$

where $A_1,\ldots,A_n$ are the number of prices for every bids, and $X_1,\ldots,X_n$ are binary $\{0,1\}$ decisions which are to be labelled. The predicate *linear_exp_i()* is defined as follow:

```
linear_exp_i([],[],0).
linear_exp_i([A|Li],[X|Vars],A*X+Lin_exp) :-
      !,
      linear_exp_i(Li,Vars,Lin_exp).
```

This second approach to modeling the constraints in CHIP is universal for single and multi unit CA. The difference between them is only in declaration of input data.

5.4. Labeling procedure

The CHIP language makes the branch and bound algorithm available as part of its compiler. It may be used in a very convenient, declarative set-up. Complex performance indices can be defined by the user and passed as a parameter to labeling. This provides maximum flexibility to the programmer while keeping a reasonable program size. There are many possibilities to do labeling. One of them is to use standard backtracking with branch-and-bound search strategies.
The general description of the branch-and-bound technique is well-known see e.g. Gonen (2000). Better approach to solve WDP is to use advanced backtracking with Forward Checking (FC) and Looking Ahead (LA) heuristics build in CHIP. FC removes values of not yet instantiated variables only on instantiated variables. FC checks only the constraints between the current variable and the future variables. LA arc-consistency is checked after each distribution step on all variables and inconsistent values are removed.
The labeling procedure for backtracking with FC and LA built-in CHIP can be realized by standard predicate *labeling(_,_,_,_).* This predicate provides a simplified and efficient way for assigning values to variables.
The *min_max()* predicates is a higher order predicate providing optimization methods in CHIP, searching among all solutions for the goal. This predicate is searching for a solution which minimizes the cost. The cost may also be bounded from below and above. By default, *min_max()* prints out solutions as they are found. This printing is most useful for the considered approach.
The advantages of CLP over traditional Operation Research techniques have been highlighted many times, see e.g. Hansen (2003).

6. CONSTRAINT LOGIC PROGRAMMING FOR THE WINNER DETERMINATION PROBLEM

The main procedure is as follows:

```
top :-
      number_of_items(Units),
      price(Price),
      data(Data),
      length(Decision,5),
      utime(_),
      Decision :: 0..1,
      X :: 0..UpperBound,
      Goal :: 0.. UpperBound,
      once(build_constraints(Data,Decision,Units)),
      once(linear_exp_i(Price,Decision,Lin_exp-)),
      once(Lin_exp #= Goal),
      X + Goal  #= UpperBound,
      min_max((labeling(Decision),_get_choice-(C)),X),
      write('Maximum:: '),writeln(Goal),
      writeln(decision-Decision).
```

`Upper Bound` is a name for the domain size. The domain of a variable can be at most 0..100000. This is a limit of the CHIP applications. If we want to receive optimal solutions, scaling of every variable `Price` may sometimes be needed.

A constraint logic programming approach to the winner determination problem in CHIP is quickly generating answers, no matter whether the allocation of goods for bidders is feasible or not.

7. A WDP EXAMPLE

An example of multi-unit combinatorial auction will be presented in detail. For six examples of single unit combinatorial auctions only the results of applying two numerical strategies of labelling are presented.
The first auction is for the sale of different number of 5 items:

Table 1. Number of items

Items	**A**	**B**	**C**	**D**	**E**
Number	5	7	3	9	4

5 bids were submitted from 5 bidders:

Table 2. Bids submit on auction

Items	**A**	**B**	**C**	**D**	**E**	**Valuations**
Bidder 1	5	7	0	9	4	100
Bidder 2	0	4	3	1	2	80
Bidder 3	2	1	0	3	0	120
Bidder 4	3	0	1	0	4	90
Bidder 5	2	1	3	4	2	95

The aim was to find an allocation of items to bidders that maximizes the auctioneer revenue. Variable declaration of this example were shown in description of modeling the WDP problem in CHIP.

The solution for this WDP is: the winner is bidder 3 with bids (A-2, B-1, D-3) and at price – 120 and bidder 5 with bids (A-2, B-1, C-3, D-4, E-2) and at price – 95.

For the next six example we use CATS (Combinatorial Auction Test Suite) see e.g.
Leyton Brown-Pearson-Shoham for generate input data. We generate random test auction with 25,50,75,100,125 and 150 items for 50 bids every auction. This experimental was running on a Pantium IV-3,06 with 1024 MB memory RAM with MS Windows XP professional operating system. We use two strategies of labeling. The first of them is presented with a continuous line in the chart and represents the *standard* labeling strategy of CHIP. The second one is the dashed line in the chart and represents *most_constrainted* strategy.

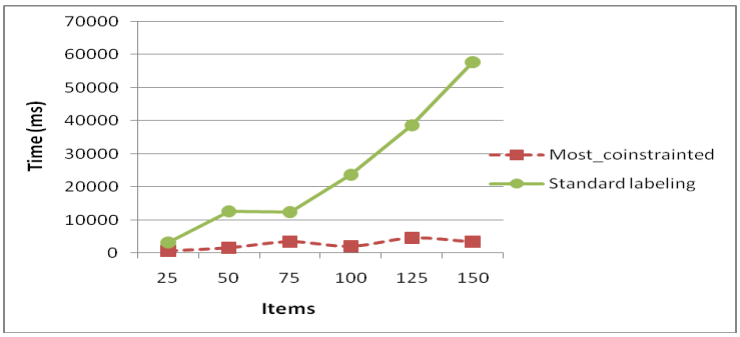

Fig. 1. Chart of solution time

We usually do not now which of those strategy is most effective for a given problem it has to be decided by "trial and error". In case of *most_constrained* labelling strategy the time needed to get the optimal solution is less then for the standard strategy (WDP problem).

8. CONCLUSION AND FUTURE WORK

It was demonstrated that Constraint Logic Programming as made available by CHIP is a suitable tool for formulating and solving a range of WDP for various combinatorial auctions. The CHIP program presented solves both single and multi-unit combinatorial auction WDP problem. The two methods of modeling of constraints for the problem were discussed and the most effective was determined. The origin of the solved problems was discussed. The first discussed example is simple but we must remember the winner determination problems are NP-hard problems. Practical size problems may be hard to solve because they cannot be solved in polynomial time. The program presented in the paper may also generate feasible solution for auctions for which the optimal solution is unobtainable because of large numbers of items and bids. This is possible thanks to the fact that the CLP program displays all feasible solutions while searching for the optimum one. Of course, the time needed to get an optimal solution depends on the number of constraints. More often then not, the more numerous and complex the constraints are, the shorter the time needed to solve the problem. The main benefit of the CLP approach is the simplicity with which additional constraints can be incorporated into the problem and the fact that there is no need to transform the original problem statement into some canonical form.

ACKNOWLEDGEMENTS

The author is grateful to prof. Antoni Niederliński for his support and contributed comments on a draft of the paper.

REFERENCES

[1] Bartàk, R.: Constraint programming: What is behind? In Proceedings of the CPDC'99 Workshop on Constraint Programming for Decision and Control, 1999, pages 7-15.
[2] Cramton, P., Shoham, Y., Steinberg, R.: Combinatorial Auctions, MIT Press, 2006.
[3] Gonen, R., Lehmann D.: Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. The Proceedings of the Second ACM Conference on Electronic Commerce (EC'00), pp. 13-20, October 2000. http://www.cs.huji.ac.il/~rgonen/
[4] Hansen, J.: Constraint Programming versus Mathematical Programming. Journal paper, Orbit 2003.
[5] Lehmann, D., Müller, R., Sandholm, T.: The Winner Determination Problem, Chapter 12 of the book Combinatorial Auctions, Cramton, Shoham, and Steinberg, eds., MIT Press.
[6] Leyton-Brown, K., Pearson, M., and Shoham, Y.: Towards a Universal Test Suite for Combinatorial Auction Algorithms, Proceedings of the 2nd ACM conference on Electronic commerce, pp. 66—76, ACM Press, 2000 year.
[7] Tatoń, T.: Zastosowanie programowania ograniczeniowego w logice dla problemu aukcji kombinatorycznych. (A constraint logic programming for a combinatorial action) Zeszyty Naukowe Politechniki Śląskiej z. 145, Wydawnictwo Politechniki Śląskiej, Gliwice 2006, strony: 217-224
[8] Toczyłowski, E.: Optymalizacja procesów rynkowych przy ograniczeniach. (Optimization of constraint market processes) II Edition, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2003.
[9] Rothkopf, M. H., Pekeć, A, Harstad, R. M.: Computationally Manageable Combinatorial Auctions, Management Science, 44, 1131-1147 (1998)

The author is a graduate of the Academy of Computer Science and Management in Bielsko-Biała (Computer Science Engineer) and the Academy of Business in Dąbrowa Górnicza Master of Management). He is a PhD student at Silesian University of Technology with special interest on combinatorial optimization problems and CPL methods.