An Implementation of the Combinatorial Auction Problem in ECL^{i}PS^e

Robert Menke and **Rina Dechter**

University of California, Irvine
Irvine, California 92717-3425
http://www.ics.uci.edu/~rmenke/
{rmenke, dechter}@ics.uci.edu

In a traditional auction, items are placed "up for bids" in an arbitrary sequence. For many bidders, this model is inadequate because the individual items increase in value when held in conjunction with other items. *Combinatorial auctions* allow bidders to bid upon multiple items simultaneously. While this resolves the problems for the bidders, it increases the problem of the auctioneer: determining the optimal selection of bids to maximize revenue is NP-complete.

(Sandholm 1999) suggests an algorithm that reduces the search space considerably. His algorithm, a DFS of the problem space using an ancillary data structure called a *Bidtree*, takes advantage of two properties of "real-life" auctions: that the bids submitted would be sparse and that the order in which bids are selected is irrelevant. The Bidtree helps select the next bid to be considered. It enforces two rules: the next bid has no items in common with the bids already selected, and that every item must eventually be considered.

The goal of this project is to evaluate the general principles and algorithms developed for constraint processing in recent years, as well as the tools and languages facilitating the use of constraints for problem solving using the auction problem as a benchmark (Dechter 1992). By comparing general constraint-processing algorithms against specific methods tailored for this task, the power of such general algorithms can be demonstrated. This project was initiated during a class project in the computer Science department, UC Irvine. Specifically, we used the constraint processing language ECL^{i}PS^e (ECL^{i}PS^e 1995) that has as its basic algorithm backtracking with forward-checking, using branch-and-bound for optimization tasks. Towards that end we had three specific subgoals: first, to implement the combinatorial auction problem in ECL^{i}PS^e; second, to implement Sandholm's solution using ECL^{i}PS^e; and third, investigate the possibility of improving the search using other heuristics.

It is important to realize that the Bidtree algorithm is optimized to do one thing: partition the set of items available into subsets that correspond to bids. It does not take into account the values the bidders have associated with each bid, nor does it consider (in this form of the algorithm) that mechanism for forward checking has been incorporated into the implementation language. Since ECL^{i}PS^e *does* support forward checking, the Bidtree algorithm simply reduces down to a static ordering of the variables.

The alternative bid selection rules used dynamic variable ordering to improve performance. The first approach was *most constrained bid* (MCB), which selected the bid whose set of items had the most non-empty intersections with all of the bids—the hope being that eliminating more feasible future bids would quickly reduce the possible revenue below the current bound, stopping further descent down that branch of the tree.

The second algorithm used the *most valuable bid* (MVB) rule. MVB selected the bid that had the largest amortized value (the value divided by the size of the set). It was hoped that the MVB selection rule would produce a higher revenue in its initial solution. This is desirable because the auctioneer may wish to stop the search before the algorithm completes. Additionally, a higher revenue discovered earlier would produce a better bound and result in faster convergence to the optimal solution.

The data sets in the full report were generated by the same methods as in the Sandholm paper, but with scaled parameters because of resource limitations. Two results using unscaled parameters are summarized in Table 1. While the MCB algorithm performed poorly, the MVB algorithm showed significant improvement over Bidtree, in time to completion and the number of refinements to the bound. (In the second example, Bidtree found the better solution before MVB but its searching of the entire space took three times as long.) The author's web page will have additional examples as they become available.

150 bids, 25 items, fixed at 3 items per bid			
Method	*CPU Time*	*Best Solution*	*Refinements*
MCB	20627.70 s	13517.25 s	29
Bidtree	18739.12 s	7847.97 s	13
MVB	2960.42 s	326.79 s	6
50 bids, 75 items, value scaled to size			
Method	*CPU Time*	*Best Solution*	*Refinements*
MCB	315.57 s	238.05 s	12
Bidtree	208.47 s	1.85 s	5
MVB	78.66 s	11.69 s	1

Table 1: Performance of the three algorithms on two separate test cases

References

Dechter, R. 1992. Constraint networks. In *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, Inc., second edition. 276–285.

1995. *ECL^{i}PS^e User Manual, v. 3.5*. Available at http://www.ecrc.de/eclipse/eclipse.html.

Sandholm, T. W. 1999. An algorithm for optimal winner determination in combinatorial auctions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 542–547.