Iterated algorithm for the optimal winner determination in combined negotiations

Samir Aknine[1]

ABSTRACT. Generally in multi-agent negotiations, agents are asked to submit various bids and counter-bids during several iterations of a negotiation process before reaching a compromise. In case of combined negotiations, agents need to solve particularly the winner determination problem WDP to find the best combination of bids in each iteration. In this paper, we tackle the problem of searching and updating the optimal solution of the WDP while moving from one iteration to another. We propose thereby an algorithm that searches and updates this optimal solution. Our algorithm is essentially based on exploring very simple graphs for shared and unshared items. These graphs speed up optimal solution searching.

1 INTRODUCTION

We are interested in searching the optimal solution, composed of one bid or a combination of bids, using a WDP algorithm in case of multi-items negotiations. Currently, several algorithms and even extensions of these algorithms [1, 2, 3] are able to compute the best bid or a combination of bids in order to maximize the utility function of an agent. In this paper we focus on the evolution of the optimal solution depending on the state of negotiation and submitted bids from one iteration to another. Our aim is to propose an algorithm to find this optimal solution in an incremental way. This work is motivated by the iterated negotiation processes found in current negotiation protocols: processes in several phases or iterations. An optimal solution is thus recomputed from one iteration to another and according to the utility reached by the agents with this solution the negotiation is stopped or resumed. The incremental searching requires an algorithm capable of updating the optimal solution with minimum computations in order to reduce its complexity. Such an algorithm will compute a new solution by taking into account only the changes that have appeared in prior iterations. The types of modification to be considered depend on the evolution of the bids. For instance, if a bid at stake becomes invalid –its sender having retracted from the negotiation-, it will imply an operation of deletion of this bid. If a bidder sends a new bid, it will imply an operation of completing the list of the bids. As WDP is a NP-Complete problem and its complexity becomes significant in combined negotiations, we claim for the necessity of introducing incremental algorithms. For n items and a negotiation protocol with k phases or levels, if an operation (deletion, new, update) is performed at each iteration, the algorithm would require more than $2^n k$ changes.

2 *WDP* ALGORITHM

The simplified algorithm that is presented in this paper applies to mono-unit reverse auctions in which a buyer agent intends to buy from suppliers who can make various bids on one or several items. The problem of the buyer agent is to find the combination of bids which makes its payoff optimal using a WDP algorithm.

[2, 3] present several algorithms for solving the WDP. We propose to extend these algorithms using our graph structures and heuristics. The simplified WDP algorithm described in this section is just an example of how we could extend the existing ones with our structures. These graphs enable us to work on an intentional representation of the bids rather than an extensional representation. These graphs as well as their properties can be applied to any current WDP algorithm. Following is an example that depicts the main steps of the basic algorithm.

Example: A buyer agent wishes to acquire 7 items {1,2,3,4,5,6,7}. In order to include all of them in his query, the agent formulates a query in a conjunctive form. It receives in return bids in the form: {1,3,5, $9} which means that the seller proposes items 1, 3 and 5 and they cost $9.

Let's suppose that it receives the following bids: O_1{1, $2}, O_2{2, $1}, O_3={4, $3}, O_4={5, $2}, O_5={6, $2}, O_6={1,2, $4}, O_7={1,5, $3}, O_8={2,5, $3}, O_9={2,4, $3}, O_{10}={3,7, $4}, O_{11}={4,5, $3}, O_{12}={1,4,5, $5}, O_{13}={2,3,4, $6}, O_{14}={2,4,5, $5}, O_{15}={2,4,5,7, $7}, O_{16}={1,3,4,5, $7}, O_{17}={1,2,4,7, $6}, O_{18}={1,2,3,4,5, $9}.

Before building the graphs, the workspace must be partitioned. The idea of this stage is to separate the various bids depending on the items they contain. This is the reason why the associated score of each item g is computed using the function f/ $f(g) = \sum_{i=1}^{n} \frac{1}{T_i}$, where n is the number of bids containing the item g and T_i is the length of the bid i containing g. In our example, we obtain the following partitions (Cf. Figure 1). The next step is the building of two graphs to speed up the search for solutions.

Definition 2 A shared item graph is a connected directed acydic graph whose first level nodes are partitions of bids, intermediate nodes contain items of these bids and whose terminal node is empty. In this graph a node n_i is connected to n_j if all the items in n_j are included in n_i.

In the shared items graph (cf. Figure 1), the second partition p_2 is connected to the node containing items 3 and 4. The node containing items 3 and 4 is connected to the one containing item 4, the latter being a subset of the former.

Definition 3 An unshared item graph is a connected directed acydic graph whose first level nodes are partitions of bids, intermediate nodes contain unshared items and terminal nodes are empty. In this graph a node n_i is connected to n_j if all the items in n_j are included in n_i.

In our example, the partition p_2 is connected to the node containing items 1, 2 and 5; as we can observe, these items belong to some bids of the partition, but not to all of them. Node (1, 2, 5)

[1] LIP6, Université Paris 6, Samir.Aknine@lip6.fr

is thus connected to node (1, 2, 3, 4, 5). The reason is that if a partition is connected to (1, 2, 3, 4, 5), none of its bids contain these items, particularly (1, 2, 5).

These graphs simplify the solution search using any method. Only n(n-1)/2 nodes are parsed instead of 2^n since the algorithm makes the search on the nodes of items rather than on the bids. They considerably improve the solution search step and are built is fast.

Only bids complying with the three following properties will be examined. It should be added that our optimal solution search algorithm builds a tree whose branches are solutions. When a bid is selected, none of the other bids of the partition it belongs to can be examined anymore. While selecting a bid, its container partition is marked using the reference of the branch in order to avoid skimming through subsequent branches.

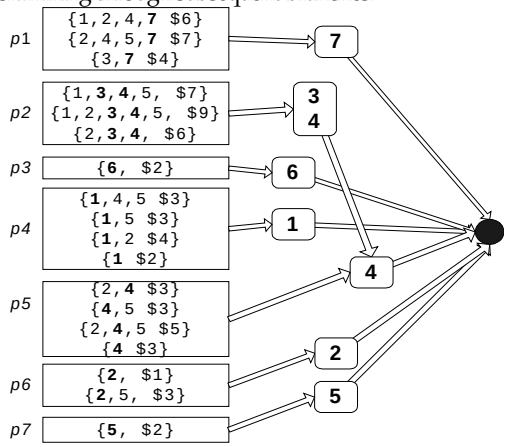

Figure 1: Shared-item graph

Property 1 If a partition p_i shares an intermediate node with another partition p_j, and if p_j is marked in the shared item graph, our algorithm will not explore p_i. Also, if one of the nodes preceding p_i, in the shared item graph contains an item in the branch under construction, p_i will not be explored anymore.

Let's suppose that a bid in the partition p_2 in our example (cf. Figure 1) has been selected. This partition will then be marked. As p_5 and p_2 have the same intermediate node 4, p_5 won't be explored. Remember that all the bids in p_2 propose items 3 and 4, and all the bids in p_5 propose item 4. Therefore, it is useless to explore p_5 as any bid selected in p_2 has already proposed item 4.

This property simplifies the search process. It keeps the algorithm from skimming through all the partitions that cannot improve the solution.

Property 2 For a partition p_j to be explored, it must be connected to a node n_i of the unshared item graph and at least one item of this node has not been added yet to the branch under construction.

Property 3 The union of the set of items in the branch under construction and that of all the partitions not already explored corresponds to the set of items mentioned in the query in the case of a conjunctive form.

This property is intended to avoid examining branches that lack some items. Let's assume a query composed of items <1, 2, 3, 4, 5, 6, 7, 8> and a branch under construction with items <1, 3, 4, 5>. If all the partitions not already explored contain only items 2 and 6, there would not be any possible solution for this branch.

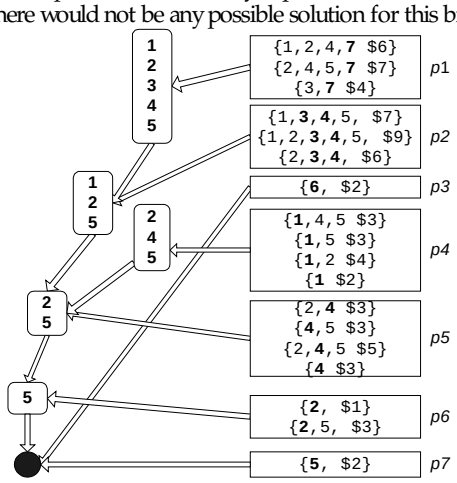

Figure 2: Unshared-item graph

Once the first step has been accomplished, i.e. graph construction, the optimal solution search is started applying the above properties. Each bid of the first partition will correspond to a sub tree having this bid as the root. The whole process leads to the following solution tree.

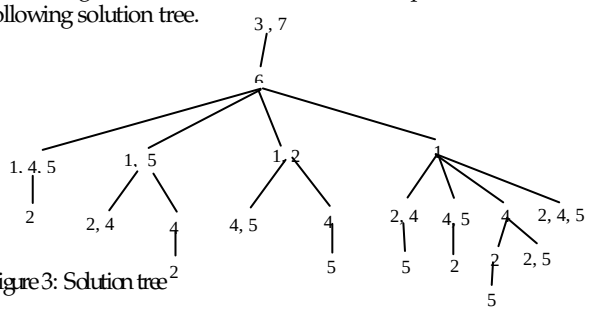

Figure 3: Solution tree

3. CONCLUSION

This paper has presented a simplified algorithm for the winner determination problem. This work addressed the problem of adapting winner determination algorithms to the constraints of the combined multi-agent negotiation. In these negotiations, the bids of the agents constantly evolve. In this context, we handle the evolution of agent's bids by computing new optimal solutions dynamically using the preceding graphs. The proposed algorithm has been implemented and tested with different distributions of bids to evaluate its performance.

REFERENCES

1. Fujishjima Y., Leyton-Brown K. and Shoham Y. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. IJCAI, pp. 548-553, 1999.

2. Sandholm T. Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence,135:1–54,2002

3. Sandholm T. and Suri S. BOB: Improved Winner Determination in Combinatorial Auctions and Generalizations. AI Journal, volume 145, pp. 33-58. 2003.