Bid evaluation for coordinated tasks: an Integer Programming formulation

John Collins and Maria Gini*
Department of Computer Science and Engineering
University of Minnesota
{jcollins,gini}@cs.umn.edu

Abstract

We extend the IP models proposed by Nisan and Andersson for winner determination in combinatorial auctions, to the problem of evaluating bids for coordinated task sets. This requires relaxing the free disposal assumption, and encoding temporal constraints in the model. We present a basic model, along with an improved model that dramatically reduces the number of rows by preprocessing the temporal constraints into compatibility constraints. Experimental results show how the models perform and scale.

1 Introduction.

Business-to-business e-commerce is expanding rapidly, letting companies both broaden their customer base and increase their pool of potential suppliers. Negotiating supplier contracts for the multiple components that often make up a single product is complicated because time dependencies introduce a significant scheduling risk. Current e-commerce systems typically rely on either fixed-price catalogues or auctions [6]. Such systems focus only on cost, which is just one factor in the complicated buyer-supplier relationship.

The University of Minnesota MAGNET (Multi-Agent Negotiation testbed) [2] system is designed to support the negotiation of contracts for coordinated tasks among a population of independent and self-interested agents. Dealing with coordinated tasks adds some major complexities to the auction model of agent interaction. First, because tasks have a temporal duration and precedence constraints, any bid selection algorithm must insure the temporal feasibility of the bids accepted. Second, because each of the tasks in the set of coordinated tasks is necessary to achieve the overall goal, any bid selection algorithm must insure coverage of all of the tasks. Third, delays and failures that might occur during the execution of the contracted tasks, can have devastating effects on the accomplishment of the overall goal. This introduces the need to assess scheduling risk and to account for risk in the bid selection algorithm.

Several papers have been published recently that deal with linear and integer programming formulations of the allocation problem in combinatorial auctions [1, 7]. This leads the the obvious question: can the MAGNET bid-evaluation problem be formulated as a linear or integer programming problem. If so, there are well-known and reasonably efficient evaluation procedures for these problems, although in general the integer programming problem is NP-complete.

2 Example

Suppose we have a job to do that involves performance of a set of coordinated tasks within a limited time frame. Also suppose that we wish to minimize the cost for completion of this job. Examples might include construction of a building or a bridge, evacuation of an island [8], or establishment of a multi-link communication channel. The resources needed to perform those tasks must be acquired from self-interested suppliers, who are attempting to maximize the value of the resources under their control. It is the job of

*Work supported in part by the National Science Foundation, awards NSF/IIS-0084202 and NSF/EIA-9986042.

a MAGNET Customer agent to use an auction process to obtain a set of commitments for those resources, that can be composed into a temporally feasible plan, at a minimum price.

Figure 1 shows an example task network. It is a directed acyclic graph, with arcs representing precedence relations. The numbers in parentheses are the expected durations of each task. We expect that this task duration data, as well as statistics on duration variability, resource availability, and supplier reliability, will be collected and made available from the MAGNET market infrastructure [2].

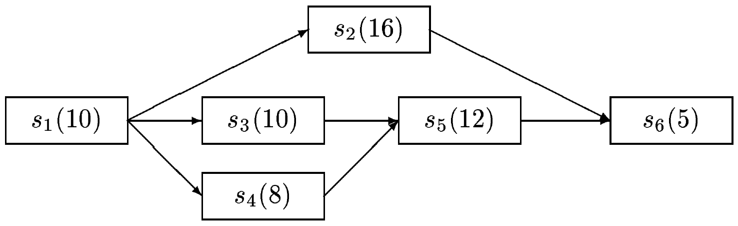

Figure 1: Example task network

Suppliers submit bids on sets of tasks based on their own resource availabilies and costs, and based on a Request for Quotes (RFQ) submitted by a potential customer. A bid includes a set of tasks and a price, along with timing data including duration and the earliest and latest times the task(s) may be started. When composing the RFQ for a plan, the customer must give suppliers some guidance about when the work must be done. This is done by specifying a *time window* for each task that gives the earliest start time and latest finish time. In generating this schedule, the customer has two conflicting goals:

1. Ensure that the bids received can be composed into a feasible plan. This can be done by specifying time windows that do not overlap.

2. Specify relatively wide, and possibly overlapping time windows, in hopes of attracting more bids and lower prices. The risk in doing this is that many bid combinations will not compose feasibly because their task time windows are in conflict.

If we take the first approach, then the bid-evaluation process reduces to a variation on the combinatorial-auction winner-determination process [11], without the free-disposal assumption[1]. The second approach leads to a much more difficult and interesting bid-evaluation problem, which is the subject of this paper.

The plan in Figure 1 has a *makespan* of 37 time units. This is the longest path through the graph. If we have a deadline for plan completion of 40 units, then we have an *overall slack* of 3 units, or about 8%.

Table 1 shows some bids that might be received for this plan. Several observations are apparent:

- Each bid may specify a "bundle" of tasks. A single price is given for the bundle.

- In this example and in the remainder of this paper, we assume that bids give early start, late start, and duration data for each task individually. This assumption can be relaxed.

- Bids b_1 and b_3 cannot both be accepted, because they both specify task s_5. Each task must be allocated to exactly one bid.

- Bids b_1 and b_2 cannot both be accepted, because the precedence relation $s_5 \prec s_6$ would be violated. This because the earliest time task s_5 could be completed under bid b_1 is 35, while the latest time task s_6 could start under bid b_2 is 33.

- Bids b_1 and b_4 cannot both be accepted. This is more subtle. The precedence relations $s_1 \prec s_4$ and $s_4 \prec s_5$ can be satisfied individually. However, when we attempt to combine the two bids, we see that, in order to satisfy $s_1 \prec s_4$, the early finish time of task s_4 is pushed back to 23.5, and $s_4 \prec s_5$ is violated.

We'll use this example to examine details of our bid-evaluation process.

[1]The free-disposal assumption states that items can remain unallocated without penalty.

Table 1: Example Bids

Bid	Task	Price	Early Start	Late Start	Duration
b_1	s_1	200	1.5	3.0	11.0
	s_3		12.5	14.0	8.5
	s_5		22.0	23.0	13.0
b_2	s_2	290	10.0	16.0	18.0
	s_6		28.0	33.0	6.0
b_3	s_4	160	9.0	13.0	6.0
	s_5		15.0	19.0	14.0
b_4	s_4	20	10.0	15.0	11.0

3 IP formulations

We start by introducing some notation. A plan consists of a set $\mathcal{S}$ of tasks $s_j, j = 1..m$. Each task s_j has a precedence set $\mathcal{P}_j = \{s_{j'} | s_{j'} \prec s_j\}$, the set of tasks $s_{j'}$ that must be completed before s_j is started. At the conclusion of some bidding process, we have a set $\mathcal{B}$ of bids $b_i, i = 1..n$. Each bid b_i specifies a set of tasks $\mathcal{S}_i$ and a price p_i. For each task s_j, a bid b_i that includes the task ($s_j \in \mathcal{S}_i$) may specify an early start time e^i_j, a late start time f^i_j, and a duration d^i_j.

We have also defined a risk factor r_i, associated with each bid, that is based on static factors, such as the reputation of the bidder. The derivation of r_i is outside the scope of this discussion, but we include it for completeness.

3.1 A straightforward model

First, we present a "direct" model, in which the coverage and feasibility constraints are directly represented. We associate a 0/1 variable x_i with each bid b_i, with the sense that in a solution where $x_i = 1$, b_i is accepted. No pre-processing is required other than composing the constraint rows.

If we include only static risk factors (those that are determined strictly by the set of bids chosen, and not by scheduling considerations), the formulation of the basic bid-evaluation problem is:

Minimize:

$$\sum_{i=1}^{n} (p_i + r_i) x_i$$

Subject to:

- Bid selection – each bid is either selected or not selected. These are the integer variables that make this an integer programming problem.

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

- Coverage – each task must be included exactly once.

$$\forall j = 1..m \sum_{i | s_j \in \mathcal{S}_i} x_i = 1$$

  Note that under the free disposal assumption, each task would be included at most once, rather than exactly once.

- Local feasibility – each task must be able to start after the earliest possible completion time of each of its predecessors. This constraint ignores global feasibility. In other words, here we are looking only at the start times of a particular task and its immediate predecessors.

$$\begin{aligned} \forall j &= 1..m, \forall i | s_j \in \mathcal{S}_i, \forall i' | s_{j'} \in (\mathcal{S}_{i'} \cap \mathcal{P}_j), \\ & x_i f^i_j \geq x_{i'} (e^{i'}_{j'} + d^{i'}_{j'}) - M(1 - x_i) \end{aligned}$$

where M is a "large" number[2], and the last term $M(1-x_i)$ is used to make the constraint satisfied in the case where $x_i = 0$.

In our example, the precedence relations between b_1 and b_4 would be expressed as

$$\begin{aligned} x_4(15.0) &\geq x_1(1.5+11.0) - M(1-x_4) \\ x_1(23.0) &\geq x_4(10.0+11.0) - M(1-x_1) \end{aligned}$$

- Global feasibility – each task must be able to start after the earliest possible completion time of each of its predecessors, where the predecessors may in turn be constrained not by their bids, but by their respective predecessors.

$$\begin{aligned} \forall j &= 1..m, \forall i | s_j \in \mathcal{S}_i, \forall i' | s_{j'} \in (\mathcal{S}_{i'} \cap \mathcal{P}_j), \forall i'' | s_{j''} \in (\mathcal{S}_{i''} \cap \mathcal{P}_{j'}), \\ & x_i f_j^i \geq x_{i'} d_{j'}^{i'} + x_{i''}(e_{j''}^{i''} + d_{j''}^{i''}) - M(1-x_i) \\ \forall j &= 1..m, \forall i | s_j \in \mathcal{S}_i, \forall i' | s_{j'} \in (\mathcal{S}_{i'} \cap \mathcal{P}_j), \\ & \forall i'' | s_{j''} \in (\mathcal{S}_{i''} \cap \mathcal{P}_{j'}), \forall i''' | s_{j'''} \in (\mathcal{S}_{i'''} \cap \mathcal{P}_{j''}), \\ & x_i f_j^i \geq x_{i'} d_{j'}^{i'} + x_{i''} d_{j''}^{i''} + x_{i'''}(e_{j'''}^{i'''} + d_{j'''}^{i'''}) - M(1-x_i) \\ & \vdots \end{aligned}$$

In our example, the infeasibility between b_1 and b_4 is captured by the constraint

$$x_1(23.0) \geq x_4(11.0) + x_1(1.5+11.0) - M(1-x_1)$$

The number of constraints generated by these formulas is highly variable, depending strongly on the depth of the plan (the length of the longest path in the precedence network), and on the detailed composition of the bids.

3.2 Collapsing the feasibility constraints

The formulation given in Section 3.1 is correct, but it can be dramatically improved, based on several observations. The first is that we can pre-process the coverage constraints to reduce the number of bids. If there is any task s_j for which only one bid b_i has been received (we'll call bid b_i a "singleton" bid for task s_j), b_i must be part of any complete solution. Bids b_k that conflict with b_i can then be discarded. In more formal terms,

$$\forall j | \sum_{i|s_j \in \mathcal{S}_i} 1 = 1, x_i = 1, \forall k | \mathcal{S}_i \cap \mathcal{S}_k \neq \emptyset, x_k = 0$$

This test is repeated until no further singleton bids are detected.

Next, we make the following observations regarding the feasibility constraints:

1. We have generated feasibility constraints between bids that cannot possibly be part of the same solution, because they contain overlapping task sets. The coverage constraints will ensure that only one bid will be chosen to cover each task. We can discard such constraints immediately. In our example, this means that we need not generate or evaluate any feasibility constraints that include both b_1 and b_3.

2. The feasibility constraints can be greatly simplified by doing the arithemetic during preprocessing, and including only those constraints that can have an impact on the outcome. This way, we eliminate all the feasibility constraints shown in Section 3.1, and replace them with a much smaller number of simple compatibility constraints. In other words, if we start with a constraint of the form

$$x_i f_j^i \geq x_{i'}(e_{j'}^{i'} + d_{j'}^{i'}) - M(1-x_i),$$

[2]our implementation uses a value of approximately 10^{12}

we compute $f_j^i - (e_{j'}^{i'} + d_{j'}^{i'})$. Just in case the result is negative, we include a constraint of the form

$$x_i + x_{i'} \leq 1$$

which will prevent both x_i and $x_{i'}$ from being part of the same solution. This simplification can be similarly applied to the global feasibility constraints. In general, such constraints tell us that for some combination of n bids, at most $n-1$ of them may be part of a solution.

In our example, we can observe the infeasibility between b_1 and b_4 during pre-processing, and rather than generate the formula given above in Section 3.1, we replace it with

$$x_1 + x_4 \leq 1$$

3. If we have successive tasks in the same bid, we can filter the bids themselves for internal feasibility prior to evaluation. After the previous step, any constraints of the form $x_i + x_i \leq 1$ represent bids x_i that can be discarded.

3.3 Minimizing completion time

If we want to minimize the time to complete the plan, we must develop an expression for the completion time. To begin with, we define t_0 as the (fixed) start time of the plan. Then we need to determine the latest time at which some task will be completed. We needn't consider all tasks, just the "leaf tasks," those that have no successors. If we ignore precedence constraints, the earliest possible completion time t_c for the plan as a whole is the maximum early finish time of any leaf task for a given bid assignment. Since in any valid bid assignment, only one bid is chosen for any given task, we can express this as

$$t_c = \max_{j | \forall k, s_j \notin \mathcal{P}_k} \sum_{i | s_j \in \mathcal{S}_i} x_i (e_j^i + d_j^i)$$

where a task s_j that has no successors is one that is in the predecessor set of no other tasks.

Unfortunately, it doesn't work to ignore precedence constraints. There may be a task in the precedence set of any given leaf task s_j whose early finish time in a given bid assignment will prevent s_j from starting at its early start time. To avoid this problem, while taking advantage of the precedence constraints we've already developed, we do two things. First, we must have a single task whose completion marks the end of the plan; to ensure that this is the case, we create a dummy task s_c, with 0 duration. We then define t_c as the start time of task s_c. We don't define a bid for this task, because we want to use its start time as a variable. We also define the plan start time t_0. Then we have to add a completion-time term to the objective function, which now reads:

Minimize:

$$W_c \sum_{i=1}^{n} (p_i + r_i) x_i + W_t (t_c - t_0)$$

where W_c is the relative weight given to cost, W_t is the relative weight given to completion time, and $(t_c - t_0)$ is the total makespan of the plan.

Next, we add an additional set of feasibility constraints, as given above in Section 3.1, to constrain the dummy task to start later than the completion times of all the leaf tasks. This set will include the local and global feasibility constraints, expanded recursively to the root tasks, substituting t_c for $x_i f_j^i$. Since t_c is a variable that appears in the objective function, its final value with be the earliest time that is greater than the maximum early completion time over all leaf tasks for any given bid assignment.

This approach does not work with the simplified form of the feasibility constraints as given in Section 3.2. There, we have discarded the temporal information in preprocessing, and are left with simple compatibility constraints. This deprives the IP solver of the information necessary to operate on completion time. An alternative approach in this case would be to make completion time be a constraint, rather than a factor in the objective function. This would typically require multiple passes, reducing the completion time until either no solution exists, or until the solution price becomes unacceptable.

4 Experimental results

To illustrate the effectiveness of our formulation, we have implemented both the original formulation given in Section 3.1, and the pre-processed formulation given in Section 3.2. Results show that our IP formulation is practical for moderate-sized problems, though it scales exponentially and the time required to find the optimum solution exhibits a very high variability. Experiments were run using dual-processor 850 MHz Linux boxes. Timings are given in wall-clock time. The MAGNET system is written in Java, and the IP solver is lp_solve, available from ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.

Each problem set consists of 200 problems, with randomly-generated plans and randomly-generated bids. Our problem generator has a large number of parameters; we kept all of them constant except for the task-count and bid-count values. Since our goal is to evaluate bids in a time-limited multi-agent interaction situation, we are primarily interested in scalability and predictability. Secondarily, we are interested in discovering measurable problem characteristics that the agent can use to tune its evaluation process "on the fly."

The process for generating problems operates as follows:

1. *Generate plan*: The desired number of tasks is generated, and random precedence relations are created between them. Tasks are randomly selected from among three "task types" that specify different values of expected duration, duration variability, and expected resource availability.

2. *Compose RFQ*: An RFQ is generated by setting time windows for the tasks in the plan, and specifying the timeline for the bidding process. Time windows are set by determining the makespan of the plan (the longest path through the precedence network) and multiplying it by a "slack" factor of 1.2, then "relaxing" the time windows for individual tasks to allow some overlap. The final result is that individual tasks are given time windows of at least 125% of their expected values (tasks not on the critical path will have longer time windows). In the test environment we ignore the bidding timeline.

3. *Generate Bids*: A specified number of attempts are made to generate bids against the RFQ. Each bid is generated by selecting a task at random from the plan, using the task-type parameters to generate a supplier time window for that task, and testing this time window against the time window specified in the RFQ for that task. If the supplier's time window is contained within the RFQ time window, we call it a "valid task spec" and add the task to the bid. If a valid task spec was generated, then with some probability, each predecessor and successor link from that task is followed to attempt to add additional tasks to the bid, and so on recursively. The resulting bids specify "contiguous" sets of tasks, and are guaranteed to be internally feasible. Finally, a cost is determined for the overall bid. Because valid task specs are not always achieved, some attempts to generate bids will fail altogether.

The first experiment compares the performance of the original IP formulation to the revised formulation. We were only able to run the original formulation on the smallest problems (5 tasks, up to 20 bids) because some problems were generating more than 10^5 rows and taking inordinate amounts of time to solve (we stopped one after 13 hours). Table 2 shows the results of this experiment. All entries are averaged across 200 problems. Key features to observe are the relative problem sizes (number of rows) and the extreme variability (given as σ(time)) of the original formulation. For the revised formulation, the reported time is the sum of preprocessing time and IP solver time. The time required for preprocessing is generally between 2 and 20 times the run time of the IP solver itself, but this appears to be time well spent. Our experience attempting to solve larger problems with the original formulation shows that the advantage of preprocessing grows dramatically as problem size increases.

The second series demonstrates scalability of the search process as the size of the plan varies, with a (nearly) constant ratio of bids to tasks (the ratio varies somewhat due to the random nature of the bid-generation process). Table 3 shows problem characteristics for this set. In the table, the "Solved" column gives the number of problems solved out of 200 (not all 200 problems were solvable), "PP Time" gives the preprocessor time, "IP Time" gives the time spent in the IP solver itself, and σ(time) gives the standard deviation of the sum of the "PP Time" and "IP Time" columns.

Table 2: Comparing IP formulations

		Original IP			Revised IP		
Task Count	Bid Count	Rows	Time (msec)	σ(time)	Rows	Time (msec)	σ(time)
5	11.4	487	195	1001	16.3	71.3	144
5	13.5	869	233	937	18.6	71.0	150
5	15.0	1265	753	5125	20.3	68.8	125
5	17.0	2271	3150	16256	22.7	95.8	279

Table 3: Task size experiment: Revised IP

Task Count	Bid Count	Bid Size	Solved	Rows	PP Time (msec)	IP Time (msec)	σ(time)
5	13.5	1.74	180	18.6	8.4	62.5	150
10	28.7	2.20	175	48.2	84.8	64.7	167
15	45.2	2.67	162	115	390	356	630
20	61.2	2.98	162	330	1711	288	2519
25	77.5	3.40	149	1030	6919	2273	24815
30	93.6	3.82	147	2296	19515	4150	71031

5 Related work

The determination of winners of combinatorial auctions [5] is known to be hard. Many algorithms have been proposed to produce good or optimal allocations. Dynamic programming [9] produces optimal allocations, but works well only for small sets of bids, and imposes significant restrictions on the class of bids. Nisan [7] formalizes several bidding languages and compares their expressive power. He analyzes different classes of auctions, and proposes an approach based on Linear Programming for bid allocation. Shoham [4] produces optimal allocations for OR-bids with dummy items by cleverly pruning the search space. Sandholm [10, 12] uses an anytime algorithm to produce optimal allocations for a more general class of bids, which includes XOR and OR-XOR bids. Andersson [1] proposes integer programming for winner determination in combinatorial auctions. The major difference is that in the cases studied for combinatorial auctions, bid allocation is determined solely by cost. Our setting is more general. Our agents have to cover all the tasks, ensure feasibility of the bids they accept, and reduce scheduling risk.

Walsh has proposed using combinatorial auction mechanisms for supply-chain formation [13] and for decentralized scheduling [14]. Neither of these proposals requires the allocation solver to deal with temporal feasibility, which is the principal problem dealt with in this work.

6 Conclusion and future work

We have shown that Integer Programming can be applied to the problem of bid evalulation in a combinatorial auction situation that includes temporal constraints among items, and lacks the free-disposal option. The formulation we present typically requires an amount of pre-processing that is significantly greater than the time required by the IP solver itself. A significant weakness of the IP approach is its high degree of variability. For an agent that must perform on a fixed time schedule, this may not be acceptable. We suggest that running a stochastic search in parallel with the IP approach may alleviate this drawback.

Future work in this area will include learning how to effectively combine IP with the Simulated Annealing algorithm we have developed [3] in a real-time agent negotiation environment, using measurable problem characteristics to guide the process. We would also like to understand how to incorporate risk factors that depend on the distribution of slack in the schedule, and how to optimize completion time without significantly increasing the complexity of the problem.

References

[1] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination. In *Proc. of 4th Int'l Conf on Multi-Agent Systems*, pages 39–46. IEEE Computer Society Press, July 2000.

[2] John Collins, Corey Bilot, Maria Gini, and Bamshad Mobasher. Mixed-initiative decision support in agent-based automated contracting. In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, pages 247–254, June 2000.

[3] John Collins, Rashmi Sundareswara, Maria Gini, and Bamshad Mobasher. Bid selection strategies for multi-agent contracting in the presence of scheduling constraints. In A. Moukas, C. Sierra, and F. Ygge, editors, *Agent Mediated Electronic Commerce II*, volume LNAI1788. Springer-Verlag, 2000.

[4] Yuzo Fujishjima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. of the 16th Joint Conf. on Artificial Intelligence*, 1999.

[5] R. McAfee and P. J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[6] Paul Milgrom. Auction and bidding: a primer. *Journal of economic perspectives*, 3(3):3–22, 1989.

[7] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proc. of ACM Conf on Electronic Commerce (EC'00)*, pages 1–12, Minneapolis, Minnesota, October 2000. ACM SIGecom, ACM Press.

[8] Martha E. Pollack. Planning in dynamic environments: The DIPART system. In A. Tate, editor, *Advanced Planning Technology.* AAAI Press, 1996.

[9] Michael H. Rothkopf, Alexander Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[10] Tuomas Sandholm. An algorithm for winner determination in combinatorial auctions. In *Proc. of the 16th Joint Conf. on Artificial Intelligence*, pages 524–547, 1999.

[11] Tuomas Sandholm. Approaches to winner determination in combinatorial auctions. *Decision Support Systems*, 28(1-2):165–176, 2000.

[12] Tuomas Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proc. of the Seventeen Nat'l Conf. on Artificial Intelligence*, pages 90–97, 2000.

[13] William E. Walsh, Michael Wellman, and Fredrik Ygge. Combinatorial auctions for supply chain formation. In *Proc. of ACM Conf on Electronic Commerce (EC'00)*, October 2000.

[14] William E. Walsh, Michael P. Wellman, Peter R. Wurman, and Jeffrey K. MacKie-Mason. Some economics of market-based distributed scheduling. In *Proc. of the Eighteenth Int'l Conf. on Distributed Computing Systems*, pages 612–621, 1998.