APPEARED IN PROCEEDINGS OF ICMAS-2000

A Combinatorial Auction for Collaborative Planning*

Luke Hunsberger
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
luke@eecs.harvard.edu

Barbara J. Grosz
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
grosz@eecs.harvard.edu

Abstract

When rational, utility-maximizing agents encounter an opportunity to collaborate on a group activity, they must determine whether to commit to that activity. We refer to this problem as the initial-commitment decision problem (ICDP). *This paper describes a mechanism that agents may use to solve the ICDP. The mechanism is based on a combinatorial auction in which agents bid on sets of roles in the group activity, each role comprising constituent subtasks that must be done by the same agent. Each bid may specify constraints on the execution times of the subtasks it covers. This mechanism permits agents to keep most details of their individual schedules of prior commitments private. The paper reports the results of several experiments testing the performance of the mechanism. These results demonstrate a significant improvement in performance when constituent subtasks are grouped into roles. They also show that as the number of time constraints in bids increases, the probability that there is a solution decreases, the cost of an optimal solution (if one exists) increases, and the time required to find an optimal solution (if one exists) decreases. The paper also describes several strategies that agents might employ when using this mechanism.*

1. Introduction

When rational, autonomous agents encounter an opportunity to collaborate on some group activity, they must decide whether to commit to doing that activity. We refer to this problem as the *initial-commitment decision problem (ICDP).* We assume the new opportunity for collaborative action arises in context: each agent may have existing commitments to other individual and group activities. We further assume that agents are utility-maximizers.

Horty and Pollack [4] have initiated research into how an individual agent may evaluate new opportunities for single-agent action in the context of existing commitments. The initial-commitment decision problem addressed in this paper is a generalization of this problem to the group context, which introduces two significant complications. First, no single agent has complete information about the existing commitments of all the agents in the group: background context is distributed. Second, the approach (i.e., choice of method and distribution of tasks) that is best for the group may not be best for any individual agent alone.

An agent participating in multi-agent planning incurs significant costs, including time and computational resources devoted to group decision-making processes; opportunity costs for commitments, not only to doing its share of tasks in the group activity, but also to supporting the actions of others; and costs of doing actions to which it commits. To decide whether to join a proposed collaboration, an agent needs to assess the impact of the collaboration on its ability to do other work. Since the planning, decision-making, and opportunity costs can be substantial, it is preferable for agents to determine some upper bound on this impact prior to committing to planning with others.

In deciding whether to commit to a new group activity, each agent must estimate two factors: (1) the potential contributions it could make to the group activity (i.e., the constituent subacts it could do or participate in) and the costs of those contributions; and (2) the possibilities for the remaining tasks to be assigned to other group members in an individually rational manner. The first factor requires that agents examine information about their individual background contexts of commitments. Because agents may be unwilling or unable to share complete information about their individual contexts, this factor is best computed "locally" by individual agents. The second factor requires a global computation that takes into account the potential contributions of all the agents.

This paper presents a mechanism that a group of agents can use to solve the initial-commitment decision problem. This mechanism, which we refer to as the *ICDP mechanism,* uses a combinatorial auction [9, 2, 7] to coordinate the sorts of local and global computations described above. Each potential contribution to the group activity is stated in terms of a locally-computed bid that specifies a set of tasks, a cost for doing those tasks, and a set of constraints on the execution times of those tasks. The global computation determines the best combination of these bids (i.e., potential contributions). It is based on an existing winner-determination algorithm for combinatorial auctions [9] that we modified to enable it to handle time constraints.

By distributing the computational burden in this way, the ICDP mechanism allows agents to maintain the privacy of their existing commitments and to focus their computational efforts on their own potential for contributing to the proposed collaboration. Furthermore, being able to condition their bids on time constraints allows agents to protect the feasibility of their existing commitments. Finally, although the global computation may be carried out centrally (as was done in our empirical investigations), it may also be carried out in a distributed fashion, with agents searching different portions of the highly-structured search space.

Section 2 describes our representation of actions and recipes. Section 3 briefly reviews standard combinatorial auctions. Section 4 defines the ICDP auction mechanism. Section 5 presents empirical results and their implications for system design. These results demonstrate a significant improvement in performance when constituent subtasks are grouped into roles. They also show that as the number of time constraints in bids increases, the probability that there is a solution decreases, the cost of an optimal solution (if one exists) increases, and the time required to find an optimal solution (if one exists) decreases. The remaining sections discuss related work and conclusions.

2. Actions, Act-types, Recipes and Roles

Our representation of actions, act-types and recipes follows Grosz and Kraus' SharedPlans theory of collaborative planning [3], but extends the representation of act-types and recipes to include roles. The experiments described in Section 5 show that the use of roles allows the ICDP mechanism to scale to larger problem instances.

Actions, Act-types and Recipes in SharedPlans. Actions are either basic or complex. A basic action is an action that may be executed at will by an individual agent under appropriate conditions; a complex action is executed indirectly using a recipe. Both basic and complex actions are classified according to their act-types. A *recipe* for a complex act-type is a set of subacts and constraints such that the doing of those subacts under those constraints constitutes the doing of an action of that type. Typically, recipe constraints

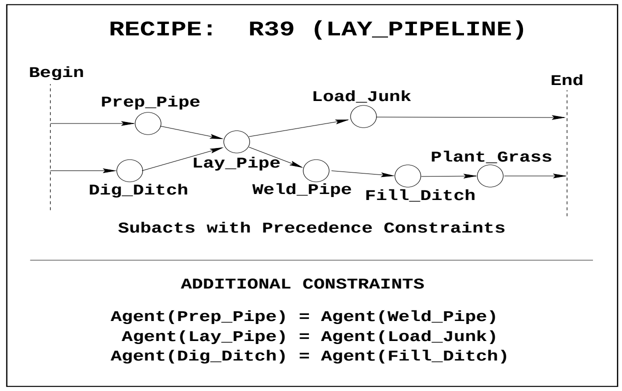

Figure 1. Sample recipe

include precedence constraints on the execution times of the various subacts; they may also include constraints that certain subacts be executed by the same agent or subgroup.[1]

Figure 1 shows a sample recipe, `REC39`, that specifies one way of doing a `LAY_PIPELINE` action.[2] Precedence constraints are indicated by arrows in the figure (e.g., the `Weld_Pipe` subact must be done before the `Fill_Ditch` subact). Although not shown in the figure, we allow precedence constraints to include *offsets.* Thus, the `Load_Junk` subact might be constrained to occur at least 20 minutes after the `Lay_Pipe` subact. Recipes may contain complex subacts; recursively choosing recipes for these subacts gives rise to a multi-level recipe hierarchy [5]. However, in this paper, we assume that recipes are fully expanded so that agents need to consider only basic subacts.

Adding Roles to Act-Types and Recipes. The value of roles may be illustrated by an example, the representation of a transaction act-type in electronic commerce. No matter which protocol (or recipe) is used to govern the transaction, some tasks must be done by the buyer, others by the seller. In addition, various preconditions, postconditions and application constraints may be succinctly stated in terms of the buyer and seller roles (e.g., the seller must own the object being sold prior to the start of the transaction). However, despite the buyer and seller roles being naturally associated with the transaction act-type, the tasks covered by each role are determined by the protocol chosen to govern the transaction. For example, in one protocol, the buyer might have to list the objects being purchased, while in another the buyer might have to go through a complex set of identification-verification steps. In addition to specifying the tasks to be covered by the act-type roles, some protocols may introduce

[1] As is standard in work on planning, act-type definitions specify preconditions, application constraints and necessary effects for an action of that type, as well as any parameters required in doing the action.

[2] We use teletype font for names of act-types, recipes and roles.

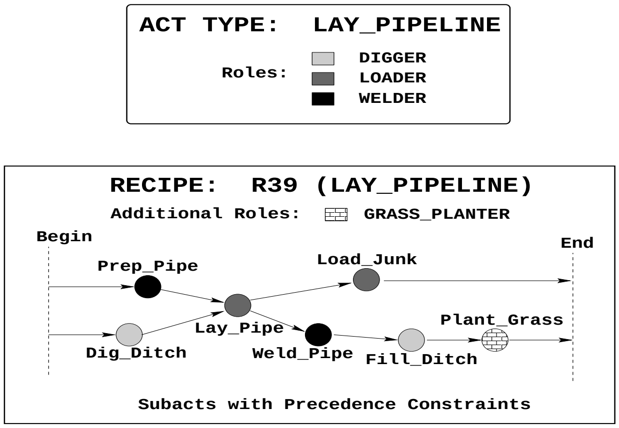

Figure 2. Act-type and recipe with roles

additional, protocol-specific roles. For example, one protocol might require an additional monitor role, responsible for carrying out a variety of transaction-monitoring tasks.

We extend the representation of act-types and recipes to include *roles.* $\texttt{ActTypeRoles}(\alpha)$ and $\texttt{RecipeRoles}(Rec)$ denote the roles associated with the act-type α and the recipe Rec, respectively. The recipe must specify the set of subacts covered by each act-type role and each recipe role. We require each subact to be covered by one and only one role. The agent filling a role is responsible for doing all the subacts covered by that role.

Figure 2 shows roles incorporated into the example from Figure 1. There are three act-type roles—`DIGGER`, `LOADER` and `WELDER`—in the `LAY_PIPELINE` act-type. In addition, the recipe `REC39` has been modified to separately specify the set of subacts covered by each role (e.g., the agent filling the `LOADER` role must do both the `Lay_Pipe` and `Load_Junk` subacts). `REC39` is further modified to include an additional, recipe-specific role that arises from this recipe's particular way of subdividing the `LAY_PIPELINE` action. The recipe specifies the responsibilities of this additional role; the agent assigned to the `GRASS_PLANTER` recipe role is responsible for executing the `Plant_Grass` subact.

Roles reduce the computational burden in two ways. First, if a particular agent finds that it is unable to do one of the subacts covered by some role, then that agent may immediately move on to considering other roles instead. Second, instead of needing to identify agents to do each of the subacts, the group need only identify agents to fill the various act-type and recipe roles; if there are many fewer roles than subacts, then there are fewer decisions to make.

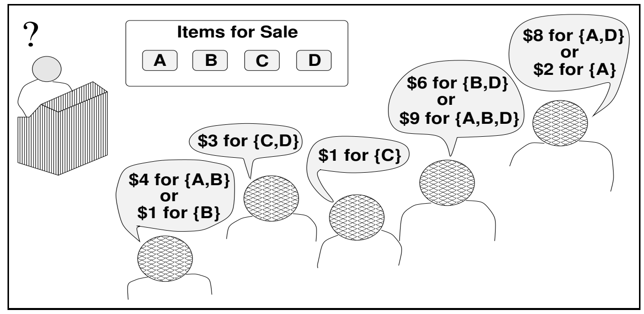

Figure 3. A combinatorial auction

3. Combinatorial Auctions

In a *combinatorial auction* [9, 2, 7], there are multiple items for sale, participants who may place bids on arbitrary subsets of those items, and an auctioneer who must determine which *awardable* combination of bids maximizes revenue. Figure 3 shows a combinatorial auction in which there are four items—A, B, C, and D—for sale and the participants have made bids such as "\$4 for $\{A, B\}$" and "\$9 for $\{A, B, D\}$."

In general, let $\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$ be the set of n items being auctioned and let $\mathbf{B}$ be the set of received bids. For each bid $b \in \mathbf{B}$, let $Items(b) \subseteq \mathbf{I}$ denote the subset of items covered by the bid and let $Amount(b)$ denote the amount of the bid. A *bid-set* (i.e., a collection of bids) is called *disjoint* if each item being auctioned is covered by at *most* one bid in the set, *covering* if each item is covered by at *least* one bid in the set, and *awardable* if it is both disjoint and covering. An awardable bid-set is also called a *solution.* For any disjoint bid-set BS, the revenue that BS, if awarded, would generate for the auctioneer is:

$$Value(BS) = \sum_{b \in BS} Amount(b).$$

The winning bid-set is an awardable bid-set that maximizes revenue: $BS^* = \underset{BS \in \mathcal{ABS}}{argmax}\, Value(BS)$, where $\mathcal{ABS}$ is the set of awardable bid-sets. The revenue generated by the winning bid-set is: $Value(BS^*)$. A winning bid-set is also called an *optimal solution.*

For the auction in Figure 3, the awardable bid-sets are:

$$\boxed{\$2 \text{ for } \{A\}} + \boxed{\$1 \text{ for } \{B\}} + \boxed{\$3 \text{ for } \{C, D\}} \quad \Rightarrow \$6$$
$$\boxed{\$2 \text{ for } \{A\}} + \boxed{\$6 \text{ for } \{B, D\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \$9$$
$$\boxed{\$4 \text{ for } \{A, B\}} + \boxed{\$3 \text{ for } \{C, D\}} \quad \Rightarrow \$7$$
$$\boxed{\$9 \text{ for } \{A, B, D\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \$10$$
$$\boxed{\$8 \text{ for } \{A, D\}} + \boxed{\$1 \text{ for } \{B\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \$10$$

The last two of these each yield the maximum revenue; thus either may be chosen to be the winning bid-set.

3.1. Existing Winner-Determination Algorithms

The general winner-determination (WD) problem for combinatorial auctions is $\mathcal{NP}$-complete [9, 2]. However, Sandholm [9] and Fujishima et al. [2] have independently presented WD algorithms that scale to problems involving scores of items and thousands of bids. The main insight is that bids in practice necessarily only sparsely populate the space of possible bids; thus the search for the winning bid-set should be restricted to the space of bid-sets composed of actual—not possible—bids.

At the core of each algorithm is a depth-first search through the space of disjoint bid-sets. Along each depth-first path in the search, a disjoint bid-set is incrementally constructed by successively appending individual bids. The path may end in a covering (and hence awardable) bid-set, or it may reach a dead-end with a non-covering bid-set for which there are no compatible bids to append. Each algorithm keeps track of the best (i.e., revenue maximizing) covering bid-set found so far and thus may be used as an anytime algorithm. Each algorithm uses an item-indexing scheme to ensure that each disjoint bid-set is generated at most once during the search. Sandholm organizes the received bids into an auxiliary data structure (called a *bid-tree*) to enable efficient generation of bids that are compatible with the current (partial) bid-set. Fujishima et al. partition the received bids into bins, each of which corresponds to a subtree of Sandholm's bid-tree.[3] In addition, each algorithm uses the same A*-admissible heuristic that estimates the revenue that items not yet covered by the bid-set might bring in; this heuristic speeds up the search by enabling pruning of partial bid-sets that are certain not to bring in as much revenue as the best solution found so far.

The differences between the approaches of Sandholm and Fujishima et al. lie mostly in their proposed improvements to their basic algorithms. We do not discuss these improvements because each sacrifices optimality in auctions involving time constraints and, in our application (as described below), agents use time constraints to protect the feasibility of their existing commitments.

4. A Mechanism for the Initial-Commitment Decision Problem

The initial-commitment decision problem (ICDP) arises when a group of agents encounter an opportunity for collaborative activity. We assume that the opportunity is in the form of a request that specifies both a time interval $[E, L]$ over which the action must be done and a payment P that the group is to receive for doing the action. If the agents can find some way of doing the action at a cost less than P, then they will profit from doing it.

In this section, we describe a mechanism that agents may use to solve the ICDP. The mechanism is based on a combinatorial auction in which agents bid on roles in the group activity. Having agents bid on roles, rather than subacts, reduces the number of items up for bid and thus enables the mechanism to scale to problems involving larger numbers of subacts. It also reduces the computational burden of bid generation because an agent that finds it is unable to carry out one of the subacts covered by some role may immediately move on to considering other roles instead.

The ICDP mechanism also allows agents to condition their bids on various time constraints, as well as on the choice of recipe for the group activity. Allowing time constraints in bids is essential in this application. It allows agents to protect the feasibility of their private schedules of existing commitments without having to reveal the details of those commitments to other agents. In addition, being able to condition their bids on the choice of recipe enables agents to generate a wider variety of bids, thereby increasing the likelihood of their finding a low-cost way of doing the group activity. For example, an agent might require only a small payment to cover a particular set of roles under `REC39`, but might be unable to cover those same roles under a different recipe. Were the agent unable to condition the bid on the choice of recipe, it would be unable to place a bid for that set of roles.

Winner determination in the ICDP mechanism is handled by an algorithm based on Sandholm's WD algorithm, but modified to be able to deal with time constraints in bids, as described below in Section 4.1.

Let $\{Rec_1, \ldots, Rec_c\}$ be the c recipes available for doing the group action. The ICDP mechanism involves c separate auctions, one for each recipe the group might use.

In our implementation, bids that are conditioned on the choice of recipe are split into multiple, recipe-specific bids; thus we henceforth assume that each bid pertains to a single recipe. For example, a sample bid pertaining to recipe `REC39` (from Figure 2) is shown below:

Roles	{WELDER, GRASS_PLANTER}
Amount	$300
GlobalConstraints	[3:30 p.m., 7:30 p.m.]
SubactConstraints	{Prep_Pipe < 4:00 p.m.}

In this bid, the bidder proposes to do the `WELDER` act-type role and the `GRASS_PLANTER` recipe role for a payment of $300 under the conditions that the `LAY_PIPELINE` group activity be done between 3:30 p.m. and 7:30 p.m. and the `Prep_Pipe` subact (one of the subacts covered by the `WELDER` role) be done before 4:00 p.m.

We define the density of a bid's constraints (DBC) as follows. Let R be the set of roles in the bid and S be the set of subacts covered by the roles in R. The bid may constrain

[3] Since the bins are coarser than Sandholm's bid-tree, additional checks are required to avoid redundant search.

the execution time of any subact covered by the bid (i.e., any subact in S) by providing lower or upper bounds; thus the bid may include up to $2|S|$ constraints. The DBC is defined to be $\frac{k}{2|S|}$, where k is the number of actual constraints in the bid. For example, the DBC for the sample bid above is $\frac{1}{6}$ because the bid contains a single subact execution-time constraint where it could have contained up to six—two for each of the three subacts covered by the bid (see Figure 2). In Section 5, the density of bid constraints will be shown to have a dramatic impact on the performance of the ICDP mechanism.

For the auction corresponding to recipe Rec_i, let

$$\mathbf{I}_i = \texttt{ActTypeRoles}(\alpha) \cup \texttt{RecipeRoles}(Rec_i)$$

be the set of roles (i.e., items) being auctioned. Let $\mathbf{B}_i$ be the set of bids received. For each bid $b \in \mathbf{B}_i$, let:

- $Roles(b) \subseteq \mathbf{I}_i$ be the subset of roles covered by the bid,
- $Amount(b)$ be the amount of the bid,
- $GlobalConstraints(b)$ be the bidder's constraints on the starting and ending times for the group activity, and
- $SubactConstraints(b)$ be the bidder's constraints on the execution times of the subacts covered by the bid.

Disjoint and covering bid-sets are defined as in a standard combinatorial auction; the awardable bid-sets, however, are defined differently. A bid-set BS is called *awardable* (with respect to recipe Rec) if, in addition to its being disjoint and covering, there exists a set of execution times for the subacts in Rec that satisfy all of Rec's precedence constraints and all the time constraints from the bids in BS.

The *cost* of an awardable bid-set is given by:

$$Cost(BS) = \sum_{b \,\in\, BS} Amount(b).$$

The winning bid-set for the auction corresponding to recipe Rec_i is given by: $\quad BS_i^* = \underset{BS \,\in\, \mathcal{ABS}_i}{argmin}\, Cost(BS)$,

where $\mathcal{ABS}_i$ is the set of awardable bid-sets. The associated cost is: $Cost(BS_i^*)$. Over all the auctions, the winning bid-set/recipe pair is an element of $\{(BS_1^*, Rec_1), \ldots, (BS_c^*, Rec_c)\}$ with the minimum cost.

4.1. The Modified WD Algorithm

To determine the winner of a combinatorial auction involving bids that include time constraints, we modified Sandholm's basic winner-determination (WD) algorithm (described in Section 3.1).[4] The modifications included adding a consistency check to the bid-set construction process and minimizing cost rather than maximizing revenue. The modified WD algorithm follows each depth-first path until: (1) the cost of the current bid-set is greater than that of the best bid-set found so far, (2) the current bid-set is found to be inconsistent, (3) the current bid-set is not covering, but there are no compatible bids to append, or (4) the current bid-set is awardable. In the first three cases, the current-bid set is pruned; in the last case, the current bid-set becomes the best found so far.

Consistency Check. The precedence constraints from the recipe determine a partial order among the recipe's subacts (e.g., as shown in Figure 2). Therefore, each constraint imposed by a bid, although directly constraining the execution time of only one subact, may indirectly constrain the execution times of numerous subacts that are precedence-related to the directly constrained subact. For example, a bid's constraint that the `Weld_Pipe` subact (from Figure 2) must occur after 2 p.m. indirectly constrains the `Fill_Ditch` and `Plant_Grass` subacts.

During the bid-set construction process, our algorithm maintains a greatest lower bound (GLB) and a least upper bound (LUB) for the execution time of each subact based on the recipe's precedence constraints and constraints from the current bid-set. Whenever a new bid is appended to the current bid-set, the effects of each new constraint on these GLBs and LUBs are propagated through the partial order network. After incorporating the new constraints, it is only necessary to check that each GLB is no greater than its corresponding LUB. If so, the constraints from the extended bid-set are consistent with the precedence constraints from the recipe; otherwise, the search must backtrack.

Minimizing Cost. Although moving from maximizing revenue to minimizing cost requires only minor changes to the algorithm, it significantly impacts the performance of the algorithm because it enables high-cost bid-sets to be pruned without requiring the sort of heuristic described in Section 3.1. If the cost of the current bid-set ever exceeds the cost of the best bid-set found so far, then the current bid-set may be pruned immediately.[5]

5. Experiments

In this section, we report studies of our modified WD algorithm. The first experiment quantifies the performance improvement that arises from using roles to bundle subacts. The remaining experiments clarify the tradeoffs that arise from allowing bids to include time constraints on the group activity. The inclusion of time constraints results in fewer consistent bid combinations and thus tends to increase the cost of an optimal solution (i.e., a least-cost awardable bid-set); in extreme cases, time constraints might result in there being no solution at all. We also examine how the number of bids affects the relationship between the time constraints in bids and the likelihood of their yielding a solution.

[4] We chose Sandholm's algorithm because his bid-tree structure ensures non-redundant search.

[5] In this paper, we do not explore the additional computational benefits that arise from using a heuristic like the one described in Section 3.1.

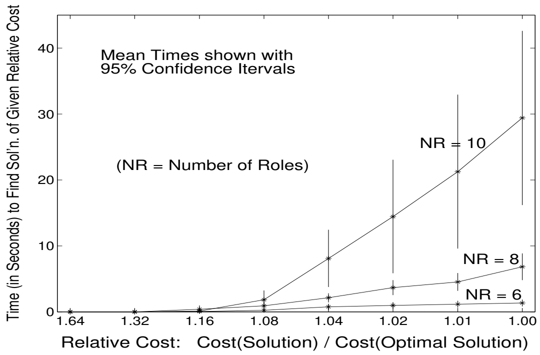

Figure 4. Varying the number of roles (NR)

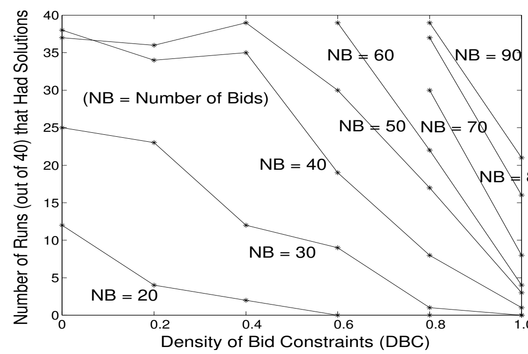

Figure 5. Varying the number of bids and the density of bid constraints

In each experiment, the group action was constrained to occur within the time interval $[0, 100]$. Recipes were generated randomly such that in each recipe: (1) the number of precedence constraints was the same as the number of subacts, and (2) subacts were assigned to roles with equal probability. The bid-generation process was simulated by randomly generating bids according to various parameters (described in each experiment). The cost of a bid was randomly generated such that the cost-per-subact was uniformly distributed between 10 and 20.

Experiment 1. The goal of this experiment was to determine the improvement in performance generated by bundling subacts into roles. The number of subacts was fixed at 40; the number of roles covering those subacts was 6, 8 or 10 (data collected for each case). Because winner determination in combinatorial auctions is exponential in the number of items being auctioned [9], we expected the performance to improve as the number of roles decreased (i.e., as the number of subacts covered by each role increased). The results of this experiment are plotted in Figure 4. Each point in the plot shows the time (averaged over 40 runs)[6] required to reach a solution of a particular *relative cost,* where the relative cost of a solution is defined as follows: $Cost$(Solution)$/$ $Cost$(Optimal Solution). Thus, the relative cost of the optimal solution is 1 (at the far right of the horizontal axis), whereas that of suboptimal solutions is greater than 1. There were 50 bids in each run, each bid covering 1 or 2 roles (uniformly distributed). Bids did not contain time constraints in this experiment.

The results clearly indicate a substantial improvement in performance as the number of roles covering the subacts decreases. If roles were not used, the number of biddable items would be the same as the number of subacts (i.e., 40), resulting in much poorer performance.

The results of this experiment also indicate that solutions with low relative costs are typically found within a relatively short period of time. For the initial-commitment decision problem, such near-optimal solutions may suffice. Once agents commit to the group activity based on a near-optimal solution, they may use this solution as a baseline while continuing to search for lower-cost solutions [8].

Experiment 2. The goal of the second experiment was to show how the likelihood of a given set of bids yielding a solution depends both on the number of bids received and the number of constraints in each bid. In this experiment, each recipe contained 40 subacts covered by 10 roles. Each data point corresponds to 40 runs in which both the number of bids (NB) and the density of bid constraints (DBC) were fixed. A DBC value of 0 represents that the bid contained no subact-execution time constraints; a value of 1 represents that the bid constrained—with both lower and upper bounds—the execution time of each subact covered by that bid's roles.[7] Each data point indicates the number of runs (out of 40) that yielded a solution. The results are plotted in Figure 5. They clearly show that the likelihood of the bids yielding a solution falls off sharply as the density of bid constraints increases. The results also indicate how many bids would be required to ensure a certain likelihood of finding a solution for a given density of bid constraints. Although these results do not directly apply to settings in which agents are focused on minimizing their own expected costs, they do provide guidance for agent design. Even in

[6] In one of the runs with 10 roles, there was no solution. For that case, the results were averaged over the remaining 39 runs.

[7] Time constraints in bids were determined as follows. First, in a random order, execution times consistent with the recipe's precedence constraints were selected for the subacts covered by the bid; for each subact S, a time point T_s was randomly chosen, uniformly distributed between the subact's GLB and LUB, and the effects of this assignment were propagated through the partial-order network of precedence constraints (as described in Section 4.1). Second, for each subact S, time constraints (simulating interactions with the agent's schedule of existing commitments) of the form $Time(S) \leq T_s + 10$ and $Time(S) \geq T_s - 10$ were generated. Each time constraint was included in the bid with probability equal to the DBC.

such settings, agents will want to identify solutions that are individually rational. These results indicate that the fewer constraints they place on their bids, the fewer bids they will need to submit, as a group, to find a solution.

Experiment 3. The goal of the third experiment was to determine how the density of bid constraints affects the cost of an optimal solution, the time required to find an optimal solution, and the time required to exhaust the search space (which is necessary to determine optimality).

The parameters were chosen such that solutions were generated in at least 90% of the runs for each DBC value. The number of subacts was 30, the number of roles was 8, the number of bids was 75, and the number of roles covered by each bid ranged from 1 to 3 (uniformly distributed).

The results of the experiment are shown in Figure 6. Plot (a) shows that the cost of an optimal solution rises as the density of bid constraints rises; plot (b) shows that the time required to find an optimal solution or to exhaust the search space decreases as the density of bid constraints rises. In both cases, the reason is that the presence of bid constraints effectively shrinks the pool of awardable bid-sets. An optimal solution in the absence of constraints might become an inconsistent, and hence non-awardable bid-set in the presence of constraints.

Implications of the Results. The results of these experiments suggest possible strategies that agents might employ when using the ICDP mechanism. For example, suppose the individual contexts of the agents were such that each agent could choose between making minimally-constrained, higher-cost bids or maximally-constrained, lower-cost bids. This would be the case if adding constraints could ensure that low-cost methods could be used to do various tasks. In early iterations of the ICDP mechanism, agents might be encouraged to generate maximally-constrained, lower-cost bids that could be examined quickly to determine whether they yielded a solution of sufficiently low cost. If not, agents could generate additional bids involving fewer constraints, thereby enlarging the pool of awardable bid-sets. As the pool of awardable bid-sets grows, the time to carry out an exhaustive search grows. However, the results of the first experiment (Figure 4) show that the modified WD algorithm tends to find near-optimal solutions very quickly; exhausting the search space tends to provide only a minimal reduction in solution cost. Thus, at each iteration, the modified WD algorithm could be run until a solution is found with cost below some threshold or until some time limit is reached.

6. Related Work

In Collins et al. [1], a Customer agent selects a recipe and issues a call for bids from a set of Supplier agents. Each bid is on a set of tasks, and specifies not only a price for doing that set of tasks, but also a price for each task if awarded separately. Bids may constrain task execution times. The Customer agent uses a generalized simulated-annealing search with heuristics based on cost, risk, feasibility and task-coverage. The primary differences between their work and ours are as follows. First, they do not bundle tasks into roles and thus lose the corresponding computational advantage. Second, they allow the Customer agent to use pieces of bids to construct awardable bid-sets; thus, a bid on n items might be split into any one of 2^n pieces. Furthermore, the cost of any piece covering fewer than n items is based on summing the individual costs of the corresponding subacts and thus ignores any sub-additivity or super-additivity relationships existing among them. (One of the main purposes of a combinatorial auction is to take advantage of such relationships.) Finally, because their heuristic is based on a combination of risk, cost, feasibility and task-coverage, their algorithm may explore vast regions of the search space involving infeasible "solutions", whereas we

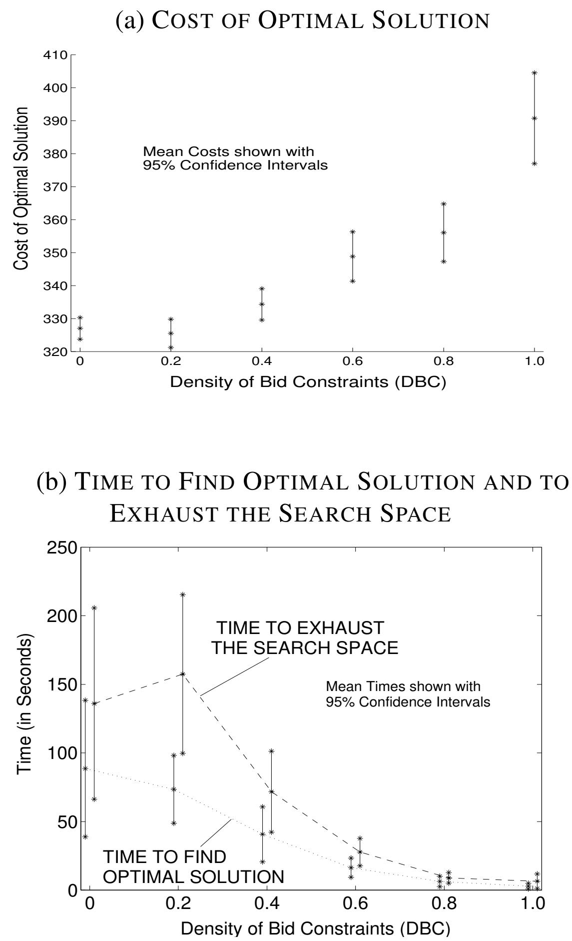

Figure 6. Plots from Experiment 3

restrict our search to the space of feasible bid-sets.

Walsh and Wellman [10], Kutanoglu and Wu [6], and Fujishima et al. [2] present market-oriented solutions to a range of combinatorial problems. Each approach involves iterative auctions. Walsh and Wellman [10] present a decentralized, asynchronous protocol for allocating and scheduling tasks. In their approach, each good (representing either a physical resource or a service provided by some agent) is auctioned in a separate (M+1)st-price auction. A separate agent is dedicated to the production of each single unit of a good. Auctions are run until they reach "quiescence" and conditions are given under which quiescence necessarily yields a valid solution. To handle time, they form (a subset of) the cross product of the discrete time line with the set of goods (resulting in a proliferation of goods, and hence auctions) and introduce special "bundling arbitrageur" agents responsible for procuring goods over various time intervals. Our approach handles time constraints without such an adverse computational impact.

Kutanoglu and Wu [6] use the iterative-auction approach (but only a single auction) to solve a distributed resource-scheduling problem in which a set of jobs must be performed, each consisting of a set of operations, each operation requiring a particular machine for some duration. They associate an agent with each job and the set of biddable items is the set of discrete machine/time-slot pairs, each pair having an associated price. For each auction iteration, each agent generates a single bid; the auctioneer examines the bids and then updates the prices in an attempt to reduce resource conflicts. The procedure stops when the auctioneer finds that all of the bids are compatible. The primary differences between our work and that of Kutanoglu and Wu are: (1) they map agents to jobs in a one-to-one fashion and use the auction to find sets of machine/time-slot pairs to satisfy the needs of each job, whereas we use an auction to find a mapping from agents to jobs; and (2) their biddable items are machine/time-slot pairs, of which there are very many, whereas our biddable items are roles, of which there are comparatively few.

Fujishima et al. [2] present a "Virtual Simultaneous Auction" as an alternative to their winner-determination algorithm for a combinatorial auction (described earlier in Section 3.1). For each original bid, they create a virtual bidder that tries to secure the items in that bid. "The simultaneous auction is repeated ... until either an optimal allocation is found or a pre-set time deadline is reached." The virtual bidders follow a simple strategy of incrementing their bids if their previous bids were insufficient to acquire the desired goods. Their experimental results showed that this approach was competitive with their basic algorithm, although not quite as fast.

7. Conclusions

In this paper, we have presented a mechanism based on a combinatorial auction that a group of agents may use to solve the initial-commitment decision problem. To make the ICDP mechanism computationally feasible for a larger class of problems, we stipulated that agents bid on roles (i.e., recipe-specific bundles of subacts). To enable agents to tailor their bids to their schedules of existing commitments, the ICDP mechanism allows agents to condition their bids on various time constraints, requiring a modified combinatorial-auction winner-determination algorithm

The ICDP mechanism distributes the computational burden by making individual agents responsible for generating bids and using a global computation to find an optimal combination of bids.

In the future, we plan to focus on the bid-generation process required by the ICDP mechanism. Horty and Pollack [4] have initiated research into how an individual agent may evaluate new opportunities for single-agent action in the context of existing commitments. Generating bids for the ICDP mechanism requires similar sorts of reasoning. We also plan to explore distributing the global winner-determination computation in the ICDP mechanism.

Acknowledgments. The research reported in this paper has been supported in part by National Science Foundation grants IRI-9618848, IIS-9978343, IRI 95-25915 and CDA-94-01024. The authors wish to thank Martha Pollack for many helpful discussions about this work.

References

[1] J. Collins, R. Sundareswara, M. Tsvetovat, M. Gini, and B. Mobasher. Search strategies for bid selection in multi-agent contracting. In *Proceedings of IJCAI-99 Workshop on Agent-mediated Electronic Commerce (AmEC-99)*. 1999.

[2] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*. 1999.

[3] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86:269–357, 1996.

[4] J. F. Horty and M. E. Pollack. Evaluating options in context. In *7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98)*. 1998.

[5] L. Hunsberger. Making SharedPlans more concise and easier to reason about. In *Agent Architectures, Theories and Languages V*, volume 1555 of *LNAI*, pages 81–98.

[6] E. Kutanoglu and S. D. Wu. On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. Technical Report 97T-012, Lehigh University, 1997.

[7] S. Rassenti, V. Smith, and R. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[8] T. Rauenbusch. Towards efficient negotiation mechanisms for collaboration (Student Abstract). In *17th National Conference on Artificial Intelligence (AAAI-2000)*. To appear.

[9] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*. 1999.

[10] W. E. Walsh and M. P. Wellman. A market protocol for decentralized task allocation and scheduling with hierarchical dependencies. In *Third International Conference on Multi-Agent Systems (ICMAS-98)*, pages 325–332, 1998.