Chapter 3

A Mechanism for the Initial-Commitment Decision Problem

When rational, autonomous agents encounter an opportunity to collaborate on some proposed group activity, they must decide, both individually and collectively, whether to commit to that activity. This thesis refers to this problem as the Initial-Commitment Decision Problem (ICDP). This chapter presents a group decision-making mechanism based on a *combinatorial auction* (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999; Rassenti, Smith, and Bulfin, 1982) that agents can use to solve instances of the ICDP. In this mechanism, called the ICDP mechanism, agents bid on sets of tasks in the proposed activity. Of central importance to the mechanism is that agents can make their bids dependent on temporal constraints, thereby enabling each agent to protect its private schedule of pre-existing commitments. In addition to solving instances of the ICDP, the ICDP mechanism can also be used to solve task-allocation problems in the context of multi-agent activities to which the group is already committed.

The rest of this chapter is organized as follows. Section 3.1 describes the Initial-Commitment Decision Problem. Section 3.2 briefly reviews standard combinatorial auctions. Section 3.3 presents a modified combinatorial auction in which agents bid on sets of roles in a proposed group activity. Section 3.4 specifies the combinatorial auction-based ICDP according to the GDMM framework from Chapter 2. Section 3.5 identifies several problems that individual agents must be able to solve to participate effectively in the ICDP mechanism. Section 3.6 discusses related work.

3.1 The Initial-Commitment Decision Problem

When rational, autonomous agents encounter an opportunity to collaborate on some group activity, they must decide, both individually and collectively, whether to commit to doing that activity. We refer to this problem as the Initial-Commitment Decision Problem. It is assumed that new opportunities for collaborative action arise in context: each agent may have pre-existing commitments to other individual and group activities. It is further assumed that agents are utility-maximizers and, thus, will only commit to group activities if their individual net profit is expected to be non-negative.

When evaluating the potential cost of participating in a proposed group activity, an agent must consider the context of its existing commitments. For example, the fact that I'm already planning to drive to the airport has a dramatic impact on my estimate of the cost of giving you a ride. Horty and Pollack (2001) initiated research into how an individual agent can evaluate new opportunities for single-agent action in the context of its pre-existing commitments. The ICDP is a generalization of that problem to the group context, which introduces two significant complications. First, no single agent has complete information about the pre-existing commitments of all of the agents in the group; in other words, the 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 single agent.

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 the actions to which it commits.

To decide whether to join a proposed collaboration, an agent needs to assess the potential impact of that collaboration on its ability to do other work. Since the planning, decision-making, and opportunity costs associated with collaborative activity can be substantial, it is preferable for agents to be able to determine some upper bound on that impact prior to committing to the group activity.

In deciding whether to commit to a new group activity, each agent must estimate two factors:

- 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
- 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 of the agents.

The proposal-based mechanism presented in Chapter 2 can be used by agents to establish their initial commitment to some proposed group activity; however, that mechanism requires that some agent be able to generate a proposal that the rest of the agents in the group would be willing to accept. To enable agents to assess the impact of such a proposal on their individual schedules, such a proposal might need to specify the recipe for the activity, a mapping from tasks to agents, and a set of temporal constraints to coordinate the execution of the activity.[1] The generation of such a proposal would require (1) knowledge about the private schedules of the potential participants which may not be available to any single agent, and (2) the ability to perform a potentially intractable computation. As a result, it is not generally practical to use the proposal-based mechanism to solve instances of the ICDP.

In contrast, the ICDP mechanism presented in this chapter uses a combinatorial auction (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999; Rassenti, Smith, and Bulfin, 1982) 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 such bids (i.e., potential contributions). It is based on an existing winner-determination algorithm for combinatorial auctions (Sandholm, 1999) that was modified to enable it to accommodate bids with temporal constraints. Notice that the ICDP mechanism not only establishes a group's commitment, but also generates a potential allocation of tasks to agents that is consistent with the private schedules of the participants.

By distributing the computational burden in this way, the ICDP mechanism allows agents to maintain the privacy of their pre-existing commitments and to focus their computational efforts on their own potential for contributing to the proposed collaboration. Furthermore, being able to condition bids on temporal constraints allows agents to protect the feasibility of their pre-existing commitments. In addition, although the global computation may be carried out centrally, it may also be carried out in a distributed fashion, with agents searching different portions of the highly-structured search space. Finally, to decide whether to commit to a proposed group activity typically does not require finding an optimal task allocation. Agents may decide to commit to an activity based on a provisional allocation of tasks that they may try to improve on later.

3.2 Combinatorial Auctions

In a *combinatorial auction* (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999; Rassenti, Smith, and Bulfin, 1982), there are multiple items for sale; there are participants who may place bids on arbitrary subsets of those items; and there is an auctioneer who must determine which *awardable* combination of bids maximizes revenue. Figure 3.1 shows a combinatorial auction in which there are four items—A, B, C, and D—for sale, and five participants who 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 submitted bids. For each bid $b \in \mathbf{B}$, let $\mathit{Items}(b) \subseteq \mathbf{I}$ 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,

[1] Such an approach would be similar to that of Kinny et al. (1994), as discussed in Chapter 1.

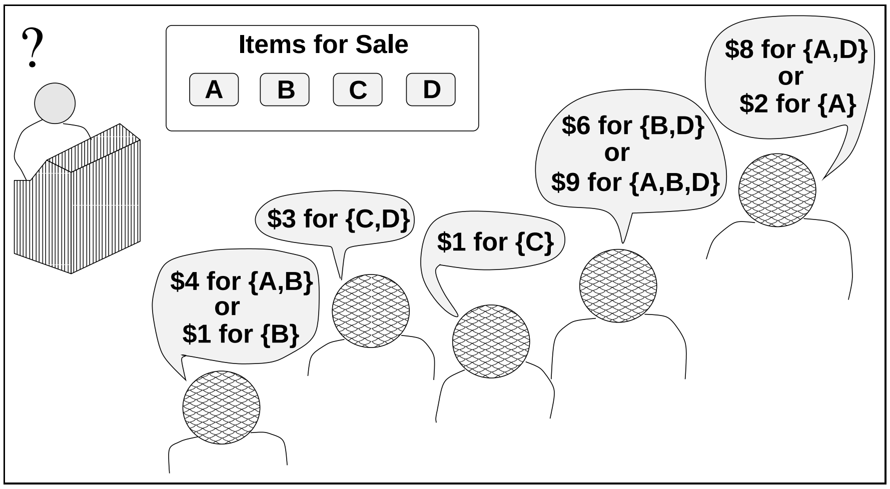


Figure 3.1: A combinatorial auction

- *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

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

The winning bid-set is an awardable bid-set that maximizes revenue:

$$BS^* \quad = \quad \underset{BS \,\in\, \mathcal{ABS}}{argmax} \; Value(BS),$$

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.*

The awardable bid-sets for the auction in Figure 3.1 are shown in Figure 3.2. The last two bid-sets shown in Figure 3.2 yield the maximum revenue (i.e., $\$10$); thus, either may be chosen to be the winning bid-set.

3.2.1 Existing Winner-Determination Algorithms

The general winner-determination (WD) problem for combinatorial auctions is $\mathcal{NP}$-complete (Sandholm, 1999; Fujishima, Leyton-Brown, and Shoham, 1999). However, Sandholm (1999) and Fujishima et al. (1999) have independently presented WD algorithms that scale to problems involving scores of items and thousands of bids. The main insight underlying these algorithms is that, in practice, bids necessarily only sparsely populate the space of possible bids; thus, the search for the winning bid-set may be restricted to the space of bid-sets composed of actual—not possible—bids.

$$\boxed{\$2 \text{ for } \{A\}} + \boxed{\$1 \text{ for } \{B\}} + \boxed{\$3 \text{ for } \{C, D\}} \quad \Rightarrow \quad \$6$$

$$\boxed{\$2 \text{ for } \{A\}} + \boxed{\$6 \text{ for } \{B, D\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \quad \$9$$

$$\boxed{\$4 \text{ for } \{A, B\}} + \boxed{\$3 \text{ for } \{C, D\}} \quad \Rightarrow \quad \$7$$

$$\boxed{\$9 \text{ for } \{A, B, D\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \quad \$10$$

$$\boxed{\$8 \text{ for } \{A, D\}} + \boxed{\$1 \text{ for } \{B\}} + \boxed{\$1 \text{ for } \{C\}} \quad \Rightarrow \quad \$10$$

Figure 3.2: The awardable bid-sets for the auction in Figure 3.1.

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.[2] 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. This chapter does not address these improvements because each sacrifices optimality in auctions involving temporal constraints and, in the current application, as described below, agents use temporal constraints in their bids to protect the feasibility of their private schedules of pre-existing commitments.

3.3 A Modified Combinatorial Auction for the ICDP

The ICDP arises when a group of agents encounter an opportunity for collaborative activity and each agent must decide whether to participate in that activity.

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

Definition 3.1 (Opportunity for Group Activity) *An* opportunity *for a complex, group activity is a specification of the following: (1) a complex act-type representing the group activity; (2) a set of temporal constraints governing the activity; and (3) a payment that a group would receive for doing the activity.*

In the presentation below, the following notation is used:

- $\alpha(Opp)$ denotes the act-type of the opportunity Opp;
- $\mathcal{C}(Opp)$ denotes the temporal constraints on time-points associated with the opportunity Opp; and
- $P(Opp)$ denotes the payment associated with the opportunity Opp.

Given an opportunity Opp, if a group of agents can find some way of doing the activity $\alpha(Opp)$, at a cost less than $P(Opp)$, while satisfying the temporal constraints in $\mathcal{C}(Opp)$, then they could profit from doing it. This section presents a modified combinatorial auction, called an ICDP auction, that agents can use to determine whether doing the activity specified in a given opportunity is feasible for them.[3] In an ICDP auction, agents bid on sets of *roles* in a proposed group activity.[4] Our use of roles is described in Section 3.3.1.

A novel feature of the ICDP auction is that agents are allowed to condition their bids on temporal constraints. Allowing temporal constraints in bids is essential in this application since agents must be able to protect the feasibility of their private schedules of pre-existing commitments without having to reveal the details of those commitments to other agents. Winner determination in the ICDP auction is handled by an algorithm based on Sandholm's WD algorithm, but modified to accommodate temporal constraints in bids, as described in Section 3.3.5.

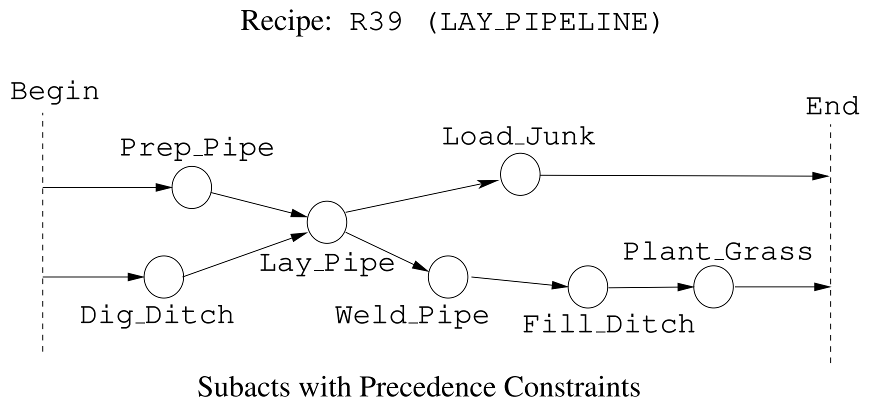
3.3.1 Roles

A role is a special type of parameter that is filled by an agent or subgroup of agents. The agent or subgroup filling a role is responsible for doing the subacts associated with that role. This chapter restricts attention to roles filled by single agents.

The value of clustering subacts into 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

[3] The same type of auction may also be used to solve task-allocation problems when a group is already committed to some activity.

[4] As will be seen, having agents bid on roles, rather than subacts, reduces the number of items up for bid and, thus, enables the auction to scale to problems involving larger numbers of subacts. It may also reduce the computational burden of bid generation, since once an agent discovers that it is unable to carry out one of the subacts covered by some role, it may immediately move on to considering other roles instead.



Additional Constraints

$$\begin{aligned}
Agent(\texttt{Prep_Pipe}) &= Agent(\texttt{Weld_Pipe}) \\
Agent(\texttt{Lay_Pipe}) &= Agent(\texttt{Load_Junk}) \\
Agent(\texttt{Dig_Ditch}) &= Agent(\texttt{Fill_Ditch})
\end{aligned}$$

Figure 3.3: A sample recipe

may be 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. Beyond specifying the tasks to be covered by the act-type roles, some protocols may introduce additional, protocol-specific roles. For example, one protocol might require an additional monitor role, responsible for carrying out a variety of transaction-monitoring tasks.

This chapter extends the representation of act-types and recipes to include *roles.* The roles associated with an act-type α are denoted by $\texttt{ActTypeRoles}(\alpha)$; the roles associated with a recipe R are denoted by $\texttt{RecipeRoles}(R)$. The recipe must specify the set of subacts covered by each act-type role and each recipe role. Each subact must be covered by exactly one role. The agent filling a role is responsible for doing all of the subacts covered by that role.

Figure 3.3 repeats the sample recipe R39 (for the LAY_PIPELINE act-type) seen previously in Figure 1.1. Figure 3.4 shows that same recipe (and act-type) modified to incorporate roles. The modified LAY_PIPELINE act-type specifies three roles: DIGGER, LOADER and WELDER. For each role, the modified recipe R39 specifies the set of subacts covered by that role. For example, the agent filling the LOADER role is responsible for doing the Lay_Pipe and Load_Junk subacts. The modified recipe also includes an additional, recipe-specific role that arises from this recipe's particular way of subdividing the LAY_PIPELINE activity. The modified recipe specifies the responsibilities of this

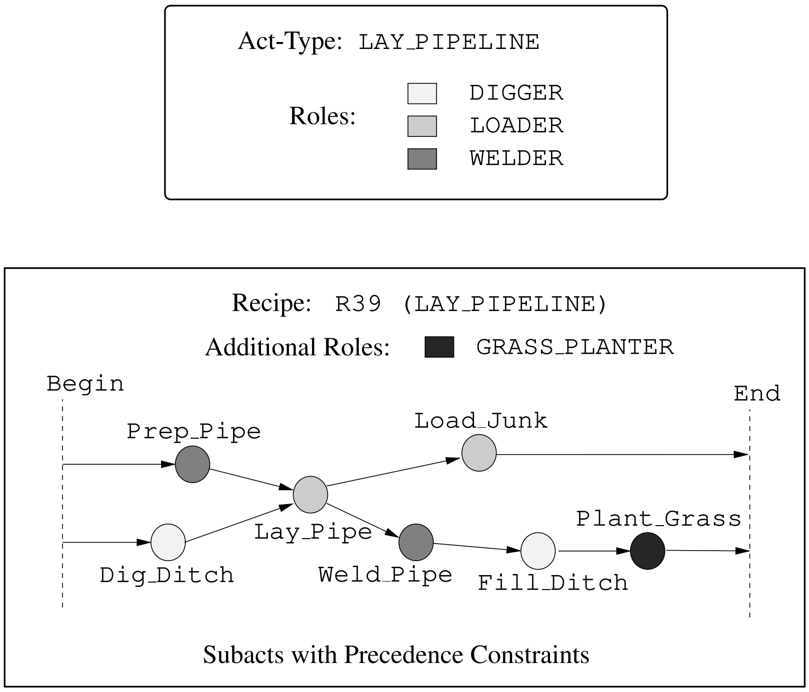


Figure 3.4: Act-type and recipe with roles

additional role: the agent filling the GRASS_PLANTER role is responsible for doing the Plant_Grass subact. Notice that the "additional constraints" in Figure 3.3 are implicit in the role specifications in the modified recipe.

Roles can 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.

3.3.2 Dealing with Multiple Recipes

The items up for bid in an ICDP auction are the roles in the proposed group activity. Since an agent's costs for covering a set of roles may be quite different depending on the choice of recipe, and since each recipe may include a different set of additional, recipe-specific roles, each ICDP auction specifies a particular recipe. If there are multiple recipes available for a given act-type, agents may hold multiple, simultaneous ICDP auctions—possibly using different agents to carry out the auctioneer function for different auctions, thereby distributing the overhead of multiple winner-determination computations.

Roles	{WELDER, GRASS_PLANTER}
Amount	\$300
Temporal Constraints	$\mathit{Begin}(\alpha) \ \geq\ 3:30$ p.m. $\mathit{End}(\alpha) \ \leq\ 7:30$ p.m. $\mathit{End}(\texttt{Prep\_Pipe}) - \mathit{Begin}(\texttt{Weld\_Pipe}) \ \leq\ 2$ hours $\mathit{Duration}(\texttt{Plant\_Grass}) \ \geq\ 1$ hour

Figure 3.5: A sample bid pertaining to the recipe from Figure 3.4

3.3.3 Bids in an ICDP Auction

Figure 3.5 shows a sample bid for an ICDP auction for the LAY_PIPELINE activity using the R39 recipe from Figure 3.4. 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 group activity occur between 3:30 and 7:30 p.m., that the pipe-preparation subact end no more than two hours after the pipe-welding subact begins, and that the duration of the grass-planting subact be at least one hour.

3.3.4 Specification of an ICDP Auction

Items up for Bid. For an auction corresponding to an act-type α and a recipe R, the items up for bid are given by

$$\mathbf{I} \ = \ \texttt{ActTypeRoles}(\alpha) \cup \texttt{RecipeRoles}(R).$$

Bids. Each bid must specify a set of roles, an amount of money, and a set of temporal constraints. For each bid b,

- $\mathit{Roles}(b) \subseteq \mathbf{I}$ denotes the set of roles covered by the bid;
- $\mathit{Amount}(b)$ denotes the amount of the bid;
- $\mathit{TemporalConstraints}(b)$ denotes the temporal constraints in the bid, which must be of the form, $t_j - t_i \leq \delta$, where t_i and t_j are time-points associated with the activity (e.g., the beginning or ending points of subacts) and δ is a real number.[5]

Awardable Bid-Sets. Disjoint and covering bid-sets are defined as for a standard combinatorial auction; however, the awardable bid-sets are defined differently. A bid-set BS is called *awardable* (with respect to recipe R) if, in addition to its being disjoint and covering, there exists a set of execution times for the subacts in R that satisfy all of the temporal

[5]This form of temporal constraint is that found in Simple Temporal Networks (Dechter, Meiri, and Pearl, 1991), as described in detail in Chapter 4.

constraints deriving from (1) the act-type α, (2) the recipe R, (3) the opportunity, and (4) the bids in BS.[6] The *cost* of an awardable bid-set is given by:

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

The Winning Bid-Set. Let $\mathcal{ABS}$ be the set of awardable bid-sets. The winning bid-set is given by

$$BS^* \;=\; \underset{BS \,\in\, \mathcal{ABS}}{argmin}\;\; Cost(BS).$$

The cost of the winning bid-set is: $Cost(BS^*)$.

3.3.5 The Modified WD Algorithm

To determine the winner of a combinatorial auction involving bids that include temporal constraints, we modified Sandholm's basic winner-determination (WD) algorithm (described in Section 3.2.1).[7] The modifications included adding a consistency check to the bid-set construction process and minimizing cost instead of maximizing revenue. The modified WD algorithm follows each depth-first path until one of the following occurs:

(1) the cost of the current bid-set is greater than that of the best bid-set found so far,

(2) the temporal constraints in the current bid-set are inconsistent with the temporal constraints associated with the proposed group activity,

(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

Determining whether the temporal constraints associated with the current set of bids are consistent with the temporal constraints associated with the proposed group activity is handled by keeping track of all of the relevant temporal constraints in a Simple Temporal Network (Dechter, Meiri, and Pearl, 1991). The number of time-points in the network depends only on the number of subacts in the recipe and is thus fixed. The number of temporal constraints is no more than $N(N-1)$, where N is the number of time-points. Whenever a new bid is appended to the current bid-set, the temporal constraints from that bid are added

[6] This satisfiability condition is easily represented using Simple Temporal Networks (Dechter, Meiri, and Pearl, 1991), as described in detail in Chapters 4 and 5.

[7] Sandholm's algorithm was chosen primarily because the bid-tree structure used in that algorithm ensures non-redundant search.

to the network and their effects propagated throughout the network. As will be seen in the next chapter, detecting the consistency of a Simple Temporal Network is easy. If the constraints from the extended bid-set do not make the network inconsistent, the search may continue; otherwise, the search must backtrack.

Experimental Evaluation of the Modified WD Algorithm

This section evaluates various characteristics of the performance of the 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 temporal constraints on the group activity. The inclusion of temporal constraints in bids results in fewer consistent bid combinations, which tends to increase the cost of an optimal solution (i.e., a least-cost awardable bid-set); in extreme cases, temporal constraints might result in there being no solution at all. This section also examines how the number of bids affects the relationship between the temporal constraints in bids and the likelihood of their yielding a solution.

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 separately for each experiment). The cost of a bid was randomly generated such that the cost-per-subact was uniformly distributed between 10 and 20.

The experiments described in this section included only unary temporal constraints in bids. In addition, the recipes and bids for each run were randomly generated. Although the resulting distributions may be different from those encountered in the context of actual multi-agent activity, they were sufficient for the purposes of these experiments. Finally, since the experiments were intended only as a pilot study to demonstrate the feasibility of the approach and to illustrate various performance tradeoffs, the implementation of the winner-determination algorithm was not optimized.

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 (Sandholm, 1999), the performance of the algorithm was expected 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 3.6. Each point in the plot shows the time (averaged over 40 runs)[8] required to reach a solution of a particular *relative cost*, where the relative cost of a solution is given by

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

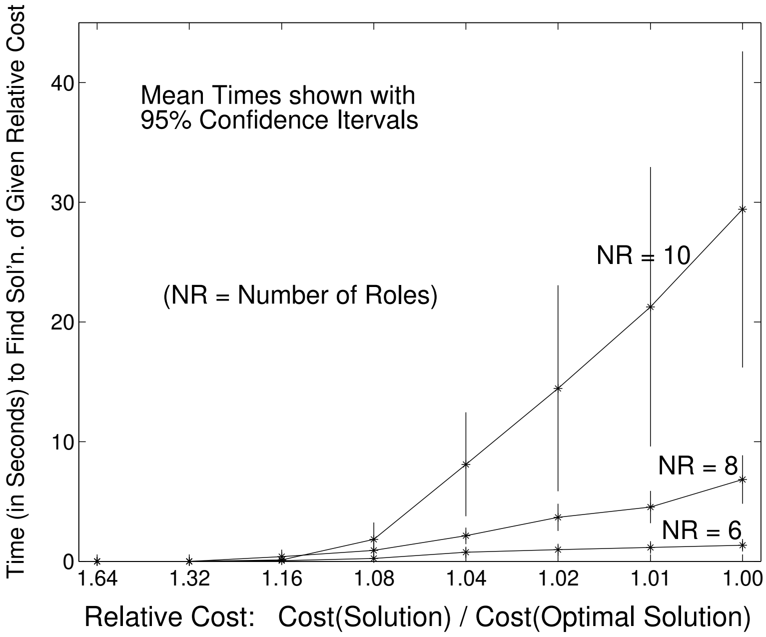


Figure 3.6: Varying the number of roles (NR)

$$Cost(\text{Solution}) \;/\; Cost(\text{Optimal Solution}).$$

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 temporal 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.

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 temporal constraints; a value of 1 represents that the bid constrained—with both lower and upper bounds—the execution

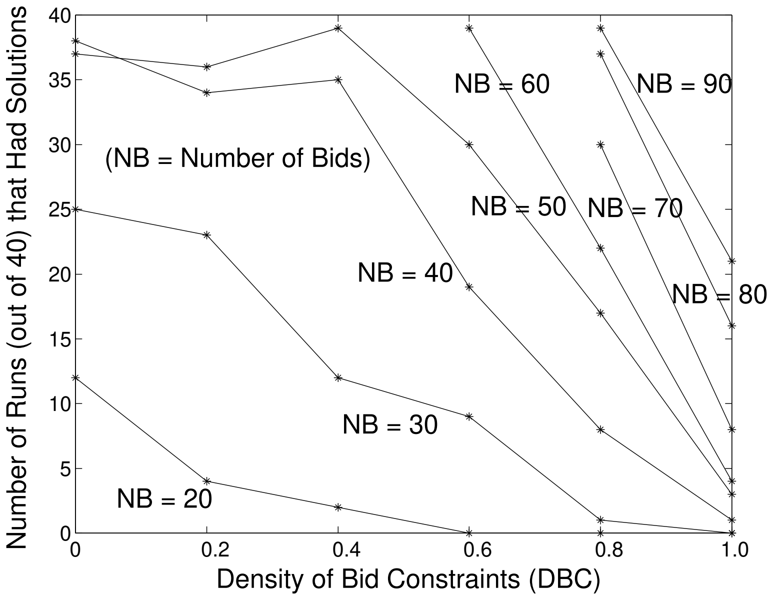


Figure 3.7: Varying the number of bids and the density of bid constraints

time of each subact covered by that bid's roles.[9] Each data point indicates the number of runs (out of 40) that yielded a solution. The results are plotted in Figure 3.7. 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 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

[9] Temporal constraints in bids were determined as follows. First, in a random order, execution times consistent with the recipe's temporal 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 greatest lower bound (glb) and least upper bound (lub), and the effects of this assignment were propagated through the temporal network (see Chapter 4). Second, for each subact S, temporal constraints (simulating interactions with the agent's schedule of pre-existing commitments) of the form $Time(S) \leq T_s + 10$ and $Time(S) \geq T_s - 10$ were generated. Each temporal constraint was included in the bid with probability equal to the DBC.

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 3.8. 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 for the ICDP. As mentioned previously, the ICDP requires only that agents determine, both individually and collectively, whether to commit to some proposed group activity; it does not require agents to commit to a particular allocation of tasks. Thus, when running an ICDP auction, agents typically do not need to find an *optimal* allocation of tasks; they need only find a *satisficing* allocation of tasks (Simon, 1969). After committing to the group activity, they may run additional auctions with the purpose of finding better task allocations.

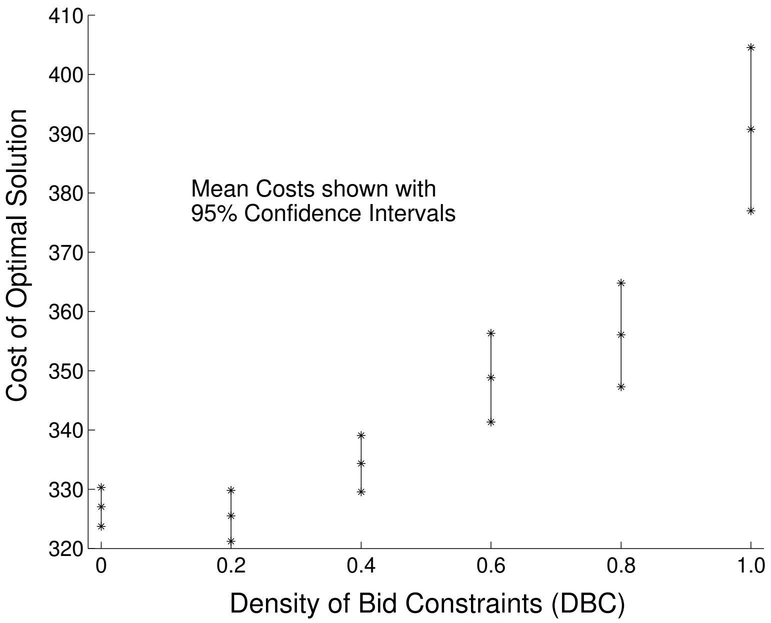
In view of these considerations, the results of these experiments suggest possible strategies that agents might employ when participating in an ICDP auction. For example, the individual contexts of the agents might be such that each agent could choose between making minimally-constrained, higher-cost 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, 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 3.6) 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.

3.4 The ICDP Group Decision-Making Mechanism

The ICDP auction described above is an example of a group decision-making mechanism. This section specifies the ICDP mechanism according to the GDMM framework presented in Chapter 2. The specification of the ICDP mechanism is similar to the specification of the proposal-based mechanism in the case of context-free proposals.

Specifying the ICDP mechanism according to the GDMM framework illustrates how that framework unifies group decision-making mechanisms for collaborative activity. Chapter 2 ended by listing several general problems that agents must solve to participate effectively in any group decision-making mechanism specified according to the GDMM frame-

(a) COST OF OPTIMAL SOLUTION



(b) TIME TO FIND OPTIMAL SOLUTION AND TO EXHAUST THE SEARCH SPACE

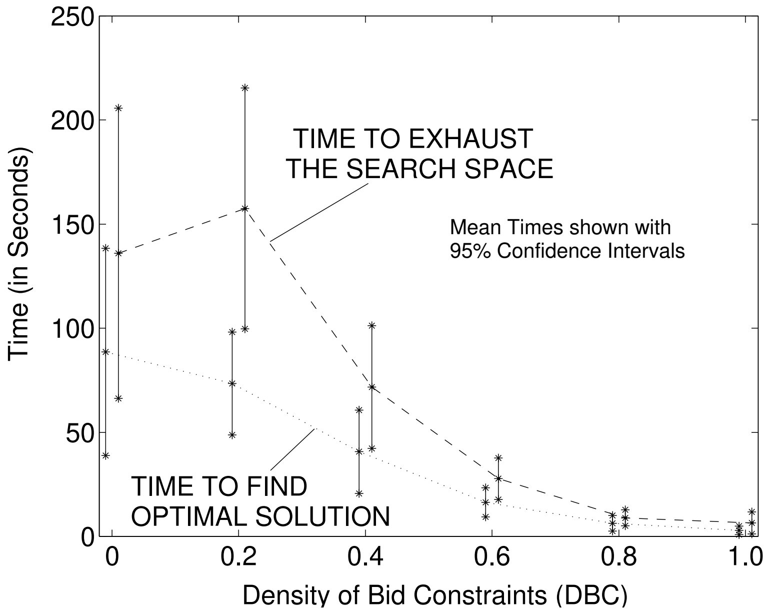


Figure 3.8: Plots from Experiment 3

work. This chapter ends by listing the more specific versions of those problems that are applicable to the ICDP mechanism. For example, agents must be able to generate temporal constraints for their bids to protect their private schedules of pre-existing commitments. In addition, they must be able to coordinate their post-auction activity, for example, by adding temporal constraints sufficient to decouple the tasks being done by different agents. Algorithms for solving the temporal-reasoning problems arising from the ICDP mechanism are presented in Chapter 5.

The specification of a group decision-making mechanism includes the classes of allowable content for the declarative speech-acts used in the mechanism and, for each such class, the conditions under which an agent is authorized to make declarations in that class.

Allowable Content of Declarations

There are three classes of allowable content for declarations in the ICDP mechanism:

- **Auction Invocation**: an agent proposes that the group hold an auction concerning a proposed group activity, using a specified recipe.
- **Bid**: an agent submits a bid to cover some of the roles in the group activity for a specified payment and subject to a specified set of temporal constraints.
- **Auction Results Announcement**: an agent announces the results of an auction that it invoked. The announcement may specify an awardable bid-set, if one exists, or the empty set, indicating no solution.

The declarations have the following forms, respectively:

- $\delta(G_1, GR, InvokedAuction(G_1, GR, I, Opp, R))$

  An agent G_1 proposes to the group GR that they hold an auction concerning the opportunity Opp (cf. Definition 3.1) using the recipe R. Should the group decide to commit to the proposed activity, the identifier I will be used for the group-related intentions; it may also be used to identify the auction.

- $\delta(G_2, G_1, Bid(G_2, G_1, GR, I, Opp, R, B))$

  An agent G_2 submits a bid B (cf. Section 3.3.4) to agent G_1 in the auction identified by I.

- $\delta(G_1, GR, AnnAucSoln(G_1, GR, I, Opp, R, BS))$

  An agent G_1 declares the results of the auction identified by I. BS is either a set of bids or the empty-set.

Authorizing Conditions

The authorizing conditions for the three classes of declarative content for the ICDP mechanism are given below. Some of the conditions refer to metric temporal constraints. Since the logic DDLTLB (cf. Section 2.1.4) does not handle metric temporal constraints, no attempt is made to codify the various temporal consistency requirements in DDLTLB. Instead, the following predicates are used, each listed with its intended interpretation:

- $\mathit{WellFormedAuction}(\mathit{Opp}, R)$ represents that the opportunity Opp is well-formed and compatible with the recipe R. In particular, $\mathit{WellFormedAuction}(\mathit{Opp}, R)$ holds only if $\alpha(\mathit{Opp})$ is a complex act-type, $P(\mathit{Opp})$ is an amount of money, $\mathcal{C}(\mathit{Opp})$ is a set of temporal constraints on the time-points defined by the act-type α, and R is a recipe for actions of type $\alpha(\mathit{Opp})$, such that the temporal constraints in the act-type α, the set $\mathcal{C}(\mathit{Opp})$, and the recipe R are mutually satisfiable.

- $\mathit{Awardable}(BS, \mathit{Opp}, R)$ represents that the bid-set BS is awardable (i.e., BS is disjoint and covering, and there exist execution times for the subacts in the recipe R that satisfy all of the temporal constraints deriving from the act-type $\alpha(\mathit{Opp})$, the recipe R, the opportunity Opp, and the bids in BS).

One of the desired properties of the auction mechanism is that no more than one announcement of an auction solution be made by the agent that invoked the auction. Thus, the authorization conditions for an *announce-auction-solution* declaration must stipulate that no previous announcements were made for that auction. Such a condition is naturally expressed using existential quantification; however, the logic DDLTLB does not provide existential quantification.

To avoid the need for existential quantification in the authorizing conditions, a special predicate $\mathit{AnnSomeAucSoln}$ (for "announce some auction solution") is introduced. To achieve the desired effect, it is assumed that whenever a more-specific $\mathit{AnnAucSoln}$ declaration is made, a less-specific $\mathit{AnnSomeAucSoln}$ declaration is simultaneously made, as represented by the following axiom:

$$\begin{aligned}\models\ & \mathit{declared}(G_1, GR, \mathit{AnnAucSoln}(G_1, GR, I, \mathit{Opp}, R, BS)) \\ & \Rightarrow\ \mathit{declared}(G_1, GR, \mathit{AnnSomeAucSoln}(G_1, GR, I, \mathit{Opp}, R))\end{aligned}.$$

Notice that the more-specific $\mathit{AnnAucSoln}$ declaration contains a variable BS that does not appear in the less-specific $\mathit{AnnSomeAucSoln}$ declaration.

- **Invoke an Auction**: Any member of the group is authorized to invoke an ICDP auction as long as it uses a recipe compatible with the well-formed opportunity.

$$\begin{aligned}\models\ & \mathit{auth}(G_1, GR, \delta(G_1, GR, \mathit{InvokedAuction}(G_1, GR, I, \mathit{Opp}, R))) \\ & \Leftrightarrow\ (G_1 \in GR)\ \wedge\ \mathit{WellFormedAuction}(\mathit{Opp}, R).\end{aligned}$$

- **Submit a Bid**: Any member of the group is authorized to submit a bid for an auction as long as the bid is well-formed and the agent that invoked the auction has not already announced the auction results.

$$\models \ auth(G, G_1, \delta(G, G_1, Bid(G, G_1, GR, I, Opp, R, B))) \\ \Leftrightarrow \begin{cases} \Diamond^{\leftarrow} InvokedAuction(G_1, GR, I, Opp, R) \\ \wedge \neg \Diamond^{\leftarrow} \ declared(G_1, GR, AnnSomeAucSoln(G_1, GR, I, Opp, R)) \end{cases}$$

- **Announce Auction Results**: The agent that invoked an auction is authorized to announce the results of that auction at most once. In the case of an announcement of no auction solution, there are no additional conditions; in the case of an announcement of an auction solution, the bid-set BS must awardable. The case of no auction solution is indicated by the bid-set BS being empty.

$$\models \ auth(G, G_1, \delta(G, G_1, AnnAucSoln(G_1, GR, I, Opp, R, BS))) \\ \Leftrightarrow \begin{cases} \neg \Diamond^{\leftarrow} \ declared(G_1, GR, AnnSomeAucSoln(G_1, GR, I, Opp, R)) \\ \wedge ((BS = \emptyset) \vee Awardable(BS, Opp, R)). \end{cases}$$

When the auction results in a solution, and thus the agents decide to commit to the proposed activity, there are two possibilities: they can decide to commit to the task allocation and temporal constraints determined by the bids in the bid-set BS or not. Should they decide *not* to commit to the task allocation and temporal constraints determined by the bids, they will need to make task-allocation decisions later.

So far, the formal specification of the auction mechanism in the case of a successful auction is merely to establish that the special predicate $AnnAucSoln(G_1, GR, I, Opp, R, BS)$ holds for some awardable bid-set BS. It does not yet establish any obligations to adopt intentions toward the proposed activity.

If the desired result is that the agents commit to the new group activity *without* committing to the task allocation and temporal constraints determined by the bid-set BS, then an axiom of the form

$$\models \quad (BS \neq \emptyset) \ \wedge \ authDecl(G_1, GR, AnnAucSoln(G_1, GR, I, Opp, R, BS)) \\ \Rightarrow \ authDecl(G_1, GR, CCR^+(GR, I, \alpha(Opp), \Upsilon(Opp, R)))$$

suffices, where CCR^+ is as defined in Section 2.2.4, $\alpha(Opp)$ is the act-type associated with the opportunity, and $\Upsilon(Opp, R)$ is an abbreviation for the following set of intention-update operations: to select the recipe R for the group activity, and to add the temporal constraints in the set $\mathcal{C}(Opp)$.

On the other hand, if the desired result is that the agents also commit to the task allocation and temporal constraints determined by the bids in the set BS, then the above axiom need only change the set of intention-update operations to include the task allocation and temporal constraints corresponding to the bids in BS, which may be abbreviated by $\Upsilon(Opp, R, BS)$.

By formally specifying the ICDP mechanism according to the GDMM framework, properties of the ICDP mechanism may be formally analyzed using techniques similar to those used in the formal analysis of the proposal-based mechanism in Chapter 2. In addition, the special $Awardable$ predicate provides a clean interface between the DDLTLB logic and the algorithms based on Simple Temporal Networks provided in Chapter 5. Ultimately, it would be desirable for agents to be able to reason about metric temporal constraints within a logic like DDLTLB; however, providing such expressivity to DDLTLB is beyond the scope of this thesis.

Chapter 2 presented a straightforward set of modifications to the context-free version of the proposal-based mechanism to enable it to accommodate context-bound proposals. A similar set of modifications may be applied to the auction-based mechanism to enable it to be used as a task-allocation mechanism in the context of an activity to which a group is already committed. In that case, the result of a successful auction would be to establish the group's obligation to do the intention-update operations corresponding to the task allocation and temporal constraints determined by the awarded bids.

3.5 Participating Effectively in the ICDP Mechanism

To participate in the combinatorial auction-based ICDP mechanism, an agent must be able to construct bids for doing new tasks that are compatible with its existing commitments. To do so requires solving the following temporal reasoning problems:

- **Temporal-Constraint-Generation (TCG) Problem**: For any set of tasks that the agent might bid on, it must be able to generate a set of temporal constraints to protect its private schedule of pre-existing commitments in case the bid is ultimately awarded.

- **Auction-Participation-Context-Generation Problem**: To keep bid-generation computations manageable, an agent should restrict the portion of its schedule of pre-existing commitments that it uses as the basis for its bid-generation computations. That portion of its schedule, which must be temporally decoupled from the rest of its schedule, is called the agent's Auction-Participation Context (APC).

Furthermore, since there may be temporal dependencies among tasks awarded to different agents, a group of agents may need to coordinate their post-auction activity, a problem this thesis refers to as the Post-Auction Coordination (PAC) Problem. This thesis identifies several strategies agents can use to solve the Post-Auction Coordination Problem: (1) they can add new temporal constraints to ensure that the tasks being done by different agents are decoupled; (2) they can add a weaker set of temporal constraints to decouple the tasks being done by *some* of the agents; or (3) they can add constraints sufficient to form a hierarchy of temporal dependence relationships that gives the greatest flexibility to the agents that need it the most.

Algorithms that agents can use to solve each of these problems are provided in Chapter 5.

3.6 Related Work

The modified combinatorial auction described in this chapter was first presented in a conference paper (Hunsberger and Grosz, 2000). This section describes related work on winner-determination algorithms in combinatorial auctions, auction-based approaches to task allocation, and market-oriented solutions to combinatorial problems, as well as subsequent work on combinatorial auctions with temporal constraints.

Other Approaches to Winner-Determination in Combinatorial Auctions.

Fujishima et al. (1999) present a *Virtual Simultaneous Auction* as an alternative to their winner-determination algorithm for a standard combinatorial auction (described earlier, cf. Section 3.2.1). For each original bid, they create a virtual bidder that tries to secure the items in that bid. Their experimental results showed that this approach was competitive with their basic algorithm, although not quite as fast.

Andersson, Tenhunen and Ygge (2000) argue that formulating the winner-determination problem for standard combinatorial auctions as an integer-programming problem can lead to competitive algorithms using standard integer-programming packages.

Sandholm et al. (2001) have developed an improved winner-determination algorithm for standard combinatorial auctions. That algorithm, called CABOB, uses numerous tricks, including treating subproblems using integer-programming techniques, to optimize its performance. Determining whether CABOB can be usefully modified to accommodate temporal constraints, as in the WD algorithm for auctions in the ICDP mechanism, is left to future work.

An Auction-based Approach to Task Allocation

In Collins et al. (2000), 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 the approach taken in this chapter 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 construct awardable bid-sets using *pieces* of bids; 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 that might exist among them. In contrast, 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 the approach taken in this chapter restricts the search to the space of feasible bid-sets.

Market-Oriented Solutions to Combinatorial Problems

Walsh and Wellman (1998), Kutanoglu and Wu (1997), and Fujishima et al. (1999) present market-oriented solutions to a range of combinatorial problems. Each approach involves iterative auctions. Walsh and Wellman (1998) 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, Walsh and Wellman 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. The approach taken in this chapter handles temporal constraints without such an adverse computational impact.

Walsh, Wellman and Ygge (2000) subsequently investigated the use of a standard combinatorial auction in a multi-level supply-chain problem in which various agents submit bids either to produce or to consume various goods. The goal of the auction-based approach was to find a set of producer and consumer agents willing to form a supply-chain in which a specified set of goods could be produced. They found that the auction-based approach invariably leads to optimal allocations when agents bid their true valuations, but that strategic bidding by producer agents, which can be quite beneficial to them, can cause the auction-based approach to fail to find solutions in some cases. Their supply-chain problem did not consider temporal constraints.

Kutanoglu and Wu (1997) use an iterative-auction approach (but with only a single auction) to solve a distributed resource-scheduling problem in which a set of jobs must be performed, each job 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 the approach taken in this chapter 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 the ICDP mechanism uses 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 the biddable items in an auction in the ICDP mechanism are roles, of which there are comparatively few.

Auctions with Temporal Constraints

Parkes and Ungar (2001) investigate the use of multiple, iterative, combinatorial auctions in a train-scheduling domain. In their work, the train-scheduling domain comprises (1) separate networks of train tracks (called *territories*), each controlled by "an autonomous dispatch agent responsible for the flow of trains [through that] territory"; and (2) a set of independent trains, each "represented by a self-interested agent that bids for the right to travel across the [global] network from its source to destination, submitting bids to multiple dispatch agents along its route as necessary."

In their decentralized approach to the scheduling problem, each dispatch agent runs an iterative combinatorial auction in which train agents bid for the right to travel across the territory controlled by that dispatch agent. Each train agent has pre-assigned entry and exit points for each territory. The global scheduling problem is to determine entry and exit *times* that will satisfy safety requirements (e.g., that trains will not collide) while maximizing "the total cost-adjusted value over all trains."

Each bid submitted by a train agent is allowed to specify unary temporal constraints of the form, 'enter before time T' or 'exit before time T'. Train agents may submit multiple bids for a given auction, but will be awarded at most one bid per auction. The auction for each territory is iterative to enable agents to coordinate their bids across several auctions.

In a suite of experiments on scenarios involving "linear chains of dispatcher territories", Parkes and Ungar found that their decentralized, auction-based approach generated better schedules than a comparable centralized approach. In addition, the winner-determination algorithm for the auction-based approach appeared to scale well "with the number of train agents and in particular with the number of dispatchers."

The approach taken by Parkes and Ungar is similar to the approach taken in this chapter in that it allows bids in a combinatorial auction to include (unary) temporal constraints. Their formulation of the winner-determination problem as a mixed-integer programming problem (using "the big M technique") could provide an alternative means of formulating the winner-determination problem for the ICDP mechanism described in this chapter. The bids submitted by train agents may be viewed as covering a *single, fixed* item (rather than an arbitrary set of items) subject to at most two unary temporal constraints, whereas in the ICDP mechanism, agents bid on arbitrary *sets* of roles, and may include numerous binary temporal constraints among time-points associated with the proposed group activity.