Designing a Family of Coordination Algorithms *

Keith S. Decker and Victor R. Lesser
Department of Computer Science
University of Massachusetts
DECKER@CS.UMASS.EDU

Abstract

Many researchers have shown that there is no single best organization or coordination mechanism for all environments. This paper discusses the design and implementation of an extendable family of coordination mechanisms, called Generalized Partial Global Planning (GPGP). The set of coordination mechanisms described here assists in scheduling activities for teams of cooperative computational agents. The GPGP approach has several unique features. First, it is not tied to a single domain. Each mechanism is defined as a response to certain features in the current task environment. We show that different combinations of mechanisms are appropriate for different task environments. Secondly, the approach works in conjunction with an agent's existing local planner/scheduler. Finally, the initial set of five mechanisms presented here generalizes and extends the Partial Global Planning (PGP) algorithm. In comparison to PGP, GPGP considers tasks with deadlines, it allows agent heterogeneity, it exchanges less global information, and it communicates at multiple levels of abstraction.

Introduction

This paper presents a formal description of the implementation of a domain independent coordination scheduling approach which we call Generalized Partial Global Planning (GPGP). The GPGP approach consists of an extendable set of modular coordination mechanisms, any subset or all of which can be used in response to a particular task environment. Each mechanism is defined using our formal framework for expressing coordination problems (TÆMS (Decker & Lesser 1993b)). GPGP both generalizes and extends the Partial Global Planning (PGP) algorithm (Durfee & Lesser 1991). Our approach has several unique features: GPGP works in conjunction with an existing agent architecture and local scheduler[1]; each coordination mechanism is defined as a response to certain features in the current task environment; the individual coordination mechanisms rest on a shared substrate that arbitrates between the mechanisms and the agent's local scheduler in a decision-theoretic manner.

The GPGP approach views coordination as *modulating* local control, not replacing it. This process occurs via a set of domain-independent coordination mechanisms that post constraints to the local scheduler about the importance of certain tasks and appropriate times for their initiation and completion. By concentrating on the creation of local scheduling constraints, we avoid the sequentiality of scheduling in the original PGP algorithm that occurs when there are multiple plans. By having separate modules for coordination and local scheduling, we can also take advantage of advances in real-time scheduling to produce cooperative distributed problem solving systems that respond to real-time deadlines. We can also take advantage of local schedulers that have a great deal of domain scheduling knowledge already encoded within them. Finally, our approach allows consideration of termination issues that were glossed over in the PGP work (where termination was handled by an external oracle).

Besides the obvious connections to the earlier PGP work, GPGP builds on work by von Martial (v. Martial 1992) in detecting and reacting to relationships (such as von Martial's "favor" relationship). GPGP also uses a notion of social commitments similar to those discussed by (Cohen & Levesque 1990; Shoham 1991; Castelfranchi 1993; Jennings 1993). Durfee's newer work (Durfee & Montgomery 1991) is based on a hierarchical behavior space representation that like GPGP allows agents to communicate at multiple levels of detail. The mechanisms presented in this paper deal with coordination while agents are scheduling (locating in time) their activities rather than while they are planning to meet goals. This allows them to be used in distributed scheduling systems, agenda-based systems (like blackboard systems), or systems where agents instantiate previous plans (like case-based planning systems). The focus on mechanisms for coordinating schedules is thus different from work that focuses on multi-agent planning (Grosz & Kraus 1993; Ephrati & Rosenschein 1994). Shoham and Tennenholtz's 'social laws' approach (Shoham & Tennenholtz 1992) can be viewed as one which tries to change the (perceived) structure of the tasks by, for example, restricting the agents' possible activities. Intelligent agents might use all of these approaches at one time or another.

*This work was supported by DARPA contract N00014-92-J-1698, Office of Naval Research contract N00014-92-J-1450, and NSF contract IRI-9321324. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

[1]Here, a "design-to-time" real-time scheduler (Garvey & Lesser 1993).

The next section will briefly re-introduce our framework for representing coordination problems, and summarize the assumptions we make about an agent's internal architecture. We then describe the GPGP substrate and five coordination mechanisms.[2] Previous work has shown how the GPGP approach can duplicate and extend the behaviors of the PGP algorithm (Decker & Lesser 1992); this paper will briefly summarize several experimental results that are reported in (Decker 1995) and concludes with a look at our future directions.

Representing The Task Environment

Coordination is the process of managing interdependencies between activities (Malone & Crowston 1991). If we view an agent as an entity that has some beliefs about the world and can perform actions, then the coordination problem arises when any or all of the following situations occur: the agent has a *choice* of actions it can take, and that choice affects the agent's performance; the *order* in which actions are carried out affects performance; the *time* at which actions are carried out affects performance. The coordination problem of choosing and temporally ordering actions is made more complex because the agent may only have an incomplete view of the entire task structure of which its actions are a part, the task structure may be changing dynamically, and the agent may be uncertain about the outcomes of its actions. If there are multiple agents in an environment, then when the potential actions of one agent are related to those of another agent, we call the relationship a *coordination relationship*. Each GPGP coordination mechanism is a response to some coordination relationship.

The TÆMS framework (Task Analysis, Environment Modeling, and Simulation) (Decker & Lesser 1993b) represents coordination problems in a formal, domain-independent way. We have used it to represent coordination problems in distributed sensor networks, hospital patient scheduling, airport resource management, distributed information retrieval, pilot's associate, local area network diagnosis, etc. (Decker 1995). In this paper we will describe an agent's current subjective beliefs about the structure of the problem it is trying to solve by using the TÆMS framework (Decker & Lesser 1993b; Decker 1995). For this purpose, there are two unique features of TÆMS. The first is the explicit, quantitative representation of task interrelationships as functions that describe the effect of activity choices and temporal orderings on performance. The second is the representation of task structures at multiple levels of abstraction. The highest level of abstraction is called a *task group*, and contains all tasks that have explicit computational interrelationships. A *task* is simply a set of lower-level subtasks and/or executable methods. The components of a task have an explicitly defined effect on the quality of the encompassing task. The lowest level of abstraction is called an executable *method*. An executable *method* represents a schedulable entity, such as a blackboard knowledge source instance, a chunk of code and its input data, or a totally-ordered plan that has been recalled and instantiated for a task. A method could also be an instance of a human activity at some useful level of detail, for example, "take an X-ray of patient 1's left foot".

A coordination problem instance (called an *episode* $\mathbf{E}$) is defined as a set of task groups, each with a deadline $\mathrm{D}(\mathcal{T})$, such as $\mathbf{E} = \langle \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n \rangle$. A common performance goal of the agent or agents is to maximize the sum of the quality achieved for each task group before its deadline.

Figure 1 shows an objective[3] task group and agent A's subjective view of that same task group. The arrows between tasks and/or methods indicate task interrelationships where the execution of some method will have a positive or negative effect on the quality or duration of another method. The presence of these interrelationships make this an NP-hard scheduling problem; further complicating factors for the local scheduler include the fact that multiple agents are executing related methods, that some methods are redundant (executable at more than one agent), and that the subjective task structure may differ from the real objective structure.

The Agent Architecture

We make few assumptions about the architecture of the agents. The agents have a database that holds their current beliefs about the structure of the tasks in the current episode; we represent this information using TÆMS. The agents can do three types of actions: they can execute methods from the task structure, send direct messages to one another, and do "information gathering". Information gathering actions model how new task structures or communications get into the agent's belief database. This could be a combination of external actions (checking the agent's incoming message box) and internal planning. Method execution actions cause quality to accrue in a task group (as indicated by the task structure). Communication actions are used to send the results of method executions (which in turn may trigger the effects of various task interrelationships) or meta-level information.

Formally, we write $B'_A(x)$ to mean agent A subjectively believes x at time t (Shoham 1991). We will shorten this to $B(x)$ when the particular agent or time is obvious. An agent's subjective beliefs about the current episode include the agent's beliefs about task groups, subtasks, executable methods, and interrelationships (e.g., $B(\mathcal{T}_i \in \mathbf{E})$. $B(T_a, M_b \in \mathcal{T}_i)$. $B(\mathsf{enables}(T_a, M_b))$).

The GPGP family of coordination mechanisms also makes a stronger assumption about the agent architecture. It assumes the presence of a local scheduling mechanism (to be described in the next section) that can decide what method execution actions should take place and when. The local scheduler attempts to maximize a (possibly changing) utility function. The current set of GPGP coordination mechanisms are for cooperative teams of agents—they assume that agents do not intentionally lie and that agents believe what they are told. However, because agents can believe and communicate only subjective information, they may unwittingly transmit information that is inconsistent with an objective view (this can cause, among other things, the phenomena of *distraction*). Finally, the GPGP family approach requires domain-dependent

[2] These five mechanisms are oriented towards producing PGP-like 'cooperative team' behavior. Mechanisms for self-interested agents are also possible.

[3] The word 'objective' refers to the fact that this is the true, real structure.

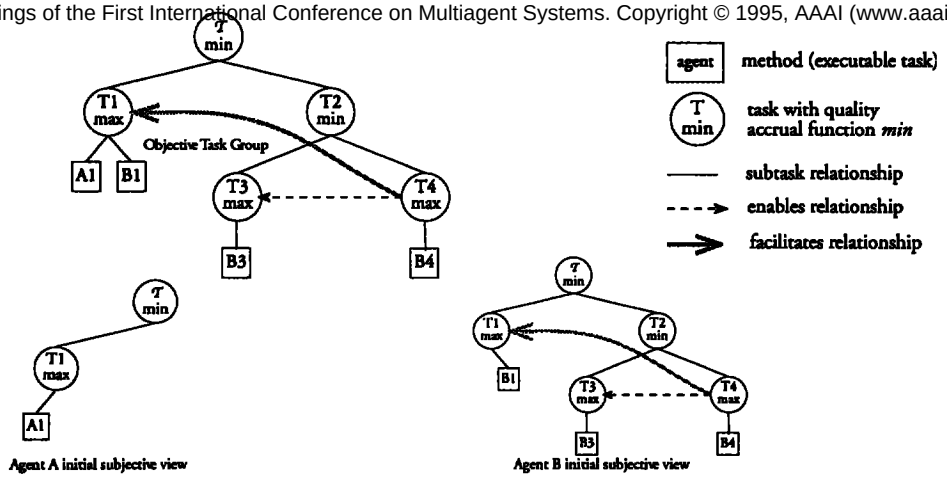

Figure 1: Agent A and B's subjective views (bottom) of a typical objective task group (top)

code to detect or predict the presence of coordination relationships in the local task structure. In this paper we will refer to that domain-dependent code as the information gathering action called *detect-coordination-relationships*; we will describe this action more in the section on Mechanism 1: Non-Local Views.

The Local Scheduler

Each GPGP agent contains a local scheduler that takes three types of input information and produces a set of schedules and alternatives. The first input is the current, subjectively believed task structure. Using information about the potential duration, potential quality, and interrelationships, the local scheduler chooses and orders executable methods in an attempt to maximize a pre-defined utility function. In this paper the utility function is the sum of the task group qualities $\sum_{T \in \mathbf{E}} Q(T, \mathrm{D}(T))$, where $Q(T,t)$ denotes the quality of T at time t as defined in (Decker & Lesser 1993b). Quality does not accrue after a task group's deadline.

The second input is a set of *commitments* $\mathbf{C}$. These commitments are produced by the GPGP coordination mechanisms, and act as extra constraints on the schedules that are produced by the local scheduler. For example, if method 1 is executable by agent A and method 2 is executable by agent B, and the methods are redundant, then one of agent A's coordination mechanisms may *commit* agent A to do method 1. Commitments are *social*—directed to particular agents in the sense of the work of Castelfranchi and Shoham (Castelfranchi 1993; Shoham 1991)). A local commitment C by agent A becomes a non-local commitment when received by another agent B. This paper will use two types of commitments: $C(\mathrm{Do}(T,q))$ is a commitment to 'do' (achieve quality for) T and is satisfied at the time t when $Q(T,t) \geq q$; the second type $C(\mathrm{DL}(T,q,t_{dl}))$ is a 'deadline' commitment to do T by time t_{dl} and is satisfied at the time t when $[Q(T,t) \geq q] \wedge [t \leq t_{dl}]$. When a commitment is sent to another agent, it also implies that the task result will be communicated to the other agent (by the deadline, if it is a deadline commitment).

The third input to the local scheduler is the set of non-local commitments $\mathbf{NLC}$ made by other agents. This information can be used by the local scheduler to coordinate actions between agents. For example the local scheduler could have the property that, if method M_1 is executable by agent A and is the only method that enables method M_2 at agent B (and agent B knows this), and $B_A(C(\mathrm{DL}(M_1,q,t_1))) \in B_B(\mathbf{NLC})$, then for every schedule S produced by agent B, $\langle M_2, t \rangle \in S \Rightarrow t \geq t_1$ (in other words, agent B only schedules the enabled method after the deadline that agent A has committed to.

A schedule S produced by a local scheduler will consist of a set of methods and start times: $S = \{\langle M_1,t_1\rangle, \langle M_2,t_2\rangle, \ldots, \langle M_n,t_n\rangle\}$. The schedule may include idle time, and the local scheduler may produce more than one schedule upon each invocation in the situation where not all commitments can be met. The different schedules represent different ways of partially satisfying the set of commitments. The function Violated(S) returns the set of commitments that are believed to be violated by the schedule. For violated deadline commitments $C(\mathrm{DL}(T,q,t_{dl})) \in \mathrm{Violated}(S)$ the function $\mathrm{Alt}(C,S)$ returns an alternative commitment $C(\mathrm{DL}(T,q,t^*_{dl}))$ where $t^*_{dl} = \min t$ such that $Q(T,t) \geq q$ if such a t exists, or NIL otherwise. For a violated Do commitment an alternative may contain a lower minimum quality, or no alternative may be possible. The function $U_{\mathrm{est}}(\mathbf{E}, S, \mathbf{NLC})$ returns the estimated utility at the end of the episode if the agent follows schedule S and all non-local commitments in $\mathbf{NLC}$ are kept.

Thus we may define the local scheduler as a function $\mathrm{LS}(\mathbf{E}, \mathbf{C}, \mathbf{NLC})$ returning a set of schedules $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$. More detailed information about this kind of interface between the local scheduler and the coordination component may be found in (Garvey, Decker, & Lesser 1994). This is an extremely general definition of the local scheduler, and is the minimal one necessary for the GPGP coordination module. Stronger definitions than this will be needed for more predictable performance, as we will discuss

later. Ideally, the optimal local scheduler would find both the schedule with maximum utility $\hat{S}_U$ and the schedule with maximum utility that violates no commitments $\hat{S}_{\bar{V}}$. In practice, however, a heuristic local scheduler will produce a set of schedules where the schedule of highest utility S_U is not necessarily optimal: $\mathrm{U}(\mathbf{E}, S_U, \mathbf{NLC}) \leq \mathrm{U}(\mathbf{E}, \hat{S}_U, \mathbf{NLC})$.

Five GPGP Coordination Mechanisms

The role of the coordination mechanisms is to provide information to the local scheduler that allows the local scheduler to construct better schedules. This information can be in the form of modifications to portions of the subjective task structure of the episode or in the form of local and non-local commitments to tasks in the task structure. The five mechanisms we will describe in this paper form a basic set that provides similar functionality to the original Partial Global Planning algorithm as shown in (Decker & Lesser 1992). Mechanism 1 exchanges useful private views of task structures; Mechanism 2 communicates results; Mechanism 3 handles redundant methods; Mechanisms 4 and 5 handle hard and soft coordination relationships. More mechanisms can be added, such as one to update utilities across agents as discussed in the next section, or to balance the load better between agents. The mechanisms are independent in the sense that they can be used in any combination. If inconsistent constraints are introduced, the local scheduler will return at least one violated constraint in all its schedules. Since the local scheduler typically satisfices instead of optimizes, it may do this even if constraints are not inconsistent (i.e. it does not search exhaustively). The next section describes how a schedule is chosen by the coordination module substrate.

The GPGP Coordination Module Substrate

All the specific coordination mechanisms rest on a common substrate that handles information gathering actions, invoking the local scheduler, choosing a schedule to execute (including dealing with violated or inconsistent commitments), and deciding when to terminate processing on a task group. Information gathering actions include noticing new task group arrivals and receiving communications from other agents. Information gathering is done at the start of problem solving, when communications are expected from other agents, and when the agent is otherwise idle. Communications are expected in response to certain events (such as after the arrival of a new task group) or as indicated in the set of non-local commitments $\mathbf{NLC}$. This is the minimal general information gathering policy. Termination of processing on a task group occurs for an agent when the agent is idle, has no expected communications, and no outstanding commitments for the task group.

Choosing a schedule is more complicated. The agent's local scheduler may return multiple schedules because it cannot find a single schedule that both maximizes utility and meets all commitments. From the set of schedules $\mathbf{S}$ returned by the local scheduler, two particular schedules are identified: the schedule with the highest utility S_U and the best committed schedule S_C. If they are the same, then that schedule is chosen. Otherwise, we examine the sum of the changes in utility for each commitment. Each commitment, when created, is assigned the estimated utility U_{est} for the task group of which it is a part. This utility may be updated over time (when other agents depend on the commitment, for example). We then choose the schedule with the largest positive change in utility. This allows us to abandon commitments if doing so will result in higher overall utility. The coordination substrate does not use the local scheduler's utility estimate U_{est} directly on the entire schedule because it is based only on a local view. The coordination substrate may receive non-local information that places a higher utility on a commitment than it has locally.

If both schedules have the same utility, the one that is more negotiable is chosen. Every commitment has a negotiability index (high, medium, or low) that indicates (heuristically) the difficulty in rescheduling if the commitment is broken. This index is set by the individual coordination mechanisms. For example, hard coordination relationships like enables that cannot be ignored will trigger commitments with low negotiability. If the schedules are still equivalent, the shorter one is chosen, and if they are the same length, one is chosen at random.

After a schedule S is chosen, if $\mathrm{Violated}(S)$ is not empty, then each commitment $C \in \mathrm{Violated}(S)$ is replaced with its alternative $\mathbf{C} \leftarrow \mathbf{C} \setminus C \cup \mathrm{Alt}(C, S)$. If the commitment was made to other agents, the other agents are also informed of the change in the commitment. While this could potentially cause cascading changes in the schedules of multiple agents, it generally does not for three reasons: first, as we mentioned in the previous paragraph less important commitments are broken first; secondly, the resiliancy of the local schedulers to solve problems in multiple ways tends to damp out these fluctuations; and third, agents are time cognizant resource-bounded reasoners that interleave execution and scheduling (i.e., the agents cannot spend all day arguing over scheduling details and still meet their deadlines). We have observed this useful phenomenon before (Decker 1995) and plan to analyze it in future work.

Mechanism 1: Updating Non-Local Viewpoints

Remember that each agent has only a partial, subjective view of the current episode. The GPGP mechanism described here can communicate no private information ('none' policy, no non-local view), or all of it ('all' policy, global view), or take an intermediate approach ('some' policy, partial view). The process of detecting coordination relationships between private and shared parts of a task structure is in general very domain specific, so we model this process by a new information gathering action, *detect-coordination-relationships*, that takes some fixed amount of the agent's time. This action is scheduled whenever a new task group arrives.

The set $\mathbf{P}$ of privately believed tasks or methods at an agent A (tasks believed at arrival time by A only) is then $\{x \mid \mathit{task}(x) \wedge \forall a \in \mathbf{A} \setminus A,\ \neg B_A(B_a^{\mathrm{Ar}(x)}(x))\}$, where $\mathbf{A}$ is the set of all agents and $\mathrm{Ar}(x)$ is the arrival time of x. Given this definition, the action *detect-coordination-relationships* returns the set of private coordination relationships $\mathbf{PCR} = \{r \mid T_1 \in \mathbf{P} \wedge T_2 \notin \mathbf{P} \wedge [r(T_1, T_2) \vee r(T_2, T_1)]\}$ between

private and mutually believed tasks. The action does not return what the task T_2 is, just that a relationship exists between T_1 and some otherwise unknown task T_2. For example, in the DVMT, we have used the physical organization of agents to detect that Agent A's task T_1 in an overlapping sensor area is in fact related to some unknown task T_2 at agent B (i.e. $B_A(B_B(T_2))$) (Decker & Lesser 1992). The non-local view coordination mechanism then communicates these coordination relationships, the private tasks, and their context: if $r(T_1, T_2) \in \mathbf{PCR}$ and $T_1 \in \mathbf{P}$ then r and T_1 will be communicated by agent A to the set of agents $\{a \mid B_A(B_a(T_2))\}$.

For example, Figure 2 shows the local subjective beliefs of agents A and B after the communication from one another due to this mechanism. We'll denote the complete set of coordination relationships as **CR**; this includes all the elements of **PCR** and all the relationships between non-private tasks.

Mechanism 2: Communicating Results

The result communication coordination mechanism has three possible policies: communicate only the results necessary to satisfy commitments to other agents (the minimal policy); communicate this information plus the final results associated with a task group ('TG' policy), and communicate all results ('all' policy[4]). Extra result communications are broadcast to all agents, the minimal commitment-satisfying communications are sent only to those agents to whom the commitment was made (i.e., communicate the result of T to the set of agents $\{A \in \mathbf{A} \mid B(B_A(C(T))\}$.

Mechanism 3: Handling Simple Redundancy

Potential redundancy in the efforts of multiple agents can occur in several places in a task structure. Any task that uses a 'max' quality accumulation function (one possible semantics for an 'OR' node) indicates that, in the absence of other relationships, only one subtask needs to be done. When such subtasks are complex and involve many agents, the coordination of these agents to avoid redundant processing can also be complex; we will not address the general redundancy avoidance problem in this paper (see instead (Lesser 1991)). In the original PGP algorithm and domain (distributed sensor interpretation), the primary form of potential redundancy was simple method redundancy—the same result could be derived from the data from any of a number of sensors. The coordination mechanism described here is meant to address this simpler form of potential redundancy.

The idea behind the simple redundancy coordination mechanism is that when more than one agent wants to execute a redundant method, one agent is randomly chosen to execute it and send the results to the other interested agents. This is a generalization of the 'static' organization algorithm discussed by Decker and Lesser (Decker & Lesser 1993a)—it does not try to load balance, and uses one communication action (because in the general case the agents do not know beforehand, without communication, that certain methods are redundant[5]). The mechanism considers the set of potential redundancies $\mathbf{RCR} = \{r \in \mathbf{CR} \mid r = \mathsf{subtask}(T, \mathbf{M}, \min)] \wedge [\forall M \in \mathbf{M}, \mathit{method}(M)]\}$. Then for all methods in the current schedule S at time t, if the method is potentially redundant then commit to it and send the commitment to Others(**M**) (non-local agents who also have a method in **M**):

$$[\langle M, t_M \rangle \in S] \wedge$$
$$[\mathsf{subtask}(T, \mathbf{M}, \min) \in \mathbf{RCR}] \wedge [M \in \mathbf{M}] \Rightarrow$$
$$[C(\mathrm{Do}(M, \mathrm{Q}_{\mathrm{est}}(M, \mathrm{D}(M), S))) \in \mathbf{C}] \wedge$$
$$[\mathsf{comm}(M, \mathrm{Others}(\mathbf{M}), t) \in \mathcal{I}]$$

See for example the top of figure 3—both agents commit to Do their methods for T_1.

After the commitment is made, the agent must refrain from executing the method in question if possible until any non-local commitments that were made simultaneously can arrive (the communication delay time δ). This mechanism then watches for multiple commitments in the redundant set and if they appear, a unique agent is chosen randomly (but identically by all agents) from those with the best commitments to keep its commitment. All the other agents can retract their commitments. For example the bottom of figure 3 shows the situation after Agent B has retracted its commitment to Do B_1. If all agents follow the same algorithm, and communication channels are assumed to be reliable, then no second message (retraction) actually needs to be sent (because they all choose the same agent to do the redundant method). In the implementation described later, identical random choices are made by giving each method a unique random identifier, and then all agents choose the method with the 'smallest' identifier for execution.

Initially, all Do commitments initiated by the redundant coordination mechanism are marked highly negotiable. When a redundant commitment is discovered, the negotiability of the remaining commitment is lowered to medium to indicate the commitment is somewhat more important.

Mechanism 4: Handling Hard Coordination Relationships

Hard coordination relationships include relationships like $\mathsf{enables}(M_1, M_2)$ that indicate that M_1 must be executed before M_2 in order to obtain quality for M_2. Like redundant methods, hard coordination relationships can be culled from the set **CR**. The hard coordination mechanism further distinguishes the direction of the relationship—the current implementation only creates commitments on the predecessors of the enables relationship. We'll let $\mathbf{HPCR} \subset \mathbf{CR}$ indicate the set of potential hard predecessor coordination relationships. The hard coordination mechanism then looks for situations where the current schedule S at time t will produce quality for a predecessor in **HPCR**, and commits to its execution by a certain deadline both locally and socially:

$$[\mathrm{Q}_{\mathrm{est}}(T, \mathrm{D}(T), S) > 0] \wedge$$
$$[\mathsf{enables}(T, \mathbf{M}) \in \mathbf{HPCR}] \Rightarrow$$
$$[C(\mathrm{DL}(T, \mathrm{Q}_{\mathrm{est}}(T, \mathrm{D}(T), S), t_{\mathrm{early}})) \in \mathbf{C}] \wedge$$
$$[\mathsf{comm}(C, \mathrm{Others}(\mathbf{M}), t) \in \mathcal{I}]$$

[4] Such a policy is all that is needed in many simple environments.

[5] The detection of redundant methods is domain-dependent, as discussed earlier. Since we are talking here about simple, direct redundancy (i.e. doing the exact same method at more than one agent) this detection is very straight-forward.

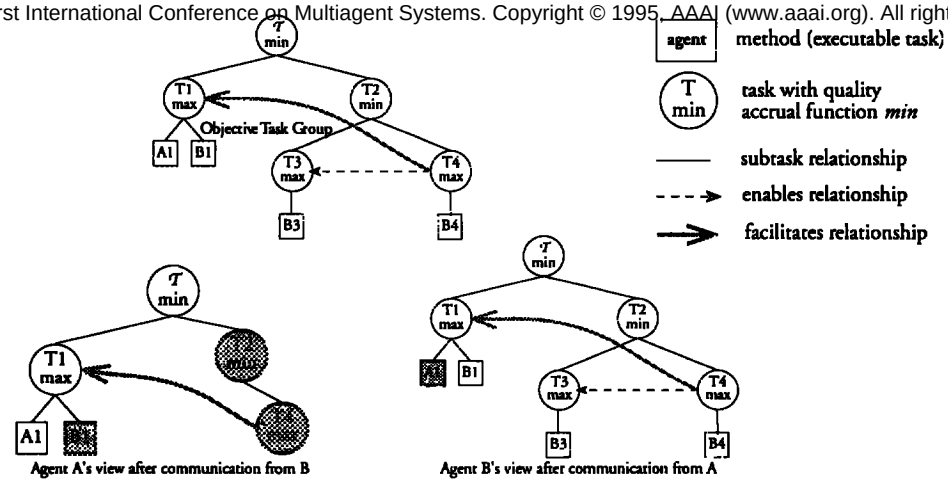

Figure 2: Agents A and B's local views after receiving non-local viewpoint communications via mechanism 1 (shaded objects). Figure 1 shows the agents' initial states.

The next question is, by what time (t_{early} above) do we commit to providing the answer? One solution, usable with any local scheduler that fits our general description of a Local Scheduler, is to use the min t such that $\mathrm{Q_{est}}(T, \mathrm{D}(T), S) > 0$. In our implementation, the local scheduler provides a query facility that allows us to propose a commitment to satisfy as 'early' as possible (thus allowing the agent on the other end of the relationship more slack). We take advantage of this ability in the hard coordination mechanism by adding the new commitment $C(\mathrm{DL}(T, \mathrm{Q_{est}}(T, \mathrm{D}(T), S), \text{"early"}))$ to the local commitment set $\mathbf{C}$, and invoking the local scheduler $LS(\mathbf{E}, \mathbf{C}, \mathbf{NLC})$ to produce a new set of schedules $\mathbf{S}$. If the preferred, highest utility schedule $S_U \in \mathbf{S}$ has no violations (highly likely since the local scheduler can simply return the same schedule if no better one can be found), we replace the current schedule with it and use the new schedule, with a potentially earlier finish time for T, to provide a value for t_{early}. The new completed commitment is entered locally (with low negotiability) and sent to the subset of interested other agents.

If redundant commitments are made to the same task, the earliest commitment made by any agent is kept, then the agent committing to the highest quality, and any remaining ties are broken by the same method as before.

Currently, the hard coordination mechanism is a pro-active mechanism, providing information that might be used by other agents to them, while not putting the individual agent to any extra effort. Other future coordination mechanisms might be added to the family that are reactive and request from other agents that certain tasks be done by certain times; this is quite different behavior that would need to be analyzed separately.

Mechanism 5: Handling Soft Coordination Relationships

Soft coordination relationships are handled analogously to hard coordination relationships except that they start out with high negotiability. In the current implementation the predecessor of a facilitates relationship is the only one that triggers commitments across agents, although hinders relationships are also present. The positive relationship $\mathrm{facilitates}(M_1, M_2, \phi_d, \phi_q)$ indicates that executing M_1 before M_2 decreases the duration of M_2 by a 'power' factor related to ϕ_d and increases the maximum quality possible by a power factor related to ϕ_q (Decker & Lesser 1993b). A more situation-specific version of this coordination mechanism might ignore relationships with very low power. Figure 3 shows Agent B making a D commitment to do method B_4, which in turn allows Agent A to take advantage of the $\mathrm{facilitates}(T_4, T1, 0.5, 0.5)$ relationship, causing method A_1 to take only half the time and produce 1.5 times the quality.

Overhead

The primary sources of overhead associated with the coordination mechanisms include action executions (communication and information gathering), calls to the local scheduler, and any algorithmic overhead associated with the mechanism itself. Table 1 summarizes the total amount of overhead from each source for each coordination mechanism setting and the coordination substrate. L represents the length of processing (time before termination), and d is a general density measure of coordination relationships. Interactions between the presence of coordination mechanisms and these quantities include: the number of methods or tasks in $\mathbf{E}$, which depends on the non-local view mechanism; the number of coordination relationships $|\mathbf{CR}|$ or the subsets $\mathbf{RCR}$ (redundant coordination relationships), $\mathbf{HPCR}$ (hard predecessor coordination relationships), $\mathbf{SPCR}$ (soft predecessor coordination relationships), which depends on the number of tasks and methods as well; and the number of commitments $|\mathbf{C}|$, which depends on each of the three mechanisms that makes commitments.

Conclusions, Status, and Future Work

This paper discusses the design of an extendable family of coordination mechanisms, called Generalized Partial Global

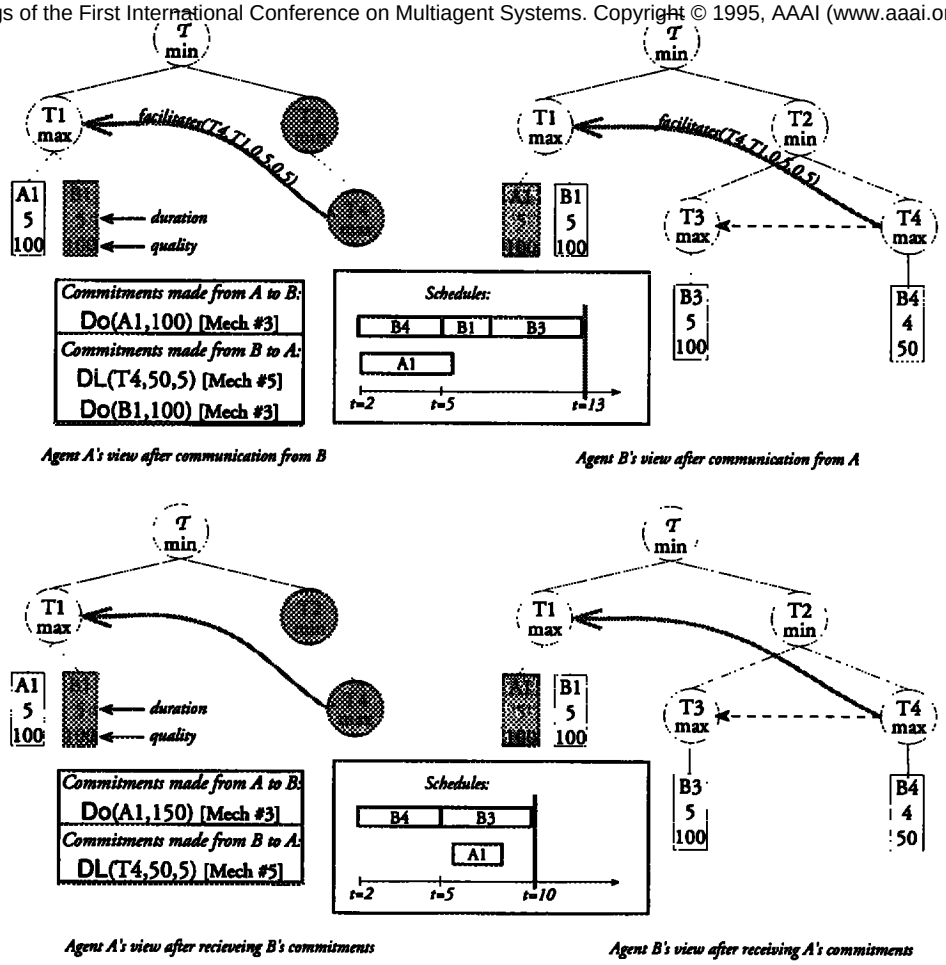

Figure 3: A continuation of Figures 1 and 2. At top: agents A and B propose certain commitments to one another via mechanisms 3 and 5. At bottom: after receiving the initial commitments, mechanism 3 removes agent B's redundant commitment.

Planning (GPGP), that form a basic set of schedule coordination mechanisms for teams of cooperative computational agents. An important feature of this approach includes an extendable set of modular coordination mechanisms, any subset or all of which can be used in response to a particular task environment. This subset may be parameterized, and the parameterization does not have to be chosen statically, but can instead be chosen on a task-group-by-task-group basis or even in response to a particular problem-solving situation. For example, Mechanism 5 (Handle Soft Predecessor CRs) might be "on" for certain classes of tasks and 'off' for other classes (that usually have few or very weak soft CRs). The general specification of the GPGP mechanisms involves the detection and response to certain abstract *coordination relationships* in the incoming task structure that were not tied to a particular domain—we have used TÆMS to describe tasks as diverse as distributed sensor networks, hospital scheduling, and Internet information gathering. A careful separation of the coordination mechanisms from an agent's local scheduler allows each to better do the job for which it was designed. We believe this separation is not only useful for applying our coordination mechanisms to problems with existing, customized local schedulers, but also to problems involving humans (where the coordination mechanism can act as an interface to the person, suggesting possible commitments for the person's consideration and reporting non-local commitments made by others) (Decker & Lesser 1995).

The GPGP coordination approach as described in this paper has been fully implemented in the TÆMS simulation testbed. Significant experimental validation of the GPGP approach is documented in (Decker 1995). These experiments have demonstrated the effective performance of the current mechanisms and the range of behaviors possible. We demonstrate that in complex task environments agents that use the appropriate mechanisms perform better than agents that do not for several performance measures. We show how to test that a particular mechanism is useful. We show that different combinations of mechanisms are, in fact, needed in different environments. We describe a way to structure the search for a particular combination of mechanisms in a particular envi-

Mechanism *setting*	Communications	Information Gathering	Scheduler	*Other Overhead*
substrate	0	$\mathbf{E}$+*idle*	L	$O(L\mathbf{C})$
nlv *none*	0	0	0	0
some	$O(d\mathbf{P})$	$\mathbf{E}$*detect-CRs*	0	$O(T \in \mathbf{E})$
all	$O(\mathbf{P})$	$\mathbf{E}$*detect-CRs*	0	$O(T \in \mathbf{E})$
comm *min*	$O(\mathbf{C})$	0	0	$O(\mathbf{C})$
TG	$O(\mathbf{C}+\mathbf{E})$	0	0	$O(\mathbf{C}+\mathbf{E})$
all	$O(M \in \mathbf{E})$	0	0	$O(M \in \mathbf{E})$
redundant *on*	$O(\mathbf{RCR})$	0	0	$O(\mathbf{RCR} * S + \mathbf{CR})$
hard *on*	$O(\mathbf{HPCR})$	0	$O(\mathbf{HPCR})$	$O(\mathbf{HPCR} * S + \mathbf{CR})$
soft *on*	$O(\mathbf{SPCR})$	0	$O(\mathbf{SPCR})$	$O(\mathbf{SPCR} * S + \mathbf{CR})$

Table 1: Overhead associated with individual mechanisms at each parameter setting

ronment, starting from certain architypical clusters of mechanisms. Our experiments have also shown the value of structuring coordination in terms of a family of algorithms and the ease of adding new mechanisms such as a cooperative load balancing mechanism. The cooperative load-balancing coordination mechanism works by providing better information to the agents with which to resolve redundant commitments.

Eventually we intend to develop a library of reusable coordination mechanisms. For example, mechanisms that work from the successors of hard and soft relationships instead of the predecessors, negotiation mechanisms, mechanisms for behavior such as contracting, or mechanisms that can be used by self-motivated agents in non-cooperative environments. Many of these mechanisms can be built on the existing work of other DAI researchers. Future work will also examine expanding the parameterization of the mechanisms and using machine learning techniques to choose the appropriate parameter values (i.e., learning the best mechanism set for an environment). Finally, we are also beginning work on using the GPGP approach in applications ranging from providing human coordination assistance to distributed information gathering.

References

Castelfranchi, C. 1993. Commitments:from individual intentions to groups and organizations. In Prietula, M., ed., *AI and theories of groups & organizations: Conceptual and Empirical Research.* AAAI Workshop. Working Notes.

Cohen, P. R., and Levesque, H. J. 1990. Intention is choice with commitment. *Artificial Intelligence* 42(3):213–261.

Decker, K. S., and Lesser, V. R. 1992. Generalizing the partial global planning algorithm. *Intnl. Jrnl. of Intelligent and Cooperative Information Systems* 1(2):319–346.

Decker, K. S., and Lesser, V. R. 1993a. An approach to analyzing the need for meta-level communication. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 360–366.

Decker, K. S., and Lesser, V. R. 1993b. Quantitative modeling of complex computational task environments. In *Proc. of the Eleventh National Conference on AI*, 217–224.

Decker, K. S., and Lesser, V. R. 1995. Coordination assistance for mixed human and computational agent systems. UMass Technical Report 95–31.

Decker, K. S. 1995. *Environment Centered Analysis and Design of Coordination Mechanisms.* Ph.D. Dissertation, University of Massachusetts.

Durfee, E., and Lesser, V. 1991. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Trans. on Systems, Man, and Cybernetics* 21(5):1167–1183.

Durfee, E. H., and Montgomery, T. A. 1991. Coordination as distributed search in a hierarchical behavior space. *IEEE Trans. on Systems, Man, and Cybernetics* 21(6):1363–1378.

Ephrati, E., and Rosenschein, J. 1994. Divide and conquer in multi-agent planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 375–380.

Garvey, A., and Lesser, V. 1993. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man, and Cybernetics* 23(6):1491–1502. See also updated UMass TR 95–03.

Garvey, A.; Decker, K.; and Lesser, V. 1994. A negotiation-based interface between a real-time scheduler and a decision-maker. UMass Technical Report 94–08.

Grosz, B., and Kraus, S. 1993. Collaborative plans for group activities. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.*

Jennings, N. R. 1993. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review* 8(3):223–250.

Lesser, V. R. 1991. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6):1347–1363.

Malone, T. W., and Crowston, K. 1991. Toward an interdisciplinary theory of coordination. Center for Coordination Science Technical Report 120, MIT Sloan School.

Shoham, Y., and Tennenholtz, M. 1992. On the synthesis of useful social laws for artificial agent societies (preliminary report). In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 276–281.

Shoham, Y. 1991. AGENT0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 704–709.

v. Martial, F. 1992. *Coordinating Plans of Autonomous Agents.* Berlin: Springer-Verlag. Lecture Notes in AI #610.