
Autonomous Agents and Multi-Agent Sytems, 9, 87–143, 2004

Evolution of the GPGP/TÆMS Domain-Independent Coordination Framework[1,2]

V. LESSER lesser@cs.umass.edu
Department of Computer Science, University of Massachusetts, Amherst, MA 01003 USA

K. DECKER decker@udel.edu
Department of Computer and Information Science, University of Delaware, Newark DE 19716 USA

T. WAGNER wagner_tom@htc.honeywell.com
Honeywell Research Lab, Minneapolis, MN 55418 USA

N. CARVER carver@cs.siu.edu
Department of Computer Science, Southern Illinois University, Carbondale, IL 62901 USA

A. GARVEY agarvey@truman.edu
Department of Computer Science, Truman State University, Kirksville, MO 63501 USA

B. HORLING bhorling@cs.umass.edu
Department of Computer Science, University of Massachusetts, Amherst, MA 01003 USA

D. NEIMAN dneiman@riverlogic.com
River Logic, Inc., Beverly, MA 01915 USA

R. PODOROZHNY podorozh@cs.utexas.edu
Department of Computer Science, University of Massachusetts, Amherst, MA 01003 USA

M. NAGENDRA PRASAD naghi_prasad@peoplesoft.com
PeopleSoft, Pleasanton, CA 94588 USA

A. RAJA anraja@uncc.edu
Department of Computer Science, University of North Carolina, Charlotte, NC 28223 USA

R. VINCENT vincent@ai.sri.com
SRI International, Menlo Park, CA 94025 USA

P. XUAN pxuan@clarku.edu
Department of Mathematics and Computer Science, Clark University, Worcester, MA 01610 USA

X. Q. ZHANG x2zhang@umassd.edu
Department of Computer Science, University of Massachusetts, N. Dartmouth, MA 02747 USA

Abstract. The GPGP/TÆMS domain-independent coordination framework for small agent groups was first described in 1992 and then more fully detailed in an ICMAS'95 paper. In this paper, we discuss the evolution of this framework which has been motivated by its use in a number of applications, including: information gathering and management, intelligent home automation, distributed situation assessment, coordination of concurrent engineering activities, hospital scheduling, travel planning, repair service coordination and supply chain management. First, we review the basic architecture of GPGP and then present extensions to the TÆMS domain-independent representation of agent activities. We next describe extensions to GPGP that permit the representation of situation-specific coordination strategies and social laws as well as making possible the use of GPGP in large agent organizations. Additionally, we discuss a more encompassing view of commitments that takes into account uncertainty in commitments. We then present new coordination mechanisms for use in resource sharing and contracting, and more complex coordination mechanisms that use a cooperative search among agents to find appropriate commitments. We conclude with a summary of the major ideas underpinning GPGP, an analysis of the applicability of the GPGP framework including performance issues, and a discussion of future research directions.

Keywords: multi-agent coordination, teamwork.

1. Introduction

Generalized partial global planning (GPGP) and its associated TÆMS hierarchical task network representation [8, 10, 11, 14, 27] were developed as a domain-independent framework for on-line coordination of the real-time activities of small teams of cooperative agents working to achieve a set of high-level goals. GPGP's development was influenced by two factors: one was to generalize and make domain-independent the coordination techniques developed in the partial global planning (PGP) framework [18];[3] the other was based on viewing agent coordination in terms of coordinating a distributed search of a dynamically evolving goal tree as pictured in Figure 1 [39]. Underlying these two influences was a desire to construct a model that could be used to explain and motivate the reasons for coordination among agents based on a quantitative view of task/subproblem dependency. As discussed in [14] we take a planning-oriented approach to coordination[4] of behaviors among agents that requires three things: specification (creating shared goals), planning (subdividing goals into subgoals/tasks, i.e., creating the substructure of the evolving goal tree) and scheduling (assigning tasks to individual agents or groups of agents, creating shared plans and schedules and allocating resources). GPGP is primarily concerned with scheduling activities rather than the dynamic specification and planning of evolving activities (e.g., such as decomposing a high-level goal into a set of subgoals that, if successfully achieved, will solve the high-level goals).[5] The basic components of the GPGP architecture and how they relate to an agent's problem-solving component can be seen in Figure 14, which will be described later.

The aim of the GPGP coordination framework is to maximize the combined utility accrued by the group of agents as a result of successful completion of its high-level goals. These goals can be independent or held jointly by two or more agents, can be time and resource sensitive, and can be of different utilities. GPGP deals with high-level coordination issues (scheduling and task selection) involving decisions for each agent about what subtasks[6] of which high-level goals it will execute, the level of resources it will expend in their execution, and when they will be executed. These

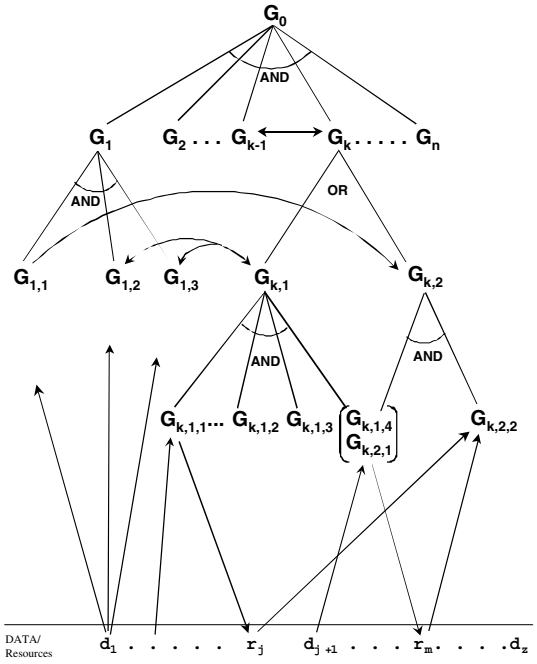

Figure 1. An *AND*/*OR* goal tree that specifies a set of subgoals that need to be solved in order to solve the top-level goal. At the bottom of the figure are raw data and resources that are needed to solve specific subgoals. The figure also represents interdependencies among goals and the fact that one subgoal may contribute to solving more than one higher-level subgoal.

decisions, if not appropriately made, can lead to the group of agents acting incoherently in achieving the desired set of high-level goals. This incoherence can be characterized by agents wasting resources by: (1) doing redundant, unnecessary or less important subtasks, (2) having nothing to do since the needed prerequisite information or external resources that are necessary for subtask execution are not available in a timely fashion, (3) being overloaded while other agents are under-loaded and, finally, (4) missing deadlines on important tasks. The need for these types of scheduling decisions occurs when agents are solving subtasks that are interdependent, either through their contention for scarce resources or through relationships arising among them because they are contributing to the solution of the same higher-level goal. GPGP is thus most appropriate for soft real-time applications operating in dynamic and resource-bounded environments where there are complex interdependencies among activities occurring at different agents. We are also assuming that (1) the application environments appropriate for GPGP will have sufficient communication bandwidth and end-to-end delays to allow for the limited communication of information among agents so that agents can have short dialogues with a limited number of agents, and (2) the time necessary for an agent to do the local planning and scheduling activities associated with GPGP reasoning is sufficiently small, compared to the granularity and deadlines of tasks that need to be coordinated over so that overhead and delays introduced by coordination are acceptable.[7]

This line of research also assumes soft real-time multi-agent applications are best structured using agents that can reason on-line about how to effectively allocate resources to multiple, time and resource sensitive activities of different utilities.[8,9] An alternative approach to this multiplexing among multiple activities in a single agent (and consequently avoiding the requirement for more complex local agent control) is to use agents that have simpler local reasoning capabilities that, for instance, only allow one goal or task to be performed at any given time. In this latter approach, the underlying operating system handles the multiplexing among the different "simple" agents' activities when these agents are hosted on the same processor node. Further, the arrival of new tasks/goals requires instantiating the appropriate "simple" agent on a specific processor node to solve the desired task. However, if there are options for not doing goals or doing them in different ways, depending on resource availability and relative importance of goals, then the former approach will permit an agent to make the decisions based on an appropriate context, while in the latter approach the "simple" agents cannot make the decision unless they get detailed feedback from the underlying operating system about the current set of activities being scheduled by other agents on the processor. Even in this case it may be hard because a "simple" agent does not have the appropriate context to understand how to balance its choice with the choice of other agents. We feel that, for many soft real-time multi-agent applications, a certain class of information is required to reason about how to balance resources among multiple tasks or goals that vie for limited resources (e.g., time). Decomposing the problem space completely to "simple" agents does not address the problem or remove the information and decision requirements. Pushing this decision down to the operating system seems inappropriate since it does not have knowledge of how a specific task/goal can be achieved in alternative ways that trade off lower resource usage for lower quality achieved in task execution.

The type of high-level coordination that we are discussing above is very different from the type of low-level, local and fine-grained coordination that commonly occurs when a protocol is created among agents to synchronize their activities. In frameworks such as COOL [2] and AGENTTALK [36] the emphasis is on the flow of messages and how the dialog between agents is structured. Such frameworks generally combine finite state machines with enhancements, e.g., exceptions, to support a flexible and explicit specification of a communication process. In contrast, in GPGP/TÆMS the focus is on a domain-independent and quantitative evaluation of the interactions among tasks and the dynamic formation of temporal constraints to resolve and to exploit these interactions. GPGP coordination is about exchanging the necessary information for a group of agents to analyze their activities so as to reformulate their local problem solving based on an understanding of the relevant activities of other agents; the purpose of this reformulation is to produce with fewer resources higher-quality solutions from the perspective of the group's problem-solving goals. While GPGP does contain protocols that define the exchange of messages and the flow of the conversation, the emphasis is not on the machinery for specifying the protocols but on the content and timeliness of the information that must be exchanged in order to create group performance that approximates what would be obtained if control of agent activities was centralized and done in an optimal manner. GPGP and work such as COOL relate in that GPGP could literally make use of

COOL or AGENTTALK to structure the flow of information between agents – GPGP would still be dealing with a higher-level optimization problem. Aspects of the control optimization problem considered by GPGP include deadlines and temporal constraints as well as resource interactions. As will be discussed in more detail later, the communication-based joint problem solving performed by GPGP also differs from the coordination that takes places in most BDI agent systems and frameworks such as TEAMCORE/STEAM [51, 62, 63],[10] ARCHON [34][11] and SHARED-PLAN [26], in that GPGP is focused on optimization – how to concurrently solve multiple, time and resource sensitive goals of varying worth within existing resource constraints so as to maximize the overall utility accrued by the agent group.

1.1. Example applications

Two recent applications of the GPGP style of coordination, which have involved dynamic supply chain management [75] and aircraft service team coordination [74], are indicative of the need for this style of coordination. In the dynamic supply chain management application, TÆMS-based agents coordinate a build-to-order production line at a manufacturing plant for a company that manufactures backpacks and outdoor products. All material flow within the network is generated on a build-to-order basis. This means that the agents are able to reduce inventory levels and introduce a new level of flexibility in material control. Note that the focus of the work, as with most GPGP-style applications, is not on establishing market prices for goods or services but on the distributed resolution of temporal or resource interaction between distributed tasks. The domain characteristics that lead to a GPGP solution include: production/consumption operations at different locations that interact (distributed dependency), deadlines on orders, distribution of control (unwillingness to allocate control to centralized party), desire to share only the scheduling information necessary to coordinate over orders (information privacy), desire to support local autonomy at each site (not constrained by other agents/sites unless interacting over a specific good), limited production capabilities (limited resources), varying amounts of time to produce different goods (temporal characteristics), a need to support low inventories (resource considerations), and increased flexibility (more dynamic response – partially remove humans from the loop and require soft real-time solutions).

In the aircraft service team coordination application, one TÆMS-based agent is assigned to each (simulated) service team. The agent's role is to interact with other agents (assigned to other service teams) and to provide scheduling-grade decision support to its assigned team. In this application the tasks of the teams interact in a spatial/temporal fashion, e.g., a given aircraft cannot be refueled while it is being rearmed or while its engines are being repaired. Motivation for a GPGP approach in this application includes: tasks performed by different teams interact in a fashion that requires proper scheduling/sequencing, performance of said interacting tasks is distributed (teams have different functions), support for incremental addition of repair teams, removal of a central point of failure or processing bottleneck, limited resources (access to zones on a given aircraft as well as repair resources, e.g., avionics modules), presence of next-mission launch times (deadlines), no guaranteed ability to

perform all repairs by all required next mission times, different amounts of time required to perform different operations on different aircraft (temporal characteristics), desire to enable each team to exercise local autonomy, peer-to-peer coordination being sufficient – not necessary to centralize and not computationally desirable for large operations.

In summary, both applications exhibit some of the important characteristics that determine when a GPGP-style approach may be relevant – distributed interacting tasks, dynamic situations that change online (making *a priori* computation less useful), a requirement for on-line/soft real-time response, and a need to keep information and control distributed.

1.2. Roadmap of paper

Since completing the first full implementation of the GPGP framework [11, 14], we have significantly extended the framework based on our experience applying it to a number of applications including information gathering and management, intelligent home automation, distributed situation assessment, coordination of concurrent engineering activities, hospital scheduling, travel planning, repair service coordination and supply chain management [12, 30, 42, 43, 45, 49, 50, 52, 74, 75]. Most of these extensions have already been discussed in published papers so the emphasis of this paper is not on providing an in-depth discussion of each extension and empirical results that examine the performance of the extension. Rather, our focus is to present a high-level overview of how this body of work fits together intellectually and how it has been used to create a framework that solves many of the issues necessary for building multi-agent applications that require complex coordination among agents.

We start out by first reviewing the basic concepts behind GPGP and TÆMS, and how the GPGP framework relates to other approaches (Section 2). In that section, we present a detailed example of GPGP and a formalization of parts of the GPGP algorithm. The following two sections discuss the four major areas where the framework has been extended. In Section 3, extensions to the TÆMS domain-independent representation of agent activities are presented. Section 4 is broken into three major subsections. The first (Section 4.1) describes extensions to GPGP that permit the representation of situation-specific coordination strategies and social laws as well as making possible the use of GPGP in large agent organizations. The second (Section 4.2) discusses a more complex view of commitments that takes into account uncertainty in commitments. The third subsection (Section 4.3) explains extensions that are related to new coordination mechanisms for use in resource sharing and contracting, and more complex coordination mechanisms that use a cooperative search among agents to find appropriate commitments, thus changing the coordination optimization from a locally biased view to a more encompassing mutually biased view based on marginal gain and marginal loss between two agents. In Section 5 we conclude with a summary of the major ideas of the paper, an analysis of the applicability of the GPGP framework including performance issues, and a discussion of future research directions. Table 1 provides a high-level description of the key concepts in the framework, some of the limitations in the original framework, and extensions to remedy these limitations.

Table 1. GPGP/TÆMS: Key concepts, original implementation limitations, recent extensions.

Key concepts of GPGP/TÆMS
Coordination as distributed optimization – quantitative view of coordination
Family of coordination mechanisms for situation-specific control
Domain-independent representation of agent tasks using a hierarchical task network
Quantitative coordination relationships among tasks
Multiple goals of varying worth, different deadlines and alternative ways of being solved
Modular interface between local agent control (planning and scheduling) and coordination mechanisms
Major limitations in original GPGP/TÆMS implementation addressed in paper
Only bottom-up coordination mechanisms
Uncertainty in agent behavior not considered in coordination decisions
Locally biased view of optimal coordination
Static situation-specific coordination
No resource-based coordination mechanisms
Coordination mechanisms use single-shot agent dialogue
Major extensions to original GPGP/TÆMS implementation
Integration of top-down and bottom-up coordination mechanisms
Uncertainty in agent behavior partially accounted for in coordination decisions
Dynamic and fine-grained situation-specific coordination
Mutually biased view of optimal coordination based on marginal gain and loss
Resource-based coordination mechanisms
Coordination mechanisms use a multi-step agent dialogue that involves cooperative distributed search

2. The fundamental GPGP approach

The GPGP approach to coordination assumes an operating environment consisting of cooperative agents that are pursuing a number of high-level goals with varying worth (importance) and time deadlines, each of whose achievement requires the (partial) satisfaction of a number of subgoals; these subgoals in turn may have their own time constraints imposed as a result of the time constraints on the higher-level goals that they are helping to achieve. This model of agent computation leads to a situation where each agent may be trying to achieve a number of subgoals, possibly associated with more than one high-level goal. We are also assuming that these subgoals may be achieved with varying levels of quality (satisfaction) which in turn will have an impact on the ultimate worth of the higher-level goal. We do not assume that every high-level goal needs to be achieved in order for the collection of agents as a whole to operate effectively.[12] We are also assuming that in application environments appropriate for GPGP, task granularity will be sufficiently large and deadlines will be appropriately loose so that the resources consumed and delays introduced by coordination activities will be tolerable.[13]

2.1. *Underlying conceptual model*

The underlying model of computation implicit in the GPGP approach to agent coordination can be represented conceptually as an extended AND/OR goal tree where the leaves of the tree are primitive (non-decomposable) activities. For each

goal that the system is solving, there is a tree that represents alternative ways of solving this goal. For example, consider the goal tree for one high-level goal as depicted in Figure 1.

In this representation, subgoals are not only related to each other by the *AND*/*OR* relationship among siblings, but also by constraints and other interdependencies among the goals. Examples of these constraints and interdependencies include: when the result of one subgoal is a precondition for another subgoal, when the result of one subgoal *if available* can improve the performance of another subgoal (for instance, by providing additional constraints), when two subgoal solutions need to be compatible (obey constraints on their solution) since they contribute to the solution of the same higher-level goal, when one subgoal produces a resource (or a state in the world) that is needed by another subgoal, and so forth.

A centralized view of the search control problem associated with solving this goal tree involves choosing, at each point, which subgoal to work on in order to produce the highest quality result in the available time. Now let us look at Figure 2, which is the distribution of this tree between two agents. Note the circles around certain goals and subgoals indicating that both agents are working on the same parts of the goal tree (jointly shared goals). The question now arises as to what type of knowledge about the current state of the problem-solving search in *Agent 1* would be helpful to *Agent 2* in deciding which order to solve its subgoals, and how the decisions among the two agents should be coordinated. For example, should they simultaneously pursue alternative ways of achieving some subgoal (e.g., $G_{k,1}^{1,2}$ and $G_{k,2}^{2}$ solving for $G_{k}^{1,2}$) or should one agent delay executing its subgoal until it gets the results of another subgoal being solved by the other agent (e.g., $G_{k,2}^{2}$ waiting for the result from $G_{1,1}^{1}$). This scheduling problem becomes significantly more complex if the agents are pursuing more than one high-level goal due to the large number of possible combinations. The computational complexity increases further when there are deadlines for the achievement of goals, there are alternative ways of achieving a goal that trade off the utility of the derived solution and the likelihood of successfully achieving a solution against resource usage, or the goals are only partially overlapping among agents. For example, *Agent 1* is pursuing a subtask associated with high-level goals 1 and 2, while *Agent 2* is pursuing a subtask associated with high-level goal 2 and goal 3. We can further complicate this problem by assuming that the goal tree is not *a priori* decomposed among the agents but is either dynamically assigned as the goal tree is being created (a top-down perspective) or the relationship among subgoals is not known *a priori*. In this latter case, the relationship among goals (e.g., G_0^1 and G_0^2 are the same goal) needs to be discovered through agent communication.

In order to operate effectively, decisions need to be made about what subgoals an agent will try to achieve, when it will attempt these goals, and how much effort it will spend on each goal. In essence, we see these control decisions as the output of an optimization problem which is trying to maximize the utility that the group of agents accrue over some time window from the (partial) satisfaction of a number of higher-level goals. As will be discussed later, additional quantitative information beyond what is specified in a classic *AND*/*OR* goal tree is necessary for appropriately formulating this optimization problem, e.g., the time it will take to achieve a subgoal and how the achievement of a subgoal (if there are alternative ways of achieving it)

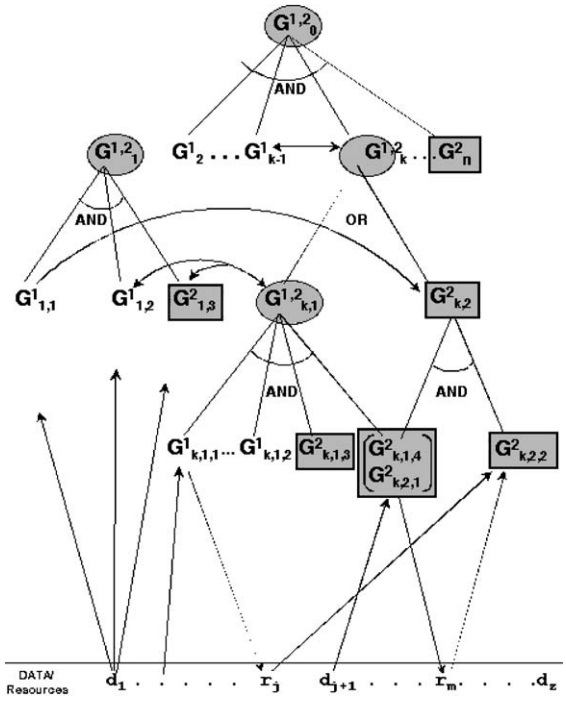

Figure 2. The goal tree pictured in Figure 1 as distributed between two agents. The superscript on a subgoal indicates which agent is responsible for solving that subgoal. A subgoal that is being solved either partly or completely by both agents will have two numbers associated with its superscript. Those subgoals have ovals around them. Subgoals being solved by only *Agent 2* have boxes around them while those being solved only by *Agent 1* have nothing surrounding them.

contributes to the utility accrued by its higher-level goal. Also needed is additional quantitative information and sequencing constraints that specify a more detailed view of the nature of the interdependencies among subgoals, their current degree of satisfaction, whether they are scheduled for execution (i.e., whether they are part of the agent's current plan to satisfy the high-level goal), and deadlines on their completion.

Our approach to this optimization problem, of necessity, needs to be distributed, quantitative and approximate because of limits and delays imposed by the agent communication medium, the uncertainty associated with agent problem-solving behavior, and the cost of decision making due to the combinatorics that result from even a relatively small number of agents and higher-level goals that are being pursued concurrently. For these reasons, GPGP is designed so that each agent optimizes its own local computational activities using optimization criteria defined by local considerations but mediated by both short-term and long-term knowledge of the activities of other agents. This knowledge will be referred to as the agent's *partial global view* (also called the agent's *subjective view* because of its potential incompleteness relative to the true global view that we have referred to in the past as the

objective view [14]). This mediation occurs as a result of GPGP imposing additional constraints on groups of these local optimization problems. These constraints connect local optimization problems that are not independent (i.e., agents who have dependencies among their activities) and represent a way of locally approximating the more global optimization problem involving the group of agents having interdependent activities.

There is a strong connection between the GPGP coordination module that is resident at each agent and a local scheduler/control module that is part of each agent's architecture. The connection between these two is necessary because the generation of suitable constraints must take into account characteristics of the current solution to the local agent optimization problem, e.g., when resources are available, when tasks will be performed, etc., and also the implications of change such as performing an unplanned task at a particular time. In our current work, the agent's local optimization expert is the Design-to-Criteria Scheduler (DTC) [67, 71, 73] or another module built on top of DTC [28], all of which are descended from the Design-to-Time (DTT) [20] scheduler.[14] The scheduler plays a critical role in coordination because the GPGP module may engage in an active search process to create constraints that are suitable to coordinate the activities of the local agent with other agents. During this search process, GPGP may query the DTC scheduler repeatedly to explore the implications of constraints, and their associated temporal/resource bindings, and to obtain feedback from the scheduler about alternative constraints that may be more appropriate [21]. In our work, the local scheduler module must also be more than a conventional scheduler (that makes sequencing decisions) since it is required to carry out planning activities like task selection. In essence, the planner/scheduler must make decisions about which one of a predefined (though potentially quite large) set of alternative approaches specified in the TÆMS task structure should be used to satisfy a goal and if there are multiple goals to solve, which ones should be solved in order to generate the maximum local utility.

The GPGP and DTC modules present in each agent thus combine to guide the agent's activities using knowledge of its own local situation and partial knowledge of the activities being carried out by other agents (i.e., partial global view). Each agent's local GPGP module also coordinates or negotiates with the GPGP modules of other agents to generate constraints on local control that will lead to more coherent agent activities. The resulting local control decisions made by GPGP and DTC are only approximately correct because of the limits on communication and processing resources. First, complete and up-to-date knowledge of all agent activities is not used due to the cost of obtaining and processing the information. Second, not all GPGP modules are involved in the decision process; only those having goals that are directly related to the goals of the agent making the local decision are.[15] Third, the GPGP modules do not engage in a complete/exhaustive negotiation (search) process that is guaranteed to find the constraints to place on local agent control that lead to the most coherent behavior. Finally, because of combinatorics, the local scheduler module is not guaranteed to generate optimal schedules [67, 71, 73].

It is important to emphasize that GPGP uses an incremental and evolving approach to coordination. This also leads to an approximation of an optimal agent

coordination policy since the choice GPGP makes about how to coordinate at one instant does not fully reflect possible changes in the coordination scenario as the result of the execution of scheduled activities nor the arrival of new goals/tasks. It is incremental and evolving, since as the execution of activities results in behaviors that do not fall into expected ranges, there are new tasks/goals that an agent encounters, or there is new or revised information that the agent receives about the state of other agents' activities, the agent will need to re-evaluate its current plans and schedules.[16] This re-evaluation may in turn cause the agent to revise its coordination strategy, which may then cause other agents to revise their plans, schedules and coordination strategies.

This incremental and evolving approach to coordination is in contrast to a distributed MDP optimal policy where all contingencies based on the execution behavior of activities in agents are pre-analyzed and factored into the design of an optimal coordination policy [5, 80, 81]. We do not feel this type of complete pre-planning for every execution contingency is feasible in environments where there is a very large number of possible scenarios over some finite horizon involving how goals are distributed among the agents and when goals become active and need to be completed. In such environments, we do not think it is practical in soft real-time to do the computation necessary to generate an optimal coordination policy so that when a new or revised coordination scenario presents itself a policy can be generated sufficiently fast to be useful [3]. Further, off-line generation of policies for all scenarios seems infeasible because of the large number of scenarios.[17] However, we are not saying that some level of contingency analysis is not worthwhile [55, 79], and in Section 4.2.2 (*Uncertainty and Dynamics of Commitments)* we will discuss how this analysis can be used to construct better coordination strategies. Further, in environments where there is more regularity in distribution and arrival rate of goals, which also implies more structure to the transitions among scenarios, a more pre-planned, contingency-based approach may be appropriate.

In essence, there is a spectrum of approaches to coordination that represent different trade-offs (optimality of coordination, computational costs, communication costs, delays, etc.) based on where they lie in a multi-dimensional space. The dimensions of this space include for example: (1) the amount, detail and accuracy of information used about the current and future set of activities for the local agent and other agents; (2) the time horizon for which a strategy is appropriate; (3) the contingencies that are analyzed and the ones that are pre-planned for; and (4) the optimality of the coordination strategy. From this perspective, GPGP makes the following choices: it only acquires information about other agents based on the existence of potential interdependencies among goals that are deemed important in the current situation; it limits the time horizon and analysis to the known current goals;[18] it does some level of analysis of contingencies in choosing local schedules and commitments but does not pre-generate alternative local schedules to handle specific contingencies, and as described above, it makes a variety of decisions that limit computational and communication costs to the detriment of the optimality of coordination. As always, there is no single best approach to coordination; the appropriateness of a specific approach to coordination depends on factors such as the size, regularity and dynamics of the coordination scenario space, the type and

degree of interdependencies among agent goals and activities, the worth structure of different goals, and the uncertainty in agent behavior, etc.

To summarize, GPGP approximates the solution to the global optimization problem of choosing and sequencing activities of a group of agents that generate the highest combined utility by breaking the problem down into a set of asynchronous and evolving local optimization problems, one per agent, where part of the optimization problem is solved by GPGP and part by the DTC scheduler. Each of these local optimization problems involve choosing and sequencing local agent activities; these local optimization problems are in turn modified to reflect the dynamically evolving interactions of activities in different agents with these local activities. This modification occurs through the addition of constraints[19] on local agent activities (commitments to other agents) and expectation of results from non-local agent activities (commitments by other agents). These modifications occur through a limited sharing of information and search by a small group of agents.

2.2. Implementation of underlying conceptual model

GPGP starts by each agent (i.e., the domain problem-solving component of the agent) constructing its own local view of the activities (goals and their associated task structures) that it intends to pursue in the short-to-medium-term time frame, and the relationships among these activities. The TÆMS representation, an enriched representation of a goal tree that includes quantitative information and temporal sequencing constraints plus more dynamic information on the state of subgoal scheduling execution, is the shared representation that is used by the problem-solving, coordination and scheduling components to communicate among themselves. This view of local agent activities may be augmented by static and dynamic information about the activities of other agents and the relationship between these activities and those of the local agent; thus, the view evolves from a local view to one that is more global. Information gathering and coordination mechanisms that are part of GPGP help to construct these more global views for an agent, and to recognize and respond to particular inter-agent task structure relationships by making commitments to other agents. These commitments result in more coherent, coordinated behavior by affecting the tasks an agent will execute, when they will be executed, and where their results will be transmitted. Integral to GPGP is the DTC domain-independent scheduler that, based on commitments, agents' goals, the local and non-local values of tasks, and other agent activity constraints, creates an end-to-end schedule of activities for the agent to perform[20] that meets real-time deadlines and addresses cost and resource constraints and preferences. In this way, GPGP coordinates the activity of agents through modulating their local control as a result of placing constraints and commitments on the local scheduler. No single coordination strategy is appropriate for all task environments, but by selecting from a set of possible coordination mechanisms (each of which may be further parameterized), we can create a wide set of different coordination responses that provide a range of trade-offs between the overhead of coordination and the optimality of coordination.

To reiterate, the central representation that drives all of the GPGP coordination mechanisms is that of the local (and non-local) task structures as specified in TÆMS. Several important pieces of static information are captured in a TÆMS representation of agent activities. Figure 11 illustrates pictorially some of these items, which include: (a) the top-level goals/objectives/abstract-tasks that an agent intends to achieve including the deadline for their completion and the earliest time they can begin being executed, (b) one or more of the possible ways that they could be achieved, expressed as an abstraction hierarchy whose leaves are basic action instantiations, called methods, (c) a quantitative definition of the degree of achievement in terms of measurable characteristics, such as solution quality and time, called quality-accumulation-functions (*qaf*'s), (d) task relationships that indicate how basic actions or abstract task achievement affect task characteristics (e.g., quality and time) elsewhere in the task structure, and (e) the resource consumption characteristics of tasks and how a lack of resources affects them.[21] The existence of interdependencies among subtasks situated in different agents as specified by *qaf*'s, task relationships and resource consumption characteristics are the triggers that are used by GPGP to establish coordination of activities among agents. It is also important to emphasize that the TÆMS representation is not intended to capture the detailed problem-solving state of an activity. The only state information easily representable is the level of resources used and the amount of quality so far achieved,[22] and it is the responsibility of the domain problem solver to update this information as methods complete execution. This choice of representation is quite unique and powerful. Unlike many systems, it deals with *worth-oriented* [58] environments where a goal is not black and white, but rather has a degree of achievement associated with it. We also allow many goals to be active at once where the overall optimization criteria is to maximize the combined utility achieved as a result of completing some of these goals. Additionally, the agent's task structure view may change over time due to uncertainty or a dynamically changing environment.[23]

Qaf's are generalizations of logical constraints among tasks and their subtasks such as *and* and *or* to quantitative functions such as *min* and *max*, respectively. The idea is to provide a repertory of such *qaf*'s that can be used to approximate how the relative success in achieving a low-level problem-solving activity will eventually affect higher-level activities of which they are a part. In a similar way, task relationships represent a quantified view of temporal constraints among activities[24] as a result of information sharing relationships. Hard relationships (e.g., *enables*) denote when the result from one problem-solving activity is required to perform another,[25] or when performing one activity precludes the performance of another (e.g., *disables*). Soft relationships, such as *facilitates* (or *hinders*), express the notion that the results of one activity may be beneficial (or harmful) to another activity, but that the results are not required in order to perform the activity. These relationships are called non-local effects (NLEs) if they are between tasks situated at different agents, and define potential places for GPGP coordination; relationships among tasks in the same agent are not of direct concern to GPGP. The quantified view of these relationships indicates how the quality of the information generated by an activity will affect the performance characteristics of the activity using this information, such as the length of its execution and the quality of its resulting solution. The quantified aspects of

these relationships are parameterized so that the user has some flexibility in specification.

Resource consumption characteristics are specified in much the same way as task relationships, except that the relationship is between a resource and a task rather than a task and a task. We have used the *produce* and *consume* relationships from task to resource to specify the amount of a resource used and produced, respectively, as a result of task execution; the *limits* relationship from resource to task has been used to represent how the lack of the required resources will affect the performance characteristics of the task such as delaying task execution or increasing the amount of time for the task to execute. The set of task and resource relationships that we have chosen is again an attempt to construct an appropriate repertory of such relationships that will allow the faithful modeling of domain problem-solving activities.

Restricting the set of *qaf*'s, task relationships and resource consumption characterizations to be predefined rather than user defined permits more computationally efficient analysis of the TÆMS structure by the DTC scheduler and GPGP but at the cost of less faithful approximation by TÆMS of domain problem-solving activity. This efficiency is important since one of the main analyses done repeatedly on the TÆMS task structure is making predictions on how the choice of particular paths through the task structure will affect the expected quality and duration of the overall task. To re-emphasize, TÆMS takes both an abstracted view of the state of domain problem solving and a local view of how intermediate activities affect the state of domain problem solving as reflected in the definition of *qaf*'s and task relationships. Based on our experiences using TÆMS in a variety of applications, we feel this abstract view of domain activities is sufficient to make effective local scheduling and coordination decisions (we discuss this issue further in Section 3.5). We will delay a more detailed discussion on TÆMS until Section 2.4, which provides a more formal view, and Section 3, which discusses extensions.

In BDI-style architectures [57], an agent selects from among its desires an intended set which are then ''planned'' for, and whose component actions are then ''scheduled'' for execution. In this paper, we view ''planning'' as the process of elucidating a task structure with a manageable number of possibilities, and ''scheduling'' as the process of selecting specific intended actions and locating them in time. In fact, planning, scheduling, and action execution can be viewed as increasingly specific and less revocable intentions to more and more precise action specifications. As stated earlier, GPGP, despite the name, has so far focused primarily on coordinating the scheduling process.[26]

The GPGP approach uses the basic TÆMS task structure representation, and adds two important extensions: partial representations of the task structures at other agents, and local and non-local commitments to task achievement. This information, as previously mentioned, will be called an agent's *partial global view* or *subjective view*. Each GPGP coordination mechanism is specified with respect to features in these dynamically changing task structures, and so provides flexible coordinated responses across different problem domains. The quantitative, worth-oriented approach has several benefits when compared to symbolic, preplanned, domain-specific approaches [33]. In particular, GPGP is not domain-specific, but

rather task-structure-characteristic-specific. The same task relationships may appear across many different domains, and the same coordination mechanisms can be used.[27] Efficient and effective coordination must account for the benefits and the costs of coordination in the current situation. The current situation includes the goals that the agent is currently pursuing, the goals it will likely pursue in the near term, the characteristics of the abstract tasks and basic actions available to achieve these goals, their relationships to other tasks (possibly at other agents), and the degree of achievement necessary for each goal. Purely symbolic reasoning about costs and benefits can be extremely complex, particularly in large systems and open environments. Thus, the ability to reason quantitatively about the benefits and costs of coordination seems essential for effective system operation where there is a large set of situations that need to be reasoned about [40].[28]

2.3. *A detailed GPGP example*

To illustrate how GPGP operates, consider the task structure shown in Figure 3. The task structure represents a global view of all of the activities being carried out by three cooperative agents, *A*, *B*, and *C*, and the interactions between these activities. In this example, the agents are fully cooperative (attempting to maximize overall utility) and working to achieve common objectives so the utility produced by one agent has equal value to another agent.[29] In this structure, task *A1* facilitates task *C1* and task *A2* facilitates task *B1* – in other words, agent *A*'s tasks interact with tasks being carried out by each of the other agents. Given this global view, control problem solving, while exponential, is straightforward. However, in GPGP we do not assume that a global view exists or that it is even composable. In large-scale multi-agent systems, particularly those embedded in dynamic environments, construction of the view is generally not possible and the combinatorics of centralized problem solving make the centralized approach intractable.

In GPGP, initially, each agent has a view of only its own local problem-solving activities. Figure 4a shows agent *A*'s initial view, Figure 4b shows *B*'s initial view, and Figure 4c shows *C*'s initial view. In this example, the agents are autonomous and operating on their own local task structures. None of the agents' activities require input from any other agent and the potential relationships between their activities are not known *a priori*. Each agent's tasks also have quality, cost, and duration characteristics, as well as potential local constraints such as deadlines and earliest start times. In this example, tasks have the following durations and deadlines: *A1* has a duration of 6, *A2* a duration of 5, *B1* a duration of 5 and a deadline of

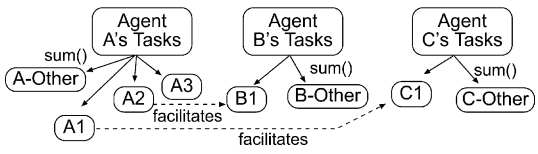

Figure 3. The hypothetical global view.

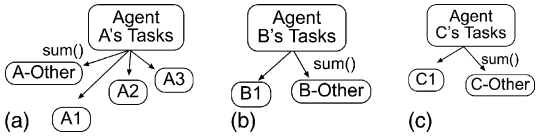

Figure 4. (a) Agent *A*'*s* initial view; (b) agent *B*'*s* initial view; (c) agent *C*'*s* initial view.

10, *C1* a duration of 4 and a deadline of 10. We will assume, to clarify the example, that the other tasks are unconstrained and ignore their characteristics, and that there is sufficient time to perform the coordination described in the example prior to the earliest start time of any of the tasks[30].

Each agent constructs a local schedule based on its local view. This is generally done by the Design-to-Criteria (DTC) agent scheduler [25]. DTC analyzes the agent's task structure and its constraints, and evaluates the trade-offs of different possible courses of action and custom tailors a schedule for the agent. Given that the agents' initial views are only of their local problem-solving activities and their local constraints, the initial schedules of the agents are shown in Figure 5. Note that time 10, which is the deadline for tasks *B1* and *C1*, is identified in all of the schedules.

If agent *A* encounters agent *B* during the course of problem solving,[31] the agents may then engage in a dialogue during which they compare their activities, or portions thereof, and look for interactions between the activities – in this way an agent incrementally constructs a partial global view of the coordination episode. In GPGP, the mechanism used to exchange and compare task structures is called *exchange-non-local-views*. It is during this dialogue that the facilitation relationship between agent *A*'s task *A2* and agent *B*'s task *B1* is discovered. To ground the example, if the agents were information-gathering agents, the facilitation might model agent *A* providing a URL to agent *B* so that agent *B* does not need to search for said URL. After the exchange, agent *A* and agent *B* have views of each other's activities as shown in Figure 6. Note that if the agents have larger task structures, it is unlikely that the agents will exchange entire task structures.[32]

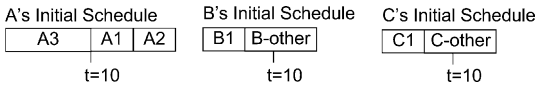

Figure 5. The agents' initial schedules.

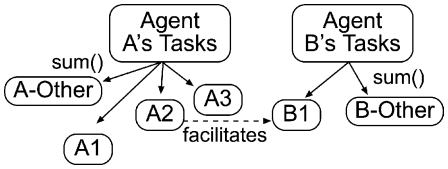

Figure 6. Agent *A*'s and *B*'s view after the information exchange.

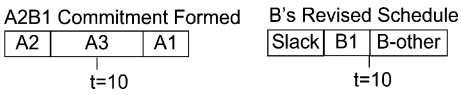

Figure 7. Agents *A* and *B* revise schedules to leverage positive task interaction.

After *A* and *B* detect the facilitation relationship they can then compute that performing *A2* before *B1*, and sending agent *B* the result from agent *A*, will improve the overall utility of their problem solving. Thus, to improve problem solving the agents deploy the GPGP *handle-soft-interactions* mechanism and agent *A* reschedules, using DTC, and offers agent *B* a *commitment* to produce a result for *A2* by time 5 so that agent *B* can execute task *B1* between time 5 and 10 and still meet its deadline. Agent *B* then reschedules, again using DTC, to take advantage of this result. Their respective schedules after the commitment is formed are shown in Figure 7.

The example thus far illustrates a single exchange, discovery, and commitment formation between two cooperative agents. However, GPGP was designed to operate in an open environment where this type of activity occurs frequently as agents discover one another and as the environment changes or the tasks being performed by the agents change. Control in this context is a difficult problem because little remains static – the goal in GPGP is to perform control from a satisficing perspective and to respond to these dynamics of the environment.

Returning to the example, let us now assume that agent *C* encounters agent *A*. As with the exchange between *A* and *B*, agents *A* and *C* will now exchange their local views. Agent *A*'s revised view is shown in Figure 8 and agent *C*'s in Figure 9. Note that agent *C* does not have a view of agent *B*'s activities – only agent *A* now has a full view of the activities happening at the other agents.

Let us assume that when task *C1* is facilitated by *A1* it will produce greater utility than *B1* when it is facilitated by *A2*. Note that because of the time 10 deadline held by both *C1* and *B1* agent *A* is unable to facilitate both tasks. In this case, agent *A*

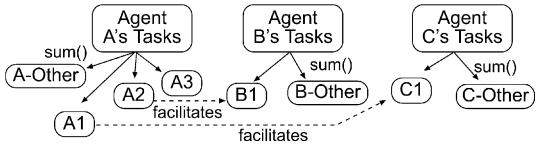

Figure 8. Agent *A's* revised view.

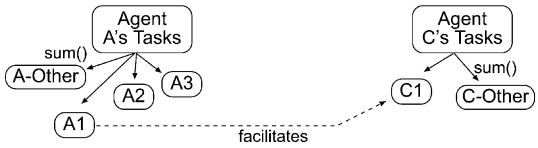

Figure 9. Agent *C's* revised view.

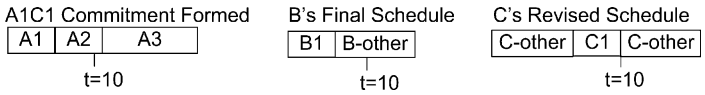

Figure 10. The final schedules of agents *A*, *B*, and *C*.

must break its commitment with agent *B*, and then reschedule in cooperation with agent *C* to form a new commitment to provide the result of task *A1* to agent *C* by time 5 so that agent *C* can perform *C1*. The act of decommitting from agent *B* may involve a penalty depending on how the decommitment is structured. In the original GPGP work, decommitment penalties were not used.[33] The revised schedules for the agents are shown in Figure 10.

This example shows the basic style of coordination in the original formulation of GPGP, which involves an incremental, dynamically evolving process. It also indicates the close interplay between the local scheduler and coordination mechanisms.

2.4. *Formalization of GPGP and TÆMS*

In order to formalize GPGP and the associated coordination problem it is solving, we must first formalize the representation of agent tasks and how task execution relates to utility maximization, using TÆMS. In utility theory, agents have preferences over possible final states (action or plan outcomes), and preference-relevant features of an outcome are called *attributes*. A substantial body of work exists on relating attribute values to overall utilities [76]. At its core, TÆMS is about specifying these attributes and the processes by which they change – what we call a model of the task environment. To simplify our discussion here we will use only two attributes, *quality* and *duration*.

A TÆMS *action* (or *executable method*) represents the smallest unit of analysis. $\mathbf{d_0}(M)$ is the *initial duration* of action M, and $\mathbf{q_0}(M)$ is the *initial maximum quality* of action M. $\mathbf{d}(M,t)$ is the *current duration*, and $\mathbf{q}(M,t)$ is the *current maximum quality* of action M at time t. $Q(M,t)$ is the *current quality* of action M. $Q(M,t) = 0$ at time t before the execution of M. If an agent begins to execute M at time t (written $\text{START}(M)$) and continues until time $t + \mathbf{d}(M,t)$ (written $\text{FINISH}(M)$), then $Q(M,\text{FINISH}(M)) = \mathbf{q}(M,\text{FINISH}(M)) = \mathbf{q}(M, \text{START}(M))$ (i.e., the current actual quality becomes the maximum possible quality). For the purposes of this simplified discussion, the amount of work done on an action M here is simply $\text{WORK}(M) = \text{FINISH}(M) - \text{START}(M)$. If there were no interrelationships (NLEs) between M and anything else, then $\mathbf{q}(M,t) = \mathbf{q_0}(M)$ and $\mathbf{d}(M,t) = \mathbf{d_0}(M)$. The execution of other actions and tasks effect an action precisely by changing the current duration and current maximum quality of the action (that is, $\mathbf{d}(M,t)$ and $\mathbf{q}(M,t)$, as specified below). For the purposes of this paper, we will also assume that $Q(M,t) = 0$ for $\text{START}(M) \leq t < \text{FINISH}(M)$; other definitions of Q are possible to represent anytime algorithms, etc. Action pre-emption and resumption may also be modeled by extending these simple definitions [14].

A TÆMS *task* or *subtask* represents a set of related subtasks or actions, joined by a common *quality accumulation function*. Given a subtask relationship Subtask(T_1,**T**) where **T** is the set of all direct subtasks or actions of T_1, if T_1 has a *min* quality accumulation function then we may recursively define $Q(T_1,t) = Q_{\min}(T_1,t) = \min_{T \subseteq \mathbf{T}} Q(T,t)$. For the purposes of evaluation, the amount of work done on a task is the sum of all the work done on its subtasks, and the finish time of a task is the latest (*max*) finish time of any subtask.

Any TÆMS action/method, or a task T containing such a method, may potentially affect some other method M through a *non-local effect e*. We write this relation (a labeled arc in the task structure graph) as NLE(T,M,e,p_1,p_2,…), where the p's are parameters specific to a class of effects. For purposes of the discussion, we assume there are three possible outcomes of the application of a non-local effect on M under our model: $\mathbf{d}(M,t)$ (current duration) is changed, $\mathbf{q}(M,t)$ (current maximum quality) is changed, or both. An effect class e is thus a function $e(T,M,t,d,q,p_1,p_2,\ldots)$:[task, method, time, duration, quality, parameter1, parameter2, …] ⇒ [duration, quality]. For example, if task T_a *enables* action M, then the maximum quality $\mathbf{q}(M,t) = 0$ until T_a is *completed*, at which time the current maximum quality will change to the initial maximum quality $\mathbf{q}(M,t) = \mathbf{q_0}(M)$. Another way to view this effect is that it changes the "earliest start time" of *enabled* method, because a rational scheduler will not execute the method before it is enabled. Thus we often write *enables* ($\boldsymbol{M}_1$,$\boldsymbol{M}_2$) to indicate that $\boldsymbol{M}_1$ must be executed before $\boldsymbol{M}_2$ in order to obtain quality for $\boldsymbol{M}_2$.

Underlying a TÆMS model is a simple state-based computation. Each method has an initial maximum quality $\mathbf{q_0}(M)$ and duration $\mathbf{d_0}(M)$ so we define $\mathbf{q}(M,0) = \mathbf{q_0}(M)$ and $\mathbf{d}(M,0) = \mathbf{d_0}(M)$. If there are no NLEs, then $\mathbf{d}(M,t) = \mathbf{d}(M,t-1)$ and $\mathbf{q}(M,t) = \mathbf{q}(M,t-1)$. If there is only one NLE with M as a consequent $NLE(T,M,e,p_1,p_2,\ldots)$, then $[\mathbf{d}(M,t),\mathbf{q}(M,t)] \leftarrow e(T,M,t,\mathbf{d}(M,t-1), \mathbf{q}(M,t-1),p_1,p_2,\ldots)$. Based on this partial formalization of the TÆMS model, we can now discuss the formalization of GPGP.

From a formal perspective GPGP is based on modifying the activity schedules of agents so as to increase the overall utility achieved in a coordination-episode. A coordination-episode (called **E**) is defined as a tuple {**A**, **T**, **M**, **t**, **R**, **D**, **SNL**, **d**} where:

- **A** is a set of agents.
- **T** is a set of TÆMS tasks.
- **M** is a set of TÆMS actions (methods).
- **t** is a mapping from tasks to sets of subtasks and/or actions.
- **R** ∈ **T** is a set of "Root tasks," (sometimes called "task groups" or "goal tasks") that represent the root task nodes of independent task trees.
- **D** is a mapping from root tasks in **R** to deadlines.
- **SNL** is a set of NLEs between elements of **T** and/or **M**.
- **d** is a mapping from every action in **M** to the agents that are capable of executing that action.

The utility associated with **E** as a result of executing actions by agents in **A** is the sum of the task group qualities $\boldsymbol{\Sigma}_{\mathbf{T}\in\mathbf{R}}\boldsymbol{Q}(\mathbf{T},\mathbf{D}(\mathbf{T}))$, where $\boldsymbol{Q}(\boldsymbol{T},\boldsymbol{t})$ denotes the quality of $\boldsymbol{T}$ at

time t as defined in [10]. Quality does not accrue after a task group's deadline. For the purposes of evaluation, a *solution* to an episode **E** can be represented abstractly as a set of *after-the-fact* schedules for each agent, indicating the START and FINISH times for each action.

Of course, such an evaluation can only take place in retrospect. Since, in general, the durations of tasks are stochastic, FINISH times can only be estimated, and with them the NLEs in **E**. In the GPGP approach, then, given an episode $\mathbf{E} = \{\mathbf{A}, \mathbf{T}, \mathbf{M}, \mathbf{t}, \mathbf{R}, \mathbf{D}, \mathbf{SNL}, \mathbf{d}\}$, each agent has a local scheduling problem (LSP) that involves choosing which methods to execute and when so as to maximize the *expected value* of the utility function. A schedule S produced by a local scheduler will consist of a subset of methods executable by the agent in question and their associated 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. Given such a schedule and the complete episode description, one can for example compute $D_{\text{est}}(T,S)$ and $Q_{\text{est}}(T,t,S)$, the expected duration of a task and the expected quality of a task at some point in time.

In all but the simplest cases of totally independent task groups or totally identical episode knowledge and capabilities, the LSP is often trivial and results in no actions and zero utility – this is because without further information no tasks requiring joint action may be achieved. In the general case, what is required from the point of view of a single agent is schedule timing information on at least some of the actions of other agents, or at least on tasks that directly relate to tasks an agent can achieve on its own.

Formally, **T**, **M**, **t**, and **SNL** induce a (often non-fully connected) directed graph with **T** and **M** as nodes and **t** and **SNL** as directed arcs connecting them. We now construct an extension of this graph to explicitly represent possible redundant actions. For every method in **M** that can be executed by more than one agent (expressed in the mapping **d**), replace it with a new task with a quality accumulation function of *max* and one unique child for each possible agent. Call this new graph G. For each agent we may now define a local subset, G_{A} of G as follows. Add all the (now unique) methods **M** for that agent, and then recursively add all the supertasks of those methods from the relation **t**. This defines the "local" subset of G for agent A. Now note two extremely important relations: first, elements of **SNL** that start in one subgraph G_{A} and end in another subgraph G_{B}, and secondly elements of either **SNL** or **t** that have one end in a subgraph G_{A} but the other end at a node that is in both G_{A} and G_{B}. These relations are called *coordination relationships* [**CR**] and are precisely the set of things that cause a problem in computing a non-zero expected value for agent utility when tasks are interdependent. Each represents an *opportunity* for coordination of some kind.

The GPGP approach then postulates for each CR some mechanism to remove the uncertainty associated with that CR, by creating a commitment to action (or non-action) possibly with an associated temporal constraint. We briefly discuss two types of commitments here, but for additional types see also [12, 79]. $\boldsymbol{C}(\mathbf{Do}(\boldsymbol{T},\boldsymbol{q}))$ is a commitment to 'do' (achieve quality for) $\boldsymbol{T}$ and is satisfied at the time $\boldsymbol{t}$ when $\boldsymbol{Q}(\boldsymbol{T},\boldsymbol{t}) \geq \boldsymbol{q}$; the second type $\boldsymbol{C}(\mathbf{DL}(\boldsymbol{T},\boldsymbol{q},\boldsymbol{t}_{\text{dl}}))$ is a 'deadline' commitment to do $\boldsymbol{T}$ by time $\boldsymbol{t}_{\text{dl}}$ and is satisfied at the time $\boldsymbol{T}$ when $[\boldsymbol{Q}(\boldsymbol{T},\boldsymbol{t}) \geq \boldsymbol{q}] \wedge [\boldsymbol{t} \leq \boldsymbol{t}_{\text{dl}}]$. 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).

A *coordination mechanism*, then, indicates for a specified subset of coordination relationships a set of commitments that should be made locally, and other actions (usually communication actions) to take in order to make the commitment social (in the sense of [33]). For example, the *hard predecessor* coordination mechanism distinguishes the direction of the relationship – it only creates commitments on the predecessors of an *enables* relationship. We will 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 $\boldsymbol{S}$ at time $\boldsymbol{t}$ will produce quality for a predecessor in **HPCR**, and commits to its execution by a certain deadline both locally and socially:

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

The next question is, by what time ($\boldsymbol{t}_{\text{early}}$ above) do we commit to providing the answer? One solution is to use the *min* $\boldsymbol{t}$ such that $Q_{\text{est}}(T,D(T),S) > 0$. 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 $C(DL(T,Q_{\text{est}}(T, D(T),S), \text{``early''}))$ to the local commitment set **C**, and invoking the local scheduler LS(**SNL**,**C**,**NLC**) to produce a new set of schedules **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 $\boldsymbol{T}$ to provide a value for $\boldsymbol{t}_{\text{early}}$. The new completed commitment is entered locally (with low negotiability) and sent to the subset of interested other agents. Other coordination mechanisms can be defined in similar ways. In this way, GPGP mechanisms generate commitments between agents in order to coordinate the schedules of agents so as to increase the utility achieved in a coordination-episode.

2.5. *Connections to joint-intention model*

The initial set of mechanisms for GPGP was derived from Durfee's work on PGP [18] in the distributed vehicle monitoring testbed (DVMT). These five mechanisms were: (1) communicate non-local views; (2) communicate appropriate results; (3) avoid redundancy; and handle (4) hard and (5) soft coordination relationships between tasks at two agents [11, 14]. These five basic mechanisms formed a domain-independent set for basic agent teamwork. As such, it is interesting to compare them to Tambe's work on flexible teamwork, implemented in the STEAM and TEAM-CORE frameworks [51, 62, 63] that is in turn based on Cohen and Levesque's work on joint-intentions[34] [46]. In particular, the five initial GPGP mechanisms do not separate the environment-centered coordination actions from actions oriented toward teamwork itself. In other words, in general, GPGP does not communicate solely to maintain some explicit model of teamwork. However, a striking example of

the relationship of GPGP to the Cohen and Levesque teamwork model is their prediction that team members should communicate when they believe that the object of a joint persistent goal has been achieved, found to be unachievable, or irrelevant. The GPGP "communicate results" mechanism contains a clause to communicate a "final result" (degrees of achievement) for every task group (implicit joint persistent goal) indicating that the agent believes no further actions can be done for that task group. Interestingly, this is one of the small generalizations that were added when creating the general GPGP mechanisms from the original DVMT PGP heuristics.

GPGP currently represents joint intentions only implicitly. There are no explicit mechanisms to guarantee agents have mutual intentions to achieve their parts of the jointly shared goal. GPGP generates commitments to accomplish subtasks of a jointly shared goal though these detailed commitments are not in response to a more encompassing commitment, explicitly held by each agent, to achieve the jointly shared goal. There is no guarantee that an agent will solve its part of a jointly shared goal even though it knows of its existence. In fact, from a social utility perspective, an agent may choose to solve other goals/tasks that it knows of rather than its part of a jointly shared goal. However, what is of concern is if the following situation occurs: an agent solves its part of a jointly shared goal while another agent chooses not to solve a critical part of the jointly shared goal, thus making the other agent's effort worthless. Additionally, there are limited mechanisms in the initial GPGP to coordinate about which alternative ways agents should achieve their local subtasks of the jointly shared goal that would be maximally beneficial to them in the context of the other tasks they are doing. This latter problem entails a complex distributed search due to the *qaf* relationships possible among distributed subtasks. We have only recently begun to solve this problem in restricted cases as will be described in the section on more complex coordination mechanisms. Of the original five initial mechanisms only the redundancy mechanism is triggered by the existence of a *qaf*'s relationship (i.e., *max*) among sibling subtasks located in different agents, and it uses a very simple heuristic not involving distributed search for making a decision.[35] Neither the lack of explicit mechanisms to generate a joint intention nor the lack of mechanisms to handle *qaf*'s such as *min* (*and*) turned out to be a serious problem in the experiments [11, 14] that we conducted. Problems did not often occur because: (1) subtasks in a jointly shared goal were often connected through interrelationships so that failing to schedule a subtask in one agent would lead another agent to not schedule its related subtask – this resulted in implicitly deciding to not solve the shared joint goal;[36] and (2) the scheduling heuristic used by the local scheduler (DTT [20]) was to first guarantee, if possible, that all local task structures achieved some minimum level of utility before attempting to maximize the overall utility achieved by all local tasks. For these reasons, the GPGP mechanisms operated correctly in most cases by either explicitly generating commitments to solve a subtask of the jointly shared goal (or not solve it) due to an interrelationship between a subtask of the shared goal held by another agent or scheduling these subtasks due to local utility concerns. We now feel that with a few additional messages being transmitted among agents,[37] GPGP could be made to work in all cases within our existing mechanisms; these messages would guarantee that when an agent made a decision not to schedule activities to generate utility for their part of a shared goal other

agents having parts of the shared goal would be notified. In essence, we are saying that the joint-intentions model is compatible and integratable with our quantitative view of coordination and the issue of implicit vs. explicit representation of jointly shared goals becomes moot as we add additional attributes to the TÆMS task structure representation necessary to support the reasoning associated with these additional messages being exchanged among agents[38]. Additionally, Tambe's STEAM/TEAMCORE work points to a need to more carefully define coordination mechanisms to separate bottom-up, environment-centered coordination activities from top-down, organizationally prescribed teamwork coordination activities – an exciting area of future work. On the other hand, the success of using GPGP environment-centered mechanisms in multiple environments demonstrates that there are indeed general mechanisms for solving domain-level coordination problems in different environments, because the task relationships in those environments can be characterized in a general way. While STEAM's teamwork rules are general, it still requires the addition of new, different, domain-specific coordination rules for each domain. Another difference is that TÆMS and GPGP deal in worth-oriented environments, thus goals are achieved to varying degrees, rather than black and white "achieved or not achieved."[39]

In the next sections we will examine how in a principled and parsimonious way the GPGP/TÆMS framework can be augmented to (1) make it a more general framework for coordination where bottom-up and top-down forms of coordination can co-exist and be intermingled and where a wider set of agent activities and their relationships can be represented, (2) allow for dynamic and finer-grained situation-specific coordination, (3) permit more effective operation in large agent societies structured as organizations, (4) take into account contingencies in agent behavior when making coordination decisions, (5) use more complex coordination protocols so that more explicit reasoning takes place about how coordination decisions affect the global utility and more optimal coordination patterns are found based on a cooperative search between agents, and (6) facilitate resource-based coordination mechanisms. We begin this discussion by first detailing extension in the representation of agent activities.

3. The evolution of TÆMS

In its original conceptualization, TÆMS was a hierarchical task representation language that featured the ability to express alternative ways of performing tasks, characterization of methods according to expected quality and expected duration – and the representation of interactions between tasks. As a result of using it in a number of applications, TÆMS has evolved considerably in its representational power and in the scope of applications that can be appropriately represented.

3.1. Addition of discrete distributions

As TÆMS has evolved, we have moved beyond simple quality/time trade-off reasoning to a multi-dimensional satisficing methodology [71]. Methods are no longer

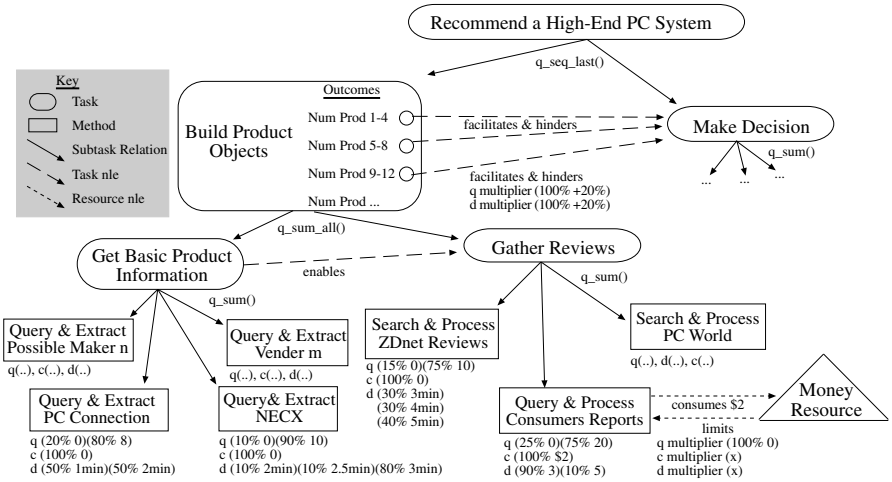

Figure 11. An example of a TÆMS task structure for an information-gathering agent.

simply characterized in terms of expected quality and duration values but are characterized statistically via discrete probability distributions in three dimensions: quality, cost, and duration; and the work is extensible to multiple other dimensions as well (see Figure 11 for an example and associated description). The switch from expected values to discrete distributions enables reasoning about the certainty [72] of particular courses of action, perhaps performing contingency analysis [55, 56], as well as the quality/cost/time trade-offs. This more complete view is crucial in highly constrained situations (e.g., where deadlines are present). For example, in meeting an important commitment, it may be preferable to choose a highly certain course of action that has only moderate overall quality to ensure that results are available by the deadline, rather than a less certain solution path, even though it may have the possibility of a very high-quality payoff. The enhancement of multiple attribute dimensions, including uncertainty, is also important because the representation enables the multi-agent system designer to better characterize the desired system performance [71, 73]. This new performance characterization takes into account both hard and soft constraints on quality, cost and duration of the schedule of tasks intended to meet a high-level goal or commitment. Additionally, the amount of uncertainty that is tolerable in meeting these constraints can be specified [44].

3.2. *Additional quality accumulation functions*

In TÆMS, progress toward the problem-solving objective is expressed and tabulated in terms of quality. Tasks accumulate quality from their subtasks according to *quality accumulation functions (qafs)*. Originally, *qafs* defined only which combinations of subtasks could be performed to achieve the parent task and how the subtask qualities are tabulated at the parent task. For example, the *sum( ) qaf* denotes that any element of the power set of the child tasks, minus the empty set, can

Table 2. Definition of quality accumulation functions.

q_min – minimum single quality of all subtasks.
q_max – maximum single quality of all subtasks.
q_sum – total aggregate quality of all subtasks.
q_last – quality of most recently completed subtask.
q_sum_all – as with *q*_sum, except all subtasks must be completed.
q_seq_min – as with *q*_min, except all subtasks must be completed in order.
q_seq_max – as with *q*_max, except all subtasks must be completed in order.
q_seq_last – all subtasks must be completed in order, and overall quality is the quality of last task.
q_seq_sum – as with *q*_sum, except all subtasks must be completed in order.
q_exactly_one – quality of single subtask. Only one subtask may be performed.
q_sigmoid – similar to *q*_sum except there is a parameterized sigmoid function that defines how quality accumulates.

be performed to achieve the parent task and that the parent task's quality is the sum of the children's qualities. In the current TÆMS, *qafs* may also define orderings.[40] In all, we have developed a suite of 11 different *qafs* to represent the needs of different application domains as seen in Table 2.

Another important enhancement is the representation of different possible outcomes for primitive activities. For example, an action may produce a result of type *A* or type *B*, or it may fail entirely. With the outcome enhancement, clients can represent cases where a result of type *A* is beneficial to some other method, say method *A*; type *B* is beneficial to yet another method, method *B*; and the failure outcome enables a recovery action. While outcomes have always been included conceptually in TÆMS, either via entries in a method's quality, cost, and duration distributions or via a contribution to the expected values in the early versions of TÆMS, this new explicit outcome representation enables the expression of task interactions that are based on particular outcomes.[41] The new outcomes feature also allows the representation of different outcomes that have different statistical characteristics, i.e., different discrete probability distributions for quality, cost, and duration.

Figure 11 shows an example information-gathering TÆMS task structure, from the BIG information-gathering agent [45] that contains some of these new extensions to TÆMS. The top-level task is to recommend a high-end PC system. It has two subtasks: one that pertains to finding information about various products and constructing models of the products, and one for making the decision about which product to purchase. The two tasks are governed by a *seq_last( ) qaf*, which indicates that the tasks must be performed in order and that the quality of the top-level task is determined by the quality returned by the last subtask executed, i.e., the decision process task.

This models the fact that the decision process described here takes into consideration the quality, coverage, and certainty of the information used to make the decision and reflects these attributes in the quality of its final decision. The task that handles finding, extracting, and building information objects *Build-Product-Objects* is decomposable into tasks for finding basic product information and for gathering reviews. At the lowest level of the tree are primitive actions (also called methods or activities) whose behaviors are specified by distributions for quality (q), cost (c) and

duration (*d*). For example, "Query & Extract PC Connection" specifies that the quality produced by this activity will be 0 (quality 0 is considered a failure) 20% of the time and will be 8 the remainder of the time when it succeeds, there will be no cost incurred in doing this activity, and finally it is equally likely that the duration of the activity will be 1 or 2 minutes.

The *sum_all ( ) qaf* denotes that both of these tasks must be performed and that the quality of *Build-Product-Objects* is a sum of the qualities produced by the subtasks. *Build-Objects* has a set of different outcomes[42] that indicate the number of objects produced during the information-gathering phase. Note the dotted edges leading from *Build-Objects* to *Make-Decision*. As previously discussed, these denote task interactions also called *non-local-effects* (or NLES). The *NLES* pictured denote facilitation and hindering effects to model the notion that the decision maker's quality is improved by more objects but that more objects also increases the time required to make the decision. *Get-Basic-Product-Information* has four children governed by a *sum( ) qaf*, which means that any element of the power set of the children (minus the empty set) may be performed and that the quality of *Get-Basic* is the sum of the qualities of its children. Note that the child tasks have different quality, cost, and duration characteristics – essentially, there are $2^4 - 1$ different alternative ways to perform *Get-Basic* and the agent will determine which approach to employ based on the current problem-solving context. For example, if time is limited, it may only get information from one fast site as opposed to a set of sites.

3.3. *Representation of non-computational resources*

The explicit representation of non-computational resources, such as network bandwidth, databases, and physical resources like printers, is another important TÆMS enhancement. To do this, we represent each relevant resource explicitly in the existing task structure, as with the Money resource in Figure 11. These nodes interact with other parts of the task structure, where methods may consume or produce resources, and the current level of the resource can affect the performance of those activities in turn. Resources are defined to be one of two varieties, consumable and non-consumable, each with minimum, maximum and current levels. The two types are differentiated in that the former requires explicit production to replenish, as with paper used by a printer, while the latter automatically restores its current level when it is no longer in use, as with the printer itself.

Resources are connected to other parts of the task structure with special resource interrelationships. A *consumes* interrelationship from a method to a resource indicates that when that method is performed, it will use up some amount of that resource. An analogous *produces* interrelationship also exists. Both include quantitative, probabilistic descriptions of how much of the resource is expected to be consumed or produced. The *limits* resource interrelationship defines how the resource itself affects the method. In this case, if the resource's current level has exceeded its minimum or maximum values, the *limits* interrelationship will activate and affect the target method's performance in much the same way that a *hinders*

interrelationship would. Multipliers defined in the *limits* interrelationship determine exactly what the effects are on the method's cost, quality or duration.

3.4. *Virtual tasks*

Figure 12 shows the same information-gathering task structure as it might be decomposed and assigned to several agents. Obviously, many such decompositions are possible; the decomposition in Figure 12 is according to the WARREN [13] approach to agent specialization in multi-agent information-gathering systems. The *Task Agent* is responsible for producing the result but *Information Agents* are the retrieval and processing experts. The interrelationships between the different task structures denote GPGP coordination points. The task structure in Figure 12 contains several different types of coordination-centric interactions. The inability of the *Task Agent* to carry out the information-gathering actions means it must contract those duties out to the specialists – this is handled by new GPGP mechanisms for virtual tasking or contracting (discussed later). The *enables NLE* between the two information agents requires the use of the original GPGP coordinate-hard-task-interaction mechanism. The monetary resource interaction requires resource centric coordination like that found in the recent hospital patient scheduling [12] and the intelligent home automation [42, 43] applications.

The task structures shown in Figure 12 provide a small-scale example of TÆMS and interaction-motivated GPGP. One can easily imagine a pool of information-gathering and decision-making agents cooperating to equip an office. Interactions would exist between information-gathering activities at multiple levels of abstraction. For example, two agents shopping for computer systems might share free-format text reviews of the structured data produced by abstraction. Interactions

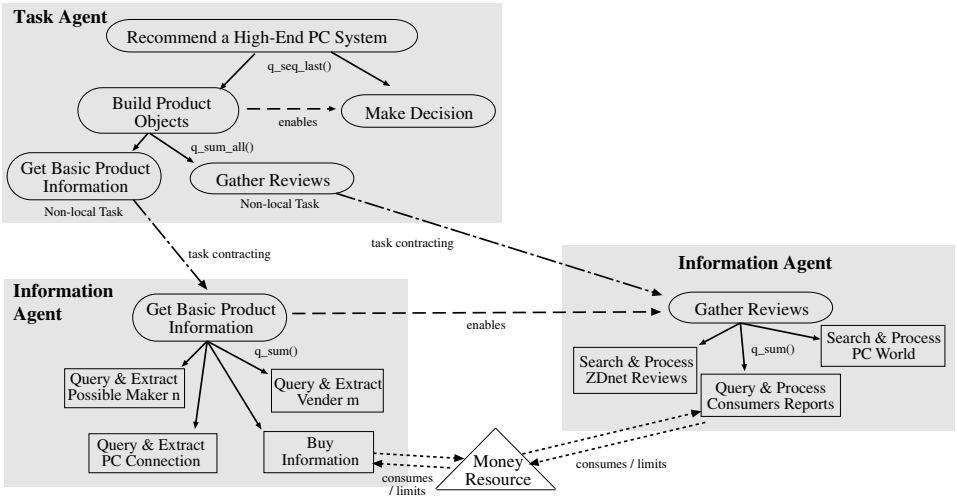

Figure 12. WARREN style model of multi-agent information gathering.

would also exist over shared resources and to support data-assimilation and data-driven decision making.

3.5. *TÆMS as an abstract model of problem solving*

There are two views of how TÆMS is related to agent problem solving. One views TÆMS as a way to represent agent problem-solving activities, that is, agents would literally use TÆMS internally. The other views TÆMS as an abstract model of agent problem-solving activities.[43] It is this latter view that is typical of our current uses in real applications: a domain problem solver uses its own internal representation for problem solving and this representation is abstracted into a TÆMS task structure for use by GPGP. In this way of using TÆMS/GPGP, we are arguing that, for most coordination needs, an appropriately abstracted view of activities is sufficient for effective agent coordination. This perspective motivates the continuous evolution of TÆMS. The addition of new features and new representations to TÆMS is caused by the constant tension among representational power of TÆMS to accurately model an agent's problem-solving activities, the ease of being able to map these activities into TÆMS, and the ability to reason without significant computational cost about these activities. There is a trade-off between representing large amounts of problem-solving possibilities/contingencies, and maintaining a somewhat unwieldy modeling framework. For example, if the problem solver enumerates all possible problem-solving actions for every step in the problem-solving process, and all possible actions caused by carrying out the first set of actions, and all possible contingencies in the case of failure or a change in the environment, the TÆMS model becomes computationally intractable.[44] However, through new constructs like iteration, outcomes, and *qaf*s that impose orderings and other semantics, the representational power of TÆMS is enhanced without increasing the computational complexity of reasoning with the TÆMS models. It also allows the domain problem solver to more accurately model key contingencies and key repetitive actions, and thus provide GPGP with more information while not overloading it with details. Without these features, an agent might have to constantly communicate with GPGP about how its view of its future activities needs to be modified as the result of its incremental execution of its tasks; this modified view may entail significant re-coordination activities since other agents may also need to be notified of these changes. Instead, GPGP can now, when it makes its original coordination decisions based on the agent's description of its activities, factor into these decisions that an agent's activities may change in certain ways as a result of the partial execution of its activities. For example, instead of establishing a commitment that is highly uncertain because there is a significant possibility that a task established to satisfy the commitment may not be completed successfully, it is now possible to analyze whether there would be another way that the agent could successfully complete the commitment with additional time. Thus, the coordination mechanism can, in making the original commitment, factor in some of the implications of the original task failing and an alternative method being required, which can result in making a different commitment that has a much higher certainty of being successfully honored.

3.6. Other extensions to TÆMS

We have also explored ideas for a number of other task relationships. We mention them here to emphasize that the TÆMS/GPGP framework can be easily extended to represent other types of task interdependencies. One relationship involves a new type of temporal synchronization among tasks that denotes the constraint that both tasks need to be started and completed within the same temporal window; the more quantitative aspects of this relationship would be the tolerable overlap among the temporal windows of tasks and how the task's solution quality is affected by the amount of overlap.[45] The need for this type of relationship is exemplified in recent work on a distributed sensor network application where measurements taken from different sensors if they are to be effectively fused need to be taken within some overlapping temporal window [31]. Another relationship indicates that there are consistency constraints among solutions to a set of tasks that will require sharing intermediate information among them to converge on a solution; the more quantitative aspects of this relationship would be the expected frequency of information exchange among tasks. Knowledge of both relationships could lead to more coordinated scheduling of task activities in the different agents.

Another interesting relationship represents a meta-level exchange of information between tasks. In this case, the relationship indicates that the completion of one task would provide information about the performance characteristics of another task; this information could then be used by the agent's problem-solving component to revise its TÆMS description of the task. This relationship is like *facilitates* in that the task can be performed without an updated view of its performance characteristics, but waiting for this information in certain situations would lead to more optimized behavior.[46]

We have also implemented a more complex view of a method's execution that permits us to model activities that do not use the processor resource for their entire duration [65]. There are two approaches that we used to represent this situation. One involved specifying an additional attribute associated with a method that indicates what percentage of the processor the method will use, and the other involved adding an attribute associated with an *enables* or *facilitates* relationship that indicated how much additional time it would take for the information generated by the method to be available for use by down-stream methods.[47] Using this information, the scheduler can then overlap method executions. In recent work [64], we have developed a more comprehensive view of method execution that allows us to reason about concurrent execution of methods where there are multiple processors and multiple copies of resources at a single agent and where methods may not be using processor resources. This allows us to build agents that can not only schedule computational activities but also act as resource controllers for shared resources [42, 43, 64].

4. Evolution of GPGP

There are three main aspects to evolution of the GPGP framework. The first concerns how the partial global view used in making coordination decisions is

constructed. It also has implications for how GPGP can interact with other components in the agent. The second involves a more complex representation of commitments. The third focuses on new and more complex mechanisms for establishing commitments among agents. These extensions have been made for a number of reasons. One reason has been to reduce the communication and computational overhead of using GPGP – the full use of all the mechanisms is not warranted in all environments as has been shown experimentally [11, 14, 48]. This is also especially important if GPGP is to operate in an environment with a large number of agents where agents take on organizational roles. Another reason is to avoid forcing users of GPGP into only one style of building multi-agent systems. The final reason is the desire to exploit the additional information representable in the extended TÆMS on uncertainty and on usage of non-computational resources for effective coordination.

It is important to mention that a version of GPGP that includes all these extensions does not currently exist. Some extensions have been implemented in different versions of GPGP while others have been developed outside of GPGP though they have used TÆMS as their basic representation of task activities and interrelationships. To our knowledge, there is no impediment to creating a version of GPGP/TÆMS that incorporates all of these extensions in a unified way.

4.1. Constructing and using the partial global view

We have significantly enriched how the partial global view used in making coordination decisions is constructed. Two issues motivated this extension. One was a desire for increased efficiency in coordination by being able to exploit default knowledge whenever it was available. The second was being able to handle not only an agent-centric and bottom-up view on how goals and their subgoals were created and distributed among the agents (which was the original orientation of GPGP) but also a hierarchical and top-down perspective on their creation and distribution. The ability to handle both perspectives in an integrated way significantly increases the scope of multi-agent applications that can be represented in GPGP.

4.1.1. Conditioned view. As mentioned earlier, TÆMS task structures are typically used to encode the different plans for achieving a goal, and the constraints and trade-offs associated with each potential plan. They are also used to describe those instances during execution where it has been determined that coordination is to take place. For instance, a TÆMS structure might indicate that one method enables another. If these two methods are to be performed by different agents, this enablement indicates a point where coordination between the two parties would be beneficial. Interrelationships in TÆMS therefore have additional meanings. A (*consumes*) relationship between a method and a resource would cause the usage of that resource to be appropriately coordinated over with other entities in the system prior to use. A hard or soft (*enables*, *facilitates*) interrelationship arising from a method or task represented at a remote agent with a local target would cause the local agent to contact the remote one to coordinate their activities in a satisfactory manner. A relationship indicating potential negative effects on a remote agent

(*disables*, *hinders*) would require the agent to first determine if performing the activity is acceptable before execution. Scheduling a method flagged as being non-local would cause the agent to coordinate over the execution of that method by a remote agent.

In this way, an agent possessing a task structure containing methods or constraints that may be satisfied by many different remote agents may actually represent quite a large range of coordination possibilities. Under some conditions the availability of so many alternatives can be beneficial, but when operating under a deadline or tight resource constraints, too many alternatives can be distracting and wasteful. What is needed then, is a way to represent an agent's true operating potential, while limiting the scope of this representation to only those parts that are reasonable and necessary.

To achieve this, we have added a second view of an agent's task structure; each agent will possess two different versions of its local task structure, called the *subjective* and *conditioned* views. The subjective view contains what the agent believes to be a complete view of its local execution alternatives. The conditioned view is a copy of the subjective view that has gone through a process of conditioning – it may contain task, method or interrelationship deletions and modifications to better adapt that structure to current operating conditions. When the conditioned view is used for plan construction, these modifications indirectly allow the problem solver performing the conditioning process to focus the attention of the scheduling and coordination mechanisms. These changes serve three purposes: to prevent certain task subtrees from being considered during scheduling, to remove interrelationships that are not to be coordinated over, and to limit the number of agents to coordinate with. In the first case, if the problem solver has determined that a particular action should not be performed, it can simply remove it from the conditioned view to prevent it from being considered. The second and third uses are more germane to the issue at hand – by convention any interrelationship present in the conditioned view indicates that the interrelationship should be coordinated over when the related tasks or methods are scheduled for execution. Removing the interrelationship or limiting the number of agents involved in its coordination will either prevent the coordination from taking place, or prevent the method from being selected for execution. By making these changes, the overhead required to reason about the task structure is reduced, and the agent's attention is more focused.

Consider the simple conditioning process that has taken place in the task structure in Figure 13.[48] On the left, the subjective view shows that resource *X*, which is needed as part of *Z's* manufacturing process, can be produced by three different agents: *P1*, *P2* and *P3*. While the local agent could elect to coordinate with all three suppliers, this is time and resource intensive, and can be avoided with the simple use of historical data. In this case, the agent knows that *P2* did a good job of producing *X* the last few times it was used, so as part of its conditioning process the other two candidates are removed, simplifying the selection process and eliminating potentially unnecessary communication. If *P2's* performance fails to meet expectations this time, the next conditioning process might elect to leave one or two of the other producers in, to increase the probability of an acceptable result at the expense of higher scheduling and reasoning costs.

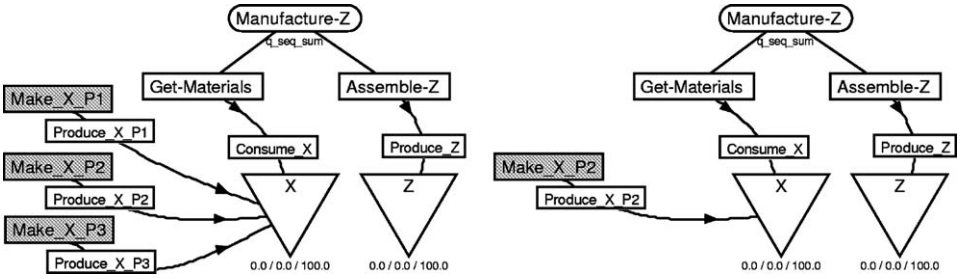

Figure 13. A simple example of a subjective and conditioned view of a TÆMS task structure.

The process of conditioning is most useful when performed dynamically at run-time by agent components capable of effectively making such decisions. For example, a problem-solving or organizational design component might enact the changes seen in the previous example after a series of negotiation sessions showed that it would be advantageous. This would be accomplished by using domain or protocol-specific knowledge to reason about the trade-offs associated with forming such a long-term relationship with another agent. Once the decision to form this relationship was made, this component would also make the necessary changes in the conditioned view. While the decision itself is relatively domain-specific, the act of removing alternatives from the conditioned view is a fairly generic one, relying only on the notion that interrelationships indicate points where coordination might need to take place. Thus, in lieu of modifying GPGP to treat some structures differently from others, conditioning modifies the basic information GPGP uses to the same effect.

At this time, much of the conditioning process we perform is accomplished by specialized components like those described above. We have, however, used diagnosis to help identify poorly conditioned structures in a more general way [28]. For example, the diagnosis engine can recognize that there are resource or enablement interrelationships present in the subjective view, but missing in the conditioned view. If the targets of these interrelationships subsequently fail to meet their expectations, one hypothesis is that the lack of those interrelationships caused necessary coordination to not take place. This could lead to a change where those conditioned changes were rolled back to correct the problem.

We have also recognized that many of the additional attributes that are dynamically attached to the TÆMS task structure as a result of coordination mechanisms (e.g., learning about the existence of non-local task relationships and the related task structure of other agents, and dynamic generation of commitments) can be statically predefined. As will be described next, this has led us to be able to not only develop highly efficient coordination protocols for specific situations but also to allow GPGP to operate in a wide range of application environments.

4.1.2. Exploiting default knowledge in coordination. We have also introduced the idea of coordination mechanisms that are based on default knowledge that allows us to represent coordination based on social laws. As previously discussed, coordination

in GPGP is achieved through the use of *commitments*, that is, inter-agent contracts to perform certain tasks by certain times. The commitments, which are dynamically constructed, generally fall into three categories: (1) *deadline* commitments are contracts to perform work by a certain deadline; (2) *earliest start time* commitments are agreements to hold off on performing certain tasks until a certain time has passed; and (3) *do* commitments are agreements to perform certain tasks without a particular time constraint. By allowing these commitments to be specified *a priori* as part of the TÆMS task structure, social law-type coordination, which has low overhead, can be achieved among agents. For example, *a priori* commitments could indicate that an agent could expect another agent to generate a specific result and transmit it to this agent by a certain time [49]. The transmitting agent, in turn, has an *a priori* commitment to generate the results by a specific time. In this way, agent activities can be coordinated without the agents exchanging information about their current activities, and then negotiating over a suitable commitment.[49] More generally, knowledge of potential task relationships can be created based on (1) organizational knowledge of the activities of other agents, (2) another agent decomposing a task structure and then assigning parts of the task structure to different agents, (3) the past history of agent interactions, and (4) a process of extensive broadcast and discovery. In summary, our goal is to have a suite of coordination mechanisms that can reason about in a controlled manner any existing information useful for making coordination decisions, and to acquire in a controlled manner additional information that is important to making the coordination decision in the current situation.

4.1.3. Bottom-up vs. top-down agent coordination. The TÆMS task structure can be viewed as not only representing an agent's problem-solving activities but also as a repository of information about how these activities relate to activities in other agents (including knowledge about in what situations coordination will take place, and what commitments have been made among agents and their current status). From this perspective, it is not important how this repository is constructed (e.g., generated locally through dynamic information sharing and negotiations with other agents, statically predefined by the system designer, generated by another agent and then transferred to this agent, etc.); this is true as long as the mechanisms that use this repository make no assumptions about how the repository was constructed and how it will evolve. On the face of it, this is not a radical shift in thinking. However, it has important implications for GPGP since it extends the range of applications for which GPGP is appropriate[50] and it allows GPGP to inter-operate with other coordination schemes.

GPGP traces its origins to PGP, which was designed to handle coordination problems occurring in the DVMT (distributed vehicle monitoring testbed [41]). In order for agents to coordinate and communicate effectively in this distributed situation-assessment application domain, they need to understand that the vehicle they are about to track is being tracked by another agent who has overlapping sensors, that the previous track positions of a vehicle they are tracking had been or is in the process of being constructed by another agent, or that the vehicle they are tracking is part of a larger pattern of vehicle movements that are being tracked by other agents.

Each of these situations can be thought of as recognizing that the tracking goals of specific individual agents were, in reality, part of a higher-level goal to track a specific vehicle or formation of vehicles. This recognition needs to be dynamic because neither the number of vehicles nor the pattern of their movements was predefined. From this perspective, the agents were dynamically recognizing and constructing, where appropriate, shared high-level goal trees from a bottom-up perspective. The GPGP mechanism-1 (''communicate non-local views'') was responsible for this bottom-up construction process.[51] If we change this problem slightly, so that once an agent recognized the track of a vehicle locally, it could make very good predictions about what and when agents will sense that vehicle, and possibly other associated vehicles. In this case, an approach to coordinating the agents would be to have the initial recognizing agent generate a goal structure for tracking the vehicle in the different agents and then appropriately distribute it to relevant agents. This goal structure would consist of individual tracking subgoals for specific agents, and would also indicate how the subgoals were related to each other. For example, two tracking subgoals (one for *agent1* and the other for *agent2*) involved tracking the same fragment of track, since both agents were capable of sensing the vehicle in a specific region. Thus, in this case the generation of the goal tree is top-down and there is no reason for agents to discover interrelationships among their goals and other agents since all this information is part of the goal structure that is distributed to specific agents.[52] In this way, GPGP can provide effective coordination independent of whether the application employs a bottom-up creation and synthesis of agent task structures, or a top-down elaboration and assignment of agent task structures.[53] In fact, we foresee applications in which both approaches to generating agent task structures co-exist and operate concurrently.[54]

4.1.4. GPGP interactions with other components. In using GPGP as part of a real multi-agent system, we have developed a more complex view of the interaction among GPGP, the scheduler and the agent's problem-solving component. We have introduced an additional component: a (domain-specific) task assessor (see Figure 14). The problem solver and its associated task assessor use TÆMS to represent the tasks that will likely need to be carried out in the future. The methods that the task assessor provides to the scheduler represent higher-level entities, which may still require that numerous decisions be made before the actual selection of executable actions is accomplished (see [45] for a description of a task assessor in a real application). It is the problem-solving component that makes these decisions – possibly in a reactive manner. This is one way that we have accommodated the need for some reactivity (without necessitating repeated reschedulings) – by having the task assessor only abstractly plan how to accomplish tasks.[55] The agents' local coordination modules interact, possibly resulting in modifications to the agents' local task structures to represent inter-agent task relationships.

This information is used in implementing domain-independent coordination protocols that generate potential non-local agent commitments. The coordination-adapted task structures together with these potential commitments are then used by each agent's real-time scheduler to find the best sequences of activities to meet both local and network-wide objectives, and in the process determine whether the new

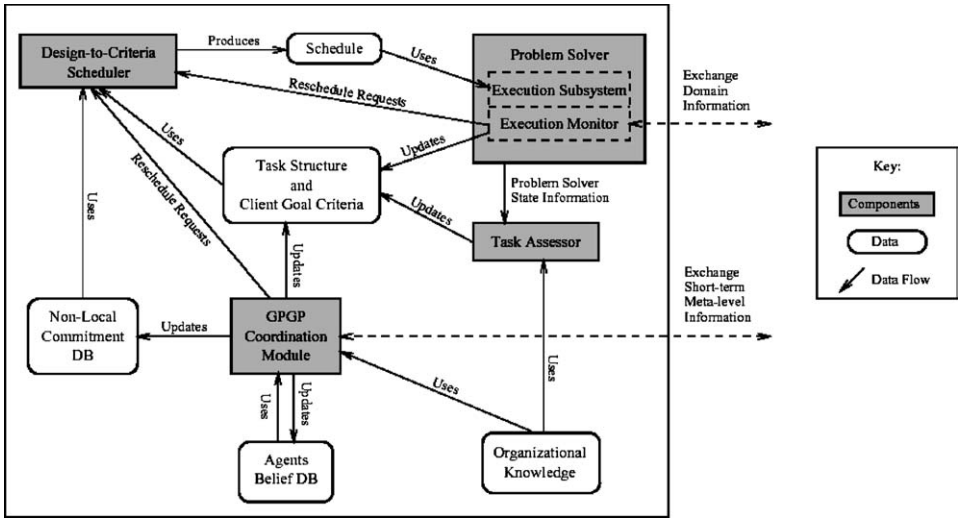

Figure 14. A typical agent architecture using GPGP.

potential commitments should be accepted and whether existing commitments should still be honored. The rejection of a potential or previously made commitment will cause the coordination mechanisms to be re-triggered; in some cases, the scheduler will offer suggestions how to modify the commitment to be more appropriate [21]. The resulting schedule of activities directs the problem solver, which ultimately determines what actions to take to best achieve a task. Organizational knowledge is used by the task assessor to condition the task structures, as described earlier, to allow for more efficient coordination by reducing the number of coordination options considered. The TÆMS description of the expected tasks created by the task assessor is used to update the current task model that is accessible to the scheduler and coordination module. This triggers the scheduler to generate a new schedule, which is passed back to the problem solver to direct the selection of actions. The schedule consists of an ordered list of methods[56] to be carried out. It implicitly represents the order in which the top-level tasks should be pursued as well as the approaches that should be used to pursue them. The scheduler also associates monitoring information with each method that indicates in what situations the problem solver should alert the scheduler that the partial execution of the method is not performing as expected.

We can also think of GPGP interacting with other coordination mechanisms (e.g., domain-specific, user-defined coordination rules built into, for instance, a BDI system integrated into the domain problem-solving component [4]) that, for instance, would generate and assign parts of goal trees to agents and make tentative commitments to the achievement of certain goals by specific agents. GPGP would then attempt to verify the feasibility of achieving these commitments in the context of the given goal structures using more detailed reasoning about resource usage and analysis of the comparative worth of achieving specific goals, and to further elaborate the commitments with, for instance, detailed temporal constraints and the level

of quality desired. This leads to a layered view of the use of GPGP [83], as pictured in Figure 15, where other non-GPGP coordination mechanisms would be embedded in the domain problem-solving component of the agent.

4.2. *More complex view of commitments*

The use of a more complex representation of commitments is motivated by two disparate issues. One was the development of coordination protocols that involved resource sharing among activities. (In the original GPGP, protocols only involved information-sharing relationships among agents.) The other was the desire to more fully exploit the knowledge about contingent/uncertain behavior that could now be represented in the extended version of TÆMS. This new knowledge involved the specification of distributions on duration, quality and cost behaviors of activities and the specification of alternative outcomes of activities.

4.2.1. Commitments for resource sharing. In the process of developing new mechanisms in GPGP to coordinate over shared resources, a new class of commitments called *don't* commitments have been introduced. They indicate to the local scheduler that a specific task cannot be executed during a certain time interval. Through the use of this type of commitment, a coordination mechanism can serialize access to a resource by a group of agents, or restrict the number of agents that will access the resource during a specific time interval. For example, we have defined a new coordination mechanism that uses a simple negotiation protocol to schedule access to any such mutually exclusive resource [12]. The idea behind this resource-constraint coordination mechanism is that when an agent intends to execute a resource-constrained task, it sends a bid of the time interval it needs and the local priority (projected utility) of its task. After a communication delay, it knows all the bids

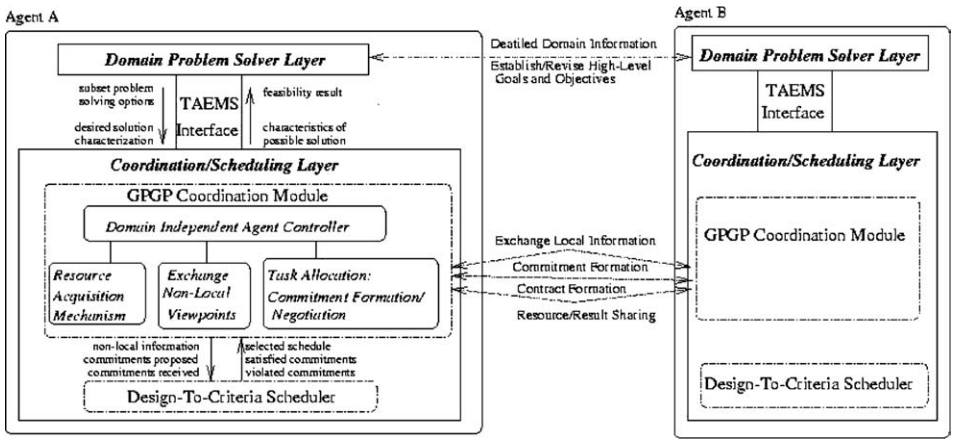

Figure 15. The layered use of GPGP.

given out by the other agents at the same time as its own bid. Since all the agents who bid will have the same information about others, if they all use the same commonly accepted rule to decide who will get the time interval, they can get the same result on this round of bidding. The agent who wins will keep its schedule and execute that task within its time interval, and everyone else will mark this time interval with a *don't* commitment, and never try to execute the resource-constrained task in it unless the owner gives it up. All the agents who didn't get their time intervals at the first round will recompute their schedules and bid again.

We have also developed a resource-coordination protocol [42, 43] that uses a different approach that involves the agent owning the resource to have a statistical model of the current reservations for this resource [64]. In this protocol, each resource request contains a priority that denotes the importance of the task for which the resource is being requested. Priority ranges are associated with the different tasks and these reflect the client's preferences. Conflicts occur when resources for a particular time period are insufficient to meet the demand. When this happens, agents holding resource confirmations for the time slot may be requested to release their reservation or may simply be told that their reservation is canceled. The affected agents can then decide to raise their priority and negate the release request or they can simply release the resource.

We have also explored another type of commitment for use in a distributed resource allocation problem [30, 31, 47]. This commitment, called a *periodic* commitment, allows us to represent commitments that reoccur at some fixed time interval. The use of this type of commitment reduces coordination costs by incurring the overhead of coordination only for the first time the commitment is established.

4.2.2. Uncertainty and the dynamics of commitments. Although commitments are often regarded as "contracts" that have a binding effect, it would be a misconception if commitments were treated as static objects. In fact, commitments are highly dynamic objects. Commitments are built on top of various sources of uncertainties in the agent problem-solving activities, therefore they are inherently uncertain. Let us look at the issue of de-commitment, for example. De-commitment often occurs for one of these reasons:

1. "Statistically unlucky": This refers to the fact that the tasks often have uncertain, sometimes undesirable outcomes (such as failure). In this case, if the task being promised in the commitment simply fails, the commitment fails because the agent does not have luck on its side for its current problem-solving episode.
2. "Infeasible": Sometimes, the task being promised in the commitment may depend on a number of conditions in order to be ready to run. These conditions include precondition tasks, and sufficient computational and non-computational resources needed for its execution. If one or more of these conditions are not satisfied, it is infeasible for the agent to execute the task successfully.
3. "Undesirable": It is also possible that although the agent can manage to satisfy the conditions for the commitment to be kept, it decides not to do so because keeping the commitment is not in the agent's current best interest; for example, the utility gain associated with the commitment has become less than the gain the

agent may receive if choosing a different action due to for instance a new task of high utility arriving at the agent after it made the original commitment.

The above discussion is focused on the agent who offers the commitment. Similarly, the agent who requested the commitment may find the commitment unnecessary if the conditions for its local use of the commitment are no longer valid, or the commitment offers no gain since the agent has decided on a more advantageous use of its resources. In these cases, the receiving agent may also prefer to retract its request for the commitment, and hence its need to de-commit. If we can predict the likelihood of these changes in the desirability of commitments, which is now possible with representation of outcomes and distributions in TÆMS, the question then arises of how to exploit this information so that agents can coordinate their activities more effectively.

As an example, let us study a commitment offered at time 0, indicating that agent X would complete task A for agent Y before time 160. The following are some scenarios that may occur due to the uncertainty and dynamics of problem solving:

Scenario 1. In typical commitment semantics, this commitment means that Y can expect to receive information about the completion of A by time 160. However, if task A has a 20% chance of failure, Y can only receive the utility gain 80% of the time. Furthermore, it means that Y needs to make contingency plans after time 160 when A fails. Thus, if Y knows about the uncertainty about this commitment, it may evaluate the commitment differently at time 0, and may have a better picture of its own overall utility.

Scenario 2. Continuing with the change added to the commitment, let us assume that task A actually depends on a preceding task B, which has a success rate of 70%, and B finishes at time 80 according to the schedule of X. Immediately we now know that 30% of the time A becomes infeasible. For Y, it would mean that the commitment has only a 56% chance of being delivered. This is another source of uncertainty for Y. New information would be added to the semantics of the commitment to indicate that there is a 30% chance of de-commit from X. More importantly, the new semantics should point out that the decision to de-commit or not will arrive at time 80. Thus, for Y, it means that another contingency schedule needs to be made at time 80. Given this information, agent Y will now have a more complete picture of the commitment at time 0.

Scenario 3. Again, following the change made in the previous scenario, we will address the issue of a commitment becoming undesirable or unnecessary. Here we need to introduce the concept of marginal gain (marginal loss). Namely, marginal gain (loss) is the difference between the agent's expected utility with and without receiving (offering) the commitment. Clearly, we need to take into account the uncertainty associated with the commitment in order to evaluate the expected utility with the commitment. For cooperative agents, the agents need to make sure that the sum of marginal gain and loss is positive, so that the total expected utility increases with the commitment. As such, if the value of the commitment decreases (in terms of the sum of marginal gain and loss), it is desirable for the agents to agree to de-commit, although the commitment is otherwise satisfiable. For example, let us assume that the task B has two outcomes when it does not fail: 40% of the time the

outcome quality is 5, and 30% of the time the quality is 2. Also, task *A*'s outcome quality depends on *B*'s outcome: *A* will have quality 6 if *B* has quality 2, and 3 if *B* has quality 2. As a result, the potential marginal gain in *Y* when *A*'s quality is 3 may be much less when *A*'s quality is 6, if *Y*'s reward depends on the quality of *A*'s outcome. In this case, agents might want to de-commit at time 80 when *B*'s outcome is 2 – if the total expected reward would be better for the agents to forgo the commitment and choose some other alternative. As such, it would be beneficial if the agent provides marginal gain/loss information in the commitment, so that the agent would be able to know the value of the commitment to the other agent. Then it could decide if the commitment is desirable/necessary, thereby allowing agents to make contingency plans earlier in the planning process.

In summary, we need to incorporate these three types of uncertainty information in the commitments, and treat commitments as dynamic objects in order to fully understand and evaluate the role of commitments, which serve as the key to agent coordination. In [78, 79] the need for more complex coordination protocols based on uncertainty of commitments is more fully addressed together with implementation details and experimental results. These experimental results indicate that the use of these new coordination protocols based on reasoning about commitment uncertainty and marginal gain and loss increases the sum of the combined utility of agents as compared to the results produced by the original GPGP protocols.

4.3. New and more complex coordination mechanisms

The development of new and more complex mechanisms for establishing commitments among agents evolved out of a desire to provide a more sophisticated and parameterized set of coordination protocols that offer a wider range of trade-offs between the overhead of coordination protocols and the optimality of the resulting coordination. In the original GPGP, the coordination mechanisms were basically single-shot in that there was no negotiation process (or distributed search) among agents. A commitment was established, or not, based on the requesting agent's local view of reasonableness, and the recipient agent's local view of whether the requested commitment was reasonable for it to commit to.[57] We call this a locally biased view of coordination optimization and it can lead to situations in which commitments though beneficial were not established. There was no idea of a cooperative search among agents to find an acceptable (or for that matter, best) commitment among agents based on some more encompassing view of the effect of the commitment on both agents. We begin this discussion by first examining how task allocation/contracting can be done in GPGP.

4.3.1. Contracting in GPGP. In the original conception of GPGP, all coordination decisions were based on agents already intending to perform certain activities as a result of the local goals that they were currently pursuing. There was no way for one agent to get another agent to generate results for it if that agent was not already intending to generate those results. Obviously, for many applications, agents may be structured so that they can pursue a large number of different types of local activities

and only when there is a specific need for the result of an activity will the agent want to execute that activity. In order to handle the coordination that arises out of this contracting model of agent interaction, we have introduced a new coordination mechanism with associated additional information in the TÆMS task structure. The development and character of this mechanism is consistent with the work described above on the use of default knowledge and situation-specific control. It is also interesting to note that this new mechanism is valuable even in applications which use bottom-up, data-directed processing strategies such as distributed situation assessment.

The new contracting mechanism models the *potential* for external interaction using "virtual tasks." A task structure is created at an agent with a "possible-task" or "virtual task" or "partially instantiated" task that abstractly describes results that could be generated by another agent. These results are described in terms of a goal that another agent could satisfy. Virtual tasks are related to an agent's task structures via *non-local-effect (NLE)* (e.g., *enables*, *facilitates*, and so on). If an agent wishes to take advantage of this *NLE*, it generates a goal, describing its desired outcome, and transmits it to the appropriate agent. The receiving agent then instantiates a task structure, based on the characteristics of the goal. The actual question of whether the receiving agent is capable of or interested in performing the task is handled by that agent's domain problem-solving component.

The purpose of the virtual task is to serve as a placeholder in scheduling. An agent can attempt to develop schedules including the contributions of the virtual task to determine whether it can indeed develop feasible schedules were that task to be executed. This can take place before the actual communication with the remote agent. It is the modeling of the utility of the virtual task using the discrete probability distribution features of the TÆMS framework that distinguishes the GPGP contracting mechanism from a conventional goal-driven system. The incorporation of the TÆMS relationships allows the system to model in advance the contributions of the actions of the remote agent, to reason about the nature and magnitude of the coordination relationships required, and to parameterize the request according to this information, as well as deadlines for transmitting the information represented by the coordination relationships.

Consider the idealized situation depicted in Figure 16. *Agent 1* has a choice of three methods for achieving task T_{11}. Two of these methods can be facilitated by the actions of other agents. Unlike the original GPGP, there is no presupposition that the tasks will actually be performed unless the agent specifically requests them. Each local method has different performance characteristics. Method M_{11}, for example, has a potential quality of 6, a duration of 8, and a cost of 20. Based purely on a local perspective, and weighing quality of solution most highly, this would be the method of choice. All other methods produce lower quality solutions, although with lower costs and/or durations. However, by understanding in detail how other agents' activities may interact with its own, the agent is able to predict that, were other agents to execute the appropriate tasks, methods M_{12} and M_{13} would be facilitated with quality/cost/duration tuples of (7,8,10) and (7,5,15) respectively. Knowing that higher quality can be achieved, *Agent 1* can now reason about the coordination issues involved in ensuring that another agent executes the appropriate task in a

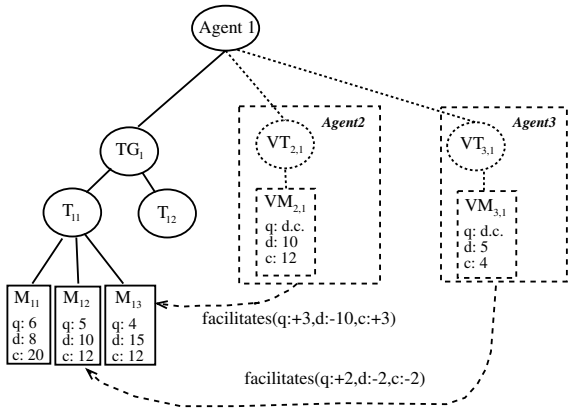

Figure 16. Example of agent reasoning about how best to contract for needed information from another agent.

timely fashion. First, it can use secondary criteria such as cost and duration to choose which task should be executed by a remote agent. Duration is of consideration both in scheduling later activities and in ensuring that sufficient time exists for a request to be transmitted, processed, and for a response to be sent. The agent may or may not have information regarding the cost of executing the tasks by the remote agents and the duration of these tasks – information that might be useful in determining *which* agent to ask for assistance.

Once a remote agent and remote task have been selected, the agent must transmit a request that the task be executed. This request can be annotated with information provided by the local scheduler, indicating the desired parameters under which the task should be executed, including but not limited to the quality, cost, and duration modifiers of the transmitted information, and the preferred finish time of the task. We next describe how this single-shot task allocation coordination mechanism can be extended to allow for a cooperative search process among agents to find an appropriate commitment and describe the conditions for an agent to accept or reject a contracting request.

4.3.2. Multi-step coordination protocols. Before discussing how to structure a cooperative search process among agents, it is crucial to understand the criteria/objectives being maximized in the search process; the criteria chosen are very strongly related to the scope and nature (different degrees) of cooperation among agents. Having appropriate criteria allows agents to reason more accurately about how a specific task allocation commitment will affect overall system performance. The most encompassing form of cooperation is what has been called "global cooperation" [38], which occurs when an agent, while making its local decision, always tries to maximize the global utility function which in GPGP is the combined sum of the utilities produced from the activities of all agents in the system. Global cooperation is unachievable in most realistic situations because of the number of agents and bounds on computational power and bandwidth. Thus we focus our approach on organizing

a cooperative distributed search among agents to find an acceptable commitment based on "local cooperation" [38]. This type of cooperation occurs when two or more agents, while negotiating over an issue, try to find a solution that increases the sum of their local utilities, without taking into account the rest of the agents in the system – we call this a mutually biased view. As mentioned briefly in Scenario 3 in Section 4.2.2 on "Uncertainty and Dynamics in Commitments," we introduce the notion of marginal utility gain and cost (loss) as a way of computing the effect of a specific commitment on the two agents involved in the negotiation.

In [84], we developed a multi-dimensional, multi-step negotiation mechanism for task allocation that uses marginal utility gain and marginal utility cost to structure this search process to find a solution that maximizes the agents' combined utility.[58] These two utility values together with temporal constraints generated as a result of an agent's local search process summarize the agents' local information used in the negotiation process, and thus reduce the communication bandwidth necessary to implement the negotiation. This is a multi-step negotiation process since agents engage in a series of proposals and counter-offers to decide whether the contractee agent will perform a task for the contractor agent. The proposals and counter-offers specify ranges of acceptable times for what is the earliest time a contracted task can be started and when it needs to be completed. Furthermore, this negotiation is over multiple dimensions (by a specified time with a certain minimum quality) rather than over a single dimension. For example, agent A wants agent B to do task T for it by time 10 and starting it no earlier than time 3, and requests the minimum quality of 8 for the task to be achieved. Agent B replies that it can do task T by time 10 but only with the quality of 6; however, if agent A can wait until time 15, it can get a quality of 12. Agent A will select the alternative it believes is better for both agents. Since the negotiation relates to both the completion time and achieved quality of the task, the scope of the search space for the negotiation is increased and thus improves the agents' chance of finding a solution that increases the combined utility.

Figure 17 shows a finite state machine (**FSM**) model that describes this multi-step negotiation protocol. The requesting agent (initiator) starts the negotiation by building a proposal (*buildproposal*) and sending this proposal (*sndmsgproposal*) to the receiving agent (responder). After receiving this proposal (*rcvmsgproposal*), the responder agent evaluates it (*evalproposal*): if the marginal utility cost is less than the marginal utility gain, it accepts this proposal (*sndmsgaccept*); otherwise, this proposal is rejected, the responder agent builds a counter-proposal (*buildcounter-proposal*) and sends it to the initiator agent (*sndmsgcounter-proposal*). When the initiator agent gets this counter-proposal (*rcvmsgcounterproposal*), it evaluates this counter-proposal (*evalcounterproposal*). If the counter-proposal is acceptable and there already are a sufficient number of solutions (a solution is an acceptable proposal if the marginal utility gain is greater than the marginal utility cost), the negotiation is terminated and the initiator agent informs the responder agent which proposal is finally chosen (*sndmsgfinish*); otherwise, the initiator agent generates a new proposal based on its previous proposal and the current proposal (*generate-newproposal*), and starts another round of communication.

This mechanism is *anytime* in character: by investing more time, the agents increase the likelihood of getting a better solution. A parameterized version of this

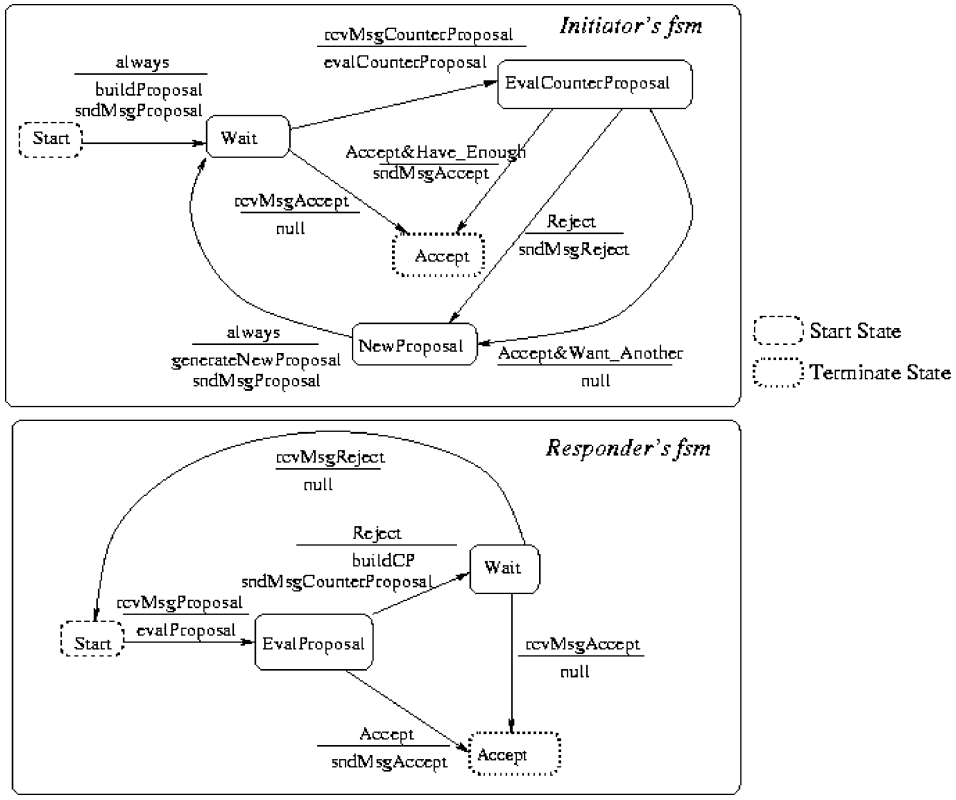

Figure 17. Multi-step negotiation mechanism.

coordination mechanism allows control over the amount of search done among agents – from a single-shot mechanism, all the way to an unlimited search for the best solution. Experimental work shows a phase-transition-like phenomenon in the operation of the negotiation mechanism: when the negotiation situation is very simple or very difficult, extra negotiation effort does not bring reasonable extra gain; when the negotiation situation is of medium difficulty, the extra gain exceeds the extra effort required by the protocols that do more search. As part of this effort, a measure of negotiation complexity that is based on local measures of agent task complexity was developed that can be used by an agent to choose the appropriate protocol through the exchange of limited meta-level information. This measure allows agents to explicitly balance the gain from the negotiation and the resource usage of the negotiation.[59]

5. Discussion, conclusions and future directions

The experiences that we have had in applying GPGP/TÆMS to a number of different applications and the ability to extend GPGP/TÆMS as described in this paper

have led us to believe that our basic approach to coordination, based on quantitative coordination relationships represented in TÆMS and the generation of commitments among agents that then constrain local agent scheduling, is a powerful and general framework for implementing sophisticated, domain-independent on-line coordination mechanisms. Further, GPGP/TÆMS can be naturally extended to be highly situation-specific where the overhead for coordination can be adjusted for the specific coordination situation. This can be accomplished by substituting *a priori* knowledge for dynamically acquired knowledge and dynamically generated commitments and the use of a *conditioned* TÆMS task structure. Additionally, GPGP has been able to be easily extended to allow the implementation of more top-down and contracting types of coordination mechanisms. These extensions are what make GPGP applicable to a wider range of multi-agent domains that require both top-down and bottom-up coordination regimes and more easily integratable into a multi-layered control architecture in which organizational design rules and/or domain-specific coordination protocols define policies for multi-agent coordination [40].

Though we have not formally shown the representational completeness of this framework, it is our intuition that without significant extensions it will be able to represent and effectively reason about most high-level coordination issues that occur in multi-agent applications. In general, the objective with GPGP coordination is to achieve coordination results that are as close as possible to optimal but to do this in a distributed, approximate, soft real-time fashion without significant agent communication. In a perfect world we would have a general purpose centralized optimal computational framework against which to benchmark all distributed techniques. However, the large feature set of TÆMS has thus far thwarted efforts to create *efficient* centralized reasoning frameworks – the implications are that optimal solutions cannot be generated for even modest problems in the general case due to computational complexity. However, in two recent experiments we have been able to benchmark GPGP against an optimal coordination strategy and have shown that its performance is close to optimal.

In the aircraft service team coordination application discussed previously [74], it was possible to construct a specialized optimal centralized scheduler due to the more limited TÆMS feature set needed by the application domain. In this case, the distributed GPGP-style approach returned results whose worst case was within 98% of optimal for generated test cases. Note that these results were produced in approximately 1/10th of the time required by the centralized scheduler and that testing was constrained to moderate sized problem instances to stay within the computational limitations of the centralized approach. (See the paper [74] for additional discussion, assumptions, and limitations.) Another experiment [78] associated with work described in Section 4.2.2 on "Uncertainty and the Dynamics of Commitments" also indicated similar results. In this case, we used a relatively simple two-agent coordination scenario that required some level of reasoning about coordination contingencies based on method outcome. The scenario consisted of two interacting TÆMS task structures (one for each agent) and the overall utility was the sum of the local utilities of the agents. We have developed a way for translating this type of problem into a centralized MDP with each action in the MDP representing the joint actions of both agents.[60] In this case, the centralized MDP did not take into account the cost

of communication between agents in order to update each other about the result of each action being executed – in essence it was assumed that a global view that included the current state of both agents could be used in making on-line coordination decisions. The results of this example scenario were that the complex commitments version of GPGP, on a statistical basis over repeated runs with different method outcomes, achieved 96% of the utility produced by the optimal centralized MDP coordination strategy. Additionally, GPGP required significantly less communication during execution than was needed when implementing the optimal coordination strategy in decentralized agents that did not have access to the global view.

Another important performance aspect to be considered in evaluating GPGP's appropriateness is its overhead, which includes its use of processor and communication resources and the delays introduced in starting up activities as a result of waiting for coordination decisions to be made. The exact details of this coordination overhead are difficult to predict since they are based on many interdependent factors: the time granularity of the primitive activities, the performance characteristics of the communication medium, the complexity of the TÆMS task structures, the frequency of interrelationships among tasks at different agents, the frequency of generation of new tasks, the need for re-coordination based on the amount and character of the uncertainty in agent behavior, and the level of default knowledge that can be exploited. Further, many of these factors can be adjusted to reflect less optimal coordination such as putting slack time in coordination commitments so as to reduce the frequency of re-coordination [16], not coordinating over relationships that do not result in significantly improved task performance, limiting the amount of effort to put into scheduling analysis so as to produce less optimal schedules more quickly, etc. Unfortunately, we have not done the experimental work necessary to understand the interplay among all these factors; however, we have shown experimentally that significant reduction in coordination overhead can occur by tailoring the use of the coordination mechanism to both the overall characteristics of the application environment (static tailoring) [11, 14][61] and the current coordination situation (dynamic tailoring) [48]. In part, this more extensive experimentation has not occurred because we never expended the considerable effort in coding required to get a highly efficient implementation of GPGP given that GPGP was never used in situations where its overhead was critical to its successful use. However, we have recently done a lot of work to speed up local agent scheduling and re-scheduling of TÆMS task structures [29, 64] in a soft real-time application. It is this type of processing that is at the heart of the computational costs associated with GPGP. Based on this experience, it is our guess that we can get coordination costs down to under 1 second, running on a current commodity processor, for generating coordination commitments for relatively small task structures.

The one caveat to our claim of representational adequacy and computation efficiency of the GPGP framework involves coordination issues involving complex, multi-stage contingencies among activities at different agents. The GPGP approach of generating and adapting a coordination strategy by incrementally reasoning about coordination relationships may not be able to generate reasonable strategies for such coordination issues. As discussed in Section 2.1, for issues involving complex

contingencies, representations and solution techniques associated with distributed Markov-Decision Processes of agent activities that take a more global and detailed view of activities may be more suitable. The drawbacks of this approach are the computational and communication demands of optimally solving coordination problems [3] in situations with only two agents that at this point precludes the use of this as an on-line framework for coordination. The hope is that approximate solution techniques will evolve for an appropriately rich class of agent coordination issues where the optimality of the solution generated can be bounded, and computational and communication requirements are sufficiently limited, making on-line coordination possible. For now, the GPGP/TÆMS coordination framework seems to be the only practical framework suitable to handle a wide range of high-level coordination issues on-line that involve quantitative (decision-theoretic) reasoning in a domain-independent manner.

Our plans for further extending GPGP fall into the following major categories: (1) to explore the interplay between local, environment-centered coordination mechanisms and coordination of large agent organizations [6, 7, 66]; (2) to expand the range of possible coordination mechanisms, including those that might make sense only in a large organizational context [66]; (3) to experiment with different coordination protocols and determine their contextually dependent efficacy; (4) to make GPGP operate in a soft, real-time environment where the true costs of coordination are factored into dynamically made decisions about what and how much effort to put into coordination [53, 54], and when[62]; (5) to allow for multiple coordination mechanisms to operate concurrently [70, 74]; (6) to explore the interplay between domain-specific coordination policies and other approaches to coordination with GPGP [4, 83]; and (7) to connect the GPGP approach to coordination with more formal models of coordination [9, 67, 78, 80].

In closing, it would be remiss of us not to mention the significance that we attach to learning. As we build multi-agent systems that operate in open and evolving environments, the systems must be able to adapt their heuristics/assumptions (and there are many in GPGP), to the current structure of the environment – not to some predefined view of the environment [28, 35, 48, 53, 54, 61].

Acknowledgments

As is obvious by the author list, the development and evolution of GPGP has been a long process that has been the focus of much of the intellectual directions of the Multi-Agent System Laboratory at University of Massachusetts at Amherst. Where appropriate, we have referenced papers by specific members of the laboratory who took major responsibility in developing a research idea explained in this paper. However, it should go without saying that there were many interchanges with, and building on ideas developed by, other laboratory members who are not directly recorded in article citations. Though they did not directly contribute to the development of GPGP, we would also like to acknowledge some long-term visitors to our laboratory, other laboratory members, and associated faculty who were part of the intellectual ferment that generated the ideas expressed here. They are: Michael

Atighetchi, Ana Bazzan, Brett Benyo, Satoru Fujita, David Jensen, Susan Lander, Qiegang Long, Roger Mailler, Lee Osterweil, Tuomas Sandholm, Jiaying Shen, Toshiharu Sugawara, Robert Whitehair and Shlomo Zilberstein. We would also like to thank Sherief Abdallah for his thoughtful comments on this paper, and recognize the contributions of Valerie Guralnik and John Phelps who are working with Tom Wagner at Honeywell Laboratories on the commercial application of the GPGP/TÆMS technology. They have contributed to the growth of TÆMS framework and the creation of new GPGP-style coordination mechanisms. Finally, it almost goes without saying, this paper owes an intellectual debt to the pioneering work of our early laboratory members, Daniel Corkill and Edmund Durfee.

Notes

1. This material is based upon work that has been sponsored by the National Science Foundation (under grant nos. IIS-9812755 and IIS-9988784) and the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF (under agreement number F30602-99-2-0525). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DARPA, Air Force Research Laboratory or the US Government. The US Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon.
2. This is a significant extension, including additional authors, of a paper with a similar title [70] written in 1998.
3. PGP took a mediation-based approach to coordination where each agent received the high-level plans of some subset of agents and then re-ordered its and other agents' plans to optimize coordinated activity based on interrelationships among activities of different agents. In a peer-to-peer agent organization, the results of this optimization only affected local agent activity. It was assumed that other agents, if they had similar information, would arrive at the same coordination strategy and thus the plans of each agent would be consistent. In PGP there were no additional mechanisms to resolve inconsistent plans that could be potentially generated by agents that had partially overlapping groups of agents that they were coordinating over. As will be discussed, GPGP takes a more distributed view of coordination that is accomplished by generating commitments among agents; thus the generation of inconsistent plans does not occur though the plans generated could be less optimal since PGP takes a more centralized view of the optimization process. GPGP also formalizes the interrelationships among agent activities in a domain-independent way, and separates coordination activities from local agent control through a bi-directional interface, thus adding another degree of domain independence.
4. There are alternative views of coordination that are not based on explicit or implicit commitments among agents but rather on statistically based views of agent interactions. This statistical view of coordination seems most appropriate in environments composed of large numbers of very simple agents.
5. Though as will be discussed, TÆMS provides a rich language for the representation of agent activities and alternative ways of achieving goals.
6. A subtask can potentially represent a significant body of code or a significant set of activities.
7. One of the directions in extending GPGP that will be discussed in Section 5 involves reducing the cost of coordination. The issue of what are acceptable overheads of coordination will be discussed further in Section 6.
8. In recent work on a distributed sensor network application [30, 31], we used an agent architecture [29, 64] with such reasoning capabilities to create a virtual agent organization of simple, single-threaded agents. These "virtual" agents were created as needed and dynamically assigned as goals to a specific agent based on gross information about current resource usage of agents and the type of resources available at each agent. In the hardware implementation of this architecture, there was one agent for

each processor node. The "real" agent then performed detailed planning/scheduling based on local resource availability and priority of these goals, and multiplexed among the different goals that it was concurrently executing in order to meet soft real-time requirements. The use of agents with this type of reasoning capability has also helped us to construct a resource negotiation protocol that operates on an abstract model of resources and does not need to resolve all conflicts to be successful [47]. In this case, the agent took on the responsibility of mapping the abstract resource allocation policy generated by the negotiation protocol into a detailed resource-allocation schedule and where possible, solved resource-allocation conflicts at the abstract level through local shifting and modification of tasks.

9. Our discussion in the paper assumes that the agents are relatively homogeneous. By relatively homogeneous, we mean that the agents can have different problem-solving architectures but all use the same set of domain-independent planner/scheduler and coordination modules to order their internal activities and communicate their intentions. However, this is not a strict requirement for using the coordination framework laid out in the paper because all that is needed to implement it is for an agent to have local scheduling capabilities and problem-solving components that can appropriately interact with the framework. This means that these local components need to be able to recognize if and how a task at another agent relates to its activities in terms of certain pre-defined coordination relationships, to understand the implications for its local problem solving of commitments specified by the coordination framework, and to be able to specify local problem-solving tasks to the coordination framework that may be involved in coordination decisions.
10. TEAMCORE/STEAM allows for solving concurrent goals as long as there is no resource contention or interdependencies among the goals, i.e., they are totally independent.
11. ARCHON does allow for some very simple scheduling of tasks taking into account their time and worth.
12. In the initial implementation of GPGP using the DTT Scheduler [20], there was a bias in local decision making towards achieving some minimal satisfaction of every high-level goal rather than maximizing the overall worth of the high-level goals achieved. In the more recent DTC Scheduler, that bias has been eliminated and the scheduler works to maximize the local utility function [71] specified by the coordination module or the agent's domain problem solver.
13. This issue will be further discussed in Section 6.
14. In the work on the DECAF agent architecture [23–25] that uses a variant of TÆMS, a simpler scheduler than DTC, called DRU, was used [22].
15. In the current implementation, even if the GPGP module is aware that there are multiple relationships among goals of different agents that need to be taken into account, it will treat them independently rather than as a unified group of relationships. There is current work on trying to take into account these other relationships [82]. Additionally, GPGP currently operates mainly on relationships that span only two agents except for one mechanism (avoiding redundant activity) that can handle the situation where two or more agents are planning to do the same activity. Again, the fact that existing GPGP mechanisms operate mainly in the context of two agents is not inherent in the approach but was initially done to simplify the protocols among GPGP modules.
16. Over the years, we have explored a number of techniques for reducing the frequency of complete re-planning, re-scheduling, and re-coordination. The ability to reduce the need for re-coordination is also strongly related to the ability of local agent control to find alternative schedules for its activities, in the event of performance deviation, without breaking existing commitments. These techniques have included adding slack time into schedules, local re-ordering of scheduled activities based on existing constraints and commitments, incremental planning and scheduling as new tasks arrive, replacement of scheduled activities with semantically equivalent activities that use fewer resources but have poorer performance [16, 19, 64].
17. In this discussion, we have not discussed the possibility of generating non-optimal distributed MDP policies on-line quickly (which, though approximate, are very good policies), or trying to dynamically adapt a smaller set of optimal policies for a selected set of scenarios to work non-optimally for a scenario in the larger set of possible scenarios. These are possible approaches that may turn out to be very useful to have in a toolkit of approaches to coordination.
18. As implemented in [66, 68], a simple model of future task arrival could be easily incorporated into GPGP through an opportunity cost model.

19. These constraints consist of both simple constraints on the temporal ordering of activities, such as the earliest time that an activity can start, the latest time it can finish, and a time interval at which the activity cannot occur and more complex constraints such as guaranteeing that an activity (or related group of activities) is completed with a certain quality by a specified time.
20. Schedules, as previously discussed, are revised or replaced as new information arrives or the problem-solving context changes. This end-to-end view, which is necessary in order to meet overall real-time and resource constraints, starts at the current time and goes to the latest finished time of the current goals. This is in contrast to the myopic view employed in [32].
21. Though resource consumption characterizations were part of the specification of TÆMS as defined in [14] they were not used as a basis of coordination in the original GPGP.
22. As will be discussed later, TÆMS has been extended to include outcomes on methods. In this way, a limited and local representation of domain state can be represented. Additionally, pseudo resources can be created to add in some capability for representing a more global domain state.
23. An agent's problem-solving component is expected to modify the task structure to reflect its revised view of its expected domain problem-solving activities.
24. Task relationships as specified currently in TÆMS only involve relationships between two tasks. There is no reason that more complex relationships irreducibly involving more than two tasks could not be created. However, we have not needed such relationships in effectively modeling subtask interdependencies for the applications we have so far studied. It should be noted however that *qaf's* represent relationships among an arbitrary number of subtasks.
25. The *enable*s relationship between tasks at different agents indicates that a result needs to be transferred, after the enabling task is completed, to the enabled task at the other agent before it can execute. There are other types of semantics that could be associated with the type of precedence relationship represented by *enables*. For instance in STEAM [51, 62, 63], there is a role dependency relationship between tasks that does not imply information transfer. Rather, it indicates that one task has created a state of the world that is necessary for the execution of the other task. A transfer of a message from one agent to the other can indicate the existence of this state of the world or it can just be observed by the dependent task. We can easily model this type of ''role dependency'' relationship in our framework, either through the use of a new relationship that is similar to *enables* but without the information transfer, or with the use of the *producer/consumer* relationship on a pseudo resource which represents the state of the world that is desired. In fact in [14, pp. 54–58], a more complete list of relationships (*facilitates*, *hinders*, *precedence*, *causes*, *shares-results*, *cancels, favors*) are specified, many of them oriented toward modeling interrelationships among non-computational activities such as ''*causes*'' where the result of completing one activity results in the completion of another activity.
26. The name GPGP derives from Durfee's Partial Global Planning [18], which by today's definition of AI planning also focuses on the scheduling end of the planning-scheduling spectrum.
27. The question of which mechanism to use when more than one is possible is a question about organizational/societal norms.
28. In fact, there are two primary classes of reasoning embodied in the GPGP/scheduler pair. The first class pertains to the reasoning required to form and propose commitments with the other agents, the second class pertains to understanding how the choice of actions is affected by commitments and when/if commitments may realistically be satisfied. In [4, 69, 83], we further separate these two different types of activities so that commitments may be specified by components other than the GPGP coordination module.
29. Because the agents are fully cooperative, in the global view, there is actually a root task common to all the agents that joins their activities. In other words, the cooperative relationship that results in the summing of the utilities produced by the individual agents is represented explicitly in TÆMS by placing a super task over the tasks of the individual agents. In this example we have omitted the root task to clarify the different world views held by the different agents.
30. Meta-level control techniques for intelligently interleaving coordination activities and domain activities will not be described in this paper. They are a subject of recent work and can be found in [53, 54].
31. We are assuming there is domain-dependent knowledge residing at each agent, which is possibly generated dynamically, that indicates which agents are likely to have activities that are interrelated to the agent's current activities or have resources that could be helpful in performing these activities. One

of the extensions addressed later in the article in the subsection on a conditioned view discusses in detail how this type of domain-dependent knowledge can be incorporated into the TÆMS representation.

32. In the experimental work cited in [11, 14], the norm for most experiments was to exchange only partial views.
33. Although decommitment penalties were not used, an alternative, more qualitative mechanism was used to make decisions about what commitments could be decommitted from, and in what circumstances. A commitment when initially constructed was marked with different ''levels of negotiability,'' which reflected its relative importance.
34. STEAM/TEAMCORE framework also extends the original joint-intentions model in some important practical ways such as providing mechanism/reasoning so that agents' can (1) switch roles if one of their teammates fails in fulfilling its role, and (2) selectively choose whether to follow the prescriptions of the joint-intention model based on a decision theoretic analysis of the costs of communication in the current situation.
35. We have also explored more complex heuristics involving load balancing [14, p. 163–164].
36. When non-local tasks were connected to local tasks as a result of task relationships such as *enables*, this would trigger a coordination mechanism to be invoked. If successful, it would result in a commitment that would guarantee that the result needed by the local task would be generated in a timely way. If a commitment could not be achieved successfully, then either an alternative way of solving the local task that did not need the non-enabled task was found, or the local task would not be scheduled for execution. This knowledge of the resultant local scheduling decision would be transmitted to the other agent so it could update the expected utility to be produced by doing its local task as part of the larger task. If this utility were zero, the other agent would not schedule tasks associated with its part of the shared goal. However, this local utility as contained in a non-local commitment at a remote agent was not always changed when it changed at the local agent because too much communication was needed.
37. We already send messages alerting agents of failed commitments. What we do not handle correctly in all situations is a case where the lack of successful generation of a commitment may still allow agents to find another way to solve their part of the joint goal, nor the case where there are no task interrelationships among the task structures that make up the shared goal and there is only a *qaf* relationship that defines the connection among activities. However, it should be mentioned that none of these issues arise when one agent contracts with another agent to do part of a higher-level goal, i.e., top-down coordination. All the problematic situations described above occur when agents need to recognize that their local activity is part of a higher-level goal to which other agents are also contributing, i.e., bottom-up coordination. In this case, the execution of their local activity does not make any sense unless other agents are committed to executing their local activity that is part of the jointly shared goal.
38. In [1], it is shown that GPGP/TÆMS with some extensions can be extended to capture the joint-intention model of STEAM.
39. In STEAM, there are some quantitative thresholds that are used to indicate whether a goal was successfully achieved, e.g., 70% of the planes reached their destination. However, to our knowledge there is no reasoning in STEAM about how to dynamically set this threshold given current conditions and how to generate different plans that will likely achieve different thresholds.
40. The addition to *qaf*s of task ordering information such as in *q_seq_sum* has provoked much discussion. One argument against inclusion of this type of information is that with the appropriate set of temporal relationships among tasks there is no need for the addition of ordering information to *qaf*s. However, we eventually came down on the side of ease of representation that occurs when in certain situations the *qaf*s contain such information.
41. However, we have not developed the equivalent of *qaf*s that are able to specify how outcomes at the method level should be propagated to outcomes of higher-level tasks. The reason is that these functions seem very domain-dependent and not amenable to systemization. This can be solved in part by having code in the domain problem solver that will compute the higher-level outcome to be associated with a specific higher-level task. This can be represented by slightly modifying the task structure such that each higher-level task that requires an outcome will have a brother task, which is a method with

the set of outcomes that are possible for its brother higher-level task. The method, when executed after its brother higher-level task completes, is in reality a call to the appropriate code in the domain problem solver that computes the desired outcome to be associated with the method. Unfortunately, this particular approach does not solve the problem completely since there is no way for the scheduler to reason about the likelihood of different higher-level tasks' outcomes given a specific plan to achieve the higher-level task. An alternative approach to solving this problem was taken in the DECAF architecture [23–25]. In DECAF, task outcomes are signaled by the child tasks (and thus the outcome probability is the same as that of the child outcome). The outcome of a parent task does not need to be signaled by the same, or even a unique, child.

42. The actual information-gathering task structure does not incorporate outcomes at the task level. This example is a conceptual abstraction of the actual class of task structures produced by the information-gathering agent's planner and is simplified for example purposes.
43. In this more abstracted view, a schedule of activities generated is more like a high-level policy indicating to the domain problem solver which approaches to problem solving to use when, and what amount of resources to expend on each approach [45].
44. Examples of modeling frameworks that have more complex control constructs are [59, 77].
45. More generally, for two tasks t_a and t_b with start and end times, you can define a class of temporal synchronization relationships based on defining four Boolean functions. $F_1^i(\text{start}(t_a), \text{start}(t_b))$, $F_2^i(\text{end}(t_a), \text{end}(t_b))$, $F_3^i(\text{start}(t_a), \text{end}(t_b))$ and $F_4^i(\text{end}(t_a), \text{start}(t_b))$ such that all need to be true for relationship i to be true.
46. The need for this meta-level relationship has been recognized in our own work [61] and by others [1].
47. We found this very useful in an information-gathering application [45] where we need to model methods that access a WWW server site for information. In this case, there was not that much processor time spent in executing the call to the server and processing the received information, but there was a significant delay before the results were returned by the server. In [14, p. 67], a variant of this idea was defined as the coordination relationship *requires-delay*.
48. This is a simplified version of the task structure used in [28].
49. An easy extension of this idea that we have not yet implemented is for either agent to notify the other that it either cannot honor the *a priori* commitment or no longer has any need for the commitment to be honored.
50. The original STEAM framework [51, 61, 62] used an approach to coordination where it was not necessary for agents to exchange knowledge about what their activity tasks were. This occurred for two reasons: one reason was that there was only one higher-level goal active at any one time, and the other was that once an agent decided to pursue its part of a higher-level goal the coordination framework made the assumption that the other agents who had roles that mandate work on this goal would also reach a similar conclusion that this specific higher-level goal was the goal that they should now work on. This type of social law can now be easily implemented in our system through appropriate generation of commitments by the domain problem solver when it generates new goals/task structures that it wants to pursue.
51. Though the PGP mainly involved the bottom–up creation of subgoals and the recognition of how these subgoals were part of a larger goal, PGP did allow for hierarchical control [18] and task contracting [17].
52. In fact, if you knew that there would be only one vehicle in the environment, then the information contained in the generated goal structure could specify commitments on exactly when certain activities would be completed. In [19], we explored such an approach to hierarchical control where significant deviations in meeting commitments would then cause the higher agent to be notified and for it to generate a new more appropriate global coordination plan. However, in a multiple vehicle environment you want the agents to coordinate with appropriate agents since one agent may be working on more than one vehicle-tracking task.
53. In early research [60], bottom-up creation/synthesis of goals was associated with a style of problem solving called *result sharing,* while the top-down elaboration/assignment of goals was associated with the style of problem solving called *task sharing*.
54. In [37], the distributed and asynchronous goal creation process possible in a complex multi-agent system was described. This process involves four major phases that include: (1) *goal formulation*, which

involves the creation of new goals from either a bottom-up or top-down perspective; (2) *active goal determination*, which involves deciding which goals should now be scheduled; (3) *goal allocation*, which involves deciding which agents will solve the goal; and (4) *goal achievement*, which involves planning how to achieve the goal, how to schedule the derived plan and then executing the plan including transmission of results. We feel that this process can be mirrored in the extended GPGP framework.

55. In recent work on SRTA, a soft real-time agent architecture [64], we developed a variety of mechanisms to avoid complete rescheduling when there are ways to do local shifting of activities – replacing activities with functionally similar but less resource-intensive activities, and inserting a precomputed sequence of activities to recover from execution failures (such precomputed recoveries may be learned over time).
56. In recent work [64, 74, 82], we have gone to a partially ordered list that is a more flexible representation.
57. For example if Task *A* in Agent *1 enabled* a Task *B* in Agent *2*, then Agent *1* would initiate the establishment of a commitment. Agent *1* in deciding when to do Task *A* would call its local scheduler to determine the earliest time that Task *A* could be done while still achieving reasonable utility for its other local activities. The results of the scheduling process would then define the completion time of Task *A* in the proposed commitment that was transmitted to Agent *2*. This proposed commitment by Agent *1* to complete Task *A* by a specified time could be either accepted or rejected by Agent *2* depending on whether it was still worthwhile for Agent *2* to execute Task *B* given that its earliest start time was the completion time of Task *A* plus some time for message transmission.
58. A very similar cooperative search process, which we have implemented, can also be used to establish coordination for non-local relationships such as *enables* and *facilitates*. We believe that a variant of this same search process can be used to implement coordination over *qaf*'s such as *min*, *max* and *sum*.
59. An interesting research question that we are just beginning to address is how this cooperative search process can be extended to more than two agents. The need for this type of a more encompassing search process occurs in a variety of different forms of multi-linking of activities among a set of agents [82].
60. The translation of a TÆMS task structure into an MDP is relatively straightforward [56]. The set of actions possible from the start state in the MDP are all the methods in the task structure that can be executed without the successful execution of a prior task. These are methods whose execution is not precluded as a result of a *qaf* relationship in the task network; the method does not have an incoming *enables* from another method that has not successfully executed and the method's earliest start time is less than or equal to the current start time. Each of the possible states that result from the execution of a method can be thought of as a partial instantiation of the starting TÆMS task structure. The reward associated with a state is the change in quality that is accrued at the root node of the task structure as a result of the particular action outcome. The reasoning about the next set of methods that can be executed from one of those MDP states can be directly computed (as described earlier) from this TÆMS task structure that represents that state. In the multi-agent case, a similar process can be followed to translate a multi-agent TÆMS scenario into an MDP based on each action in the MDP representing a set of actions – one for each agent. From this perspective, TÆMS can efficiently describe, in terms of the number of nodes needed in the task structure, a class of MDPs.
61. This work indicated that, in environments with either high frequency of task arrival with tight deadlines or low frequency of task arrival with loose deadlines, using the full range of GPGP coordination mechanisms was not cost effective. Using the full set of coordination mechanisms was most appropriate in environments with medium-frequency task arrival and medium task deadlines.
62. In [53, 54], a meta-level agent control architecture is detailed that makes decisions about how much effort should be spent on coordination based on the structure, importance, and deadlines of current tasks. For example, if a task has a deadline in the near future, the meta-level controller can choose not to perform the task in a way that will require coordination with other agents. Similarly an agent may not engage in a coordination protocol if its current tasks have tight deadlines and the potential gain from performing the coordination does not justify having one or more of its current tasks missing their deadlines.

References

1. S. Abdallah, N. Darwish, O. Hegazy, and I. Talkhan, "Monitoring and synchronization for teamwork," in *Proceedings of the 17th ACM Symposium on Applied Computing* (*SAC2002*), 2002.
2. M. Barbuceanu and M. S. Fox, "COOL: A language for describing coordination in multi-agent systems," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, 1995, pp. 17–24.
3. D. S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of Markov decision processes," in *Proceedings of the 16th International Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, 2000, pp. 32–37.
4. R. H. Bordini, A. L. C. Bazzan, R. de O. Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser, "AgentSpeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, Part 3: ACM Press, July 2002, pp. 1294–1302.
5. C. Boutilier, "Sequential optimality and coordination in multiagent systems," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (*IJCAI99*), 1999.
6. D. D. Corkill and V. R. Lesser, "The use of meta-level control for coordination in a distributed problem-solving network," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 1983, pp. 748–756.
7. D. Corkill, "A framework for organizational self-design in distributed problem-solving networks," Ph.D. thesis, University of Massachusetts/Amherst, 1983.
8. K. Decker and V. Lesser, "Generalizing the partial global planning algorithm," *Int. J. Intell. Cooperative Inf. Syst.*, vol. 1, no. 2, pp. 319–346, 1992.
9. K. Decker and V. Lesser, "An approach to analyzing the need for meta-level communication," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, vol. 1, 1993.
10. K. Decker and V. Lesser, "Quantitative modeling of complex environments," *Int. J. Intell. Syst Accounting, Finance, Management*, vol. 2, no. 4, pp. 215–234, *Special Issue on Mathematical and Computational Models of Organizations*: *Models and Characteristics of Agent Behavior*, 1993.
11. K. Decker and V. Lesser, "Designing a family of coordination algorithms," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: San Francisco, 1995, pp. 73–80.
12. K. Decker and J. Li, "Coordinating mutually exclusive resources using GPGP," *Auton. Agents Multi-Agent Syst.* Special Issue: Best of ICMAS'98 – Part II, vol. 3, no. 2, pp. 133–158, 2000.
13. K. Decker, M. Williamson, and K. Sycara, "Intelligent adaptive information agents," *J. Intell. Inf. Syst.*, vol. 9, pp. 239–260, 1997.
14. K. S. Decker, "Environment centered analysis and design of coordination mechanisms," Ph.D. thesis, University of Massachusetts/Amherst, 1995.
15. K. S. Decker, "Task environment centered simulation," in M. Prietula, K. Carley, and L. Gasser (eds.), *Simulating Organizations*: *Computational Models of Institutions and Groups*, AAAI Press/MIT Press, 1997.
16. E. Durfee and V. R. Lesser, "Predictability versus responsiveness: Coordinating problem solvers in dynamic domains," in *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988, pp. 66–71.
17. E. H. Durfee and V. R. Lesser, "Negotiating task decomposition and allocation using partial global planning," in M. Huhns and L. Gasser (eds.), *Distributed Artificial Intelligence*, vol. 2, Pitman Publishing Ltd.: London, 1989, pp. 229–244.
18. E. H. Durfee and V. R. Lesser, "Partial global planning: A coordination framework for distributed hypothesis formation," *IEEE Trans. Syst. Man, Cyber.*, vol. 21, no. 5, pp. 1167–1183, 1991.
19. S. Fujita and V. R. Lesser, "Centralized task distribution in the presence of uncertainty and time deadlines," in *Proceedings of the Second International Conference on Multi-Agent Systems*, AAAI Press: CA, 1996, pp. 87–94.
20. A. Garvey and V. R. Lesser, "Design-to-time scheduling and anytime algorithms," *SIGART Bull.*, vol. 7, no. 3, 1996.

21. A. Garvey, K. S. Decker and V. R. Lesser, "A negotiation-based interface between a real-time scheduler and a decision-maker," *Proceedings of Workshop on Models of Conflict Management in Cooperative Problem Solving*, AAAI Press: Seattle, 1994.
22. J. Graham, "Real-time scheduling in distributed multi-agent systems," Ph.D. dissertation, University of Delaware, January 2001.
23. J. Graham and K. S. Decker, "Towards a distributed, environment-centered agent framework," in N. Jennings and Y. Lesperance (eds.), *Intelligent Agents VI*: *Agent Theories*, *Architectures*, *and Languages*, LNAI #1757, Springer, 2000, pp. 290–304.
24. J. Graham, K. S. Decker, and M. Mersic, "DECAF: A flexible multi-agent system architecture," *Auton. Agents Multi-Agent Syst.*, accepted for publication.
25. J. Graham, D. McHugh, M. Mersic, F. McGeary, M. Windley, D. Cleaver, and K. S. Decker, "Tools for developing and monitoring agents in distributed multi-agent systems," in T. Wagner and O. Rana (eds.), *Infrastructure for Agents*, *Multi-Agent Systems*, *and Scalable Multi-Agent Systems*. *Lecture Notes in Computer Science #1887*, Springer, pp. 12–27, 2001.
26. B. J. Grosz and S. Kraus, "Collaborative plans for complex group action," *Artif. Intell.*, vol. 86, no. 2, pp. 269–357, 1996.
27. B. Horling, et al. "The TÆMS white paper," *Technical Notes of Multi-Agent Systems Lab*, Department of Computer Science, University of Massachusetts, 1999.
28. B. Horling, B. Benyo, and V. R. Lesser, "Using self-diagnosis to adapt organizational structures," in *Proceedings of the Fifth International Conference on Autonomous Agents*, Montreal, 2001, pp. 529–536.
29. B. Horling, V. Lesser, R. Vincent, and T. Wagner, "The soft real-time agent control architecture," in *Proceedings of the AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. July 2002. Also available as UMass Computer Science Tech Report 02–14.
30. B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. R. Lesser, "Distributed sensor network for real-time tracking," in *Proceedings of the Fifth International Conference on Autonomous Agents*, Montreal, 2001, pp. 417–424.
31. B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. R. Lesser, "Using autonomy, organizational design and negotiation in a distributed sensor network," *Distributed Sensor Networks*: *A Multiagent Perspective*, pp. 139–183, 2003.
32. F. F. Ingrand, M. P. Georgeff, and A. S. Rao, "An architecture for real-time reasoning and system control," *IEEE Expert*, vol. 7, no. 6, pp. 34–44, 1992.
33. N. Jennings, "Commitments and conventions: The foundation of coordination in multi-agent systems," *Knowl. Eng. Rev.* vol. 8, no. 3, pp. 223–250, 1993.
34. N. Jennings, "Controlling cooperative problem solving in industrial multi-agent systems using joint intentions," *Artif. Intell.*, vol. 75, no. 2, pp. 195–240, 1995.
35. D. Jensen, M. Atighetchi, R. Vincent, and V. R. Lesser, "Learning quantitative knowledge for multiagent coordination," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.
36. K. Kuwabara, T. Ishida, and N. Osato, "AgentTalk: Coordination protocol description for multiagent systems," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, p. 455, 1995.
37. B. Lâasri, H. Lâasri, S. Lander, and V. R. Lesser, "A generic model for intelligent negotiating agents," *Int. J. Intell. Cooperative Inf. Syst.*, vol. 1, no. 2, pp. 291–317, 1992.
38. S. Lander and V. R. Lesser, "Sharing meta-information to guide cooperative search among heterogeneous reusable agents," *IEEE Trans. Knowl. Data Eng.*, vol. 9, no. 2, pp. 193–208, 1997.
39. V. R. Lesser, "A retrospective view of FA/C distributed problem solving," *IEEE Trans. Syst., Man, Cyber.*, vol. 21, no. 6, pp. 1347–1362, 1983.
40. V. R. Lesser, "Reflections on the nature of multi-agent coordination and its implications for an agent architecture," *Autonomous Agents and Multi-Agent Syst.*, vol. 1, pp. 89–111, 1998.
41. V. R. Lesser and D. D. Corkill, "The distributed vehicle monitoring testbed: A tool for investigating distributed problem-solving networks," *AI Mag.*, vol 4, no. 3, pp. 15–33, 1983.

42. V. R. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S. X. Q. Zhang, "The intelligent home testbed," in *Proceedings of the Autonomy Control Software Workshop* (*Autonomous Agent Workshop*), 1999.
43. V. R. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S. X. Q. Zhang, "A multi-agent system for intelligent environment control." in *Proceedings of the Third International Conference on Autonomous Agents*, ACM Press: New York, NY, 1999, pp. 291–298.
44. V. R. Lesser, K. S. Decker, N. Carver, A. Garvey, D. Neiman, M. Nagendra Prasad, and T. Wagner, "Evolution of the GPGP domain-independent coordination framework," University of Massachusetts/Amherst CMPSCI Technical Report 98-05, 1998.
45. V. R. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner, and S. X. Q. Zhang, "BIG: An agent for resource-bounded information gathering and decision making," *Artif. Intell.*, vol. 118, no. 1–2, pp. 197–244, 2000. (Elsevier Science), Special Issue on Internet Information Agents.
46. H. J. Levesque, P. R. Cohen, and H. T. Nunes, "On acting together," in *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990, pp. 94–99.
47. R. Mailler, R. Vincent, V. R. Lesser, and B. Horling, "Cooperative negotiation for soft real-time distributed resource allocation," in *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems* (*AAMAS 2003*).
48. M. V. Nagendra Prasad and V. R. Lesser, "Learning situation specific coordination in cooperative multi-agent systems," *Autonomous Agents Multi-Agent Syst.*, vol. 2, pp. 173–207, 1999.
49. M. V. Nagendra Prasad, K. S. Decker, A. Garvey, and V. R. Lesser, "Exploring organizational designs with TÆMS: A case study of distributed data processing," in *Proceedings of the Second International Conference on Multi-Agent Systems*, 1997.
50. T. Oates, M. V. Nagendra Prasad, and V. R. Lesser, "Cooperative information gathering: A distributed problem-solving approach," *IEE Proc. Software Eng.* Special Issue on Agent-based Systems, vol. 144, no. 1, pp. 72–78, 1997.
51. D. Pynadath and M. Tambe, "An automated teamwork infrastructure for heterogeneous software agents and humans," *J. Auton. Agents Multiagent Syst* (*JAAMAS*), 2002.
52. Y. Qian, E. Albert, T. Wagner, J. Phelps, and G. Beane, "A modified architecture for constructing real-time information-gathering agents," in *Proceedings of Agent-Oriented Information Gathering Systems*.
53. A. Raja, "Meta-level control in multi-agent systems," Ph.D. thesis, University of Massachusetts/Amherst, 2003.
54. A. Raja and V. Lesser, "Automated meta-level control reasoning in complex agents," in *Proceedings of Workshop on 'Agents and Automated Reasoning' in the Eighteenth International Joint Conference on Artificial Intelligence* (*IJCAI 2003*), 2003.
55. A. Raja, V. R. Lesser, and T. Wagner, "Toward robust agent control in open environments," in *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, 2000, pp. 84–92.
56. A. Raja, T. Wagner, and V. R. Lesser, "Reasoning about uncertainty in agent control," in *Proceedings of the 5th International Conference on Information Systems, Analysis, and Synthesis*, Computer Science and Engineering, Part 1, vol. VII, 2001, pp. 156–161.
57. A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: San Francisco, 1995, pp. 312–319.
58. S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, MIT Press: Cambridge, MA, 1994.
59. Z. Rubinstein, "The Hurrier I Go, The Behinder I Get," in *Proceedings of Artificial Intelligence and Manufacturing*, IJCAI & AAAI, 2001, pp. 97–102.
60. R. G. Smith and R. Davis, "Frameworks for cooperation in distributed problem solving," *IEEE Trans. Syst., Man, Cyber.*, vol. SMC-11, pp. 61–70, 1981.
61. T. Sugawara and V. R. Lesser, "Learning to improve coordinated actions in cooperative distributed problem-solving environments," *Mach. Learn.*, vol. 33, no. 2–3, 1998 (Kluwer Academic Publishers).
62. M. Tambe, "Agent architectures for flexible, practical teamwork," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

63. M. Tambe, "Towards flexible teamwork," *J. Artif. Intell. Res.*, vol. 7, pp. 83–124, 1997.
64. R. Vincent, B. Horling, V. R. Lesser, and T. Wagner, "Implementing soft real-time agent control," in *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001, pp. 355–362.
65. T. Wagner and V. R. Lesser, "Design-to-criteria scheduling for intermittent processing," University of Massachusetts/Amherst Computer Science Technical Report 1996-81, 1996.
66. T. Wagner and V. R. Lesser, "Relating quantified motivations for organizationally situated agents," in N. R. Jennings and Y. Lesperance (eds.), *Intelligent Agents VI*: *Agent Theories*, *Architectures*, *and Languages*, Springer-Verlag, vol. 1757, 2000, pp. 334–349.
67. T. Wagner and V. R. Lesser, "Toward soft real-time agent control," in T. Wagner and O. Rana, (eds.), *Lecture Notes in Computer Science*: *Infrastructure for Large-scale Multi-Agent Systems*, Springer-Verlag, vol. 1887, 2001.
68. T. Wagner and V. R. Lesser, "Evolving real-time local agent control for large-scale multi-agent systems," in J. -J. Meyer and M. Tambe (eds.), *Intelligent Agents VIII*: *Agent Theories*, *Architectures*, *and Languages*, Springer-Verlag, LNAI vol. 2333, pp. 51–68, 2002.
69. T. Wagner, B. Benyo, V. Lesser, A. Raja, P. Xuan, and X. Q. Zhang, "GPGP^2: Supporting situation specific coordination protocols," UMASS Computer Science Technical Report #98-042, 1998.
70. T. Wagner, B. Benyo, V. R. Lesser, and P. Xuan, "Investigating interactions between agent conversations and agent control components," in F. Dignum and M. Greaves (eds.), *Issues in Agent Communication*, Springer-Verlag, vol. 1916, 2000, pp. 314–331.
71. T. Wagner, A. Garvey, and V. R. Lesser, "Complex goal criteria and its application in design-to-criteria scheduling," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.
72. T. Wagner, A. Garvey, and V. R. Lesser, "Leveraging uncertainty in design-to-criteria scheduling," UMass Computer Science Technical Report 1997-11, 1997.
73. T. Wagner, A. Garvey, and V. R. Lesser, "Criteria directed task scheduling," *Int. J. Approximate Process.*, Special Issue on Scheduling, vol. 19, pp. 91–118, 1998.
74. T. Wagner, V. Guralnik, and J. Phelps, "A key-based coordination algorithm for dynamic readiness and repair service coordination," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS'03*), ACM Press, 2003.
75. T. Wagner, V. Guralnik, and J. Phelps, "TAEMS agents: Enabling dynamic distributed supply chain management," *J. Electron. Commerce Res. Appl.*, 2003 (Elsevier).
76. M. Wellman and J. Doyle, "Modular utility representation for decision-theoretic planning," in *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, 1992, pp. 236–242.
77. A. Wise, A. G. Cass, B. Staudt-Lerner, E. K. McCall, L. J. Osterweil, and S. M. Sutton, Jr., "Using little-JIL to coordinate agents in software engineering," in *Proceedings of the Automated Software Engineering Conference* (*ASE 2000*), 2000, pp. 155–163.
78. P. Xuan, "Uncertainty handling and decision making in multi-agent cooperation." Ph.D. thesis, University of Massachusetts/Amherst, 2002.
79. P. Xuan and V. R. Lesser, "Incorporating uncertainty in agent commitments," in N. R. Jennings and Y. Lesperance (eds.), *Intelligent Agents VI*: *Agent Theories*, *Architectures*, *and Languages*, Springer-Verlag, vol. 1757, 2000, pp. 57–70.
80. P. Xuan and V. R. Lesser, "Multi-agent policies: From centralized ones to decentralized ones," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS 2002*), Part 3: ACM Press, July 2002, pp. 1098–1105.
81. P. Xuan, V. R. Lesser, and S. Zilberstein, "Communication decisions in multi-agent cooperation: Model and experiments," in *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001, pp. 616–623.
82. X. Zhang and V. Lesser, "Multi-linked negotiation in multi-agent system," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS 2002), Bologna, Italy, 15–19 July, 2002, pp. 1207–1214.

83. X. Q. Zhang, V. R. Lesser, and T. Wagner, "A two-level negotiation framework for complex negotiations," in *Proceedings of IEEE/WIC International Conference on Intelligent Agent Technology* (*IAT 2003*), Halifax, Canada, IEEE Computer Society Press, 2003.
84. X. Q. Zhang, R. Podorozhny and V. R. Lesser, "Cooperative, multistep negotiation over a multi-dimensional utility function," in *Proceedings of the IASTED International Conference*, *Artificial Intelligence and Soft Computing* (*ASC 2000*), IASTED/ACTA Press, 2000, pp. 136–142.