Analyzing a Quantitative Coordination Relationship [1]

Keith Decker and Victor Lesser
Computer and Information Science Department
University of Massachusetts

UMass Computer Science Technical Report 91–83
May 10, 1993

Abstract

Coordination is a crucial behavior in cooperative distributed problem solving (CDPS). Analyzing coordination requires an understanding of the interplay between the agents, their problem, and their environment. The core behaviors of *distributed coordination* in CDPS systems are the coherent specification and scheduling of tasks over the set of distributed agents working on sets of interrelated problems. The complexity of, and uncertainty about, the problem interrelationships make distributed task coordination difficult. This paper describes a causal model of this process that links the interrelationships, called *coordination relationships*, to the local scheduling constraints of distributed agents. Besides coordination relationships, environmental uncertainty and the lack of infinite computational resources also make distributed coordination difficult.

It is not only the presence or absence of a coordination relationship that is important, but its quantitative properties: how likely is it to appear, how significant is its effect, etc. These aspects determine the usefulness of a particular coordination relationship in the context defined by an environment, a problem to be solved, and an agent architecture. This paper discusses the analysis of coordination relationships, using as an example our abstract model for the *facilitates* relationship. We detail the derivation and assumptions of this model and apply it to the the design of a generalized coordination module that is separate from, and interfaces cleanly with, the local scheduler of a CDPS agent. A set of simulation experiments is described that test our assumptions and design process in the coordination of a group of real-time problem-solving agents.

[1] ***To appear in the journal Group Decision and Negotiation, in 1993.*** This work was supported by DARPA contract N00014-92-J-1698, and partly by the Office of Naval Research contract N00014-92-J-1450, and NSF contract CDA 8922572. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

1 Introduction

The core of the distributed coordination problem in Cooperative Distributed Problem Solving (CDPS) systems [9, 17] is the coherent specification and scheduling of tasks over the set of distributed agents working on sets of interrelated problems. From a purely computational standpoint, coordination allows better use of computational resources by intelligent scheduling, e.g., reducing needless redundancy, scheduling the arrival of needed results, and load balancing. From a problem-solving perspective, it also allows the detection (and thus the potential for resolution) of some kinds of uncertainties and inconsistencies.

The partial global planning (PGP) approach to distributed coordination [11] increased the coordination of agents in a network by such methods as providing predictive results, avoiding redundant activities, shifting tasks to idle nodes, and indicating goal compatibility. It achieved this by recognizing certain *coordination relationships* among tasks in the Distributed Vehicle Monitoring Testbed (DVMT) environment and producing the appropriate scheduling constraints. In fact, because the local scheduler was so simple, the PGP mechanism supplanted it, recording and responding to many of the appropriate scheduling constraints itself. This work has identified coordination techniques that are helpful, but did not provide a deep analysis of when and why they are appropriate. We believe that the right way to think about coordination is through abstractions of these coordination relationships, defined in a domain-independent way. These relationships need to be quantified, not just identified. Earlier work [8, 10] has shown that a weakly qualitative approach to answering questions about coordination can lead to unsatisfying answers—that different coordination algorithms (organizations, communication patterns, etc.) are better or worse depending on the situation. These experiences and the large parameter spaces involved have led us to a more quantitative approach. It is the quantified features of the relationships that determine their usefulness in the context of a particular environment, problem, and agent architecture. A more complex view of coordination is necessary to develop general theories for the application of coordination algorithms in different environments. Quantitative coordination relationships are a step toward a theory of coordination in cooperative distributed problem solving. After developing a quantitative theory, qualitative statements may be derived from it.

We are now focusing on generalizing the partial global planning mechanism. This process involves identifying the types of coordination relationships that are used by the basic PGP algorithm and that exist but that are not used [6], developing a conceptual model that clearly specifies the roles of a PGP-style coordination algorithm as identifying coordination relationships and producing behaviors (primarily the creation and refinement of local scheduling constraints), and generalizing the partial global planning algorithm itself (GPGP) [6]. Our current approach views the coordination mechanism as *modulating* local control, not supplanting it—a two level process that makes a clear distinction between coordination behavior and local scheduling [3]. By concentrating on the creation of local scheduling constraints, we avoid the sequentiality of scheduling in partial global planning that occurs when there are multiple plans. By separating coordination from local scheduling, we can also take advantage of advances in real-time scheduling algorithms to produce CDPS systems that respond to real-time deadlines. We define these coordination relationships in a domain-independent manner; for example, one task may provide results to another task that *facilitate* the second task

by allowing the second task to exploit the provided result to increase the quality or reduce the duration of the second task. It is not only the presence or absence of a coordination relationship that is important, but its quantitative properties: how likely is it to appear, how hard is it to detect, how significant is its effect, etc. It is these aspects that determine the usefulness of a particular coordination relationship in the context of an environment, problem to be solved, and agent architecture.

This paper will focus on one coordination relationship, the *facilitates* relationship[1]. This relationship (along with *overlaps*) was a major contributor to the results achieved with partial global planning. We first discuss our conceptual model and develop a detailed instantiation of it for the *facilitates* relationship. This more detailed model, including such characteristics of the *facilitates* relationship as how likely is it to appear and how significant is its effect, is then used to inform the design of the *facilitates*-responding portion of the generalized partial global planning algorithm. Several hypotheses about our environmental assumptions and our design are articulated, and experimental results are presented.

2 Conceptual Model and Environmental Assumptions

Our approach to distributed task coordination rests on a conceptual, causal model of the generalized PGP-style distributed coordination process (see Figure 1). Information flows from the environment through the agent's coordination mechanism and local scheduling mechanism, eventually influencing performance. First, a given environment and/or task domain, in conjunction with an agent's architecture, induces certain coordination relationships (CRs) between tasks in that environment. An agent follows some *coordination algorithm* that detects or hypothesizes CRs and reacts accordingly—the algorithm produces certain behaviors, for instance the creation and refinement of local scheduling constraints (other possible behaviors include communication, negotiation, and the creation of internal data structures). We can relate how the behavior of the coordination algorithm (creating and refining constraints) affects the agent's scheduling behavior by basing our analysis on certain properties of the local scheduling mechanism. Finally, the scheduling behavior affects the performance of the agent and any organization of which it is a part not only by ordering and executing tasks but also through incurring costs, such as communication and time.

The primary interruption to the flow of information just described is uncertainty. Uncertainty also flows from the environment (the open systems concept[15]), and through the above mechanisms to create local scheduling uncertainty. Less uncertainty in the environment means less uncertainty in the existence and extent of the CRs, less uncertainty in local scheduling, and therefore less complex coordination behavior (communication, negotiation, partial plans, etc) is needed (for example, one can have cooperation without communication [14]). Even given complete certainty, the lack of infinite computational resources/time results in the necessity of satisficing, not optimal, behaviors. Even if agents had instant access to the complete state of the composite system it does not mean that the environment provides sufficient computational resources or time to the agents to exploit that voluminous information[2].

[1]Other potential coordination relationships include *inhibits*, *cancels*, *constrains*, *causes*, *enables*, and *subgoal* [6].

[2]Our description of the coordination process is consistent with social views of organiza-

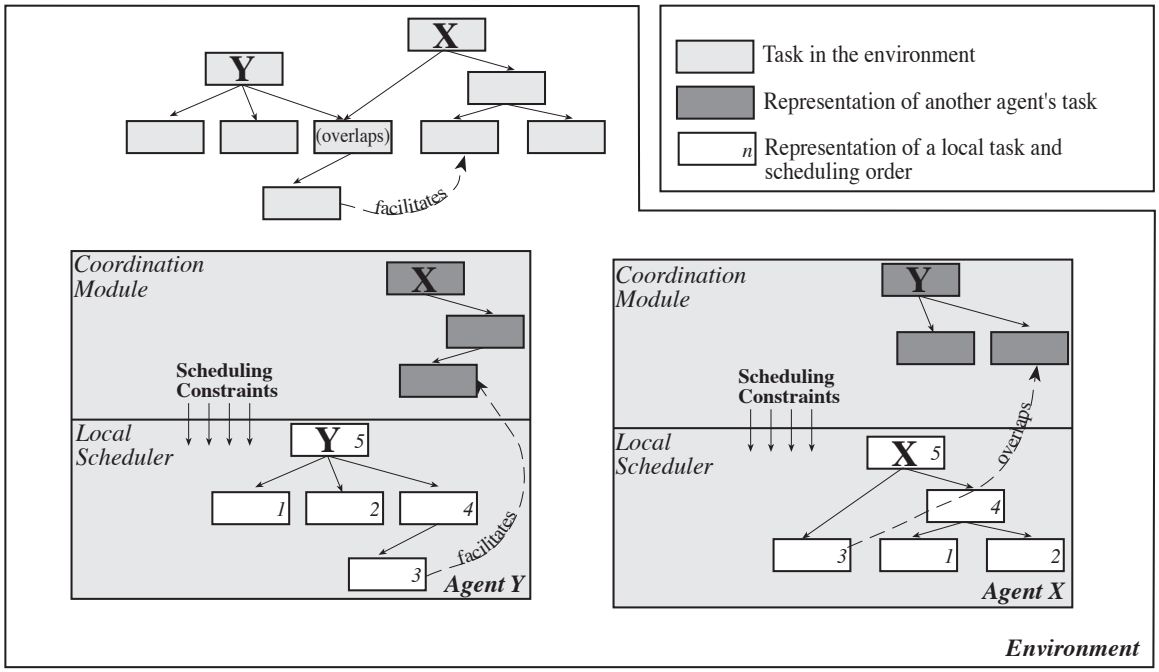

Fig. 1: Conceptual view of the distributed coordination process

For example, the abstract domain of distributed search contains several potentially uncertain coordination relationships [17]. The primary uncertainty lies in the subgoal relationship: how a particular task relates to the problem as a whole. Will this task be a part of a final solution? Are there multiple paths to the goal? How much effort will it take? A secondary uncertainty lies in the presence of a *facilitates* relationship: whether the amount of processing is greatly affected by the order in which goals are solved. Ignoring these relationships will affect the amount of resources the agents use through undesirable redundant processing, idleness, and distraction.

2.1 Environmental Assumptions

To build a model of coordination, we have made several assumptions about generic CDPS environments. We assume that the agents share a common language for communicating results, as well as other information such as abstract goals needed for coordination. We also assume, as the original PGP algorithm did, that there is a global utility function shared by the agents. The interesting problems arise when agents do not have equal or up-to-date access to all the necessary information. Other than having a common language, the agents may be heterogeneous and have different capabilities and responsibilities.

We are developing a formal model of task environment structures for CDPS and single-agent scheduling tasks [5]. This model has three levels: the *objective* level, describing a particular problem solving instance; the *subjective* level, describing what objective level information is available when and at what cost to agents; and the

tional coordination mechanisms or behaviors: the use of rules, regulations, and standards; the creation of supervisory and decision-making hierarchies; and specialization or departmentalization. Organizational structure should be viewed as part of the coordination algorithm.

generative level, describing statistical characteristics of the other two levels across particular instances of a domain.

The *objective* level describes the essential structure of a particular problem-solving situation or instance over time. It focuses on how task interrelationships dynamically affect the *quality* and *duration* of each task. Briefly, the basic model is that task groups $\mathcal{T}$ occur in the environment at some frequency, and induce tasks T to be executed by the agents under study. Task groups are independent of one another, but tasks within a single task group have interrelationships. Each task group has a deadline $\mathcal{D}(\mathcal{T})$. The *quality* of the execution or result of each task influences the *quality* of the task group result $\mathcal{Q}(\mathcal{T})$ in a precise way [5]. These quantities, deadline and quality, can be used to evaluate the performance of a system.

An individual task that has no subtasks is called a method M and is the smallest schedulable chunk of work (though some scheduling algorithms will allow some methods to be preempted, and some schedulers will schedule at multiple levels of abstraction). There may be more than one method to accomplish a task, and each method will take some amount of time or other resources and produce a result of some *quality*. The term *quality* in the model summarizes several possible properties of actions or results in a real system: certainty, precision, and completeness of a result, for example[7]. Quality of an agent's performance on an individual task is a function of the timing and choice of agent actions ('local effects'), and possibly previous task executions ('non-local effects'). When local or non-local effects exist between tasks that are known by more than one agent, we call them *coordination relationships*.

The environment is also characterized at the *subjective* level by a mapping from objective structures in the environment to subjective structures available to the agents. Subjective level characteristics include the distribution of tasks to agents, and the cost of detecting a coordination relationship. The subjective level can also characterize the uncertainty an agent has over the presence of tasks, their duration, quality, and so forth, but we will not use these features in this paper.

The *generative* level characterizes a generic CDPS environment by statistically characterizing such objective and subjective level features as the frequency of occurrence of a class of tasks, how pressed for time that class of tasks typically is (a relative measure based on the average deadline), and how amenable it is to trading off time for quality (the particular mix of methods available to each task, each with its own average duration and result quality; c.f., approximate processing algorithms [7], anytime algorithms [1]). There may also be a biased task class/agent distribution (certain agents tend to receive certain classes of tasks); *a priori* likelihoods for the presence of a CR (such as *facilitates*); costs for detecting a CR; and, in this case, the estimated effect of a *facilitates* relationship on the facilitated task's duration and quality.

Agents themselves are also part of the environment. Agents have local scheduling algorithms about which we want to make certain limited, explicit assumptions dealing with how the local scheduler responds to particular constraints (see Section 2.3). A final assumption is that each agent has the same coordination algorithm.

2.2 The *Facilitates* coordination relationship

Informally, the *facilitates* coordination relationship is defined so that Task A *facilitates* Task B if the result of Task A can be used to increase the local utility of B. This can occur because the two tasks are interdependent. The measure of utility is often some

trade-off between increased quality and decreased time to solution. The *facilitates* relationship, therefore, has two parameters (called *power* parameters) ϕ_{d} and ϕ_{q}, that indicate the effect on duration and quality respectively. The effect varies not only through the power parameters, but also through the quality of the *facilitating* task available when work on the *facilitated* task starts. Let T_a *facilitate* T_b with some duration power ϕ_{d} (let $\phi_{\mathrm{q}} = 0$ for this example). If T_a is completed with maximal quality, and the result is received before T_b is started, then the duration $\mathbf{d}(T_b)$ will be decreased by a percentage equal to the duration power ϕ_{d} of the *facilitates* relationship. If the result received is of less than maximal quality, ϕ_{d} is reduced proportionately:

$$\mathbf{d}(T_b) \;=\; \mathbf{d}(T_b) \;-\; \mathbf{d}(T_b) \cdot \phi_{\mathrm{d}} \cdot \frac{\text{quality received}}{\text{maximal quality}}$$

This model assumes that the proportional reduction is linear. Communication after the start of processing has no effect; the effect on quality when $\phi_{\mathrm{q}} \neq 0$ is computed analogously. A fuller treatment of the mathematical details can be found in [5][3].

Two tasks can be mutually facilitating; either there will be a clear preference for one over the other via the shared global utility function, or they are co-routining—iteratively exchanging results while executing in parallel. A side effect of the *facilitates* relationship can be that the existence of a facilitates relationship from A to B increases the likelihood that task B will produce results that contribute to a final or otherwise usable solution (called "goal compatibility").

When faced with A *facilitates* B, the partial global planning algorithm makes sure that the result of A is communicated to the agent performing B, and delays the start of B until after A is completed. The generalization of this is that the detection of this relationship affects the local schedulers of the two agents by adding or refining constraints on what approximations to use, deadlines, ready times, expected durations, and even what tasks to schedule.

2.3 Local Scheduling

How can we say anything useful about the effect of a generalized partial global planning (GPGP) algorithm on performance without knowing the details of the agent's local scheduling algorithm? The key is to make any claims contingent on properties of the local scheduling mechanism. In our case, we call these claims *admissibility* and *bounded rationality* properties.

We have chosen our particular approach for two reasons. First, it allows us to draw a clear line between the "coordination" part of an agent architecture and the rest of the agent. This validates our claim that we are examining general issues of coordination and the effect of environmental factors on coordination, while still allowing us to ground experiments in real, implemented agent architectures. It also is appealing from an asynchronous, layered subsumption architecture point of view [2]. Secondly, it allows us to take advantage of the advances currently being made in planning and scheduling. The original PGP mechanisms had to deal with a rather unsophisticated local scheduling mechanism; to do so, it often kept track of and enforced constraints

[3]Note, for example, that we can represent the cases where ϕ_{d} and/or ϕ_{q} are *negative*, resulting in 'negative facilitation'. Such a relationship may be useful for modeling the phenomena of *distraction* [3].

on its own. More sophisticated "real-time" schedulers actually make the coordination task easier by directly interpreting most of the needed coordination behaviors (in the form of scheduling constraints) [4].

Scheduling constraints may be hard or soft. *Admissibility* refers to hard constraints. A local scheduling algorithm is admissible with respect to a hard constraint if it always produces a feasible schedule if that is possible. *Rationality* refers to utility-maximizing behavior of the local scheduler with respect to soft constraints (assume each constraint has some utility associated with it). In many systems, perceived utility is represented by an objective function or evaluation function.

For some combinations of simple constraints and utility functions, there exist well-behaved local scheduling algorithms. For example, if $r(T_i)$ is a hard ready (start) time constraint on task T_i, and $D(T_i)$ is a soft deadline constraint on task T_i, and $r(T_i) < r(T_j) \rightarrow D(T_i) \leq D(T_j)$ ("any task that starts later than task T_i has a deadline later than task T_i's deadline"), and the objective function is to minimize the number of tardy jobs (those that miss their deadlines) then there exists a $O(n^2)$ local scheduling algorithm that is admissible (with respect to ready times) and rational (with respect to soft deadlines) [16]. As a designer of a coordination algorithm, we now have some powerful information for designing coordination algorithms.

For many scheduling problems a useful (i.e., non-exponential) local scheduling algorithm cannot be proven to be rational. Often heuristic methods are used to come up with a schedule; for example, the original PGP mechanisms included a hill-climbing scheduler. In such cases, the scheduler often *satisfices* (searches to a prespecified aspiration level), rather than optimizes, and the local scheduler can be considered to be (informally) *boundedly rational*. Instead of assuming that any soft constraint the coordination algorithm adds is always obeyed, the coordination algorithm can instead depend on an event being signaled if the constraint is or will be violated.

It is the uncertainty in the environment that makes the agents unable to "pre-program" all of their behaviors in advance and thus requires the dynamic recognition of and reaction to coordination relationships. A local scheduler usually has considerable uncertainty over the constraints that it uses to schedule, and about whether it has all the constraints that it needs. Without uncertainty there is little need to detect and communicate CRs at runtime. For example, the *facilitates* relationship can affect the local scheduler's beliefs in what the proper deadlines and ready times for tasks are, what level of quality (approximation) is appropriate, task durations, and even what tasks to schedule. Different CRs will affect different constraints, and we may be able to prove that in certain environments we do not need to detect certain coordination relationships because they are subsumed by others or are too weak.

3 Design of a Coordination Module for *Facilitates*

Our conceptual model of coordination leads to the design of a coordination module to handle, for example, the *facilitates* relationship. Similar types of analysis will occur for other relationships. This paper does not discuss how to relax constraints in overconstrained situations resulting from many different relationships, but the discussion at the end of this section on breaking commitments indicates the direction of our thoughts. According to our causal model, a coordination module has three basic decisions to make: when to detect and communicate about the CR, what local

scheduling constraints to add when it is detected, and what to do when an error occurs. For different environments, this results in a design that is more or less complex. In some environments, the *facilitates* relationship does not exist or is never useful to detect. In others, it is a useful constraint but not worth the trouble of negotiating or rescheduling when a problem occurs. In still others, the relationship affects problem solving so significantly as to make it worthwhile to expend considerable effort into achieving the scheduling commitments that it entails. A key point is that coordination relationships are abstractly defined in a domain-independent way, so we can catalogue coordination strategies and characterize them by potential features of an environment and task.

When should an agent test for a relationship? An agent can potentially test for the presence of the *facilitates* relationship between any new local task and the tasks it knows about from other agents. The costs per task here include the processing cost of testing for the relationship between two tasks and the cost of communication to get more information if the test result is too uncertain (C_{detect}). Note that the test itself is domain specific, even though the *facilitates* relationship is general. If the relationship is present, the benefits include some increased utility (reduction in time, increase in quality of the facilitated task) (B_{fac}). The benefits will accrue fully only if the scheduling constraints implied by the new detected relationship can be incorporated, and only if the facilitating task is accomplished.[4] Costs are also incurred when the relationship is present, including the direct costs of communicating the result, adding the scheduling constraints, and updating the models of each others' tasks, and the indirect costs of delaying some tasks (C_{fac}). Using the *a priori* likelihood Pr_{fac} of a *facilitates* relationship from the environmental model, we can build an expression for the form of the expected utility of the relationship, $\text{Pr}_{\text{fac}} * (B_{\text{fac}} - C_{\text{fac}}) - C_{\text{detect}}$. In many environments, such as the DVMT, this expression will be relatively static, with C_{detect} relatively constant and with B_{fac} and C_{fac} depending on many environmental factors, including the task at hand, the current system utilization, etc. (see the experiments in Section 4.2). If C_{detect} is not relatively constant, then it is beneficial to consider separately the decision to detect a relationship and the decision to communicate about it.

When a relationship is received from another agent, what should be done? There are four basic cases:

Need more communication: In this case the other agent was not certain enough about the presence of the relationship to make any commitments and has requested more information. The second agent can use the task information from the first agent to make more detailed tests, at deeper levels of detail, than the first agent could. The second agent can then verify the presence or absence of the relationship, and if it is present appropriately inform the first agent, updating its model, and continuing with one of the other cases below [21]. By communicating only when potentially necessary, and at successively greater levels of detail, overall communication is reduced [6, 10].

[4] For some relationships, there is also a chance the benefit will accrue serendipitously without the detection of the relationship. When possible, tasks can be structured to take advantage of this fact, which may reduce coordination costs considerably.

Other agent task A facilitates local task B: Delay the start of task B to the committed completion time of task A. Update the predicted duration of B. May also increase the belief that B is part of some solution (goal compatibility).

Local task A facilitates other task B: Commit to a communication deadline for the result of task A (based on any deadline information known about task B) and commit to a certain level of quality (approximation) for task A.

Mutual Facilitation: Either the benefits clearly imply a winning order for the two tasks or the two tasks should really co-routine. In the second case the tasks should be constrained to execute concurrently and micro-scheduled to exchange partial results with one another while executing.

How do we compute how long to delay a task B that is facilitated by a task A? For example (see Figure 2), if

- utility is a function of result quality only (as opposed to a function of both quality and missed deadlines)
- the local scheduler does not assure the minimal quality of a task result
- *facilitates* affects only the durations of tasks directly

then agents must attempt to produce their highest quality solutions. Suppose there are several different tasks B_i such that A *facilitates* each B_i. If we estimate the latest start time for each task B in order to produce a highest quality solution, and estimate the finish time for task A, then all tasks B with latest start times after the estimated finish of A (B_2 and B_3 in the figure) should be delayed to the minimum latest start time of those tasks (B_2 in Figure 2; times must be calculated taking communication time into account if it is substantial). If the local scheduler *were* boundedly rational with respect to quality constraints, then the estimate of the latest start time of B could be modified by (made later by) the estimated effect of the *facilitates* relationship for the maximally assured minimal quality of the result of task A.

What should be done when something goes wrong? One key to handling errors is keeping track of the benefit derived from the commitments currently made. The other key is to keep track of when handling an error is worthwhile. Error handling is also where the most complex decision making occurs. If there are no errors, the coordination behaviors can easily be shown to have a positive effect on system performance. There are three basic causes of errors in the coordination behaviors involved with the *facilitates* relationship, although the basic question always comes down to "when should I break a commitment"?:

Scheduling problems: Given that A *facilitates* B, either the agent with A cannot commit to the scheduling constraints on A needed to accomplish B, or B cannot be delayed to the committed completion of A. The first thing to note is that fixing this may be more expensive in processing time than letting the agents act in an "uncoordinated" manner. If the benefit is clearly worthwhile, then in the second case (where B cannot be delayed) there can be a one-shot negotiation where the agent with B proposes the latest start time for B (given that the result from A is received). If this fails, and in the first case, it may be possible to reduce the quality of the result of A, or deliver a partial result early, and still save time or increase the quality of the result of B. Finally, the only option left is to break an existing commitment, as discussed below.

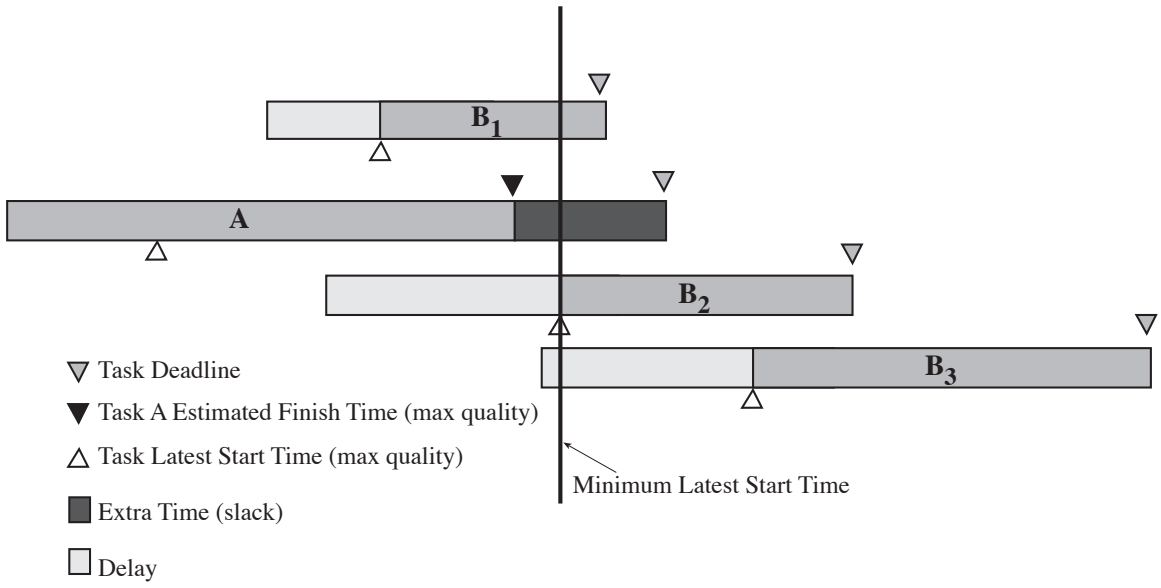

Fig. 2: Calculating Delays

Task failure: In the same situation, what happens if task A fails? The agent with A can possibly still fulfill its original commitment by trying another method. If this is not possible, then the agent with A must notify the agent with task B that it was unable to keep its commitment. The 'failure' will in some domains still produce some partial result that may help with task B. If such a partial result is not available or not useful, the second agent can (if it is still worthwhile) try to delay B further so that A can be completed. If this fails then the only option left is to break an existing commitment.

Non-intentional break in a commitment: This could occur for many external reasons, but the primary local manifestation is that the boundedly rational local scheduler has broken a commitment because it did not have enough time to come up with another schedule. Again, the affected agent must decide if it is worthwhile to pursue the matter, and if so, the only recourse is for some other commitment to be (intentionally) broken.

How does an agent decide what commitment should be broken, if it arrives at this decision? Breaking a commitment is complex because each commitment may be part of a web of commitments spreading across many agents, following the network of CRs that tie together the interdependent subproblems. An agent without a global view may not be fully aware of this web of commitments. Balancing this view of an agent unsure of what constraint to remove or weaken for fear of everything collapsing all around it is the inherent resiliency of the satisficing local scheduler that will try to complete tasks without exploiting the CR. While it is possible for a break in a commitment to impact many other unknown agents, a break in the commitment of least benefit is not liable to have this effect due to the slack and resiliency of each successive local scheduler. Slack has long been viewed as a coordination behavior and as such should be an organizational parameter [8].

This treatment does not give full justice to the analysis of this particular problem, to which we will have to devote a future analysis of its own.

4 Hypotheses and Experiments

As we flesh out our conceptual model to a set of analytic components by making various assumptions, we develop a number of hypotheses about these components that can be experimentally verified. This section will concentrate on two quantitative properties of the *facilitates* relationship: how likely it is to appear and how significant is its effect. This paper does not deal with the third property, how hard the CR is to detect. We would like to experimentally verify our formulations. To do so we are taking a tack somewhat between the analytical but perhaps too simplified approach of Malone [18] and the non-analytical but probably too specialized approach of the DVMT, by using a statistical simulation of a large class of CDPS environments.

4.1 The Simulation

The simulation is driven by the environment, task, and agent characteristics we discussed in Section 2.1; the tasks represent abstract computations. In the experiments below, there were two abstract task classes. One class of tasks had a mean time between arrivals of 40% less than the other. Each task arrives uniformly randomly at an agent. Facilitation relationships are generated between tasks with a base probability that decreases linearly with the difference between task arrivals[5]. For example, if there are 642 tasks generated, and a base probability of 0.5, interrelated tasks are grouped into clusters approximately distributed as in the histogram in Figure 3. If A *facilitates* B and C, and C *facilitates* D, that cluster is of size 4. This gives an indication of the webs of commitment that may potentially exist.[6]

Each agent uses a sophisticated "design-to-time" (DTT) real-time local scheduler based on the concept of approximate processing[7, 12]. The DTT scheduler will choose a method for a task based on the amount of time available for that task and the other tasks currently on the agenda. The DTT scheduler may change the method being used during execution at a task monitoring point; in the experiments described here 50% of the work done before changing methods is lost. The DTT scheduler is boundedly rational for deadline constraints, and was modified to be boundedly rational for delay constraints (delaying the start of a task until the result of another task is received). Each class of tasks had 5 methods of varying quality and duration; each task execution was monitored by the DTT scheduler three times. The actual duration and quality values for each method for each task are randomly generated from normal distributions with means equal to the estimated values and variances as specified

[5] An experiment not reported here showed that in an environment where the probability of a facilitates relationship drops off exponentially instead of linearly, the system response characteristics are similar to a linear environment with the *same number of detected relationships*.

[6] The total number of ways to distribute k tasks to n agents is n^k. The number of ways to distribute k tasks to i agents where each agent gets at least 1 task (surjections) is $i!S(k,i)$, where $S(k,i)$ are the Stirling numbers of the second kind. So the expected number of n total agents that are involved in a k-cluster is:

$$\sum_{i=1}^{n} i\frac{\binom{n}{i}i!S(k,i)}{n^k}$$

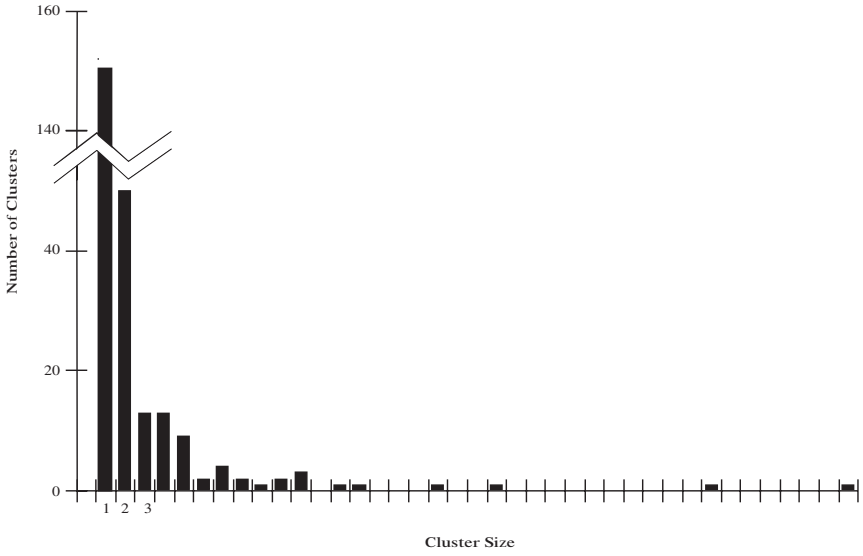

Fig. 3: Histogram of sizes of task clusters for $\mathrm{Pr}_{\mathrm{fac}} = 0.5$, $N = 642$.

for the method/task combination. The shared global utility function by which agents are judged is the total quality of their individual task solutions. The design-to-time scheduling algorithm used by each agent ensures that under normal circumstances (a required utilization (system load) of less than 3) less than 2% of the tasks ever miss a deadline. This is a property of the existence of low cost/low quality approximations.

Observations in the experiments that follow are made from average responses over 5 statistically generated runs of 1000 simulated world time units. The number of tasks that are actually generated depends on their arrival rate, which was varied. We also varied the *a priori* likelihood ($\mathrm{Pr}_{\mathrm{fac}}$) of a *facilitates* relationship between two tasks of the same class and the significance of the effect it has on the time of the *facilitated* tasks (called the *power* of the CR and measured by the percent reduction in time facilitated).

Each run consists of two sets of 4 agents. Each of the two agent-sets receives exactly the same set of tasks at exactly the same times—one agent-set uses the original DTT scheduling algorithm and no communication, the other agent-set uses the DTT scheduling algorithm modified to be boundedly rational with respect to *delay* constraints and to *always* detect and communicate coordination relationships. Each of the four agents in each agent-set receives precisely the same set of tasks as its counterpart in the other agent-set. The coordinating agents calculate delays as described in Figure 2, and handle the potential errors as described in Section 3. The coordinating agents do not enter into negotiation, however, but rather simply break the failed commitment.

The primary performance metric in the experiments presented here is the percentage increase in quality between each pair of agents that received the same task set (a paired response), and then averaged across the agent pairs. This metric is indicated in the figures by APQI (average percent quality increase). The other variable that was manipulated was the mean time between arrivals of the tasks; we can then compare relative increases in quality based on the average utilization required by that set of tasks. When the average required utilization is greater than 1, it means that it is

impossible to complete all tasks without approximating some of them.

There are three characterizations of this simulated environment that make it different from the actual environment of a system such as the DVMT:

- In interpretation environments like the DVMT, not all tasks need to be done, but in our simulation they do.
- Our simulation assumes that approximating a result has only a local effect on quality, as opposed to reducing the quality of a whole group of related tasks.
- Our simulation does not contain a model of the relationship between the fact that a task does or doesn't need to be done, and the fact that a *facilitates* relationship does or doesn't exist (goal compatibility).

On the other hand, our measurements are against a real-time scheduling algorithm that likewise does not take these factors into account. The addition of these extra characteristics to the task structure requires the addition of new coordination relationships (especially *subgoal*).

4.2 When to detect and communicate *facilitates*

The first suite of experiments (Figures 4 and 5) involves the effect of the two quantitative properties of *facilitates*, likelihood (Pr_{fac}) and duration power (ϕ_{d}). Our hypothesis is that the simple expected utility model given in Section 3 can be instantiated as a decision rule for each agent as to whether or not that agent should detect, communicate, and react to the *facilitates* relationship. The alternative is that the simple linear relationship is in fact not linear, or is drowned out by secondary characteristics such as the cluster size of the *facilitates* relationship or scheduler errors which cause the benefits of the facilitation relationship (B_{fac}) to become non-linear with respect to the power of the relationship.

To test this hypothesis, we first gathered raw data from 90 paired-response simulations of 4 agents that *never* detect, communicate, or react to the *facilitates* CR, and 4 agents that *always* detect, communicate, and react—5 simulations at each of 6 different duration powers and 3 different likelihoods (Figure 4). Each data point in the figure is the average of 20 computed percent quality increases (5 experiments of 4 paired agents each). All 90 experiments were simulated with the same frequency of task arrival (an average required utilization of 1.5—too high to allow all tasks to be completed at maximum quality without communication, but not so high as to saturate the DTT scheduler). Direct communication costs and detection costs were fixed at 0. Examining Figure 4, we find that below a power of 10%, in this example, exploiting the *facilitates* relationship costs more in *indirect* costs than it is worth. The indirect costs arise primarily from agent's delaying tasks and rearranging their schedules unnecessarily.

In Section 3 we postulated the form of the expected utility of detecting, communicating, and reacting to the CR as $\text{Pr}_{\text{fac}} * (B_{\text{fac}} - C_{\text{fac}}) - C_{\text{detect}}$. To instantiate this model for this experiment, we let $C_{\text{detect}} = 0$, make the benefits proportional to the duration power ($B_{\text{fac}} = \beta\phi_{\text{d}}$) and the costs constant (since the arrival rate was held constant) ($C_{\text{fac}} = \text{c}$). Applying linear regression we achieve a fit that explains 98% of the observed variance R^2 ($\beta = 1.013$, $c = 17.03$, both parameters are significant at the $\alpha = 0.01$

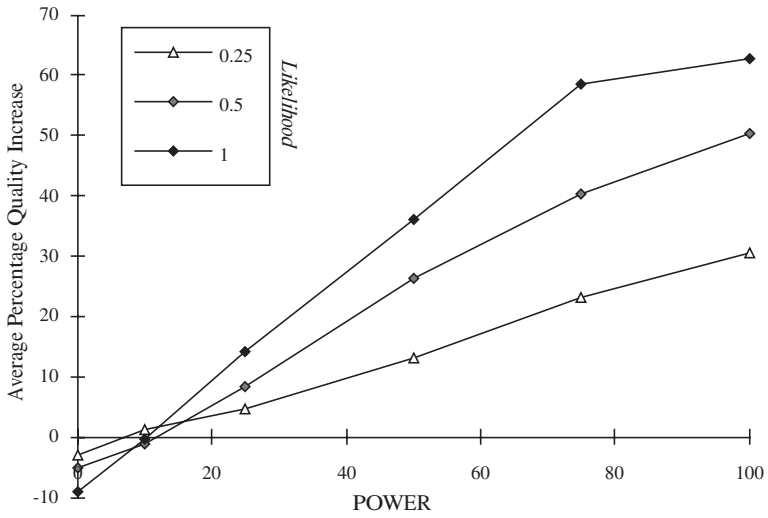

Fig. 4: The effect of the power of the *facilitates* relationship on relative quality at different likelihoods

level). Thus this formula could be used as a decision rule by each agent to decide if it is worthwhile to detect, communicate, and react to the coordination relationship.

Figure 5 is a depiction of the surface defined by the three curves in Figure 4, via cubic interpolation. It succinctly shows how the positive effect of duration power grows with the likelihood. Note the flattening at the high end of the power and likelihood scales, caused by a ceiling effect since in this region all tasks are being accomplished at maximum quality (and so one cannot do any better).

4.3 *Facilitating* Real-time Performance

The second suite of experiments (Figure 6) shows that the average relative increase in quality in the results of communicating agents versus non-communicating agents grows with the required utilization of the system (load). The harder the task set is, the more important detecting the *facilitates* CR is, even at low power. The left side of Figure 6 shows duration power versus the relative quality increase for several required utilizations (system loads)[7]. The right side of Figure 6 shows the effect of required utilization on relative quality for duration powers of 25% and 50%. The *a priori* likelihood of the presence of the CR was fixed at 0.5 for Figure 6. We would expect both the benefits and costs to change when the required utilization changes; the question is how much. The data from these experiments is not enough to disprove that the increase in quality grows linearly.

These figures show the increase in quality, but just as important is the decrease in missed deadlines (not shown here), which is similar. At high loads (high required utilization) the non-coordinated agent set reaches an asymptotic quality performance (which is less than maximum quality across loads; see Figure 7), and an unbounded number of missed deadlines. This figure depicts the absolute quality response of a single agent system over approximately 250 runs. The response has three major com-

[7]Two data points on the utilization = 6 line, (power = 75, APQI = 555) and (power = 100, APQI = 1010), were left out for clarity.

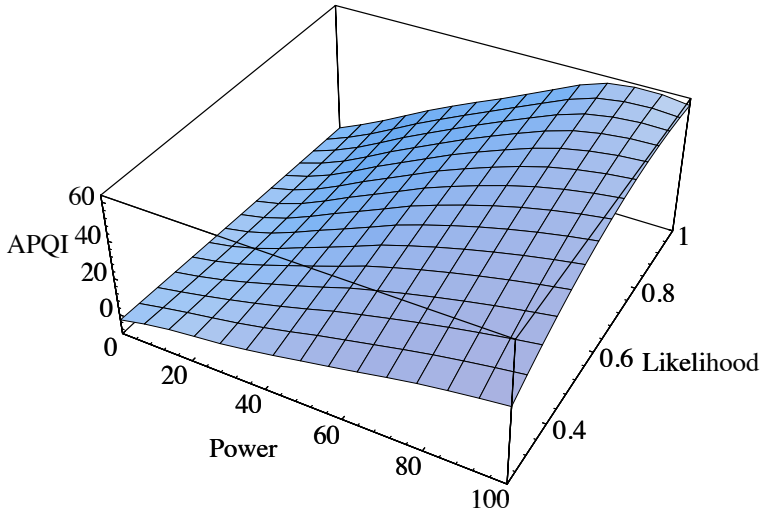

Fig. 5: The effect of the power of the *facilitates* relationship on relative quality at different likelihoods

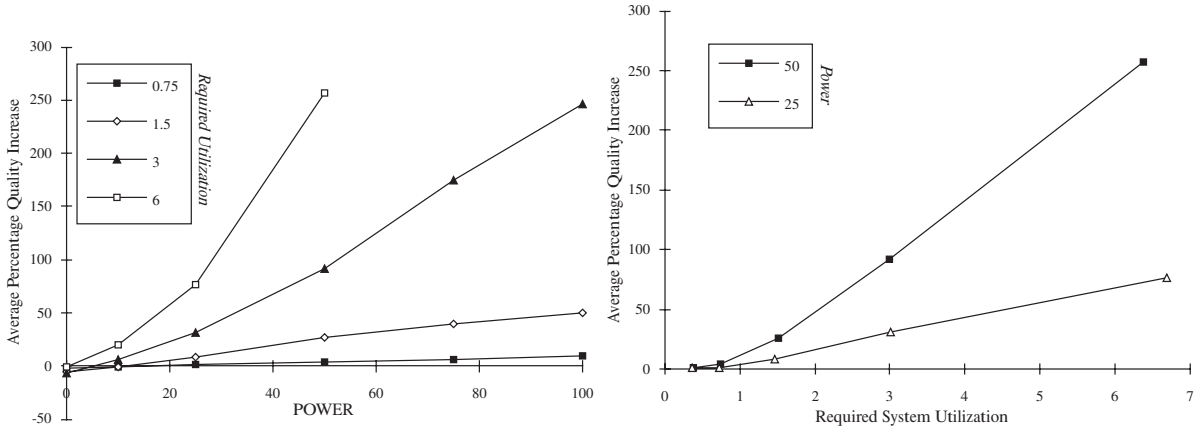

Fig. 6: The effect of power and required utilization (system loads) on relative quality

ponents: up to around a utilization of 1, absolute quality grows quickly as the number of tasks increases—each task is usually done at maximum quality; from around 1 to around 5, the DTT scheduler begins to trade off low quality, fast approximations for maximum quality, slow methods to avoid missing deadlines; above 5, every task is scheduled with the fastest method (resulting in asymptotic quality performance) and missed deadlines grow without bound.

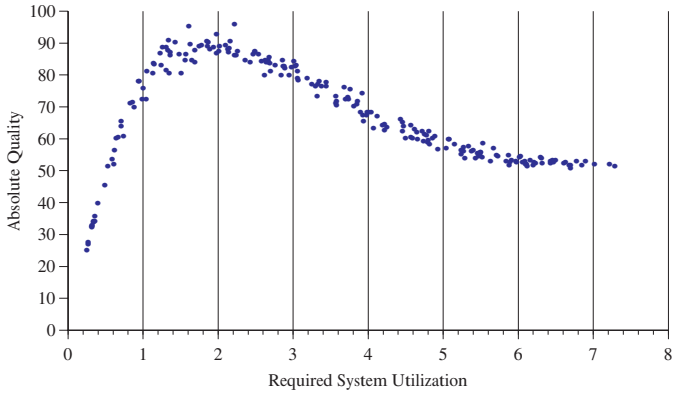

Fig. 7: Effect of system load on the absolute quality for a one agent system

The effect of detecting, communicating about, and reacting to the *facilitates* CR is to move this curve upward. Even though the indirect costs of delaying tasks due to exploiting facilitation may increase under heavy loads, the average relative quality increase remains at 0 (rather than becoming negative) because the communicating agents can do no worse than the non-communicating agents, who are continuously executing tasks with the fastest, minimal quality method. The indirect costs can show up in more missed deadlines before the non-communicating schedulers become saturated near a utilization of 5. Figure 8 shows that while the *relative* performance of the coordinating agents grew in Figure 6, the *absolute* performance actually levels-off (note that the 'max-quality' line represents all tasks being completed at maximum quality, which is an impossible ideal to ever achieve for a required utilization greater than one).

4.4 Delay

We ran a final suite of experiments to validate the effect of the delay time on performance. Assume again that task A *facilitates* tasks B_i as shown in Figure 9. The possible amounts by which to delay a task are bounded below (number 1 in Figure 9) by task A's estimated finish time for some quality, and bounded above (number 3 in Figure 9) by the minimum latest start time of all tasks B that can be started after A finishes *calculated to include the predicted effect of receiving the maximum quality result of task* A. For example, since the result of task A will reduce the amount of time required for tasks B_i, each task's latest start time would increase (move to the right) in Figure 2. Our choice, to delay to the minimum latest start time computed as if the result of A will *not* be received, is somewhere in between (number 2 in Figure 9).

We would like to show how this choice has the highest expected utility for the agent pair. Given an environment where power $\phi_{\mathrm{d}} = 50\%$, likelihood $\mathrm{Pr}_{\mathrm{fac}} = 0.5$, and

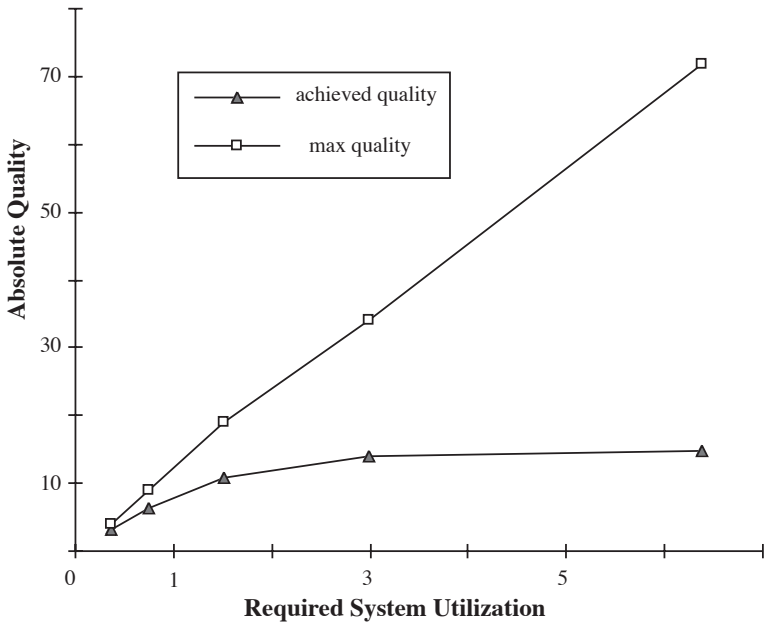

Fig. 8: Effect of system load on the absolute quality

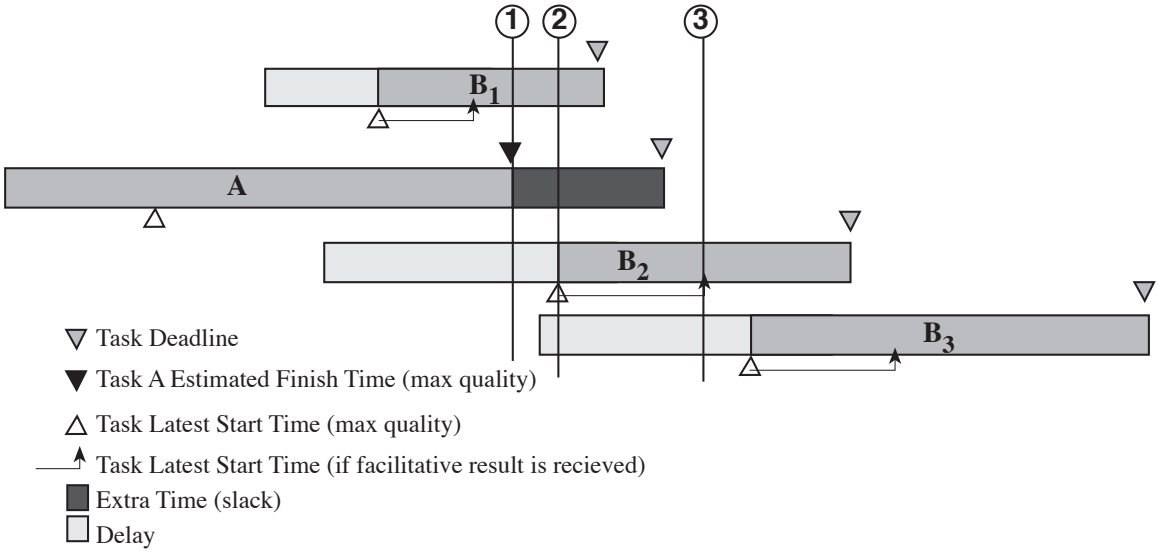

Fig. 9: Other ways of calculating delays.

a mean time between arrivals of 2.5, we achieve the following average percentage quality increases:

	Shortest Delay (1)	Normal Delay (2)	Longest Delay (3)
APQI	19.1	26.4	22.9

When an agent commits to finishing task A early, it does so with a soft deadline—the task is scheduled to finish at the soft deadline time but monitoring will not switch to a faster, lower quality method unless the hard deadline is threatened. To commit to a hard deadline in order to take advantage of a *facilitates* relationship would be a complex decision, as the agent would have to weigh the cost of a potential local loss of quality with the potential gain in quality (through reduced duration) at the remote node. In general, shortening the delay hurts quality because often the *facilitating* task A does not quite finish on time (it does not have a hard deadline) and so the *facilitated* task begins without A's result. Lengthening the delay can also hurt quality because sometimes task A does not complete with the required quality (remember, the current scheduler is not boundedly-rational with respect to minimum needed quality), and so the quality of any delayed task B suffers. Delay time is similar to the slack time that was investigated in Durfee and Lesser's predictability vs. responsiveness experiments[8].

5 Discussion and Conclusion

These experiments show that abstract simulations can produce interesting phenomena and can be used to validate hypothesis about coordination strategies. Our simulations lead us to believe that relatively simple models may be sufficient to explain and predict many observed coordination behavior characteristics in real environments. Such models can then be used both to design coordination algorithms for specific environments, and to build more sophisticated agents that can better analyze and apply the range of coordination behaviors available to them in a particular situation.

There are several short term directions we are currently pursuing with respect to this work. The first is in verifying our model of the expected utility of communicating the *facilitates* relationship, and the second is building a low-level model for the necessary task delay times to analytically explain our experimental observations. We need to modify the DTT real-time scheduler to handle quality constraints, and to exploit information about tasks arriving in the future (as did the original DVMT approximate processing scheduler [4, 12]).

The task environment model used for the experiments in this paper has a very simple view of task interactions (see the characteristics mentioned in Section 4.1, for example), but we believe decision rules of the form described in this paper will be useful in more complex environments. We have developed a much more comprehensive model (introduced in [5]) of task environments that allows a mathematical specification of task interactions for either analysis or simulation. We are using this model to analyze problems in CDPS, parallel scheduling, and real-time scheduling. For example, we are building a much more detailed model of distributed sensor network task environments, and are using this model to *predict* the effect of various agent organizations and show precisely when meta-level communication can improve performance.

Les Gasser and Michael Huhns, in the front matter of the 1989 collection *Distributed Artificial Intelligence, Vol. II*, include in a list of issues requiring further research [13]:

Deep Theories of Coordination: Researchers in DAI have developed several weak and highly constrained theories of coordination, which provide guidance and some techniques for designing DAI systems. In general, these are still too specialized and project-specific. We still have no broadly useful definitions of terms such as *coordination*, *cooperation*, or *interaction*. To be sure, we do have the "cooperation without communication", "rational deal-making", and "probabilistic interaction" theories of Rosenschein, Genesereth, Breese, and Ginsberg [20, 14, 19], but these are bound by highly restrictive assumptions. The promising "metalevel control" and "partial global planning" techniques of Lesser and his colleagues [3, 11] have not yet become full-fledged coordination theories that can guide us to other new practical coordination techniques.

This paper outlines a methodological approach toward building a theory of coordination. We have discussed our conceptual model of coordination, how it can be applied to the design of a coordination algorithm, and experiments that begin to verify and concretize that model. Among the key ideas presented here is the environmental analysis of CDPS systems that characterizes the features of the external environment, the problems being solved, and the architectures of the agents so that they can be used to design effective coordination algorithms. Also key is the separation and flow of information from the coordination relationships in the environment through the coordination algorithm to the local scheduler by means of additional or strengthened scheduling constraints.

Other work has had the characteristic of showing coordination techniques that are helpful, but not providing a deep analysis of when and why they are appropriate. We think that the right way to think about coordination is through the general coordination relationships we have discussed. These relationships are a step toward a theory of coordination in cooperative distributed problem solving.

Acknowledgments

The authors would like to thank Alan Garvey for his work on the real-time scheduler, and for producing the histogram in Figure 3.

Endnotes

* This work was supported by DARPA contract N00014-92-J-1698, and partly by the Office of Naval Research contract N00014-92-J-1450, and NSF contract CDA 8922572. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

[1] Other potential coordination relationships include *inhibits*, *cancels*, *constrains*, *causes*, *enables*, and *subgoal* [6].

[2] Our description of the coordination process is consistent with social views of organizational coordination mechanisms or behaviors: the use of rules, regulations, and standards; the creation of supervisory and decision-making hierarchies; and specialization or departmentalization. Organizational structure should be viewed as part of the coordination algorithm.

[3] Note, for example, that we can represent the cases where ϕ_{d} and/or ϕ_{q} are *negative*, resulting in 'negative facilitation'. Such a relationship may be useful for modeling the phenomena of *distraction* [3].

[4] For some relationships, there is also a chance the benefit will accrue serendipitously without the detection of the relationship. When possible, tasks can be structured to take advantage of this fact, which may reduce coordination costs considerably.

[5] An experiment not reported here showed that in an environment where the probability of a facilitates relationship drops off exponentially instead of linearly, the system response characteristics are similar to a linear environment with the *same number of detected relationships*.

[6] The total number of ways to distribute k tasks to n agents is n^k. The number of ways to distribute k tasks to i agents where each agent gets at least 1 task (surjections) is $i!S(k,i)$, where $S(k,i)$ are the Stirling numbers of the second kind. So the expected number of n total agents that are involved in a k-cluster is:

$$\sum_{i=1}^{n} i \frac{\binom{n}{i} i! S(k,i)}{n^k}$$

[7] Two data points on the utilization = 6 line, (power = 75, APQI = 555) and (power = 100, APQI = 1010), were left out for clarity.

References

[1] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, August 1989.

[2] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[3] Daniel D. Corkill and Victor 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*, pages 748–755, August 1983.

[4] Keith Decker, Alan Garvey, Marty Humphrey, and Victor Lesser. A blackboard system for real-time control of approximate processing. In *Proceedings of the 25th Hawaii International Conference on System Sciences*, January 1992. Extended version to appear in the International Journal of Pattern Recognition and Artificial Intelligence 7(2) 1993.

[5] Keith S. Decker, Alan J. Garvey, Victor R. Lesser, and Marty A. Humphrey. An approach to modeling environment and task characteristics for coordination. In Charles J. Petrie, Jr., editor, *Enterprise Integration Modeling: Proceedings of the First International Conference*. MIT Press, 1992.

[6] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(2), June 1992.

[7] Keith S. Decker, Victor R. Lesser, and Robert C. Whitehair. Extending a blackboard architecture for approximate processing. *The Journal of Real-Time Systems*, 2(1/2):47–79, 1990.

[8] E. Durfee and V. Lesser. Predictability vs. responsiveness: Coordinating problem solvers in dynamic domains. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 66–71, August 1988.

[9] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Cooperative distributed problem solving. In A. B. Barr, P. Cohen, and E. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume 4, pages 83–147. Addison Wesley, 1989.

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

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

[12] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6), 1993. Special Issue on Scheduling, Planning, and Control.

[13] L. Gasser and M. N. Huhns, editors. *Distributed Artificial Intelligence, Vol. II*. Morgan Kaufmann, 1989.

[14] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein. Cooperation without communication. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, PA., August 1986.

[15] Carl Hewitt. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47(1):79–106, 1991.

[16] H. Kise. A solvable case of the one-machine scheduling problem with ready and due times. *Operations Research*, 26(1), 1978.

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

[18] Thomas W. Malone. Modeling coordination in organizations and markets. *Management Science*, 33:1317–1332, 1987.

[19] J. S. Rosenschein and J. S. Breese. Communication-free interactions among rational agents: A probabilistic approach. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence, Vol. II*. Pitman Publishing Ltd., 1989.

[20] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, August 1985.

[21] Frank v. Martial. *Coordinating Plans of Autonomous Agents*. Springer-Verlag, Berlin, 1992. Lecture Notes in Artificial Intelligence no. 610.