Using Partial Global Plans to Coordinate Distributed Problem Solvers

Edmund H. Durfee and Victor R. Lesser
Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

Communicating problem solvers can cooperate in various ways, such as negotiating over task assignments, exchanging partial solutions to converge on global results, and planning interactions that help each other perform their tasks better. We introduce a new framework that supports different styles of cooperation by using *partial global plana* to specify effective, coordinated actions for groups of problem solvers. In this framework, problem solvers summarise their local plans into *node-plant* that they selectively exchange to dynamically model network activity and to develop partial global plans. However, because network and problem characteristics can change and communication channels have delays and limited capacity, problem solvers' models and partial global plans may be incomplete, out-of-date, and inconsistent. Our mechanisms allow problem solvers to agree on consistent partial global plans when possible, and to locally form partial global plans that lead to satisfactory cooperation even in rapidly changing environments where complete agreement is impossible. In this paper, we describe the mechanisms, knowledge representations, and algorithms that we have developed for generating and maintaining partial global plans in a distributed system. We use experiments to illustrate how these mechanisms improve and promote cooperation in a variety of styles.

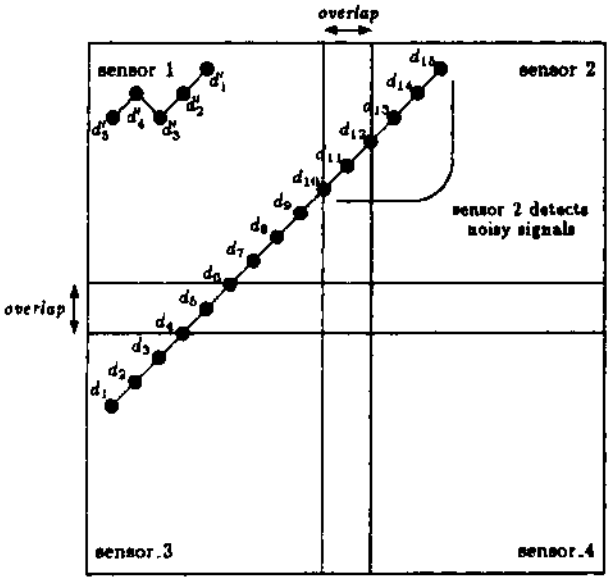
I. Introduction

Communicating problem solvers can pool their resources and expertise, work in parallel on different parts of a problem to solve it faster, avoid harmful interactions such as resource conflicts or working at cross-purposes, and promote helpful interactions such as moving information to where it is most needed or tasks to where they can best be performed. The style in which the problem solving *node$* should cooperate depends on problem domain and environmental characteristics, and there are a variety of styles for which different mechanisms have been developed. Sometimes the nodes channel all of their information to coordinating nodes that generate and distribute multi-agent plans [Cammarata *et* a/., 1983, Corkill, 1979, Georgeff, 1983, Konolige, 1984, Steeb *et* a/., 1986]. When communication is very expensive, however, nodes should work relatively independently and selectively exchange their local solutions to converge on global solutions in a functionally-accurate/cooperative (FA/C) manner [Corkill, 1983, Lesser and Corkill, 1981]. Alternatively, they may negotiate in small groups to contract out tasks in the network [Davis and Smith, 1983, Smith, 1980].

The trouble with having different mechanisms for each style of cooperation is that some distributed problem solving situations call for several styles simultaneously. For example, consider a vehicle monitoring problem where nodes track vehicles moving through an area monitored by acoustic sensors. A sample problem situation (Figure 1) has four problem solving nodes, each connected to a different sensor (node i to sensor i) that supplies it with signal information at discrete times. A node tries to combine signals into tracks, which we represent as d_i-dj whered, is data for the track's first sensed time and d_j is data for its last. When all the data is present, then from a local perspective: node 1 plans to develop two tracks $(d_1'-d_8'$ and $d_4-d_{12})$; 2 plans for one track $(d_{10}-d_{15})$, but because its sensor is faulty it will need much time to filter out noisy data; 3 plans for one track (d_1-d_6); and 4 has no local plans. From a global perspective: node 1 should first work on track d_4-di_2 since it has more global significance (joining the tracks of 2 and 3), and also should quickly generate and send a predictive result (like the short track ds-d_9 which borders on node 2's view) to help 2 disambiguate its noisy data; 2 should expect this predictive information; 3 should take responsibility for the data that both it and 1 sense (d_4-d_6) since 1 has more data to process; and 4 should take on some tasks, either by getting data from other nodes, or by acting as network coordinator, or both. The nodes therefore need to negotiate about task assignments, exchange partial results to converge on global solutions, and plan interactions that will help each other perform their tasks better.

Instead of developing a hybrid system that uses different mechanisms for different styles of cooperation, we have constructed a unified framework that supports all these styles through the use of *partial global plana* that specify how sets of nodes will act and interact. Each basic style

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8306327, by the National Science Foundation under Support and Maintenance Grant DCR-8318776, by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041. Edmund Durfee was also supported by an IBM Graduate Fellowship.



The four overlapping sensors detect signal data at discrete sensed times (the dots with associated times). Sensor.2 is faulty and not only generates signal data at the correct frequencies but also detects noisy signals at spurious frequencies

Figure 1: Four-Sensor Configuration with Sensed Data.

of cooperation can be viewed as using partial global plans in some way. In the multi-agent planning style, the partial global plan is the multi-agent plan, while in the contracting style each contract is a partial global plan involving a pair of nodes. The FA/C style has an general, implicit partial global plan (in the form of an organizational structure) that predisposes nodes to exchange and integrate information. Within our new framework, nodes always use partial global plans to coordinate their behavior as best they can, and their style of cooperation depends on how they form, exchange, manipulate, and react to partial global plans. This framework lets nodes converge on common plans for network activity in a stable environment (where their plans do not change because of new data, failed actions, or unexpected effects of their actions). However, when network, data, and problem solving characteristics change and when communication channels have delay and limited capacity, the framework allows nodes to locally respond to new situations and to cooperate effectively even when they have inconsistent partial global plans.

Our approach to coordination emphasizes sophisticated local control: a problem solver uses its current local view of network activities to control its own actions. Therefore, although problem solvers selectively exchange information about local plans and partial global plans, each may use this information differently and asynchronously. Coordination is part of each problem solver's control activities and is interleaved with problem solving. This paper describes how this view of coordination through partial global plans has been implemented and shows how the new mechanisms allow nodes to coordinate in a variety of ways. In the next section, we briefly review how the problem solvers in our experimental domain plan their local activities. In section 3, we describe the implemented mechanisms for integrating local plans into partial global plans, covering such issues as how and where local plans are sent, which nodes develop partial global plans, what information about partial global plans is exchanged, and what happens if nodes have incomplete, obsolete, or inconsistent partial global plans. In Section 4, we use experimental results to evaluate the ability of the mechanisms to improve coordination and allow different styles of cooperation, and we discuss how network and problem characteristics affect how (and whether) nodes develop consistent views. Finally, we summarize our approach in Section 5, citing its strengths, weaknesses, and remaining open problems.

II. Distributed Vehicle Monitoring

The Distributed Vehicle Monitoring Testbed (DVMT) simulates a network of vehicle monitoring nodes, where each node is responsible for a portion of the sensed area and where the nodes develop partial tracks in parallel and exchange these to converge on a complete map of vehicle movements [Lesser and Corkill, 1983). A node applies signal processing knowledge to correlate the data, attempting to recognize and eliminate errorful sensor data as it integrates the correct data into an answer map. Each problem solving node has a blackboard-based architecture [Erman et al., 1980], with knowledge sources (KSs) and blackboard levels of abstraction appropriate for vehicle monitoring.

Each node has a *planner* that uses an abstract view of the problem solving state to plan sequences of actions for resolving uncertainty about the potential solutions to develop and for developing them [Durfee and Lesser, 1986, Durfee and Lesser, 1987]. The abstract view is built by clustering related data into a hierarchy of abstractions, and it allows the planner to recognize long-term problem solving goals (to track some type of vehicle through a particular region). In the sample situation (Figure 1), for example, node 1 develops an abstract view indicating that two vehicles may have passed through its area and roughly where they were at each sensed time. The planner develops a plan to form a track satisfying each goal: it sketches out its long-term activities (the order it will process data); it builds predictions about how long each of these activities will take and about their likely results; and it details a sequence of short-term actions to process the next data. The planner adds detailed actions to the plan incrementally because how (and whether) it processes subsequent data can depend on the results of earlier actions and on unexpected changes to its data. Thus, the node interleaves plan execution with plan generation, monitoring, and repair so that it can respond to unexpected situations. Node 1's plan to form track d'_1~d'_5, for example, eventually fails because the KSs that know about allowed vehicle movements do not give credibility to hypotheses involving such sharp zigzagging.

To coordinate nodes whose plans may change at any time, we could force nodes to pursue their plans no matter what happens: nodes that commit to plans are much more predictable [Fikes, 1982). Unfortunately, they are also unresponsive to changing circumstances. But nodes with the flexibility to respond to unexpected situations risk disrupting coordination when they take unpredictable actions that can lead them into interfering with each other or ignoring important tasks because they falsely assume other nodes are doing them. What we need are mechanisms that allow nodes to exchange useful information about their views and to use whatever local information they have to find the best balance between predictability and responsiveness.

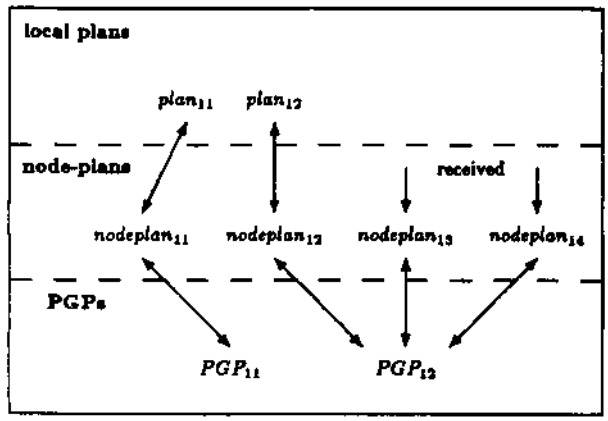
III. Implementation

To model each other and modify their local plans based on their perceived opportunities for cooperation, nodes must integrate and reason about large amounts of information. Implementing the concept of coordination through partial global plans is thus a difficult task, and this section describes important issues and algorithms we have identified.

A. Network Models

To locally plan actions that help overall network problem solving, a node needs a dynamic model of network activity (Figure 2). The node has *local plans* based on its own knowledge and local view (from the clusters of its abstraction hierarchy). For example, node 1 in Figure 1 will have two local plans, and the one to form the track d_4–d_{12} is outlined in Figure 2. The node's planner summarizes each of its local plans into a *node-plan* that specifies the goals of the plan, the long-term order of the planned activities, and an estimate of how long each activity will take (based on the time costs of the plan's past and current activities). The planner uses this information to generate the node-plan's *activity-map:* a series of activities, where each activity has a predicted starting time, ending time, and result track. The node-plan for node I's local plan to form track d_4–d_{12}, for example, is outlined in Figure 2. Since node-plans have much less detailed information than local plans and do not point to local data structures, nodes can cheaply exchange them and can reason about each other's node-plans as they can their own. Thus, nodes can communicate node-plans to build up models of each other [Corkill, 1979, Georgeff, 1984, Konolige, 1984].

A node's planner scans the model of the network to recognise *partial-global-goals* (PGGs). A PGG is global in the sense that it may (but does not necessarily) encompass the local goals of several nodes, and is partial in that only part of the network might participate in it. The planner identifies PGGs by comparing local goals and using simplified knowledge (in this domain, about allowable vehicle movements) to determine whether they are part of a larger goal (tracking the same vehicle). For each PGG, the planner forms a *partial-global-plan* (PGP) that represents the concurrent activities and intentions of all of the



$plan_{12}$: cluster information = (cl_{11} ... cl_{1n})
goal track = ((4 $region_4$) ... (12 $region_{12}$))
goal vehicle types = (1 2 ...)
long-term times order = (4 5 ... 12)
long-term cost estimates = ((4 c_4) ... (12 c_{12}))
past actions = (4 $action_1$ ... $action_i$)
pending actions = (4 $action_{i+1}$ ... $action_n$)

$nodeplan_{12}$: goal track = ((4 $region_4$) ... (12 $region_{12}$))
goal vehicle types = (1 2 ...)
long-term times order = (4 5 ... 12)
long-term cost estimates = ((4 c_4) ... (12 c_{12}))
activity-map = ((t_0,$t_0 + c_4$,d_4)
($t_0 + c_4$,$t_0 + c_4 + c_8$,d_4-d_8) ...)
plan = $plan_{12}$

PGP_{12}: participants = ($plan_{12}$ $plan_{21}$ $plan_{31}$)
goal track = ((1 $region_1$) ... (15 $region_{15}$))
goal vehicle types = (1 2 ...)
activity-map = ((t_0,$t_0 + c_4$,d_4) (t_0,$t_0 + c_1$,d_1) ...)
s-c-graph = (3 d_1-d_8) + (1 d_4-d_9) → (1 d_1-d_9) ...
communication = ((3 1 d_1-d_8) (1 2 d_1-d_9) ...)

The network model of node 1 from Figure 1 it graphically depicted along with simplified views of the data structures. A local plan has pointers to local data clusters, a goal track (a region for each sensed time) and vehicle types, the long-term order for processing data (data for sensed time t, then j, etc) and the estimated cost for each (time-cost pairs), and finally the specific KSs to process the next data. A node-plan has the local plan's goals and long-term information, and has an activity-map (each activity has a start-time, end-time, and result-track). A PGP points to participating plans, combines their goals, interleaves their node-plans' activity-maps, and has a solution-construction-graph and communication predictions (for example, node 3 forms d_1–d_9 and node 1 forms d_4–d_9 and node 1 combines them into d_1–d_9).

Figure 2: An Example of a Node's Network Model.

nodes that are working in parallel on different parts of the same problem (to potentially solve it faster). For example, the PGP to generate the overall track d_1-d_{15} (Figure 1) by combining the plans of nodes 1, 2, and 3 is outlined in Figure 2. The planner interleaves the participating node-plans' activity-maps to recognize the relative timing of the nodes' activities and to discover how activities might be reordered to avoid harmful interactions (such as performing redundant activities) and to promote helpful interactions (such as providing predictive information sooner).

Given a suitably ordered set of activities for the participating nodes, the planner uses this view of how nodes will act to develop expectations about how they will interact. It estimates when a node will complete a group of activities that together form a sharable result, and forms a *aolution-construction-graph* that indicates how and where results should be integrated. For example, in the situation of Figure 1, node 3's track d_1-d_3 and node l's track d_4-d_9 may be combined at node 1 to form track d_1-d_9 (Figure 2). The planner uses its network model to assign integration tasks to nodes with available computation resources or suitable expertise. Thus, while the activity-map provides details about how each node will form its own results, the solution-construction-graph provides a high-level view of how the nodes are pooling resources and working together. This view of node interactions helps nodes avoid wasting communication resources because they can better identify important communication actions (to send track d_1-d_3 from node 3 to node 1 as in Figure 2, for example).

To summarize, a network model has three types of information:

local plan: The representation of a plan maintained by a node that is pursuing the plan. Contains information about the plan's objective, the order of major plan steps, how long each is expected to take, and detailed actions (KSs) that have been taken or will be taken.

node-plan: The representation of a plan that nodes communicate about. Contains information about the plan's objective, the order of major plan steps, and how long each is expected to take. Details about short-term actions are not represented.

PGP: The representation of how several nodes are working toward a larger goal. Contains information about the larger goal, the major plan steps that are occurring concurrently, and how the partial solutions formed by the nodes should be integrated together.

A PGP can be formed for any number of nodes with compatible local goals. Initially, a node's PGPs correspond only to its local plans, but, as information from other nodes arrives, it builds larger, more encompassing PGPs. Because nodes build their network models asynchronously and over time, they may be incomplete or out-of-date and cooperating nodes may have inconsistent PGPs. The extent and quality of a node's network model and how it is formed depends on the *meta-level organization:* the communication topology, capacity, delay, and reliability; the coordination responsibilities of different nodes; the credibility that a node has in coordination information from other nodes (which determines their authority relationships); and so on. The distributed problem solving network is therefore organized both in terms of problem solving responsibilities (the domain-level organization) [Corkill, 1983, Corkill and Lesser, 1983] and in terms of coordination responsibilities (the meta-level organization).

We assume in this paper that the meta-level organization is statically defined during network creation. A node sends its node-plans to those nodes specified in the organization, perhaps to a particular coordinator-node, or maybe to all other nodes so that they recognize P G P S individually (as in Figure 2 where node 1 receives node-plans from 2 and 3 and forms its own PGPs). When it coordinates other nodes, a node may send them PGPs to guide their actions, but two nodes with equal authority may also exchange PGPs to negotiate about (converge on) a consistent view of coordination. A node that receives a node-plan or PGP considers the sending node's credibility when deciding how (or whether) to incorporate the new information into its network model: it can follow a highly-rated PGP from a much trusted coordinator-node, but may disobey coordination requests if the credibility and ratings of its local information is superior. The meta-level organization therefore influences how nodes may converge on consistent views of network activity and how responsive they will be, allowing various levels of autocracy and democracy, obedience and insubordination.

Although they often convey similar information, node-plan and PGP messages serve different purposes. In some organizations, a node might not have authority to locally change a received PGP even if it believes the changes represent improvement. By sending its local view as a node-plan, it might persuade a node with more authority to change the PGP, or it could persuade other low-authority nodes that they should all change their PGPs together (in a kind of "grass-roots" movement). PGP messages say how nodes *art* working together, while node-plan messages provide context for deciding how nodes *might* cooperate. Finally, nodes could send only some of a PGP's information (leaving out the planned activities of some participants) so that recipient nodes have insufficient context to recognize and suggest improvements. This enforces consistent views and reduces computation at recipient nodes, since they blindly follow the PGP and cannot explore alternatives.

B. Group Activity

A PGP's activity-map interleaves the concurrent activities of the participating node-plans, and each activity has an estimation of when it begins, when it ends, what task (part of a track) it is working on, and what results (tracks) it will produce. The planner scans the activity-map to find activities that are more useful (such as activities that form important results to share) or less useful (such as activities that unnecessarily form redundant results). Each activity is rated based on attributes such as its expected time costs, its expected result quality, how it will be affected

by preceding activities, and how it will affect later activities. The planner attempts to reorder activities to move more highly-rated ones earlier in the plan. For example, node 1 (Figure 1) has a plan to build track d_4 d_{12} and, since it locally has no reason to prefer certain activities over others, it chose (4 5 6 7 8 9 10 11 12) as the ordering (Figure 2). When the activities of nodes 2 and 3 are incorporated in the PGP, however, node I's activities are no longer equally rated: because they provide node 2 with predictive information, the activities for generating tracks neighboring node 2 are more highly rated (for example, the partial track d_8–d_9), but because they may generate redundant results, the activities for times 4-6 and 10-12 have their ratings lowered. By rating activities in alternative orderings, the planner determines that node I's activities should be reordered to (9 8 7 10 11 12 6 5 4).

To reduce planning overhead, the planner does not guarantee an optimal ordering (which would require a large search) but instead uses a less costly hill-climbing algorithm that generates a satisfactory ordering: it begins with the activity-map built from the node-plan activity-maps (which it expects each node to currently be following), and rates the activities. It then reorders the activities so that the most highly rated occur earlier, and then rates the activities in this new order. Because the ratings of activities depend on their relative ordering, reordering them may reduce ratings of some activities and raise ratings of others. The sum of the ratings before and after reordering are compared: if the original ordering has a higher total, then it is used; if the new order is better, then the process repeats until no better ordering can be found.

The planner uses the activity-map to form the solution-construction-graph. It first identifies the earliest times that different pieces of the overall solution will be generated and at what nodes, and then determines when and where they should be integrated into a single answer. For example, after reordering the activity-map as described above for the PGP to form track d_1–d_{15} (Figure 1), the planner makes a solution-construction-graph specifying that tracks *d1-dB* from node 3 and d7-d11 from node 1 should be combined at node 1, and the resulting track d_1–d_{11} should be combined at node 2 with that node's track d_{12}–d_{15}. Alternatively, in a slightly different network where node 4 has very good integration expertise (KSs), the solution-construction-graph has nodes 1, 2, and 3 send their tracks to 4 for integration. The planner builds the solution-construction-graph by: finding the pair of partial results that can be combined earliest (time both are at an integrating node plus an estimate of how long it will take, depending on its expertise, to combine them); adding the combination as a new partial result; and then repeating this process until a complete result is formed. This inexpensive, iterative algorithm generates a graph that is acceptable although possibly non-optimal.

The solution-construction-graph improves communication decisions since a node has a more global view of where results are needed than it has with a more local view [Durfee *tt* a/., 1985b, Durfee *et* a/., 1987]. For example, it knows that track d_1–d_6 should be sent from node 3 to node 1. A node's planner can also make better local decisions by identifying whether it is or is not responsible for a particular result, and how much time it has to generate that result. In situations with multiple solutions, integration responsibilities for a solution are assigned to one node so that others can more quickly move on to other solutions. Also, since two partial results cannot be integrated until they are both at the integrating node, the planner may identify cases where the node has some time to spare: it predicts that *n* time-units are needed to form and send the result, but that the other result will not be ready for n 4-1 time-units. Our mechanisms allow the planner to work on important activities for other PGPs (perhaps generating predictive information) during the other *i* time-units, treating the *n* + *i* time-units as a window in which the task to generate the result can be moved around [Vere, 1983].

C. Planning Node Activities

The planner reasons about the concurrent actions of nodes and about their potential interactions to find the next problem solving action for the node to take, as shown in Figure 3. It first uses any received network information (node-plans and PGPs) to update the network model. It then finds the current-PGP: the PGP that specifies activities that the node should do at this time. The local plan that contributes to this PGP is updated and the next action is found. For example, if the PGP indicates that the local plan should develop data in a different order, the plan's long-term information is changed to reflect this and, if necessary, detailed short-term actions are found for the next data to process. When it is updated, the local plan may become inactive—it may not yet have data in the area where it is expected to work—so several PGPs may need to be tried before one with an active plan is chosen and node problem solving can continue (Figure 3, steps 2d-f). Finally, any highly-rated PGPs or node-plans that have been altered are sent to whatever other nodes should be informed (based on the meta-level organization). Sending only highly-rated information can reduce communication costs and the number of PGPs (combinations of node-plans), but may cause views to be inconsistent or important PGPs to be missed. A parameter decides how highly rated information must be to be sent. When more or less complete communication would improve planning, the experimenter (and in future implementations perhaps the node itself) may alter this parameter.

When finding the current-PGP, the planner first updates its set of local plans based on any new data from its sensors or other nodes (Figure 3, step 2a). The new data modifies the abstraction hierarchy, and the planner forms new plans for new potential solutions and modifies existing plans whose clustered information has changed. Node-plans are created for any new plans and the node-plans of modified plans are updated. The planner then updates the network model using new and updated node-plans either formed locally or received (Figure 3, step 2b). It updates the PGPs of any updated node-plans and uses new node-plans to either update existing PGPs (if com-

A node's planner will:

1. receive network information;
2. find the next problem solving action using network model:
   (a) update local abetract view with new data;
   (b) update network model, including PGPs, using changed local and received information (factoring in credibility based on source of information);
   (c) map through the PGPs whose local plans are active, for each:
      i. construct the activity-map, considering other PGPs;
      ii. find the best reordered activity-map for the PGP;
      iii. if permitted, update the PGP and its solution-construction-graph;
      iv. update the affected node-plans
   (d) find the current-PGP (this node's current activity);
   (e) find next action for node based on local plan of current-PGP;
   (f) if no next action (local plan inactive) then go to 2b (since local plans may have changed), else schedule the next action;
3. transmit any new and modified network information.

Figure 3: The Principal Planning Activities.

patible) or to generate new PGPs (if incompatible with all current PGPs). If any PGPs have been modified or created (as a result of changed node-plans or reception of credible PGPs), the planner then checks the set of PGPs, merging together any that are now compatible and separating any that are no longer compatible. For example, if the same vehicle passes through a node's area twice, then the node initially develops separate PGPs for the two potential tracks. If it later receives node-plans from other nodes indicating that its two potential tracks are connected, then it merges the PGPs into a single largeT PGP.

Once the network model is updated, the planner proceeds to find the current-PGP. It first finds the PGPs to consider (leaving out PGPs that have already failed to generate useful actions because their local plans are inactive) and orders them. It also decides what nodes to plan for—usually just the current node, but if this node is also to coordinate others it should plan for them as well. The planner then steps through the PGPs from highest rated down (Figure 3, step 2c), updating their activity-maps and solution-construction-graphs, until all the desired nodes are planned for or no PGPs remain. For example, when the nodes it should plan for do not all participate in the same PGPs, the planner must update multiple PGPs until a current activity is found for each node.

When updating a PGP, the planner first generates a current activity-map by interleaving the activities of each of the participating plans (Figure 3, step 2ci). For the local plan, the planner uses the past and predicted steps straight from the plan data structure. For non-local plans, the planner can get the activities from two potential sources: from a received node-plan (if there is one) or from the activity-map of the PGP (it may have been received from a node that had the node-plan). If the planner has information from both sources, it chooses between them using the information accompanying received node-plans and PGPs specifying how current and credible the model of the plan is. If the planner has neither source, it must have received the PGP (it could not have formed it locally without the node-plan) with the sending node intentionally holding back information so that this node could not generate its own (possibly better) activity-map but instead must blindly follow the activity-map supplied with the PGP.

If it forms one, the planner checks the activity-map against any PGPs that it has already planned for. A node that participates in this PGP may also be part of a previously formed PGP, and the activity-maps are compared to make sure the node is not expected to do two things at once. When there is a conflict, the node's activities in this (less highly-rated) PGP are moved to future, non-conflicting times. The planner then uses the hill-climbing algorithm previously described to reorder activities for better coordination (Figure 3, step 2cii). The sum of the activity ratings for the new activity-map is multiplied by the node's credibility in its own plans, and this value is compared with the value of the previous activity-map (if any).[1] If the value of the new activity-map is higher, the planner updates the PGP with the new activity-map (Figure 3, step 2ciii), forms the solution-construction-graph and communication expectations, and modifies its local plans and their node-plans based on the better activities (Figure 3, step 2civ). When the activity-map is improved but the credibility factors (authority relationships) do not allow the planner to change the PGP, the planner can transmit the node-plans that it believes should be modified: it assumes that if nodes with authority have the same view of these node-plans that it has, then those nodes will modify the PGP appropriately and the modified PGP will eventually be sent back to this node.

Once all of the PGPs have been updated and an action has been found for the node, the final step is to send out any important modified node-plans and PGPs, as described previously (Figure 3, step 3). To make problem solving decisions based on the best, most up-to-date view of network activities, a node invokes the entire series of activities—from modifying the network model, to developing PGP activity-maps, to sending out new information—each time it needs to choose an action to take. If the node wants to conserve computational resources, it can do these activities less often at the cost of possibly making poorer control decisions. Because nodes are asynchronously performing coordination activities and are interleaving these activities with problem solving, they must each balance the costs and benefits of these mechanisms.

[1] A PGP has a previous activity-map if it was received or previously formed locally, and its value it the turn of its activity ratings multiplied by the credibility of the node that generated it.

IV. Evaluation

To evaluate the planning mechanisms, we must consider their ability to improve coordination, their ability to permit cooperation in a variety of ways (depending on the organization), and their costs (computation and communication). We have implemented the mechanisms as described in the DVMT, and here we summarize some initial empirical findings. This discussion concentrates on how well the implementation meets the goals of improving coordination and allowing cooperation in a variety of styles, and only briefly addresses the costs of these mechanisms.

Using the situation in Figure 1 (which was constructed deliberately as a challenge to coordinate), we show how the new mechanisms improve network problem solving by considering three degrees of partial global planning: where no network information is exchanged (nodes have only local views); where network information is exchanged so that nodes can rate their plans based on global significance but *cannot* change local plans based on this view; and where nodes *can* change their local plans by reordering activities. In all the experiments, the domain-level organization lets nodes exchange hypotheses so any node can potentially form solutions. The first meta-level organization that we explore is broadcast—nodes broadcast their node-plans to each other (with a simulated communication delay) and individually form PGPs. Although all of the data is present at the start of problem solving, nodes can have inconsistent network models because their plans change over time (for example, when node 2 receives predictive information from node 1 it reduces the predicted time needs for processing its data). The results of the experiments are summarized in Table 1, experiments E1-E3. For each experiment we show the amount of time the network needed to find a solution (where each time unit corresponds to the execution of a KS) and the average plan-message traffic (node-plans and PGPs) in the network. The results show that the new mechanisms can substantially reduce the solution time at the cost of increasing communication (recall that a node currently transmits every highly-rated node-plan or PGP that it modifies without regard to the significance of the modification). We have found similar results in other problem situations.

A broadcast organization allows inconsistencies and uncoordinated behavior because a node's view of its own plans is more up-to-date than its view of other's plans. In one problem situation, for example, two nodes that had both planned to work on the same data assumed that the other would work there and changed their plans. The messages about the changed plans incurred communication delays, so time elapsed before the nodes recognized that the one of them involved in less highly-rated PGPs (as determined by their local models of network activity) should change its plan back again. In situations where their models are inconsistent, however, both or neither may change plans. A centralized meta-level organization can reduce such inconsistencies. In experiment **E4,** nodes 1-3 send node-plans to 4 which in turn sends PGPs back. Nodes 1-3 cannot locally modify the PGPs they get from 4, so that they are all following consistent (received) PGPs. This organization involves less communication than the broadcast but takes longer to respond to new situations (and to initially get multi-node PGPs to nodes 1-3) because of combined communication delays to and from node 4. It still performs fairly well because the nodes are following completely consistent views. This can be compared to a "ring" organization (experiment E5), where a node sends node-plans only to its clockwise neighbor. The ring organization also uses less communication than the broadcast, but performs worse than either of the other organizations. Delays in propagating network information around the ring cause nodes to have inconsistent views more often and for longer periods of time, impairing coordination.

Expt	Organisation	Mechanisms	Time	AveMsgs
E1	broadcast	local only	139	0
E2	broadcast	PGPs, no reorder	46	3.4
E3	broadcast	all	27	4.6
E4	centralised	all	30	3.0
E5	ring	all	38	3.0

Legend

Organisation:	broadcast of node-plans, central coordinator, or node-plans passed around a ring
Mechanisms:	local only (no node-plan communication) PGP, no reorder (PGPs, no local changes) all (all mechanisms for forming PGPs)
Time:	Earliest time a solution was found (how many KSs each node ran)
AveMsgs:	Average number node-plan and PGP messages transmitted in the network at any time.

Table 1: Experiment Summary.

Our experiments indicate, not surprisingly, that the quality of coordination depends substantially on how consistent the different nodes' network models are. Inconsistencies can have many causes such as communication delays and inaccurate estimates of when various pieces of the solution will be formed. When estimates are inaccurate, the communication and integration decisions may need to be modified and the changed plans communicated. So long as updated information is exchanged, nodes can in time recover from unexpected situations and once again coordinate their activities. The planner also can develop a more forgiving PGP by enlarging the duration estimates for activities, and thus providing some leeway in predicted interactions. The resulting PGP is less likely to need alteration, but also may have less crisp interactions between nodes. The balance between forming PGPs that anticipate incorrect predictions versus using communication to update PGPs depends on the communication resources and performance requirements.

Developing and maintaining PGPs involves computation and communication overhead. PGPs substantially reduced the number of KS executions needed to solve our complex experimental situations, but, for simpler cases where nodes have little uncertainty about how to coordinate, the new mechanisms may introduce unnecessary overhead. The tradeoffs in improving problem solving at

the cost of increased planning overhead in a single node have been studied [Durfee and Lesser, 1986], but the evaluation becomes much more complex in a distributed system (Durfee *et al.*, 1985a] because of issues such as communication/computation tradeoffs, reliability, maintenance of consistent views, and assignment of planning responsibilities. For example, a broadcast is more costly than a centralised organization in both communication and computation (since each node forms similar PGPs) but can be much more responsive and reliable (since the network does not depend on a single coordinator).

Prescribing appropriate meta-level organizations and styles of cooperation for generic problem situations depends on the evaluation criteria considered, and is beyond the scope of this paper. What is important, however, is that our framework lets us explore different styles of cooperation using one set of mechanisms. We can have nodes pass around node-plans and plan for themselves, or send node-plans to a node that plans for everyone, in which case we can even force them to sit idle waiting for these PGPs by having them give no credibility to their locally developed PGPs. Or we could let nodes exchange PGPs so that they negotiate on a consistent global view by adopting the most highly-rated version of a PGP. Nodes can also use node-plans and PGPs to form contracts. A node with a particular task can generate a PGP activity-map that has the task being performed in the future at some remote nodes.[2] The PGP is sent to these nodes which locally develop node-plans representing these potential future tasks. These node-plans are modified and rerated to reflect the node's view of its own activities; for example, if it may form an activity-map for the node-plan indicating that the received tasks could not be performed until much later in the future. The nodes return these node-plans to the original node, which adopts the best one and follows it by sending task information (data) to the chosen node. In essence, one node requests bids for a cooperative plan and each of the others bids on how it expects it could cooperate. The node that likes the interaction most (or dislikes it least) is awarded the task. Moreover, because the returned node-plan conveys information about how the task fits into the bidding node's more global view, the originating node could use the node-plans it gets back to recognize that the task is unimportant and should not be awarded at all.

Depending on its relative credibility in received versus locally generated PGPs, a node could respond to an important local development by breaking its contract or disobeying a superior. The PGP-based framework for coordination permits different degrees of commitment for the various styles of cooperation. The nodes can be organized as predictable team-players or as locally responsive skeptics. In fact, a node with authority could elicit activity it desires from other nodes by misrepresenting itself or the network—changing its node-plans or PGPs based on some local goal of how it wants the network to behave regardless of network goals. Our framework therefore not only allows nodes with common goals to work as a better team, but also lets nodes with competing goals satisfy their own goals by misrepresentation ("lying") and exerting authority ("threats") [Rosenschein and Genesereth, 1985].

[2] The node's planner thus not only reorders activities but also finds possible reassignments that move computational load from a bottleneck node to nodes that are not participating in highly-rated PGPs. Making such reassignments is a complex problem for which we as yet have only primitive mechanisms.

V. Conclusion

A node in a distributed problem solving network must not only solve problems in its task domain but must also plan its activities to coordinate with others. However, planning is only important insofar as it improves domain problem solving: nodes should only plan as much or as little as they need to (or are able to) depending on the problem and network characteristics. The approach that we have described is therefore not strictly a distributed planning system because nodes may solve their domain problems without ever developing an overall network plan. Although our framework allows organizations where nodes can cooperatively develop and converge on a consistent distributed plan *before* domain problem solving begins, it emphasizes that planning usually occurs *during* problem solving and that nodes in an uncertain environment will build and modify their plans for network activity over time. When communication and computation resources are limited and when problem and network characteristics change rapidly, nodes simply may not be able to plan for optimal cooperation. Since the purpose of the network is to solve the domain problem, nodes need not cooperate optimally so long as they cooperate well enough to form acceptable solutions. What nodes need is the flexibility and local sophistication to use whatever information they have to decide how best to cooperate at a given time.

In this paper we introduced a new framework that uses partial global plans to promote many different styles of cooperation. Nodes build PGPs by exchanging short summaries of their local plans and recognizing when a group of nodes should work together. A node can hypothesize how the cooperating nodes could best interact and modify its local actions accordingly. We outlined important issues and algorithms in implementing these mechanisms, and showed experimentally how these mechanisms work.

Several avenues for further research remain. Nodes should be able to alter their meta-level organization: they need to exchange and reason about pertinent information to find a suitable style of cooperation for their current situation. Nodes also should use computation and communication resources effectively by selectively transmitting only plan information that will significantly impact network behavior and by selectively applying planning mechanisms to maximise the improvement in problem solving while minimizing the overhead costs. Finally, since nodes cooperate in so many different styles in this framework, we should try to develop some rules-of-thumb about how nodes should cooperate for generic problem situations.

Although our mechanisms have been implemented in a specific problem domain, we believe that they are applicable to distributed problem solving systems in general. Our description has stressed the basic knowledge representations and reasoning that goes on, and though the decisions about how goals and activities interact and how they fit into this representation is domain dependent, the planning and communication algorithms are not. We expect to use these mechanisms for coordinating concurrently running KSs in a multiprocessor blackboard-based problem solver, and, through this paper, we hope to encourage their application in other distributed AI systems as well.

Acknowledgments

We would like to thank Reid Smith for many suggestions concerning the content and presentation of this paper.

References

[Cammarata *et ai,* 1083] Stephanie Cammarata, David McArthur, and Randall Steeb. Strategies of cooperation in distributed problem solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pages 767-770, August 1983.

[Corkill, 1070] Daniel D. Corkill. Hierarchical planning in a distributed environment. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 168-175, August 1070.

(Corkill, 1083] Daniel David Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks.* PhD thesis, University of Massachusetts, Amherst, Massachusetts 01003, February 1083. Available as Technical Report 82-33, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1082.

[Corkill and Lesser, 1083] 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 748-756, August 1083.

[Davis and Smith, 1083] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence,* 20:63-100, 1083.

[Durfee and Lesser, 1086] Edmund H. Durfee and Victor R. Lesser. Incremental planning to control a blackboard-based problem solver. In *Proceedings of the Fifth National Conference on Artificial Intelligence,* pages 58-64, August 1086.

[Durfee and Lesser, 1087] Edmund H. Durfee and Victor R. Lesser. *Incremental planning to control a time-constrained, blackboard-based problem solver.* Technical Report 87-07, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, February 1087.

[Durfee et a/., 1085a] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. *Coherent Cooperation Among Communicating Problem Solvers.* Technical Report 85-15, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, April 1085. Also to appear in *IEEE Transactions on Computers.*

[Durfee *et ai,* 1085b] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Increasing coherence in a distributed problem solving network. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* pages 1025-1030, August 1085.

[Durfee *et ai,* 1087] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Cooperation through communication in a distributed problem solving network. In Michael N. Huhns, editor, *Distributed Artificial Intelligence,* Pitman, 1087. (In press. Also to appear as Chapter 7 in Scott P. Robertson, Wayne Zachary, and John Black, editors, *Cognition, Computing, and Cooperation: Collected works on cooperation in complex systems,* in press).

[Erman *et ai,* 1080] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys,* 12(2):213-253, June 1080.

[Fikes, 1082] R. E. Fikes. A commitment-based framework for describing informal cooperative work. *Cognitive Science,* 6:331-347, 1082.

[Georgeff, 1083] Michael Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pages 125-120, August 1083.

[Georgeff, 1084] Michael Georgeff. A theory of action for multiagent planning. In *Proceedings of the Fourth National Conference on Artificial Intelligence,* pages 121-125, August 1084.

[Konolige, 1084] Kurt Konolige. A deductive model of belief. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pages 377-381, August 1084.

[Lesser and Corkill, 1081] Victor R. Lesser and Daniel D. Corkill. Functionally-accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-II(I):81-06, January 1081.

[Lesser and Corkill, 1083] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring test bed: a tool for investigating distributed problem solving networks. *AI Magazine,* 4(3):15-33, Fall 1083.

[Rosenschein and Genesereth, 1085] Jeffrey S. Rosenschein and Michael R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* pages 01-00, August 1085.

[Smith, 1080] Reid G. Smith. The contract-net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers,* C-20(12):1104-1113, December 1080.

[Steeb *et ai,* 1086] Randall Steeb, Stephanie Cammarata, Sanjai Narain, Jeff Rothenberg, and William Giarla. *Cooperative Intelligence for Remotely Piloted Vehicle Fleet Control.* Technical Report R-3408-ARPA, Rand Corporation, October 1086.

[Vere, 1083] Steven A. Vere. Planning in time: windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-5(3):246-267, May 1083.