Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation

Edmund H. Durfee
Dept. of EE and CS
University of Michigan
Ann Arbor, MI 48109

Victor R. Lesser
Dept. of Comp. and Info Science
University of Massachusetts
Amherst, MA 01003

Keywords: Cooperative Problem Solving, Distributed AI, Interpretation, Vehicle Monitoring, Distributed Computing, Blackboard Systems, Planning, Negotiation

Abstract

For distributed sensor network applications, a practical approach to generating complete interpretations from distributed data must coordinate how separate, concurrently-running systems form, exchange, and fuse their individual hypotheses to form consistent interpretations. Partial global planning provides a framework for coordinating multiple AI systems that are cooperating in a distributed sensor network. By combining a variety of coordination techniques into a single, unifying framework, partial global planning enables separate AI systems to reason about their roles and responsibilities as part of group problem solving, and to modify their planned processing and communication actions to act as a more coherent team. Partial global planning is uniquely suited for coordinating systems that are working in continuous, dynamic, and unpredictable domains because it interleaves coordination with action and allows systems to make effective decisions despite incomplete and possibly obsolete information about network activity. We have implemented and extensively evaluated partial global planning in a simulated vehicle monitoring application, and have identified promising extensions to our framework.

[0]This research has been sponsored, in part, by the National Science Foundation under CER Grant DCR-8500332, by the National Science Foundation under under RIA Grant IRI-9010645 and CTCT Grant IRI-9015423, and by the Office of Naval Research under University Research Initiative Grant Contract N00014-86-K-0764, and under DARPA Contracts N00014-79-C-0439 and N00014-89-J-1877.

1 Introduction

While interpreting data from a geographically distributed network of sensors can be done centrally, a centralized approach suffers from several disadvantages. First, the central interpreter is a single point of failure, meaning that if the one central system fails then the entire sensed area becomes unmonitored. Second, the central interpreter is a potential bottleneck as several concurrently active phenomena in different parts of the network all compete for the single, central processing resource. Third, the communication needs in a centralized system are high, as large amounts of raw sensory data must travel long distances to the central interpreter.

In distributed sensor interpretation, interpretation systems are distributed around the sensor network, such that each system is responsible for only a local portion of the overall area. These systems interpret their local information and exchange only their most abstract interpretations in order to combine local views into a global interpretation. This approach increases reliability by allowing graceful performance degradation as systems fail. That is, despite the loss of one interpretation system, the working systems still monitor the remainder of the area. Because the distributed systems interpret their local data in parallel, there is a speedup in forming overall interpretations and there is no bottleneck. Finally, raw sensory data travels much shorter distances to local interpretation systems, and only highly processed interpretations need to be transmitted over long distances between interpreters. Thus, distributed sensor interpretation has many advantages over centralized interpretation.

To attain these advantages, however, the distributed interpretation systems must work together in a coordinated fashion to use their combined resources wisely and to integrate the partial local interpretations into a coherent overall view. What makes this coordination a challenge is that each system has its own local information and objectives. The collection of interpretation systems needs to not only solve interpretation problems, but also coordination problems. Coordination problems include how to:

- reconcile differences in interpretations;
- send information to guide other systems' interpretation processes;
- take advantage of received information to attend to promising interpretations and data;

- decompose and relocate interpretation tasks to exploit other systems' resources;
- coordinate the formation and exchange of partial and complete interpretations to form useful hypotheses in a timely manner.

We have developed partial global planning as a general coordination framework for solving these problems. Using a distributed vehicle monitoring application as the context for discussion, we have previously described our framework at a conceptual level [13], and at this level have illustrated its ability to balance predictability and responsiveness among problem solvers [15] and to enable task passing as well as result sharing [16]. In contrast, our objective in this paper is to solidify how the conceptual framework maps into the particular application domain. Seen another way, we are delving into the characteristics of distributed hypothesis formation in the vehicle monitoring domain in order to motivate a more complete description of the algorithms and data structures that transform our framework's concepts into practice. We present the concurrent activities of cooperating interpretation systems working in a specific experimental scenario to show how our framework improves coordination, and we summarize additional experiments involving larger numbers of cooperating systems. When networks become larger, our framework's ability to organize the cooperating systems hierarchically becomes increasingly important, but limitations in how concurrent activities are represented in our current framework become critical. These limitations are fundamental to all current frameworks, and we outline how our current research directions are addressing these and other issues.

1.1 Distributed Vehicle Monitoring

The partial global planning framework is particularly suited to coordinating problem solving in distributed sensor networks (DSNs) that are employed in applications whose characteristics include the following:

- more data could be sensed in an area than can be exhaustively processed in a timely manner;
- sensor and environmental noise generate data that need not be processed;

- correlations between data sensed at nearby locations provides constraints on whether and how that data should be processed;

- sensor overlap leads to possibilities for duplication of processing effort;

- sensing demands in a given area can vary over time.

The emphasis in such networks is thus not only on fusing data to form encompassing hypotheses, but also on using local and received information to guide processing decisions and to balance processing responsibilities. In the remainder of this paper, when we refer to a DSN, we are assuming a network with these characteristics.

One example application of such a DSN is distributed vehicle monitoring, where the distributed sensors attempt to identify and track vehicles as they move through an area. Figure **??** illustrates a simple network with four overlapping sensors. For simplicity, each sensor's range is represented as a two-dimensional, square area. As vehicles move through the overall area, they pass among the ranges of different sensors. A sensor collects data—in this case acoustic data—at discrete sensed times (indicated by dots on the vehicle tracks, where a dot corresponds to signals over a spectrum of frequencies and where the size of a dot indicates the sensed signals' strengths), and sends this data to its associated interpretation system(s). In our example, each sensor is connected to a different interpretation system (node), such that sensor-1 is connected to node-1, sensor-2 to node-2, and so on. The goal of the network as a whole is to converge on a consistent map of vehicle movements, generally by integrating the partial tracks formed at different nodes into a single complete map at any of the nodes or into a consistent set of local maps distributed among the nodes. The map information is passed on to some user of the DSN.

**** Figure ?? about here ****

We view the vehicle monitoring task as a search through the space of possible interpretations of sensory data to find plausible, consistent interpretations of that data. An important consideration in vehicle monitoring is that large amounts of potentially noisy data arrive continuously, so that a strictly data-directed approach that exhaustively processes all of the data is prohibitively costly and impractical. Fortunately, the interrelationships between data can be exploited to improve the signal to noise ratio by using partial interpretations to make predictions about subsequent data—predictions that can be used to focus processing on relevant and limited portions of the space of all possible interpretations. As vehicles move between the sensory ranges of different nodes, therefore, it is imperative that nodes exchange high-level, partial interpretations that supply predictive information to guide each other into developing important, consistent overall interpretations.

Thus, nodes must carefully and intelligently allocate processing resources to build useful hypotheses about vehicle movements quickly and to ignore noisy and errorful data. Although our framework supports tasks where sensor data arrives during interpretation activities, we will assume that all of the data shown in Figure **??** arrives at the nodes before the nodes begin processing the data. This simplification clarifies the discussion below, but we have applied our framework to problems where data arrives incrementally as well [10].

If each node works independently, then node-1 will process the more strongly sensed data in the upper left corner first (the d' track), eventually recognizing that the signals cannot be correlated into a believable track. The signals thus must correspond to sensor noise or echoes in the environment. Node-1 then processes the data that is part of the longer (d) track spanning the ranges of sensors 1-3. Node-2, meanwhile, has a large amount of noisy sensor data, and must spend substantial amounts of time forming alternative interpretations (identifying different possible vehicle types). Node-3 has clear data and quickly forms its piece of the overall track. Node-4 has no data to process.

By having the nodes coordinate their activities, we would hope that:

- node-1 would give preference to interpreting the data that is more likely to be part of a jointly-developed track, even though this data is less strongly sensed, because confidence in a track corroborated by several nodes is higher;
- node-1 would quickly form a partial interpretation near node-2's range, so that it can give node-2 predictive information about what type of vehicle it should track to be consistent with what node-1 is tracking;
- node-2 should use this predictive information from node-1 to focus first on interpretations that can extend the partial interpretation it gets from node-1;
- nodes 1-4 should evaluate how processing load is distributed among themselves to propose reallocations that would take advantage of otherwise idle resources (node-4);

- nodes 1-4 should decide when and where their local interpretations should be transmitted such that a node with available processing resources receives hypotheses to integrate in a timely manner.

Our partial global planning framework gives nodes the ability to make coordination decisions such as these so that the network as a whole can quickly develop the most promising interpretations of the data with less wasted computation and communication.

1.2 Partial Global Planning Overview

A principal concern in distributed sensor interpretation thus is how to control distributed activity to: (1) efficiently generate good candidate hypotheses; (2) filter out noise and data errors; and, (3) combine evidence and hypotheses from multiple sources into coherent interpretations. If uncontrolled, the systems might waste computing and communication resources by working at cross-purposes, misallocating processing tasks, exchanging useless information, and mistiming the exchange of information such that some systems are waiting inordinate amounts of time for results from others.

Our partial global planning framework is especially suited to coordinating distributed interpretation systems. Using our framework, interpretation systems that are individually capable of processing locally-received sensor data can be combined into a cooperative distributed sensor network. Partial global planning gives an individual system the ability to (1) represent its own expected interpretation activities, (2) communicate about these expectations with others, (3) model the collective activities of multiple systems, (4) propose changes to one or more systems' interpretation activities to improve group performance, and (5) modify its planned local activities in accordance with the more coordinated proposal. Moreover, because new sensor data can arrive at different sites asynchronously, our framework allows systems to incrementally and adaptively coordinate their activities as circumstances change. Hence, partial global planning is a flexible framework for improving coordination, leading to optimal group performance in static domains (where nodes can accurately model how they will work on stable sets of data), and to satisfactory performance in rapidly evolving situations.

In the next section, we review prior technology for coordinating multiple AI systems, with particular emphasis on research in distributed interpretation systems. We then describe the foundations of the partial global planning framework, showing how it generalizes and encompasses many of the past approaches. Next, we describe a prototype implementation of partial global planning for a DSN task, and show how it can promote coherent teamwork among systems that are performing distributed hypothesis formation. Our experiments in evaluating this prototype are discussed, and we conclude by outlining our current research directions for improving on and extending the partial global planning framework.

2 Coordinating Problem-Solving Systems

The partial global planning framework builds on ideas from the fields of artificial intelligence (AI) and distributed computing. AI contributes techniques for flexibly applying multiple sources of knowledge to identify, reason about, and resolve uncertainty in possible interpretations of data and possible ways of coordinating. Distributed computing exploits the inherent parallel nature of distributed sensing tasks to increase the reliability and speed of processing. Our approach falls in the area where these fields overlap, called cooperative distribute problem solving (CDPS), which is concerned with how multiple intelligent systems can reason about their individual and collective behaviors in order to cooperatively solve large problems [19].

CDPS has garnered a small but growing research community over the last 10 years [1, 22, 25]. While most CDPS researchers have AI backgrounds and use AI methodologies, they have been drawn to the study of CDPS from different motivations. Some have uncovered CDPS issues when trying to apply AI techniques to inherently distributed problems, while others have been motivated by having distributed computing technology in need of applications. Some face pragmatic issues of extending the capabilities of existing knowledge-based systems by allowing those systems to work together, and others use CDPS as a methodology for testing theories about human cooperation.

We can characterize the past CDPS approaches that are relevant to the DSN domain into 4 categories: contracting, result-sharing, organizing, and planning.

2.1 Previous Approaches

Contracting. Contracting views CDPS as a process where large tasks are decomposed, subtasks are distributed among appropriate problem solvers, subtasks are achieved in parallel, and subtask results are routed to suitable nodes who synthesize the results of larger tasks from subtask results. Hence, the underlying perspective is one where there is one big problem and many potential problem solvers, and the goal of coordination is to utilize the problem solvers to the utmost. This view of coordination as decomposing and distributing tasks is often called *task-sharing* [35].

Research in this area has principally focused on subtask allocation, and the most well-known research developed the Contract-Net protocol that allowed nodes to use a bidding process to distribute tasks [7, 34]. This technique was applied to a DSN task, with particular emphasis on initializing the network so that appropriate nodes were allocated to different sensor areas to ensure coverage. The overall task, that of building a map of vehicle movements through a large area, is given to a monitor node, which acts as the interface between the user and the network. The monitor node decomposes the overall area into subareas, and for each subarea announces a request for bids from other nodes who might take responsibility for that subarea. Nodes respond with bids that indicate their suitability for the task based on how well they fall within the area to be monitored. The monitor node awards the task to the node with the best bid. A subarea manager then further decomposes the task into subtasks for detecting groups of related signals and for tracking different types of vehicles. These tasks are contracted out using the bidding protocol, and the process continues until all tasks have been decomposed to a primitive level and contracted out. At this point, the network is ready to begin monitoring the area because each node has gotten its responsibilities and each task has been assigned to some node.

Contracting represents a flexible technique for making pairwise allocation decisions, where contractors use their bids to have input into what tasks they are assigned, and managers award tasks to the best of the bidders. Thus, contractors and managers mutually select each other. Mutual selection is a characteristic of the protocol that sets it apart from simpler mechanisms for coordination where “masters” unilaterally decide what tasks “slaves” will perform. Other techniques involving contracting and negotiation have addressed applications for communication network management [3], for manufacturing systems [32], and for choosing a leading controller for rerouting air traffic [2].

Result-Sharing. While the emphasis in contracting is in distributing tasks that are assumed to initially arise in one location, result-sharing concentrates on problem domains where tasks are inherently and possibly unpredictably distributed [35]. In such domains, a number of problem solvers are distributed such that each has its own local information that it uses to solve problems, but the problems that the nodes are solving are potentially related. Because subproblems can be related differently at different times, and because nodes might have several different subproblems to solve at any given time, a node cannot locally determine which other nodes are currently working on related problems. Lacking a more global context for processing its data, a node initially can only form tentative partial solutions from its data. The nodes must then engage in an iterative exchange of their tentative partial solutions, so as to identify relationships between their local problems and to converge on consistent local solutions. Because of the uncertainty they face, nodes must be able to recover from incorrect initial decisions and tolerate inconsistencies in their partial solutions. Result-sharing through iterative exchanges of tentative, uncertain information has been termed *functionally-accurate, cooperative* [27].

Considering how differently task-sharing and result-sharing approaches view cooperative problem solving, it is interesting to note that DSNs have been a fertile domain for studying each approach. Result-sharing DSN research has assumed that monitoring nodes are distributed throughout the overall area to begin with, and that there does not have to be a hierarchical management structure above them as occurs in a contracting approach. Instead, each node tracks vehicles within its area, and exchanges information with other nodes to build more global maps. The challenge in coordinating nodes is in giving each enough knowledge to anticipate which other nodes could use its partial results, and to even focus its local processing on generating partial results that are of interest to others. Providing the additional knowledge for this type of reasoning has been a motivation behind the organizing approach to CDPS, as well as to our partial global planning approach.

Organizing. Nodes that are organized have some general long-term knowledge about each other's roles, interests, and responsibilities in network problem solving. Organizational knowledge can be coupled with result-sharing techniques to guide nodes' local processing and communication activities, and can also be combined with task-sharing to help nodes focus bid requests and other messages toward a relevant subset of the network. CDPS researchers have used organizational concepts to propose alternative perspectives on how groups of problem solvers coordinate their actions, including using organizational structuring as a form of meta-level control [5], viewing coordination as a process of organizing based on settling and unsettling sets of questions [23], and applying organizational intuitions such as the scientific community metaphor [26].

Of particular interest to DSN applications is the work on organizational structuring to control how nodes in a DSN form and exchange partial solutions [6]. This work explored differences between hierarchical and lateral styles of problem solving, and showed how a result-sharing approach could be constrained to work effectively. One interesting observation that arose from this work is that nodes in a network need some amount of "skepticism" regarding information they receive from each other to avoid conforming prematurely to one interpretation before independently searching for promising alternatives.

Another observation that this work led to was that, to be generally effective, an organization cannot restrict the roles of the nodes too much, but instead must give each enough freedom to find activities that it can perform in a range of situations. On the other hand, the more freedom a node has, the more difficult it is for other nodes to anticipate its actions [17]. Thus, it appeared that additional, more dynamic, mechanisms must augment the use of organizations—mechanisms that allow nodes to communicate about which of their possible roles they were currently playing.

Planning. Planning, when applied to coordination, means developing an explicit plan that accounts for nodes' actions and interactions in achieving specific goals. Planning in CDPS, as in most of AI, has traditionally been viewed as developing an ordered set of operations to achieve the desired goal(s), assuming that the only changes to the state of the world are due to the planned operations. When multiple agents are involved, the plans usually include synchronization actions between the agents to enforce important orderings between their actions. Multiagent plans can be formed by a single, centralized agent [2, 24], or the planning itself can be distributed among multiple agents [4, 33].

One reason why multiagent planning paradigms have not seen much application to DSN work is that multiagent planning has concentrated on issues of resource conflicts, and particularly on synchronizing the actions of agents to avoid simultaneously attempting to access a non-sharable resource (such as a tool or a location in the workspace). This emphasis on conflict avoidance is not as critical in DSNs, because the separate systems in DSNs can work independently. Instead, DSNs need techniques for actively promoting cooperation rather than for simply preventing conflicts. A second reason that multiagent planning has had little application to DSNs is that multiagent plans are coordinated at very detailed levels which requires very accurate and stable models of the goals and actions in the domain. Because DSN tasks are very dynamic and unpredictable, planning for DSN interpretation tasks cannot assume detailed coordination, but instead must coordinate at a more abstract level in order to maintain flexibility and recover from unexpected events. Thus, although multiagent planning has been used in simplified air-traffic control problems [2], traditional multiagent planning systems are unsuited for the dynamic behavior of DSNs.

2.2 DSN Requirements

To be fully successful in the type of DSN task exemplified by distributed vehicle monitoring, a technique for coordination must incorporate the strengths of all of these CDPS approaches, allowing nodes to:

- dynamically decompose and reassign interpretation tasks and responsibilities to utilize network resources;
- individually and collectively adapt quickly to changing data;
- exploit their most current knowledge in deciding their own roles and the roles of others in group problem solving;
- balance the need to conform as a collective whole with the need to explore locally promising interpretations;
- selectively decide what partial interpretations to communicate and with whom;

- organize their overall problem-solving responsibilities in many alternative ways;
- tolerate inconsistencies in their interpretations and in how they view coordination.

To make these requirements more concrete, consider what they mean in terms of the distributed vehicle monitoring problem shown in Figure **??** (Section 1.1). In this example, the separate nodes should dynamically reassigning tasks and responsibilities so that node-4's processing power is not wasted. The nodes might assign tasks to process particular data (so node-1 might send some of its data to node-4), or they might assign responsibilities, such as making node-4 in charge of integrating the partial interpretations developed at the other three nodes. Node-1 should send a hypothesis about a short partial track to node-2, and we would want node-2 to adapt its interpretation plans quickly to this new data by focusing its attention on processing only its data that could extend the received partial track. Node-2 should have enough individual authority to change its plans without waiting for approval for the change from every other node first, but we want it to alert the other nodes of the unilateral change so that they can adapt their plans as well. Although nodes might have obsolete views of each other, they will eventually exchange updated information and should react to the most up-to-date information by improving their local plans and their coordination decisions. As an example of balancing conformity with autonomy, nodes 1-3 will determine that they should work together on forming the long track, but each should have the flexibility to break this commitment if it later senses a more important vehicle.

The nodes should not exchange every partial interpretation they form: If node-3 forms the track including data d_1 and d_2, it should not send this to another node because no other node can use this information. Node-3 will later extend this track to include d_3, and then d_4, and so on. Node-3 should therefore make a judicious decision about when it has built a sufficiently large partial result to make communication worthwhile [29]. As data arrives and problem solving proceeds, nodes that initially were underutilized might become overly burdened with processing data. For example, if data arrives over time, node-2 is initially idle, but once it begins receiving data the noise in its data overwhelm it quickly. Coordination techniques must allow nodes to modify local plans and propose individual and organizational responsibilities based on such changes. Finally, the coordination decisions and the underlying problem-solving mechanisms must be sufficiently robust to tolerate incorrect or out-of-date information, such as when node-2 uses the partial track from node-1 to focus its resources on processing compatible data first before considering unlikely alternative interpretations based on the noisy data.

Thus, distributed interpretation, as embodied in DSN applications, demands task-sharing and result-sharing, organization and planning. We have developed partial global planning as a unified framework in which we integrate all of these different approaches, and where the strengths of different approaches can dominate under appropriate conditions.

3 Partial Global Planning

Partial global planning brings together a variety of previously distinct coordination approaches by proposing a unifying perspective. This perspective starts with a planning view, where coordination is a matter of explicitly planning cooperative interactions. However, unlike traditional plans that rigidly dictate specific actions at specific times, our plans are more fluid and adaptable to changing information and circumstances. Plans, in our view, detail a node's problem-solving strategy and its expectations but, although a node attempts to follow this strategy closely as long as it is appropriate, the node also has the ability to change strategies as problem solving progresses. While carrying out a plan, moreover, a node can flexibly elaborate details of the plan to meet the needs of the current situation.

Coordination thus entails sharing enough tentative plan information so that at least one node can establish a sufficiently global view to recognize how changes to local plans could improve coordination among nodes—changes such as having node-1 quickly form and share predictive information with node-2 in Figure **??**. Note that any number of nodes could potentially collect plan information from others; the decision as to which node or nodes should coordinate plans depends on domain requirements and constraints. Also, a node does not need a completely global view in order to improve coordination. A node only needs to know about the part of the network that could affect it. As a node collects plan information from other nodes, it combines its *partial* knowledge about the more *global* situation into partial global plans (PGPs) that represent the collective activities of the nodes. The node then can propose changes to the PGPs (and in turn to its own local plans or to the local plans of other nodes) to improve group problem solving.

The partial global planning framework integrates organizational concepts by introducing two types of organizations. One organization specifies the long-term problem-solving roles and responsibilities of nodes. A node uses this organizational knowledge to constrain its search for appropriate tentative plans, and to guide its expectations of other nodes' plans. The second organization, called the *meta-level organization*, gives nodes a framework for deciding how to solve coordination problems. Therefore, two problems are being solved simultaneously in this type of network: the task-level problem (building a map of vehicle movements) and the meta-level problem (deciding how to coordinate problem solving in order to effectively solve the task-level problem).

By combining planning and organizational concepts, our partial global planning framework enables nodes that are working on potentially related pieces of the same problem to exchange information in an organized way in order to plan joint activities to solve that problem. Result-sharing is thus much more coordinated, leading nodes to exchange appropriate results at the right time without unduly wasting network communication and computation resources. What is less obvious is that partial global planning is also a powerful framework for task-sharing. To see this, first consider that an agreement over the exchange of tasks—a contract—is essentially a shared plan of action: The manager plans to send the task to the contractor, the contractor plans to then perform the task and return the result, and the manager plans to collect the result and use it in some way. However, consider how inflexible a contract looks when viewed as a plan, providing no room for concurrent activity or counterproposing.

In partial global planning, task-sharing is coordinated by allowing nodes to propose (and counterpropose) potential plans that involve the transfer of tasks among themselves. For example, if a node wants help in processing half of its data, it builds a potential plan indicating that, while it works on half of the data, some other (unspecified) node is working on the other half. It can pass this proposal to every other node or only to nodes that it believes are underutilized. A recipient node can accept the proposal by substituting its name in for the unspecified node, or it can counterpropose by both substituting its name and by modifying the plan (such that it only accepts a third of the data instead, for example). Not only does the plan explicitly represent concurrency among local and transferred tasks, but it also allows more flexibility in contracting because it permits counterproposing.

With this high-level view of how partial global planning brings together different approaches to coordination into a single framework, we now go into more detail about the conceptual basis and the algorithms of partial global planning. The development begins by looking at how an individual node models itself by building tentative plans, and then addresses how nodes use organizations and communication to exchange coordination information. We then examine how nodes model group activity by integrating local plans into partial global plans, followed by how nodes improve coordination by searching for useful modifications to their partial global plans. Specifics about an implementation of partial global planning for the distributed vehicle monitoring task are given in Section 4.

3.1 Self-Modeling Using Plans

The better a node can predict its future actions, the better it will be at coordinating those actions with the actions of others. In a vehicle monitoring task, for example, nodes that can predict when they will form their pieces of an overall interpretation can decide how to exchange results as efficiently as possible. Unfortunately, most domains are uncertain and dynamic, making individual and collective planning more difficult. Vehicle monitoring in a DSN is an example of a very uncertain and dynamic domain, where a node's data set can change unexpectedly because new vehicles (or noise) can be sensed. Even when the data set remains unchanged, moreover, the node could alter how it expects to process the data due to unanticipated partial interpretations it forms.

Our approach to these problems is to have a node use approximate knowledge to quickly form very rough characterizations about all of the possible interpretations of its data. These characterizations represent possible problem-solving goals, and the node builds tentative plans, each of which attempts to achieve (or refute) one or more of these goals. Because several goals might call for processing some common subset of data, one plan might initially be formed to pursue those goals, with the expectation that the plan might later be split into separate plans when the common data has been processed. Hence, although planning relys on having coarse approximations of potential goals, the perceived goals can change over time and planning will adapt. We describe techniques for economically forming and using approximations elsewhere [9, 30].

A plan represents future activity at two levels of detail. At a high level, it outlines the major steps it expects to take in achieving the goal(s). This high-level plan represents a long-term problem-solving strategy that not only guides detailed planning decisions, but also is the view that can be shared with other nodes to give them a clear indication of what problem-solving activities the node will be engaged in. A plan also contains details of specific primitive actions to achieve the next major plan step. As a plan is pursued, new details are added incrementally [12]. Details can also be added in reaction to a changing situation without affecting the major plan steps. This gives a node the ability to continue predictably following its long-term strategy (and thus to behave as other nodes expect it) while still retaining the flexibility to react to minor unexpected contingencies. When the situation changes radically, such as when very different data arrives or problem-solving actions create partial results that deviate wildly from expectations, the long-term strategy can change, and entire plans can change by having separate plans merge together or by having a single plan split into plans for pursuing different subsets of the original plan's goals. At any given time, the plans are ranked and stored on an agenda.

We can specify a plan as a tuple of the form:

$$\langle\ n\ t_{create}\ G\ S_{lt}\ D_{st}\ P_t\ P_c\ r\ \rangle$$

Each plan has its own unique **name** (n) and has a record of its **creation time** (t_{create}). The plan's **goals** (G) are a set of objectives ($\{g_1, \ldots, g_n\}$) to achieve, or to prove are not worth attempting to achieve. In interpretation tasks, the goals correspond to rough characterizations of the data that indicate potential interpretations.

The representation of the **long-term-strategy** (S_{lt}) is domain-dependent, but for interpretation tasks has the general form of a ordered list of **planned-actions** for data processing. A planned-action a_i represents a major step in the plan, and has the form $\langle\ D_i\ P_i\ t_{est-start}\ t_{est-end}\ abres_{est-partial}\ \rangle$ indicating the set of data D_i to be processed, the procedures P_i to be applied to the data, the estimated start ($t_{est-start}$) and end ($t_{est-end}$) times of the major step, and an estimate of the characteristics of, and confidence in, the eventual abstract partial result ($abres_{est-partial}$) that will be developed at the conclusion of the major plan step. The estimates are derived from the plan's predictions P_t and P_c (defined below). The order of the actions is computed by an algorithm that uses three general-purpose heuristic computations: (1) to prefer actions that concurrently achieve multiple goals; (2) to prefer actions expected to require less resources (especially time); and (3) to prefer actions that will strongly verify or refute that some goals are worth pursuing. Each of these heuristics returns a numeric rating for each of the actions, and these are weighted and summed to rate the entire action. The long-term-strategy is thus the list $\{a_1\ a_2\ \ldots\ a_n\}$ such that (for all j between 1 and $j-1$)

$$\sum_{i=1}^{3} w_i h_i(a_j) > \sum_{i=1}^{3} w_i h_i(a_{j+1}).$$

The **short-term-details** (D_{st}) corresponds to a set of primitive problem-solving operations, where new operations are added to this list incrementally. Again, the specification of an operation is implementation dependent. For interpretation tasks, an operation indicates the specific data objects to be processed, the exact operator to be applied to the data, and precisely the expected results of the processing. Results of one operation are typically used as data for subsequent operations, assuming that data is processed in stages.

The plan's **time-predictions** (P_t) and **confidence-predictions** (P_c) contain estimates of how long each major step of the long-term-strategy will take and the expected outcome of each step. For interpretation tasks, this corresponds to expectations about when successive abstract partial interpretations will be formed and how strongly believed those interpretations are expected to be. Estimates are based on default knowledge that is updated during problem solving so that past experiences can affect predictions. A plan's **rating** (r) reflects the importance of pursuing the plan. Several factors go into the calculation of rating, including the confidence-predictions (prefer plans that are expected to form higher confidence solutions) and the time-predictions (prefer plans that will form solutions sooner). By pursuing highly-rated plans, a node attempts to form promising solutions in a timely manner.

As a plan is executed, its slots are updated. New short-term-details are added when needed, experiences can affect the time- and quality-predictions, partial results of the plan can lead to changes in long-term-strategy or in goals, and ratings can change as a consequence of partial results, arrival of new data, or costs of past processing.

3.2 Organizations and Communication

We assume that nodes are initialized with commonly-known organizational information about roles, interests, and responsibilities. For example, when sensors in a DSN are statically arranged, then a node knows not only the region that it is responsible for and what types of problems it can solve, but also which nodes are responsible for neighboring regions and what their capabilities are.

But just knowing the *possible* roles and responsibilities of other nodes does not guarantee coordinated problem solving. For example, when a node recognizes that a vehicle it is tracking has entered the sensor region of a neighbor, should it surmise that the neighbor is now tracking the vehicle? The neighbor might be, but it might also be performing tasks that it considers more important instead, such as tracking vehicles that the first node has no knowledge of, or integrating important partial tracks from another group of nodes. Thus, while an organization improves coordination by providing guidelines for how nodes might cooperate, it also gives nodes enough flexibility to potentially interact in uncoordinated ways. Using organizations that restrict nodes to very narrow ranges of tasks would remove this flexibility, but introduces different inefficiencies as very specialized nodes sit idle whenever their particular capabilities are not needed. We thus can conclude that relatively static organizations should be flexible to allow nodes to undertake whatever tasks are currently pending, but nodes should be able to dynamically update each other regarding which of their possible roles they are currently filling [17].

A node's plans embody the situation-specific expectations of activity that nodes need to refine their organizational knowledge dynamically. In our partial global planning framework, therefore, nodes communicate selected plan information in order to model near-term node activities within the more general, long-term organizational structure.

But while this approach seems sensible, it brings up new coordination questions such as what plan information should be sent, when, and to whom? While blindly sharing all plan information with all nodes whenever plans are updated ensures nodes of having models of each other, this essentially result-sharing strategy is very inefficient. Nodes should selectively encode and exchange plan information in a structured manner to maximize coordination with minimal additional communication and computation overhead. When viewed this way, deciding how to coordinate is itself a distributed problem that nodes must solve, and we again turn to using organizational information to guide group problem solving.

The nodes' **meta-level organization** specifies, for each node:

1. nodes it has authority over – it receives plan information from these nodes, identifies

how related plans should change to improve coordination, and sends modified plan information back;

2. nodes that have authority over it – it sends plan information to these nodes, receives modified plan information back, and can adopt the modified plans;

3. nodes that have equal authority – it sends plan information to and receives plan information from these nodes, and each node can use this plan information however it wants to locally.

The meta-level organization can indicate centralized coordination (where one node has authority over all others), hierarchical coordination (where some number of middle management levels lie between the top node and the bottom nodes), and lateral coordination (where all nodes have equal authority). Typically, a node is only informed about other nodes with equal authority if the node and the other nodes have no common "supervisor" at a higher level. That is, nodes that cannot communicate vertically through the authority structure can communicate horizontally. This allows nodes to be organized laterally in a single level or oligarchically (where the top nodes of several separate hierarchies coordinate laterally). The organizational structure also allows several nodes to have authority over the same node, in which case the node uses criteria such as plan ratings or recency to decide between conflicting plan modifications from other nodes.

While the task-level and meta-level organizations can have many similarities, it is also possible for them to be very different, such as when the meta-level organization designates that one node should act as a central coordinator (building partial global plans and telling other nodes how to change their local plans), but the task-level organization does not concentrate the collection and integration of partial tracks on any single node (see Section 4.2.2).

3.3 Group-Modeling as Plan Integration

The meta-level organization specifies to whom a node should send its plan information, but we must also indicate *what* information to send. The information that nodes should exchange should be geared towards coordinating their group activities without getting lost in the details of what each other is doing. In partial global planning, therefore, the information nodes exchange indicates the goals, long-term-strategy, and rating of a plan.

Whenever a node receives plan information from another node, it first attempts to relate the plan's goals with goals of other plans it knows about. Goals can be related in various ways [8], often based on characteristics of the domain. In vehicle monitoring, a typical relationship between goals to generate partial tracks is that they could be part of some larger goal to generate a complete track. Using goal-relationship knowledge, the partial global planning mechanisms group plans together whenever the plans are potentially pursuing some common larger goal. In effect, this explicitly links together plans for generating mutually constraining and corroborating results. The mechanisms then build a PGP to represent the group goal and the planned activities for achieving the goal. A PGP can be represented as a tuple of the form:

$$\langle\ n\ t_{create}\ P_{component}\ G\ S_{lt}\ I_{lt}\ r\ \rangle$$

When the planning mechanisms create a PGP, they give it a unique name (n), and store its creation-time (t_{create}), a set of pointers ($P_{component}$) to its component plans ($\{p_i \ldots p_j\}$), and the larger, more encompassing goals (G) it was created to pursue. Initially, the long-term-strategy (S_{lt}) is simply the union of the planned-actions of the separate plans, sorted by the estimated end-times of the actions. This effectively represents the interleaved activities of the participating nodes.

To determine *when* to send partial results, the partial global planning mechanisms can analyze the long-term-strategy to find the estimated times at which partial results are expected to have been formed, and then can determine which partial results should be transmitted to which nodes at approximately what times. Decisions about which partial results to transmit are based on the conflicting desires of trying to send predictive results in a timely manner (which leads to early transmissions) and trying to send few, more complete, results (which leads to delaying transmissions) [18]. The partial global planning mechanisms explicitly consider both of these desires as they search through the sequence of planned-actions to identify predictive results and to find the portions of the overall result that each participating node should form so that the complete result can be constructed most quickly.

Having found the set of partial results that should be integrated, the mechanisms use statistics about communication delays between nodes to plan out the exchange and integration of results to form a complete solution. This planned set of long-term interactions (I_{lt}) between the nodes is represented in the PGP. The algorithm for computing the interactions, shown in Figure **??**, essentially constructs a binary tree starting with the leaves, and returns the root representing the integrated solution. For example, in one of the experimental runs (Experiment E1.2) described later, which is based on the scenario of Figure **??**, the interaction strategy that the algorithm generates is represented in Figure **??**.

** **Figure ?? about here** **

** **Figure ?? about here** **

As a node follows its local plan and generates partial results, it checks the planned interactions of the associated PGP to determine whether the partial result should be transmitted, and if so to what node.

3.4 Coordination as Search Through Alternative PGPs

When initialized, a PGP represents the currently predicted activities of a group of nodes based on their initial plans. However, it is possible that the nodes could pursue the PGP's goal more efficiently if one or more nodes change their plans. The partial global planning mechanisms attempt to modify PGPs to improve coordination by searching through a portion of the space of alternatives to identify better ones.

3.4.1 Coordination Through Task Reordering

One algorithm involves searching through alternative orderings of the planned-actions to reduce the time or communication needs of forming the complete result. This amounts to improving how each node focuses its resources into forming and sharing results. The mechanisms rate each action in the sequence based on several factors, including whether the action extends a partial result (vehicle track hypothesis) using data not yet processed by any other node, whether the action produces a partial result that might help some other node in forming its partial results, and how long the action is expected to take. The mechanisms then calculate the rating of the ordering as the sum of the ratings of each individual action. A hill-climbing algorithm, shown in Figure **??**, explores alternative orderings.

** **Figure ?? about here** **

Because an action's rating depends on what actions precede and succeed it, swapping actions using this algorithm will affect the ratings of those actions, and thus can increase (or decrease) the individual and overall ratings. This algorithm will thus find an improved ordering (based on the rating factors), but is not guaranteed to find an optimal ordering because it might arrive at a local maximum. Because PGPs are formed in dynamic situations, however, investing the much greater time to optimize the ordering is not justified, since the optimized ordering can quickly become obsolete.

3.4.2 Coordination Through Task Reallocation

The second important algorithm for improving coordination involves task-decomposition and task-sharing. When nodes exchange information about their planned activities, a node with no planned activities sends an empty plan, indicating that it is idle. The partial global planning mechanisms search for possible decompositions and allocations of tasks to make better use of idle resources (Figure **??**). Similarly, when an idle node receives a proposal in the form of a PGP, it updates the PGP based on its local knowledge about its capabilities, responsibilities, and commitments. The updated PGP represents a counterproposal that it sends back. Note that, in determining its commitments, a recipient node can use whatever information it has to predict future commitments of its resources (such as when it extrapolates a received partial track to determine that a vehicle might be coming its way), and these will factor into its counterproposal. The experiments we describe in Section 4.2 do not focus on issues in task relocation, but we have detailed them elsewhere [16].

**** Figure ?? about here ****

3.4.3 Autonomy and Conformity

When a node receives a PGP from a node that has authority over it, it can adopt that PGP and modify its local plans accordingly in order to conform to the coordination decisions of the higher authority node. If the higher authority node has correctly discerned the situation and made appropriate coordination decisions, then conforming is the proper response of a node. Unfortunately, however, a higher authority node might have incomplete or out-of-date information, so by conforming a node could doom itself to acting ineffectively.

In the partial global planning framework, the authority of a node is represented as a weighting factor. Thus, when a node receives a PGP from another node and must decide whether or not to conform, it multiplies the rating of the received PGP with the weighting factor, and conforms if that product is greater than the rating of its current local PGPs. This gives nodes the autonomy to act on highly-rated local plans instead of blindly following the possibly outdated and incompletely informed commands of another node.

3.5 Execution, Monitoring, and Recovery

Partial global planning is an ongoing process throughout the course of problem solving. Although our descriptions have been simplified by describing the process as if partial global planning occurs in discrete stages (local planning, communication, initializing PGPs, modifying PGPs, etc.), the mechanisms are actually geared for much more dynamic, asynchronous, distributed systems. A node uses whatever information it has about its local plans and PGPs to decide what problem-solving and communication actions to take at any given time. As time passes, it might receive more data from its sensors, partial results from other nodes, or PGP information from others, all of which are integrated into its current view of problem solving and coordination, and any of which might cause it to change its own plans. Unlike traditional planning approaches where plans are completely laid out before any action begins, partial global planning assumes that changes in plans are inevitable. Planning in the current situation should not incur excessive (mostly unnecessary) overhead, plans should be adaptable, and plans should be monitored and updated as circumstances change.

As plans are executed, they might form unexpected results, fail to form desired results, or take longer (or shorter) than anticipated to form expected results. Because nodes coordinate at the level of major plan steps rather than at individual operations, adapting to minor deviations in plans can be restricted to local modifications such as adding actions to form a desired result in a different way or deleting actions when the result they were intended to form has been formed serendipitously in some other way [14].

However, these local deviations can impact coordination when a node can no longer form and transmit an anticipated partial result at the expected time. It is tempting to insist that such deviations be propagated to PGPs, which are then transmitted to appropriate other nodes, possibly leading to revisions in how nodes should coordinate their group activities. The trouble with this attitude is that propagating changes internally and externally involves a commitment of computation and communication resources that might outweigh any benefits of better coordination. Sometimes it is better to accept minor inefficiencies in coordination rather than incurring the major overhead of resolving those inefficiencies, particularly when the change triggered by one node causes other nodes to change their plans in a (possibly cyclic) chain reaction.

Partial global planning provides two techniques for striking a balance between responding to important deviations and predictably following old PGPs when deviations are minor. One technique is to define a threshold value for how much a plan can deviate (in terms of when a partial result will be formed) before deviations should be propagated and PGPs should be revised. This technique leads to nodes potentially having very obsolete models of each other, and in fact, a node could have two very different representations of its own plan—a representation of the modified plan that it is actually following and a representation of the original plan that other nodes are still trying to coordinate with.

The second technique for balancing responsiveness and predictability is to build more robust PGPs, where a certain amount of time "cushion" is added to plan steps to increase the chances that planned deadlines will not be exceeded. This technique reduces the number of times plans deviate from expectations, and thus the overhead spent in checking whether a deviation is significant. However, this technique also makes the interactions between nodes less crisp, so more robust plans are often also less efficient.

4 Prototype Implementation and Evaluation

The partial global planning framework has been implemented and evaluated for coordinating multiple AI (blackboard) systems in a simulated distributed sensor network task.

4.1 Implementation

Our prototype has been implemented in Lisp and studied using a simulation testbed that models a distributed vehicle monitoring task.

The Distributed Vehicle Monitoring Testbed. The distributed vehicle monitoring testbed (DVMT) is a flexible, instrumented research tool for studying cooperative distributed problem solving [28]. The DVMT simulates a distributed sensor network, where each sensor detects acoustic signals and sends signal information to one or more problem-solving systems for interpretation. As vehicles move among the sensors, information about a signal's approximate location, frequency class, and strength is supplied at discrete times to the corresponding interpretation nodes.

Local Hypothesis Formation. A node is an AI system based on the blackboard architecture originally built for speech signal interpretation [21, 31]. In the blackboard paradigm, a number of processing elements, called knowledge sources (KSs), communicate through a shared data structure (the blackboard) to incrementally construct interpretations of data. When initial data appears on the blackboard, KSs that can process that data build intermediate interpretations of it (such as grouping related signals together) and post these interpretations on the blackboard, which are then processed by other KSs (that might match signal-groups to vehicle types, or string together sequences of partial interpretations into vehicle tracks) until overall interpretations are generated.

In practice, most blackboard systems are implemented on serial machines, so KSs cannot act in parallel. Instead, each KS is given a chance to inspect the blackboard, and then the KS rates how important it is that it be given a chance to act. Possible KS executions are stored on an agenda based on their ratings, and the most highly rated is allowed to execute. It in turn generates new blackboard entries, which trigger additional possible KS executions which are added to the agenda, and the process repeats.

Besides modularity in breaking the interpretation process into several nearly independent KSs, the blackboard architecture also has the advantage that it opportunistically explores multiple potential solutions, as KSs are applied to the most important (highest confidence) entries on the blackboard at any given time. However, this flexibility can make a blackboard system's behavior appear highly erratic, as it executes different KSs and jumps between alternative interpretations.

To enable a blackboard system to behave more predictably while still retaining some opportunistic capabilities, we implemented local planning mechanisms based on the concepts in Section 3.1. The details of these mechanisms are given elsewhere [14], but the upshot is that the blackboard system can plan and represent its near-future problem-solving activities, both at a long-term strategy level, and at a detailed level.

Distributed Hypothesis Formation. The meta-level organization, plan integration, and plan modification mechanisms have been implemented in the DVMT nodes, using information about the domain and about the time needs and capabilities of the blackboard system in order to build rating functions for PGPs and individual planned actions. While tailoring the PGP mechanisms to this application requires examining many domain-level details, the concepts and algorithms outlined in Section 3.4 form the core of the implementation. Details on the implementation are given elsewhere [10].

One of many possible distributed hypothesis formation strategies is implicit in our implementation of the knowledge-based heuristics that guide local and partial global planning. Specifically, the strategy that we focused on was to form complete hypotheses as quickly as possible by having nodes coordinate such that each has its own unique areas of responsibility. But other strategies might be equally valid, such as strategies where nodes coordinate such that they first concurrently process data in overlapping areas to corroborate their results, and then extend these results into non-overlapping areas. Such a strategy emphasizes taking more time in order to build very high-confidence solutions, whereas the strategy we implemented was to quickly build solutions with adequate confidence. We can move between different strategies by changing heuristics such as heuristics for rating action orderings.

4.2 Experiments

Using the DVMT, we have experimented extensively with our implementation of partial global planning in order to evaluate both its versatility (for coordinating in many different ways) and its practicality (for coordinating without requiring more overhead than it saves). In this section, we first investigate whether partial global planning indeed allows nodes to coordinate in the variety of ways needed, and to do this we go beyond our previous descriptions [13] by providing a detailed look at the concurrent behaviors of nodes. We then explore how alternative organizations perform as the problems are scaled up, leading to important observations that have motivated much of our ongoing research.

4.2.1 Flexible Coordination and Meta-Level Organizations.

Recall that the vehicle monitoring problem introduced back in Figure **??** places a variety of demands on how nodes coordinate. Node-1 should ignore the strongly-sensed but noisy data in the upper-left corner in favor of cooperating with nodes 2 and 3. Moreover, node-1 should build its portion of the track so as to send a partial track to node-2 early on to help node-2 disambiguate its data. Node-2 should take advantage of this data. Finally, node-4 is an available resource that the other nodes could take advantage of.

We measure the quality and costs of coordination along four dimensions. One dimension is the simulated solution time of the problem-solving network, where a KS requires one simulated time unit to execute, and where communication between nodes takes two simulated time units. Nodes execute their KSs concurrently, so if the simulated solution time is t, it means that each node executed at most t KSs. A lower simulated solution time for solving the same problem means that nodes made better decisions as to which KSs to execute and that they distributed the load better to enable more parallelism.

The other dimensions measure the overhead of partial global planning to determine whether its benefits outweight its costs. We measure the actual runtime of our simulation to discover whether the amount of computation needed by our implementation of partial global planning is less than the computation it saves by reducing the number of KSs executed. We measure the number of messages exchanged to determine how much additional communication (of plan information) is required to improve coordination. Finally, we measure the memory requirements, considering that nodes using partial global planning must store PGP information but, if they execute fewer KSs, must store less information on the blackboard.

The experimental results are summarized in Table **??**. The first set of experiments compares the network performance of nodes that only plan locally and never exchange plan information (E1.1) with nodes that can perform partial global planning using different meta-level organizations including lateral (E1.2), central (E1.3), and ring (E1.4). E1.1 represents a baseline of performance, in which a node works independently and broadcasts a partial solution it forms only when it cannot locally improve on the local solution. As a result, nodes duplicate effort in overlapping areas, communication is unfocused, and information that can guide nodes into forming compatible results is not exchanged in a timely manner.

**** Table ?? about here ****

Providing nodes with a meta-level organization to enable them to exchange PGPs allows them to coordinate their activities much more effectively to reduce the simulated solution time. Among the improvements to coordination are: node-1 can ignore the noisy data and send a short partial track (covering sensed times 8 and 9) to node-2 very quickly; node-2 uses the partial track to disambiguate its information, and nodes 1-3 can exploit node-4's resources by sending it their partial results for integration. The extent to which these improvements are achieved, and the overhead of achieving them, depend on the particular meta-level organization. In brief, a lateral organization excels in terms of minimizing the simulated run time with the central organization slightly behind due to the extra delays in waiting for PGP information from the central coordinator (node-4, because it is otherwise least busy). The ring organization, in which the four nodes pass a single collection of PGPs from one to the next as if around a ring, is very much inferior along this metric because the extra delays in propagating information around the ring means that nodes are more often basing their decisions on outdated information. In actual runtime, the centralized organization is best because only one node (node-4) is incurring the overhead of integrating plans and searching for better PGPs, while in the other organizations each node performs these tasks. Because it also runs many more KSs, the ring organization requires more time to run than the lateral organization. The ring organization requires the least communication, however, because PGPs are batched together and circulate in a predictable fashion. The lateral organization requires the most communication because each node exchanges plan information with every other node. Finally, the storage requirements for the lateral and centralized organizations are comparable, while the ring organization uses more storage because it executes more KSs and builds more partial interpretations.

To investigate the impact of partial global planning on the actions and interactions of nodes more fully, consider the concurrent processing and communication activities of nodes for experiments E1.1-E1.3 (Figures **??**, **??**, and **??**). These figures indicate the activities of each node over sequential time intervals, represented by the data involved in the hypotheses each node is forming. Because several knowledge sources must act on data to process it into a high-level interpretation, the reader will observe that nodes must process the same data over a series of time intervals before it is ready to be integrated into an extended track. Before investigating the details, moreover, we must remind the reader that, even with partial global planning, we cannot expect optimal coordination because: nodes find improved collective activities using a satisficing, hill-climbing algorithm; nodes only exchange tasks when one of them is a severe bottleneck relative to the others; nodes cannot always predict the outcomes of their activities accurately; nodes will withhold information about minor deviations from their plans; and communication delays will enable nodes to enter chain reactions of PGP changes. The behaviors of the nodes should not be viewed as optimal given our complete view of the problem, but instead should be considered within the context of the limitations of computation and communication in which they work.

First, consider the baseline case where nodes work independently on their data and exchange partial results only when those results cannot be extended locally (Figure **??**). After receiving their sensor data at sequential times from 1-16 the nodes begin problem solving. Node-1 works on the track in its upper left corner first, which it completes at the time interval beginning at time 34. It sends this information to the other nodes, and locally takes the final step of posting this track as a solution (time 35). Meanwhile, node-2's noisy data supports seven different vehicle types equally well, so it is forming all alternative hypotheses for d_{10}. Node-3 is forming its part of the overall track, and node-4 is idle. When node-3 completes its partial track ($d_1 - d_6$), it sends the information to the other nodes, and node-1 aborts its redundant activities in the overlapping areas and focuses on extending the received track. Nodes 2 and 4 post the received track as a possible solution (time 42). Node-3 goes on to hypothesize a track for a weakly supported vehicle type, while node-1 eventually forms and transmits the track $d_1 - d_{12}$. When node-3 receives this, it focuses on extending the track into $d_{13} - d_{15}$, building the overall track at time 79 and posting it as (the correct) solution over time interval 80-81.

**** Figure ?? about here ****

Inspecting the behaviors of nodes in E1.2 (Figure **??**), we note that initially (time 16) the nodes pursue their best local plans, but at time 18 they receive PGP information from each other, and each fits these pieces together to identify how it should change its plans. Node-1 starts to work on $d_8 - d_9$ (more globally relevant that $d_1' - d_5'$), and node-2 focuses on data (d_{13}) beyond the overlapping area to avoid redundancy. Node-1's predictive result $d_8 - d_9$ is received by node-2 at time 27, allowing node-2 to develop a separate plan for processing only a subset of the data in $d_{10} - d_{15}$ that could be compatible with the received partial result. Node-2 has already processed that data for d_{13}, so it works on $d_{14} - d_{15}$ (avoiding the overlapping area until later). Meanwhile, nodes 1 and 3 continue forming partial tracks. Because of inaccurate predictions, node 2 believes nodes 1 and 3 are farther along than they really are, and sends $d_{13} - d_{15}$ off to node-4 at time 35, because node-4 was initially slated to do the integration. When node-3 completes its plan, and when node-1 modifies its plan due to received information, the PGPs at the various nodes change asynchronously. While this impairs coordination, the nodes still eventually converge and node-1 builds the overall solution.

**** Figure ?? about here ****

The nodes in experiment E1.3 (Figure **??**) get a later start at being coordinated, because node-4 receives their individual PGPs at time 18 and sends back coordinated PGPs that do not arrive until time 20. Node-4 assigns the overall integration task to itself because of its available resources. Unlike in the lateral organization where asynchronous PGP changes led to changing (and at times inconsistent) views about which of the nodes should do the integration tasks, the central organization enforces consistent views among the subordinate nodes (1-3) because all changes to the PGPs are made and broadcast by node-4. Thus, even though it underestimated the time at which node-1 would receive $d_1 - d_6$ from node-3, node-4 still accumulates the relevant partial results and integrates them together.

**** Figure ?? about here ****

4.2.2 Scale-Up Effects in Larger Networks.

When scaling up to larger problems, a practical concern for experimentation is the overwhelming complexity of analyzing the concurrent behaviors of a large number of very different nodes to determine the quality of coordination. To simplify the experimental analysis, one useful tool is to generate problem-solving situations involving symmetries, so that nodes can be divided into equivalence classes, and we can investigate coordination based on these classes rather than on the individuals. This desire motivates the environment used for our larger experiments, depicted in Figure **??**, which involves 10 overlapping sensors, arranged diagonally. Two vehicles move in parallel among the nodes, but while the upper vehicle consistently generates moderately sensed data, the lower vehicle track alternates between strongly and weakly sensed sections. The overall confidence in the moderately sensed track is greater than that for the other track; an effective problem solving network should derive both solutions, deriving the better track first.

**** Figure ?? about here ****

For brevity, we will not discuss the details of the nodes' behavior here, but instead we will concentrate on how well partial global planning scales up in these examples depending on the meta-level organization employed. The task-level organization used in these experiments is a lateral organization (any node can integrate partial solutions) and can be varied independently of the meta-level organization (Section 3.2).

The results are summarized in Table **??**. As in the previous experiments, we use the performance of the nodes without any partial global planning (where nodes only plan locally) as a baseline (E2.1), and contrast this performance to the performance of the lateral (E2.2) and central (E2.3) organizations. Experiments E2.2 and E2.3 reconfirm that partial global planning improves network problem-solving performance, again at the cost of increased communication because of the need to exchange PGP information. In contrast to the experiments discussed before (Table **??**), however, in the larger network a centralized organization was better in all ways compared to a lateral organization. As expected, concentrating the communication and coordination tasks at one node means less overall network communication and computation, as shown by the **Comm** and **Rtime** data. Because overall network communication in a lateral organization increases quadratically with the number of nodes while communication in a central organization grows linearly, the difference in communication overhead between these becomes more pronounced in larger networks. Similarly, in a lateral organization all nodes combine and coordinate PGPs, leading to inefficiencies due to redundant work. Again, these computational inefficiencies increase as the number of nodes increases. Finally, in this experimental scenario, the more rapid response time possible with a lateral organization is offset by degraded coordination because of the inconsistent information caused by communication delays. The more consistent views enforced by a central organization lead to better simulated solution times.

**** Table ?? about here ****

However, even in the central organization, the scale-up effects are daunting because one node has to integrate and coordinate the PGPs of all ten nodes. The combinatorics become substantial, and will only get worse with even larger networks. To partially address this problem, we used a standard organizational approach practiced by people: we employed a hierarchical organization. In this experiment (E2.4), the ten nodes are split into two groups of 5 (nodes 1-5 and 6-10), and each group has a central coordinator (nodes 1 and 6). These 2 central coordinators collect and coordinate the PGPs of their 5 nodes, and then pass these coordinated PGPs on up to a single top-level coordinator (node 3), who then resolves any coordination difficulties between the two groups. When investigating the performance of this organization (Table **??**), we note that the simulated performance is comparable despite the additional layer of command (and the delay it implies), and so is the amount of communication (although greater than the centralized case). The real benefit of this organization is in reducing the actual computational resources consumed by the network. Because each of the middle-level coordinators must integrate and coordinate only 5 nodes' PGPs, they incur less combinatoric overhead than a single coordinator for 10 nodes does (E2.3). Furthermore, while the top-level coordinator must integrate the PGPs for the groups together (and recall these PGPs indicate the activities of every node in each group), the top-level manager's task of coordinating the nodes is simplified because middle-level coordinators have already resolved coordination problems within each group.

Experiment E2.4 thus illustrates the power of hierarchical organizations in decomposing complex coordination tasks into more manageable and tractable chunks. However, the hierarchical organizations that people use go beyond the "preprocess and pass up" mechanisms that partial global planning does. That is, our partial global planning mechanisms currently allow middle managers to coordinate subgroups, but then all the details of the subgroups are passed up the hierarchy. In human hierarchical organizations, the manager of a group seldom gives full details to a superior, and in fact a manager's job revolves around his or her ability to summarize the important aspects of a group's behavior for a superior, as well as to take abstract coordination guidelines from a superior and translate them into more detailed instructions for subordinates. This capability is an important direction of our ongoing research [20].

5 Conclusions

Partial global planning combines abilities for task-sharing, result-sharing, planning, and organizational structuring into a single unified framework that is both powerful and practical. Conceptually, partial global planning highlights how an intelligent system must intertwine modeling other agents and anticipating what they will or should do with making local decisions about what to do next. Our experiences have shown that coordination is not a separate phase in group activity—it is not a kind of post-processing on local decisions—but instead is an integral part of decision making. As a result, partial global planning represents a new perspective on coordination: rather than the traditional view of providing some protocol or language between systems that enables them to coordinate, partial global planning emphasizes that coordination arises out of local reasoning. This view of coordination as something that emerges out of sophisticated local control decisions rather than as something imposed on individuals by some externally defined protocol or set of rules can lead to important new directions and insights.

The assumptions underlying partial global planning are met in many types of DSN tasks. These assumptions include: the ability to roughly characterize (or cluster) tasks or data to identify potential processing goals; the ability to estimate the time needed for achieving goals based on having performed similar tasks in the past; the ability to efficiently represent and communicate potential goals and their time needs; and the ability to reorder goals to improve efficiency. DSN tasks such as vehicle monitoring meet these assumptions because the repetitive nature of the application domain (similarities in how data at subsequent time frames are processed) facilitates generalization and summarization of possible goals and their time needs, and the order in which data is integrated will generally affect only the timeliness of a solution rather than its correctness. At the same time, the needs of this type of DSN (as described in Section 2.2) are well matched to the strengths of partial global planning.

One direction that we are pursuing is in generalizing partial global planning to enable coordination in pursuing a wider range of goals and more varied relationships between the goals of different agents, including competitive goals among heterogeneous agents [8]. Moreover, because the timing of interactions is critical to effective coordination, we have been investigating the use of approximate processing techniques to enable agents to meet their time constraints [9, 30], and for treating time constraints as being socially imposed and, hence, negotiable [11]. We have also been extending the representation to allow intelligent agents to communicate and reason about not only their plans and goals, but also about their temporal and spatial relationships, their memberships in temporary or permanent teams, and their long-term motivations that led them to adopt their current goals and plans [20]. This work builds on partial global planning to more completely combine theories from organizational science and operations research with AI concepts.

Finally, we should emphasize that our partial global planning framework is uniquely suited to coordinating problem solvers engaged in cognitive tasks such as distributed interpretation and hypothesis formation, because of the way it interleaves coordination, planning, and execution. Although we can impose organizational constraints such that nodes do not take any problem-solving actions until a fully coordinated PGP is worked out, this is seldom done because we are working under the assumption that the problem situation can change dynamically and unexpectedly, so that reacting to new events and recovering from incorrect decisions is a fundamental part of coordination. While this is fine for cognitive tasks where a system can pursue an alternative solution path simply by working in a different part of its solution space, it might be less effective in physical domains where recovering from an incorrect decision might involve undoing several actions, some of which might in fact be irreversible. To understand how partial global planning needs to be extended for coordination in such applications, we are beginning to explore coordination issues in cooperative robotics domains. The fact that partial global planning is undergoing many extensions and improvements indicates that this flexible and practical framework for coordinating distributed interpretation systems is a fertile foundation for building theories and techniques for coordination in other domains as well.

Acknowledgements

We would like to thank Dan Corkill, Krithi Ramamritham, and Reid Smith for detailed discussions about the research presented in this paper, and the members of the distributed AI group at the University of Massachusetts for intellectual and software support. We also thank three anonymous reviewers for helpful comments that have helped to improve this

paper.

References

[1] Alan H. Bond and Les Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[2] 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, Karlsruhe, Federal Republic of Germany, August 1983. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 102–105, Morgan Kaufmann, 1988.).

[3] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 367–384. Morgan Kaufman, 1988.

[4] Daniel D. Corkill. Hierarchical planning in a distributed environment. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 168–175, Cambridge, Massachusetts, August 1979. (An extended version was published as Technical Report 79-13, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, February 1979.).

[5] 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, Karlsruhe, Federal Republic of Germany, August 1983. (Also appeared in *Computer Architectures for Artificial Intelligence Applications*, Benjamin W. Wah and G.-J. Li, editors, IEEE Computer Society Press, pages 507–515, 1986).

[6] Daniel David Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks.* PhD thesis, University of Massachusetts, February 1983. (Also published as Technical Report 82-33, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1982.).

[7] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, pages 63–109, 1983.

[8] Keith S. Decker and Victor R. Lesser. Some initial thoughts on a generic architecture for cdps network control. In *Proceedings of the 1989 Distributed AI Workshop*, pages 73–94, September

1989.

[9] 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.

[10] Edmund H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers, 1988.

[11] Edmund H. Durfee. Towards intelligent real-time cooperative systems. In *Notes of the 1990 AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*, pages 29–33, March 1990. (Available in University of Maryland Systems Research Center technical report *Planning in Uncertain, Unpredictable, or Changing Environments* (SRC-TR-90-45), James Hendler (ed.)).

[12] 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, Philadelphia, Pennsylvania, August 1986.

[13] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 875–883, Milan, Italy, August 1987. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 285–293, Morgan Kaufmann, 1988.).

[14] Edmund H. Durfee and Victor R. Lesser. Incremental planning to control a time-constrained, blackboard-based problem solver. *IEEE Transactions on Aerospace and Electronics Systems*, 24(5):647–662, September 1988.

[15] Edmund H. Durfee and Victor R. Lesser. Predictability versus responsiveness: Coordinating problem solvers in dynamic domains. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 66–71, August 1988.

[16] Edmund H. Durfee and Victor R. Lesser. Negotiating task decomposition and allocation using partial global planning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*, pages 229–243. Pitman, 1989.

[17] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36(11):1275–1291, November 1987. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 268–284, Morgan Kaufmann, 1988.).

[18] 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*, Research Notes in Artificial Intelligence, chapter 2, pages 29–58. Pitman, 1987. (Also in S. Robertson, W. Zachary, and J. Black (eds.), *Cognition, Computing, and Cooperation*, Ablex 1990.).

[19] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Cooperative distributed problem solving. In Avron Barr, Paul R. Cohen, and Edward A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume IV, chapter XVII, pages 83–137. Addison-Wesley, 1989.

[20] Edmund H. Durfee and Thomas A. Montgomery. A hierarchical protocol for coordinating multiagent behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 86–93, July 1990.

[21] 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 1980.

[22] Les Gasser and Michael N. Huhns, editors. *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*. Pitman, 1989.

[23] Les Gasser and Nicolas Rouquette. Representing and using organizational knowledge in distributed AI systems. In *Proceedings of the 1988 Distributed AI Workshop*, May 1988.

[24] Michael Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 125–129, Washington, D.C., August 1983. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 200–204, Morgan Kaufmann, 1988.).

[25] Michael Huhns, editor. *Distributed Artificial Intelligence*. Morgan Kaufmann, 1987.

[26] William A. Kornfeld and Carl E. Hewitt. The scientific community metaphor. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):24–33, January 1981. (Also published

in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 311–320, Morgan Kaufmann, 1988.).

[27] Victor R. Lesser and Daniel D. Corkill. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):81–96, January 1981.

[28] Victor R. Lesser and Daniel D. Corkill. The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15–33, Fall 1983. (Also published in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, pages 353–386, Addison-Wesley, 1988 and in *Readings from AI Magazine: Volumes 1–5*, Robert Engelmore, editor, pages 69–85, AAAI, Menlo Park, California, 1988).

[29] Victor R. Lesser and Lee D. Erman. Distributed interpretation: A model and experiment. *IEEE Transactions on Computers*, C-29(12):1144–1163, December 1980. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 120–139, Morgan Kaufmann, 1988.).

[30] Victor R. Lesser, Jasmina Pavlin, and Edmund H. Durfee. Approximate processing in real-time problem solving. *AI Magazine*, 9(1):49–61, Spring 1988. (Also published in *Blackboard Architectures and Applications*, V. Jagannathan, R. Dodhiawala, and L. Baum, editors, Academic Press, 1989.).

[31] H. Penny Nii. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2):38–53, Summer 1986.

[32] H. Van Dyke Parunak. Manufacturing experience with the contract net. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 10, pages 285–310. Pitman, 1987.

[33] Jeffrey S. Rosenschein and Michael R. Genesereth. Communication and cooperation among logic-based agents. In *Proceedings of the Sixth Phoenix Conference on Computers and Communications*, pages 594–600, Scottsdale, AZ, February 1987.

[34] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

[35] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):61–70, January 1981. (Also

published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 61–70, Morgan Kaufmann, 1988.).