Planning Coordinated Actions in Dynamic Domains

Edmund H. Durfee and Victor R. Lesser

COINS Technical Report 87-130

December 1987

This paper also appears in the Proceedings of the DARPA Knowledge-Based Planning Workshop, Austin TX, December 1987.

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8306327, by the National Science Foundation under Support and Maintenance Grant DCR-8318776, by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract N00014-79-C-0439. Edmund Durfee also received support from an IBM Graduate Fellowship.

Planning Coordinated Actions in Dynamic Domains

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

Abstract

To cooperate effectively, intelligent agents must behave predictably in order to coordinate their planned activities. In dynamic domains where agents cannot fully predict future situations, however, plans about possible future actions and interactions must be tentative and modifiable, so that agents are free to respond to unexpected situations. Coordination in dynamic domains thus involves balancing predictability and responsiveness: agents must be predictable enough to anticipate and plan future interactions while being responsive enough to react to domain dynamics.

The partial global planning approach to coordination provides a framework for flexibly balancing predictability and responsiveness. In this approach, each individual agent plans its activities incrementally: it predictably pursues a sequence of major plan steps but responsively adds detailed actions to achieve those steps, interleaving planning and execution. The agents communicate summaries of their local plans to build up *partial global plans* (PGPs) that specify cooperative actions and interactions. In dynamic domains where local plans change, agents must determine when responding to changes is worth the time and effort of reformulating PGPs, and when working predictably with slightly out-of-date PGPs is more cost-effective. In this paper, we briefly outline the partial global planning approach, discuss how it flexibly balances predictability and responsiveness, and experimentally show how different balances affect behavior in a simulated problem solving network.

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8306327, by the National Science Foundation under Support and Maintenance Grant DCR-8318776, by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract N00014-79-C-0439. Edmund Durfee also received support from an IBM Graduate Fellowship.

Introduction

Coordination requires predictability. If it cannot predict the actions of others, an agent cannot take local actions that lead to coordinated interactions. Coordination is therefore easiest to achieve when agents commit themselves to explicit, globally-known plans. However, committing to such plans prevents agents from dynamically responding to unexpected situations they might encounter. To work effectively in dynamic domains, agents must be responsive, and thus unpredictable to a certain extent. Coordination in dynamic domains thus requires that the agents find a suitable balance between responsiveness and predictability.

In a distributed problem solving network, each agent is a problem solving *node* that must work with other nodes to solve network problems. A node must be responsive because its subproblems may change over time: it may get new knowledge or information, causing it to pursue different subproblems or to develop unexpected subproblem solutions. To cooperate in a coherent team, however, a node must be able to predict (at least roughly) the types of subproblem solutions that various nodes are developing and when they will be formed, which in turn means that nodes must form tentative plans. Because plans must permit both predictability and responsiveness, planning should be incremental: a node should roughly sketch out major plan steps ahead of time and work predictably on these steps, but it should detail actions for these steps incrementally as the plan is pursued, taking unexpected situations into account when doing so.

Once nodes plan their local activities, they must then reason about how their planned local actions contribute toward solving network problems. By exchanging summaries of their plans, nodes can identify when their local plans contribute to solving more global problems. Nodes develop *partial global plans* (PGPs) to represent the combined activities of some *part* of the network that is developing a more *global* solution. PGPs use the predictions from local plans to determine how and when nodes could act and interact as an effective team. Because local plans can dynamically change, however, and because communication about these changes takes time, PGPs might at times be based on incomplete, inconsistent, and out-of-date information. When this occurs, nodes may not work together as a fully coherent team, but the network still generally performs better with faulty PGPs than with no PGPs at all [Durfee, 1987]. Since PGPs are tentative and may change often, a node should not incur the high computation and communication overhead of developing optimal PGPs when those PGPs may shortly be discarded. The gains of partial global planning in improving cooperation must be weighed against its costs, and nodes should work toward satisfactory, not optimal, cooperation in dynamic domains.

Responsiveness can be detrimental if the overhead of propagating changed plan information and modifying PGPs exceeds the gains in reacting to the changes. On the other hand, by being completely unresponsive, nodes may pursue inappropriate plans: in essence, they commit to their initial plans and work as a coordinated team, even though their plans may now be unsuitable. Thus, either extreme—completely responsive or completely predictable—leads to trouble. The partial global planning approach allows the network to strike a balance, so that nodes can respond to "significant" deviations between their planned and actual problem situations, while ignoring "negligible" deviations. Moreover, nodes can develop more robust PGPs which anticipate possible deviations and which need no modification when such deviations occur.

In the remainder of this paper, we describe the partial global planning approach and experimentally show how it allows us to strike different balances between predictability and responsiveness. In Section 2, we outline the experimental testbed and briefly describe how partial global planning has been implemented within this testbed. Section 3 discusses in more detail the mechanisms for planning node interactions and how they permit different balances between predictability and responsiveness. Next, Section 4 presents experimental results that illustrate how different balances affect network performance and coordination overhead, and how an appropriate balance depends on the problem situation. Finally, in Section 5 we summarize our findings and outline our current research directions.

Partial Global Planning in the DVMT

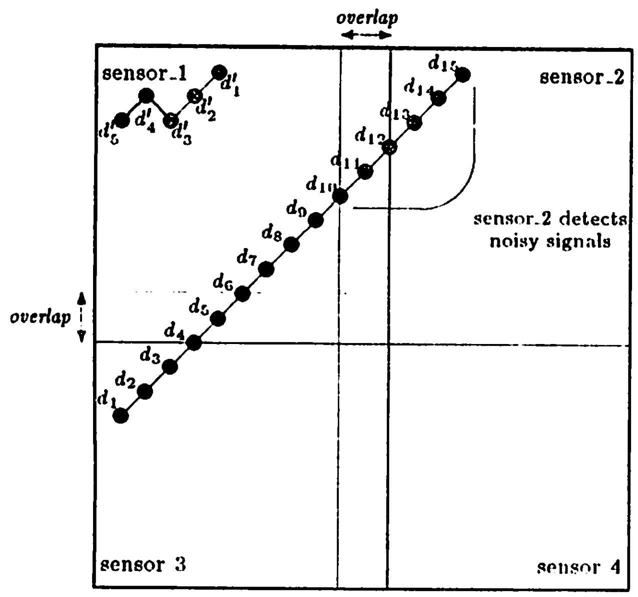
The Distributed Vehicle Monitoring Testbed (DVMT) simulates a network of vehicle monitoring nodes that track vehicles moving through an area monitored by acoustic sensors [Lesser and Corkill, 1983, Lesser *et al.*, 1987]. The acoustic sensors and the problem solving nodes are geographically distributed, so that each node receives signal information from only a local subset of the sensors. Nodes individually track vehicles through their own sensed areas and then exchange their partial tracks to converge on a complete map of vehicle movements. A node applies signal processing knowledge to correlate its sensor data, attempting to recognize and eliminate errorful sensor data as it integrates the correct data into an answer map.

Each problem solving node has a blackboard-based architecture [Erman *et al.*, 1980], with knowledge sources and blackboard levels of abstraction appropriate for vehicle monitoring. A *knowledge source* (KS) performs the basic problem solving tasks of extending and refining *hypotheses* (partial solutions). The architecture includes a goal blackboard and goal processor, and through goal processing a node forms knowledge source instantiations (KSIs) that represent potential KS applications on specific hypotheses to satisfy certain goals. KSIs are prioritized based both on the estimated beliefs of the hypotheses each may produce and on the ratings of the goals each is expected to satisfy.

An example problem situation (Figure 1) has four problem solving nodes, each connected to a different sensor (node i to sensor i) that supplies it with signal information at discrete times. A node tries to combine signals into tracks, which we represent as d_i–d_j where d_i is data for the track's first sensed time and d_j is data for its last. Let us assume that all of the data is present when network problem solving begins. Given its basic blackboard-based architecture, each node with data opportunistically pursues its best KSIs, but since in this situation the data for each sensed time is equally strongly sensed, all KSIs are equally rated and a node chooses where it works randomly.

Because the nodes have such short-sighted views, they fail to work purposefully toward important long-term network goals, such as forming d_1–d_{15}. In the example situation, node 1 could work on 14 different data points, and without a planning component it cannot recognize that in fact it should plan on forming 2 different partial tracks (d'_1–d'_5 and d_4–d_{12}). If node 1 not only forms these local plans but also receives information about the plans of nodes 2 and 3, moreover, then it could recognize that one of its plans is related to the plans of those nodes (to form d_1–d_{15}). With this information, node 1 could decide how it should develop partial results to help those other nodes (especially node 2 which must disambiguate substantial noisy data) and avoid redundant effort on the shared data in the overlapping sensor areas.

We have developed planning mechanisms that let nodes plan their actions and interactions. These mechanisms



The four overlapping sensors detect signal data at discrete sensed times (the dots with associated times). Sensor_2 is faulty and not only generates signal data at the correct frequencies but also detects noisy signals at spurious frequencies.

Figure 1: Four Node Environment (A).

are particularly effective in domains where nodes are initially uncertain about what local and network goals to pursue, where partially achieving these goals helps to resolve this uncertainty, where unpredictable changes to problem situations may affect goals and plans, and where nodes must roughly coordinate their actions to achieve network goals. To work in such domains, the planning mechanisms make several assumptions. First, they assume that a goal may be achieved in several ways (through different sequences of actions) although some ways may be better (faster, cheaper) than others. This assumption permits incremental planning: the planner can take actions with only a sketchy view of future activities because it assumes that later it can flexibly plan actions that are compatible with earlier actions and that further the achievement of its goals.[1] The second assumption is that nodes can make rough predictions about what results they may form and when, so that they can anticipate how and when they may interact. Third, the mechanisms assume that nodes can tolerate imperfect coordination: if nodes fail to coordinate as well as they might (for example, they may have different views of network activity) then network performance will suffer, but eventually the nodes will coordinate (after communicating about their views) and achieve network goals.

Local Planning. Our local planning mechanisms allow nodes to identify and plan for long-term goals while still retaining the flexibility to react to unexpected situations [Durfee and Lesser, 1986, Durfee and Lesser, 1987a, Durfee, 1987]. A node's planner begins by identifying possible long-term goals: it uses approximate domain knowledge to make a quick pass at interpreting the available data and to build a representation of the data explicitly intended for guiding control decisions. This representation takes the form of an abstraction hierarchy, where data are clustered into more and more abstract groups based on approximate domain knowledge about relationships between data.

The abstraction hierarchy indicates possible long-term solutions to work toward, but because the possible solutions are based on approximate knowledge, the planner cannot be sure which (if any) of these possible solutions will actually satisfy the more discriminating knowledge contained in the KSs. Faced with this uncertainty, the planner generates plans to pursue one or more possible solutions in such a way as to reduce the planner's uncertainty about which possible solutions are worth pursuing. As it follows plans, the planner cannot assume that the actions it takes will achieve the results it expects (those results might be unachievable), so it must monitor and repair plans, or even abort them if they cannot be completed. Moreover, since a node may receive more data over time from its sensors or from other nodes, the planner must modify long-term goals and plans to achieve them.

Because a node faces so much uncertainty about how (and whether) it will ever complete its plans, planning must be responsive and incremental. Rather than planning all of its future actions in detail only to have to discard the plan due to an unexpected event, a node only plans details for the near future. To insure that these details lead to long-term goals, a node also sketches out entire plans at an abstract level. The long-term views of plans are less likely to become obsolete over time because they only predict sequences of major plan steps, not details of how those steps will be achieved. In the DVMT, for example, a major plan step corresponds to integrating data at a particular sensed time into a track, such as extending d_i–d_j into d_{j+1}. To complete a major step of extending a track, a node typically executes several KSs, depending on the details of the data that must be integrated.

The planner employs heuristics to form sequences of

[1] Without this assumption, planning would be fruitless in dynamic domains: if near-term and distant-future actions constrain each other such that near-term actions are taken that depend on *specific* future actions, then when unexpected situations make those future actions impossible the planner cannot recover.

major plan steps that lead most effectively toward isolating and constructing promising eventual solutions. These heuristics include preferring steps that further several possible solutions simultaneously (to avoid premature commitment to a single possibility), preferring steps that are most quickly achieved (to generate partial solutions as quickly as possible), and preferring steps that emphasize differences between possible solutions (to help discriminate between possibilities).

The planner uses a plan's sequence of major steps to detail specific actions for the near future. For the next major step it should achieve, the planner generates a sequence of specific actions (each corresponding to the application of a KS). Using models of KS activities, the planner builds expectations about the results and time needs of each specific action, and makes sure that the result of the last action is expected to achieve the desired result of the major plan step. For example, in the DVMT, the last action in a sequence should expect to generate an extended track.

As each detailed action is consecutively pursued, the planner monitors execution by comparing the actual result of the KS with the expected result from the KS models. When these disagree, the planner attempts to repair the plan by inserting additional actions that may achieve the desired result.[2] If it can find such actions, then the plan is pursued further, but if it cannot then the plan is aborted. So long as actions continue to achieve the major plan steps, the plan is followed by detailing actions for the next major step when the previous one is achieved. By postponing detailed planning until necessary, the planner can detail actions based on the most up-to-date problem information, including any unexpected events that might have occurred (some KSs may have formed unexpected results or new data may have arrived from sensors or other nodes).

The planner thus interleaves generating, monitoring, repairing, and updating plans with plan execution. Through this interleaving, the planner remains responsive to unexpected situations that can arise in dynamic domains. The planner also provides predictability, however, since it not only develops a sequence of major plan steps, but also develops predictions about the results of those steps and roughly how long they will take, based on the results and time needs of similar steps in the past and on models of KSs [Durfee and Lesser, 1987a]. Incremental planning thus provides predictability at a coarse level and responsiveness at a detailed level.

[2]For example, if a KS fails to integrate data into a longer track, the planner finds other KSs that might succeed at the integration.

Partial Global Planning. For effective coordination, a node should adapt its plans to the needs of the network. To each node, therefore, we have added the ability to form and reason about *partial global plans* (PGPs) that represent how nodes should work together as a team [Durfee and Lesser, 1987b, Durfee, 1987]. Partial global planning is an integral part of a node's control activities: local and partial global planning work in tandem to maintain a view of network activity as PGPs and to modify local plans based on network needs. A node's partial global planning responsibilities depend on the *meta-level organization* that specifies the coordination roles of the nodes in the network (as opposed to the domain-level organization which specifies their problem solving roles). The meta-level organization declaratively indicates which nodes are responsible for acquiring plan information and for forming and distributing PGPs, and we can easily experiment with different meta-level organizations to determine, in a given network problem situation, whether plan information should be broadcast and every node should form PGPs, whether a single node should form and distribute PGPs for the network, or whether some other arrangement leads to the best performance.

The first step in partial global planning is to provide one or more nodes with information about the plans of various nodes. This information does not need to include details about short-term actions: not only are such details quickly outdated, but they are also irrelevant since nodes should only attempt to coordinate their more gross outward behavior—their major plan steps—and how they internally achieve this behavior is unimportant for coordination. With this information, a node forms *models* of other nodes and of itself (by saving summaries of its own plans). The partial global planner scans these models to find cases where several nodes are working on parts of some larger *partial global goal*.

The partial global planner forms PGPs to achieve the partial global goals. It uses information about the major plan steps (including the rough predictions about how long each will take) to interleave the expected activities of the participating nodes. It then scans this *plan-activity-map* to find activities that are more useful (such as activities that form important results to share) or less useful (such as activities that unnecessarily form redundant results). Each activity is rated based on attributes such as its expected time needs, its expected result quality, how it will be affected by preceding activities, and how it will affect later activities. The partial global planner attempts to reorder activities to move more highly-rated ones earlier in the PGP. Having formed a new order, it once again rates the activities (since changing their relative order can alter their ratings) and reorders them again. This hill-climbing procedure continues until a reordering does not increase the sum of the ratings of all activities. The procedure is *not* guaranteed to find an optimal ordering since it does not search the entire space of possible orderings. However, because plan-activity-maps may be reordered many times in dynamic situations, cheaply finding a good ordering using hill-climbing is more cost-effective than investing the time needed to find an optimal ordering.

With a reordered plan-activity-map, the partial global planner modifies the local plans to reflect the reordering of major plan steps and to detail actions for the next major plan step if it has changed. The partial global planner also forms a *solution-construction-graph* that indicates which sharable partial results will be generated by individual nodes, and when and how these results should be exchanged and integrated into complete results. The solution-construction-graph therefore represents expected interactions between nodes and is used by nodes to plan their communication actions. Because these expected interactions are based on predictions about what partial results nodes will generate and when, unexpected changes to problem solving activities can change how nodes view their interactions and can disrupt coordination. Whether nodes should bother to reformulate their PGPs based on dynamic changes to their problem, however, depends on how they balance predictability and responsiveness.

Predictability and Responsiveness

The purpose of partial global planning is to improve network coordination without introducing excessive computation and communication overhead. Because nodes must coordinate in dynamically changing situations, partial global planning does not expect to optimally coordinate nodes: it works under the assumption that, because of domain dynamics and communication delays and errors, nodes cannot avoid forming PGPs based on potentially incomplete, inconsistent, and out-of-date information. In dynamic domains, nodes may never be able to reach agreement on how they should coordinate, so each must use its local view to decide on actions and interactions that it believes will lead to effective coordination. If they communicate enough information about their plans and partial results, nodes will converge on network solutions in a functionally-accurate, cooperative manner [Lesser and Corkill, 1981]. Partial global planning increases the rate of convergence by allowing nodes to more fully represent and reason about actions and interactions that will likely lead to cooperation.

Because PGPs represent only rough expectations about network coordination, nodes should anticipate deviations from these expectations, or should at least tolerate such deviations. If a node changes its local plans because it gets unexpected information, this change to local plans can affect PGPs. Sometimes such a change is significant: the node might now expect to form drastically different results or to form its results at a substantially different time for its current PGP; or it might redirect its attention from one PGP to another, disrupting how a group of nodes expected to coordinate. On the other hand, a change might be negligible: a node might expect to form a result at a different time, but this change either might not affect other nodes at all (they should not change their plans at all) or the changes it triggers in other nodes cause only minor improvements to coordination, at the cost of having to reformulate the PGPs.

For example, node A might initially expect to generate d_1–d_2 at time 6, while node B expects to generate d_3–d_6 at time 12. The PGP indicates that node A should send d_1–d_2 to B, and it will arrive at time 8 (due to communication delays) but will not be integrated with B's result until sometime after time 12. If node A determines that it had underestimated the time it would need to form its result, and in fact it cannot get its result to node B until time 12, this change is unimportant from node B's perspective since it will not disrupt coordination at all. Alternatively, if node A cannot form d_1–d_2 until time 20, this could significantly disrupt coordination: rather than waiting to receive and integrate d_1–d_2 at time 22 (due to communication delays), perhaps node B should send d_3–d_6 to A so that A can integrate the results as soon as it forms d_1–d_2 at time 20. Finally, perhaps node A cannot get d_1–d_2 to node B until time 13. Is a difference of 1 time unit worth the effort of communicating about plans and recomputing PGPs, or can this minor deviation from expectations be ignored and the minor inefficiency tolerated? In fact, the overhead of communication and computation to respond to the deviation might be more costly than the inefficiency caused by the deviation, especially in dynamic domains where deviations are common.

Partial global planning allows nodes to achieve a balance between predictability and responsiveness so that nodes can work as a reasonably effective team without wasting resources by continually making minor improvements. This balance is specified to nodes as a tolerance for deviations in the timing of their planned interactions. Nodes use this tolerance in two ways. First, as they pursue and modify their plans, they compare these plans with the expectations in the PGPs and *respond* to deviations. Second, when they develop PGPs they use the tolerance to *predict* (plan for) possible deviations, so their PGPs are more robust and less likely to need modification. These techniques for balancing predictability and responsiveness are described in more detail below.

Responding to Deviations. If a node that has been cooperating with other nodes suddenly begins pursuing another plan because of unexpected information generated locally or received from elsewhere, then that node must inform the other nodes of the change. If either the old plan or the new plan is part of some larger PGP that other nodes share, then those nodes must be informed of the change or else they will anticipate interactions that may never come about. Switching from one plan to another is thus a significant deviation of behavior, and the partial global planning component of the node transmits information about the change to other nodes (based on the meta-level organization).

Even when a node consistently pursues the same plan, however, its actual behavior may deviate from its predicted behavior: the predicted time needs of major plan steps are, after all, only approximations based on models of KSs and on the time needs of similar (not identical) steps in the past. Moreover, during plan monitoring and repair additional actions may be inserted that increase the time needed to complete plan steps. The deviations in *when* plan steps will be completed does not affect the overall goals of PGPs, but can change *how* nodes view their interactions. When a PGP is initially formed, the plan-activity-map and solution-construction-graph reflect a good (not necessarily optimal) sequence of node actions and interactions to generate a complete solution. A node that fails to act and interact when it is expected can cause inefficient network behavior as some nodes sit idle because either they worked faster or others worked slower than expected.

Nodes must be sensitive to deviations in their plans to avoid such inconsistencies. However, if they are over-sensitive, they may communicate about negligible changes to their plans where, after all the effort to reformulate better PGPs, the nodes end up planning to interact exactly as before. Worse yet, when one node changes its plans, the modification to the PGP can trigger another node to change its plans, which modifies the PGP further and triggers changes in other nodes, and so on. Such a chain-reaction of minor changes to plans can be very expensive in overhead and have little if any benefit. Nodes may even oscillate between several different PGPs as these changes are propagated. Although the oscillation must eventually cease,[3] the nodes would work as a better team if they simply chose one of these PGPs and stuck to it.

Nodes need to know when deviations are negligible and should be ignored, so that they can dampen their reactions to deviations. The partial global planner considers a deviation between actual and predicted times to be negligible if that difference is no larger than the *time-cushion* [Durfee, 1987]. The time-cushion is a user-specified parameter (although we eventually hope to have the partial global planner compute it dynamically) that represents negligible time. It is the time-cushion that balances predictability and responsiveness, since a small time-cushion forces nodes to respond more frequently to deviations while a large time-cushion allows them to continue working on their plans in essentially the way that they had expected to despite deviations.

For nodes to develop consistent PGPs, they need to have consistent views of the network. This means that a node's model of itself should agree with how other nodes model it. When a node's plan deviates from expectations, it therefore has a choice. It could summarize the modified plan, using that summary to change its model of itself and also sending the summary off to other nodes that are responsible for developing PGPs. On the other hand, it could determine that the changes to the plan are negligible, based on the time-cushion. In this case, the node does *not* modify its model of itself even though it has a more up-to-date view of its plans. When forming and reasoning about PGPs, the node must view itself as it believes others view it, based on the information it has sent out. If nodes build PGPs based on views that they do not share, then they may not only develop inconsistent PGPs (which can occur when they do share their views because of domain dynamics and communication delays) but they may *never* converge on consistent PGPs (as they will eventually if they share their views).

As a consequence, nodes in our network maintain two possibly different views of themselves: a view of their internal problem solving (represented by their local plans), and a view of themselves as part of the network (represented by their models of themselves). How far these views can diverge depends on the time-cushion. With a time-cushion of 0, any deviation in local plans causes nodes to change their models so that they have as accurate a view of each other as possible. As the time-cushion grows, the possibilities for differences increase, so that nodes may be coordinating based on outdated views of their plans. An appropriate choice of time-cushion allows nodes to coordinate based on acceptably

[3]Because the nodes are constructing partial solutions, they make progress over each oscillation so eventually nodes complete their plans despite oscillations. This assumes that activity is constructive; if nodes could undo each other's actions, then the oscillations could go on indefinitely. For such domains, nodes would need additional mechanisms to recognize cyclic activity and terminate it.

accurate models of themselves and others without the overhead of updating their models and PGPs for insignificant plan deviations.

Planning for Deviations. The partial global planner also uses the time-cushion when forming PGPs in the first place in order to build more robust expectations about coordination. When building the solution-construction-graph, for example, the partial global planner uses the time-cushion to build more robust expectations about where and when the partial results from nodes should be integrated. With a time-cushion of 0, the partial global planner builds a solution-construction-graph where partial results are combined as early as possible. With larger time-cushions, however, the partial global planner has more flexibility. It may determine that node A could integrate partial results at time t while node B could integrate the results at time $t + i$. If i is no greater than the time-cushion, then the partial global planner considers the difference between when the nodes could integrate the results to be negligible. To choose where to integrate the results, the partial global planner determines how busy the alternative nodes are by examining the PGPs to see what important partial results they are expected to form. Because nodes may take more time than expected to form partial results, the partial global planner takes the strategy that, if several nodes could integrate results at about the same time, it should assign the integration task to the least busy node because it is least likely to encounter delays that will postpone the integration. The resulting PGP is therefore more robust to changes in plans.

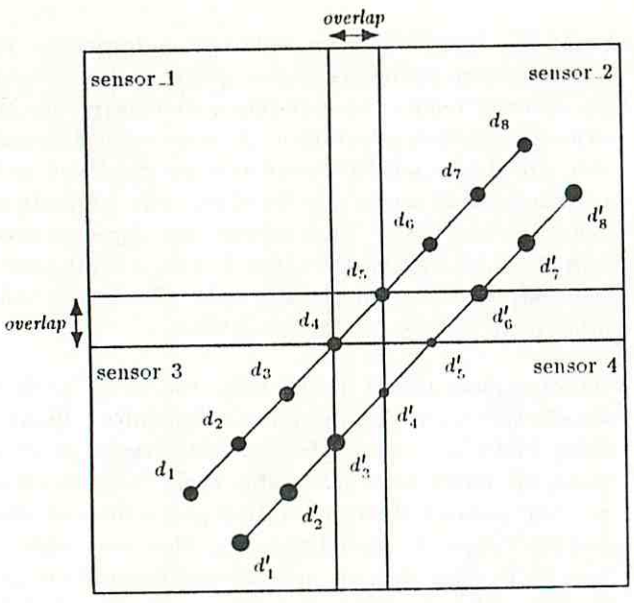
The partial global planner also uses the time-cushion to build more robust PGPs when it decides to delay acting on one PGP to assist in another. For example, a node may generate a partial result for a highly-rated PGP, and should send this result elsewhere for integration. However, according to the PGP, it will send this result much earlier than it needs to, because the other results for integration will not be available for some time. In this situation, the node may choose to postpone forming the result: it can work on other results for a while and still generate the result so that it arrives at the integrating node when it is needed. Because of the uncertainty of predictions, however, the node might add some "cushion" to the expected time needs to form the result, just in case. If the node has a small time-cushion for this value, it implies that it expects small deviations from predictions, while a larger time-cushion gives more leeway. Thus, a large time-cushion leads to more robust PGPs that anticipate possible plan deviations.

As a final example of how the partial global planner plans for deviations from expected performance, the solution-construction-graph can anticipate the possibility of node failures by building redundancy into the expected solution integration. A user-specified parameter called the *solution-construction-redundancy* indicates how many nodes should redundantly integrate results [Durfee, 1987]. This redundancy improves reliability by insuring that the network will generate overall solutions even if an integrating node fails because some other node will also do the integration.

Building more robust PGPs helps the nodes work as an effective team despite domain dynamics. Because these PGPs are applicable in a wider range of situations, the nodes need not modify their PGPs as often, and this reduces the computation and communication overhead of partial global planning. However, more robust PGPs often degrade network performance because they let nodes coordinate less crisply, allowing them be less precise about when they interact so that some nodes may sit idle, waiting for others. Building in redundancy among the nodes can also degrade performance when nodes unnecessarily duplicate each other's results. The partial global planner must therefore balance the costs and benefits of building robust PGPs, because making overly predictable PGPs degrades the network's ability to advantageously respond to specific situations.

Experiments

We have extensively evaluated the benefits and costs of partial global planning [Durfee, 1987]. In this section, we discuss a small subset of experiments that illustrate how different balances between predictability and responsiveness affect network performance in a few situations. These experiments employ two network problem solving environments. Environment A was shown in Figure 1, while environment B is shown in Figure 2. Each of these environments involves a network of four nodes, where each node is connected to a different sensor. Environment A tests how well the partial global planner distinguishes between more or less globally important plans (node 1 has one plan that is more globally important than another), how it allows nodes to provide predictive information (node 1 should send the short track d_8–d_9 to node 2 to help it disambiguate its data), and how it avoids redundant activity in overlapping areas. In environment B, two vehicles pass among the nodes and the network should find both solutions. Environment B tests how well the partial global planner allows different subsets of nodes to work on different PGPs simultaneously and how it allows nodes to avoid redundancy despite the high degree of data overlap. In these simulated networks, 1 time-unit corresponds to



The four overlapping sensors detect signal data at discrete sensed times (the dots with associated times). Two vehicles move in parallel from the lower left to the upper right corners.

Figure 2: Four Node Environment (B).

the time needed to execute 1 KS. It takes 2 time-units for a message to get from one node to another.

Our experiments explore two different meta-level organizations. In the *broadcast* meta-level organization, each node broadcasts summaries of its local plans and develops PGPs based on the summaries it forms locally and receives from other nodes. In the *centralized* organization, a single node is responsible for forming and distributing PGPs. In environment A, node 4 is the coordinator (nodes 1–3 send their plan summaries to 4 which forms PGPs and sends them back to 1–3) while in environment B, node 1 is the coordinator.[4]

For the four combinations of environments and meta-level organizations, we run three experiments: setting the time-cushion to 0, 1, or 2 time-units. For comparison, we also run experiments with only local planning (no coordination though PGPs) and with neither local nor partial global planning. In these experiments, we measure four things [Durfee, 1987]. First, we measure the simulated runtime of the network. Since each time-unit corresponds to executing a KS, the simulated runtime corresponds to the number of KSs run by the nodes, so a lower simulated runtime means that the nodes made better, more coordinated decisions about how to solve network problems. Second, we measure the actual runtime of the simulation. Given the current implementation of the KSs and the planning mechanisms, this measure indicates how much computation was performed in the network on both problem solving and planning (the time spent context-switching to simulate the network is negligible) to understand whether the computation overhead of planning is worthwhile. Third, we measure communication of hypotheses and of plan information to roughly determine the communication needs of the network. Fourth, we measure the number of data structures generated, including hypotheses, goals, KSIs, plans, and PGPs to roughly estimate the storage overhead of the planning mechanisms.

[4]The node chosen as coordinator is the node with the least data to process.

The experimental results are summarized in Table 1. We begin with environment A. First, note that without any planning at all, the simulated and actual runtimes are very high, as are the number of hypotheses communicated and the amount of storage (E1). Introducing local planning substantially reduces all four measurements (E2). Partial global planning (E3–E8) makes further substantial reductions to simulated runtime because the nodes' control decisions are more coordinated. Because computing PGPs requires computation, however, the overhead of partial global planning means that savings in actual runtime are less substantial. Moreover, partial global planning requires significant communication about plans and PGPs, so overall communication overhead rises despite the reduction in hypotheses exchanged. Whether the improvements to coordination are worth the communication depends on the relative cost of communication. Finally, partial global planning reduces storage needs despite building more plan information because fewer KSs are executed, resulting in fewer hypotheses, goals, and KSIs.

Looking more closely at the effects of the time-cushion, we begin with environment A using a broadcast organization (E3–E5). As the time-cushion increases, several trends become apparent. First, the quality of coordination decreases because nodes build PGPs that tolerate less crisp interactions and because they do not adapt the PGPs to changing circumstances as often so that they continue with PGPs that may not be the best they could form. Second, the computation overhead is substantially reduced, since nodes do not recalculate how they should coordinate as often. Third, the communication overhead is also significantly reduced, since nodes update each other (by transmitting plan summaries) less often. Fourth, the storage overhead slightly increases due to the extra problem solving caused by less precise coordination: the extra storage is attributable to more hypotheses, goals, and KSIs, while the coordination storage is essentially the same (since updated plan summaries replace earlier versions). The same trends

Table 1: Experiment Summary.

Ex	En	Org	TC	ST	RT	H-r	M-r	T-r	Store
E1	A	no	-	171	465	44	-	44	3593
E2	A	lo	-	81	76	17	-	17	1688
E3	A	bc	0	43	76	5	63	68	1280
E4	A	bc	1	46	64	5	54	59	1352
E5	A	bc	2	47	57	4	42	46	1357
E6	A	cn	0	45	59	6	65	71	1306
E7	A	cn	1	48	52	4	48	52	1331
E8	A	cn	2	49	50	4	35	39	1347
E9	B	no	-	84/44	221	117	-	117	3256
E10	B	lo	-	30/44	42	24	-	24	1173
E11	B	bc	0	25/34	45	6	95	101	1015
E12	B	bc	1	25/34	37	5	54	59	1006
E13	B	bc	2	26/39	39	7	63	70	1093
E14	B	cn	0	32/41	42	8	85	93	1057
E15	B	cn	1	26/35	32	7	49	56	985
E16	B	cn	2	32/47	39	4	41	45	1136

Abbreviations

En:	The problem solving environment
Org:	The meta-level organization used, if any: no = no planning, lo = local planning only bc = broadcast, cn = centralized
TC:	The time-cushion used (if any)
ST:	The simulated time to find solution(s); if more than one, earliest time for each is given (better-sol/worse-sol).
RT:	The actual experimental runtime (in minutes).
H-r:	Number of hypotheses communicated.
M-r:	Number of meta-level messages (plan summaries and PGPs) communicated.
T-r:	Total number of messages communicated.
Sto:	The total number of structures stored.

are seen with the centralized organization (E6–E8).

In environment B, the same basic differences are seen between having no planning (E9), having only local planning (E10) and having partial global planning (E11–E16). However, when the time-cushion is varied, different phenomena are encountered. In the broadcast organization, the best time-cushion is 1 (E12). A lower time-cushion (E11) does not improve coordination (solution time) while it does introduce substantially more computation and communication overhead (because nodes unnecessarily update their plan summaries and PGPs more often). Meanwhile, a higher time-cushion (E13) degrades coordination because nodes do not adequately adapt to incorrect predictions about when they will exchange results. By the time nodes do respond to inappropriate PGPs, they have already wasted time on unnecessary actions (either duplicating each other's results or forming results for inferior plans while waiting for results from others) so network computation is increased due to this extra work. In addition, when a node does finally react to deviations in its local plans and updates its plan summaries and PGPs, the exchange of the changed plan summaries causes other nodes to change their plans, and these cause other nodes to further change, so on. This chain-reaction of changes increases the meta-level communication so that nodes communicate more despite the higher time-cushion (comparing E13 with E12).[5]

With a centralized organization, a lower time-cushion actually degrades coordination (E14), because nodes are *too* responsive. Specifically, the more constant stream of updated plan information received by node 1 (the coordinating node) causes it to change the network PGPs and nodes oscillate between coordinating one way and then another. For example, the expectation about whether node 3 or node 4 will integrate d'_1–d'_2 and d'_3–d'_6 changes several times, where sticking to either decision would have been resulted in better performance. A higher time-cushion (E16) also degrades coordination, but this time because nodes are not responsive enough. In the broadcast organization (E13), nodes build their own PGPs and this introduces inconsistencies that can trigger a chain-reaction of updated plans whenever one node changes its plans. Such chain-reactions do not occur with a centralized organization, because only one node (in this case node 1) forms PGPs for the network: it determines how all of the nodes should respond to a changed plan and imposes this view on the nodes so that they cannot respond for themselves. As a consequence, the nodes must communicate less (comparing E16 with E15, as opposed to E13 compared with E12). In turn, the PGPs formed by node 1 are modified much less frequently, so the nodes pursue PGPs based on outdated information and solution time (relative to E15) suffers as a result. Because the network invokes more KSs, overall network computation increases when compared to E15 despite the lower partial global planning overhead. Whether the savings in communication warrant this choice of time-cushion over the time-cushion of 1 (E15) depends on the available network resources.

Conclusions

Our experimental results show that partial global planning improves network coordination, but it also introduces overhead in computation, communication, and storage. Partial global planning also allows us to strike different balances between predictability and responsiveness in the network, but as we have seen the balance chosen results in both benefits and costs. By increasing

[5]Most of this extra communication activity occurs near the end of network problem solving when some nodes have finished their local responsibilities for important PGPs and begin pursuing and communicating about less highly-rated plans.

responsiveness by lowering the network's view of "negligible" time, we were sometimes able to improve coordination so that the network works as the most coherent team possible. This comes at the cost, however, of more communication and computation as nodes must reformulate their PGPs. In addition, sometimes nodes can be too responsive, so that they jump from one view of coordination to another and end up working less effectively. However, increasing predictability by raising the networks view of negligible time can reduce the costs of coordination since nodes are less prone to communicate about changed plans and reformulate PGPs, but as a result the network does not coordinate as well as it potentially could.

Our results thus point out two very important facts. First, there is no correct balance between responsiveness and predictability that is independent of the problem situation. Second, and consequently, planning mechanisms for coordinating agents in dynamic domains must have the flexibility to strike different balances between predictability and responsiveness. Our partial global planning approach has such flexibility. By allowing nodes to plan their activities incrementally, the approach permits sufficient predictions about node activities without stifling a node's ability to respond to unexpected events. By reasoning about the more gross aspects of node behavior and by flexibly ignoring deviations in plans, the partial global planning approach coordinates nodes without incurring excessive overhead by appropriately balancing the benefits of better coordination against the costs of achieving that coordination.

Balancing predictability and responsiveness is only one of the issues that partial global planning allows us to explore [Durfee, 1987]. From the perspective of an individual node, we have studied issues in meeting deadlines, in deciding when to terminate problem solving, in monitoring and repairing plans, and in modifying local plans as information arrives at different rates. From a network perspective, we have explored how the choice of meta-level organization affects the costs and benefits of cooperation, and how partial global planning allows us to achieve a variety of cooperative goals such as avoiding unnecessary redundancy, balancing problem solving load, exploiting different nodes' expertise, and improving network reliability. In short, the partial global planning approach provides the versatility to coordinate agents in a wide variety of cooperative activities, and the implementation of the approach in the DVMT allows us to experimentally evaluate its benefits, costs, and limitations. Our current and future research directions involve further extending and testing our approach to better understand how artificially intelligent agents should coordinate their activities in dynamic and uncertain domains.

References

[Durfee, 1987] Edmund Howell Durfee. *A Unified Approach to Dynamic Coordination: Planning Actions and Interactions in a Distributed Problem Solving Network.* PhD thesis, University of Massachusetts, Amherst, Massachusetts 01003, September 1987. (Also published as Technical Report 87-84, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, September 1987.).

[Durfee and Lesser, 1986] Edmund H. Durfee and Victor R. Lesser. Incremental planning to control a blackboard-based problem solver. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 58–64, August 1986.

[Durfee and Lesser, 1987a] 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*, 1987. In press. (Also published as Technical Report 87-07, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, February 1987.).

[Durfee and Lesser, 1987b] 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, August 1987.

[Erman *et al.*, 1980] 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.

[Lesser and Corkill, 1981] 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.

[Lesser and Corkill, 1983] 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.

[Lesser, *et al.*, 1987] Victor R. Lesser, Daniel D. Corkill, and Edmund H. Durfee. *An Update on the Distributed Vehicle Monitoring Testbed.* Technical Report 87-111, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, 1987.