Predictability Versus Responsiveness: Coordinating Problem Solvers in Dynamic Domains

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

Abstract

Coordination in dynamic domains involves balancing predictability and responsiveness: agents must be predictable enough to anticipate and plan future interactions while being responsive enough to react to unexpected situations. The partial global planning approach to coordination provides a framework for flexibly balancing these opposing needs. In this approach, agents communicate about their current local plans to build up partial global plans (PGPs) that specify cooperative actions and interactions. When their plans change, agents must decide whether the time and effort of reformulating their PGPs is worthwhile, or whether working predictably with slightly out-of-date PGPs is more cost effective.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.

1 Introduction

Coordination requires predictability. If unable to predict each other's actions, agents cannot coordinate their interactions. Coordination is therefore easier when agents commit themselves to explicit, globally-known plans. However, committing to such plans prevents agents from dynamically responding to unexpected situations. To work effectively in dynamic domains, agents must be responsive, and thus unpredictable to a certain extent. Coordination in dynamic, uncertain domains thus requires that the agents suitably balance responsiveness and predictability.

In a distributed problem-solving network, for example, each agent is a problem-solving **node** that works with other nodes to solve network problems. A node must respond to changing subproblems: it might get new knowledge or information that causes it to pursue different subproblems or develop unexpected subproblem solutions. To cooperate with others, however, a node must predict (at least roughly) what subproblems other nodes will be solving and when, which in turn means that nodes must form tentative plans. Nodes therefore must have a framework for coordination that allows them to tentatively plan coordinated interactions and to modify their plans in response to unanticipated situations.

The **partial global planning** approach is a flexible framework for coordination where nodes can balance their needs for predictability and responsiveness differently for different situations. In this framework, nodes exchange information about their tentative local plans and develop **partial global plans** (PGPs) to represent the combined activities of some *part* of the network that is developing a more *global* solution. A node's PGPs indicate its current view of how nodes should coordinate on forming larger solutions. Because local plans can change and communication about these changes takes time, however, a node's PGPs might at times be based on incomplete, inconsistent, and out-of-date information. Such PGPs can degrade network problem-solving performance because nodes might not work as a coordinated team, but on the other hand the communication and computation overhead for forming and maintaining the best possible PGPs might be prohibitively high. In dynamic domains, nodes should strive for satisfactory, not optimal, cooperation by balancing predictability and responsiveness. The partial global planning approach allows nodes to strike a balance so that they incur the planning overhead only for "significant" deviations from planned activity, and so that they develop more robust PGPs that need less modification when deviations occur.

2 Partial Global Planning in the DVMT

To study and evaluate our approach to coordination, we have implemented the partial global planning framework in the Distributed Vehicle Monitoring Testbed (DVMT), which simulates a network of vehicle monitoring nodes that track vehicles moving through an acoustically sensed area [Lesser and Corkill, 1983]. The acoustic sensors and problem-solving nodes are geographically distributed, so that each node receives signals from a local subset of sensors. A node has a blackboard-based problem-solving architecture with knowledge sources and levels of abstraction appropriate for vehicle monitoring. Nodes apply their signal processing knowledge about the characteristic sounds and movements of vehicles in order to correlate their sensor data, integrating this data into larger, more abstract *hypotheses* (partial solutions) about vehicle movements. By exchanging the high-level hypotheses formed from their in-

This research was sponsored, in part, by the National Science Foundation under CER Grant DCR-8500332, and by the Office of Naval Research under University Research Initiative Grant Contract N00014-86-K-0764, and under Contract N00014-79-C-0439. Edmund Durfee also received support from an IBM Graduate Fellowship.
Edmund Durfee will be with the Department of Electrical Engineering and Computer Science at The University of Michigan beginning Fall 1988.

dividual sensor data, nodes use their knowledge about vehicle movements to integrate their partial hypotheses into a complete answer map.

Nodes act by forming local hypotheses and interact by exchanging hypotheses not only to converge on overall solutions but also to provide information that helps others solve local subproblems. By coordinating their actions and interactions, they avoid duplicating effort in tracking vehicles through overlapping sensed areas and they share partial tracks in a timely manner to resolve uncertainty about their information. Nodes should consider their local expertise and available computing resources when deciding which local subproblems to solve and where to assign future subproblems such as integrating partial results from several nodes. Because subproblems, expertise, and other resources may be inherently but possibly unevenly distributed, nodes need to coordinate for result-sharing *and* task-sharing [Davis and Smith, 1983; Durfee and Lesser, 1988b].

Each node has a local planner that balances the needs for predictability and responsiveness by planning *incrementally* [Durfee and Lesser, 1986; Durfee and Lesser, 1988a]. For predictability, the planner sketches out a sequence of major plan steps that will lead to possible problem solutions. In the DVMT, the major plan steps correspond to extending partial tracks into new time frames (such as extending the track d_i–d_j into d_{j+1}, where d_k is data sensed at time k). A major plan step might take several processing actions to analyze the new data, filter out noise, and integrate the correct data into the track. For responsiveness, the planner only details specific actions for achieving a major plan step when that step must be taken, so the choice of actions depends on the current situation. Thus, the planner interleaves planning and execution, and can add new actions to the plan when planned actions fail to achieve their desired results. For each major plan step, the local planner also roughly estimates what partial results will be formed and when, based on models of problem solving and on past problem-solving experience.

Each node also has a partial global planner (PGPlanner) as an integral part of its control mechanisms [Durfee and Lesser, 1987]. The PGPlanner forms **node-plans** to summarize a node's local plans, where a node-plan specifies the possible solutions being developed by the plan, and the plan's major steps, including the predictions about when the steps will be taken and their expected results. Nodes do not communicate about their detailed actions because this information is frequently changed and quickly outdated. Where a node sends its node-plans depends on the **meta-level organization** that specifies the coordination roles of the nodes (as opposed to the domain-level organization that specifies their problem-solving roles). The meta-level organization might have nodes send their node-plans to some coordinator nodes that decide how they should work together, or it might have nodes simply broadcast the plan information to all nodes so that each can develop a complete model of the network.

A node's PGPlanner scans its current network model to identify when several nodes are working on goals that are pieces of some larger **partial global goal**. By combining information from individual plans, the PGPlanner builds PGPs to achieve the partial global goals. The PGPlanner forms a **plan-activity-map** from the separate plans by interleaving the plans' major steps using the predictions about when those steps will take place. Thus, the plan-activity-map represents concurrent node activities. To improve coordination, the PGPlanner reorders the activities in the plan-activity-map using expectations about their costs, results, and utilities. Rather than examining all possible orderings, the PGPlanner uses a hill-climbing procedure to *cheaply* find a better (not always optimal) ordering. From the reordered plan-activity-map, the PGPlanner modifies the local plans to pursue their major plan steps in a more coordinated fashion. The PGPlanner also builds a **solution-construction-graph** that represents the interactions between nodes. By examining the plan-activity-map, the PGPlanner identifies when and where partial results should be exchanged for the nodes to integrate them into a complete solution, and this information is represented in the solution-construction-graph.

3 Predictability and Responsiveness

PGPs represent only rough expectations about network coordination, and nodes should anticipate, or at least tolerate, deviations from these expectations. If a node changes its local plans because it gets unexpected information, these changes can affect PGPs because nodes might abandon one PGP in favor of another or might develop and transmit partial solutions at times significantly different from when originally planned. Partial global planning allows nodes to balance predictability and responsiveness so that they respond to significant temporal deviations without wasting resources to make minor improvements. This balance is specified as a tolerance representing negligible time. Nodes use this tolerance as they pursue and modify their plans to detect when deviations exceed this tolerance, and when this happens nodes *respond* to the deviations. In addition, nodes use this tolerance when they develop PGPs in order to *predict* (plan for) possible deviations, so their PGPs are more robust.

3.1 Responding to Deviations.

A node that has been cooperating with other nodes might suddenly begin pursuing another plan because of unexpected information generated locally or received from elsewhere. 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 to another plan is thus a significant deviation of behavior that nodes should communicate about and respond to.

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. Moreover, additional actions may be added to a plan when planned actions fail to form desired results. 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. For example, to cooperatively form d_1–d_6, 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 after time 12. If node A has underestimated the time it needs to form its result, and in fact it cannot get its result to node B until time 12, this change is negligible since it will not affect when node B will integrate the results. Alternatively, if node A cannot form d_1–d_2 until time 20, this change could significantly disrupt coordination: rather than waiting to receive and integrate d_1–d_2 at time 22, 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, if node A cannot get d_1–d_2 to node B until time 13, is the 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?

To avoid inefficiencies, nodes must be sensitive to plan deviations, but must not be overly sensitive or else they will communicate about negligible changes to their plans where, after all the effort to reformulate better PGPs, the nodes interact no better. 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 or no benefit. Nodes may even oscillate between several different PGPs as these changes are propagated. Although the oscillation must eventually cease,[1] the nodes would work as a better team if they simply chose one of these PGPs and stuck to it.

To dampen their reactions to deviations, nodes need to know when deviations are negligible and should be ignored. The PGPlanner considers a deviation between actual and predicted times to be negligible if that difference is no larger than the time-cushion [Durfee, 1988]. The **time-cushion** is a user-specified parameter (although we eventually hope to have the PGPlanner 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.

When one of its local plans deviates from expectations, a node must decide whether to respond to improve network coordination. However, the computation and communication costs in making this decision can be high. To completely identify the deviation's consequences, the node cannot assume that its PGPs and models of other nodes are complete and up-to-date, so it must communicate with other nodes. Alternatively, the node could reduce costs by determining the deviation's significance based on only its local view. It could determine how the deviation could affect the PGP(s) that the plan contributes to, and how these effects might influence other participating nodes, and how these other nodes might as a result deviate in other PGPs, and so on. In effect, nodes would duplicate much of the same processing they would perform if they had simply assumed that the deviation was significant and propagated its effects. Rather than incurring this computational overhead in exploring all of the repercussions of a local deviation, our current implementation instead simply compares the deviation in the local plan with the time-cushion, resulting in less informed but also less costly decisions about when to respond to temporal deviations.

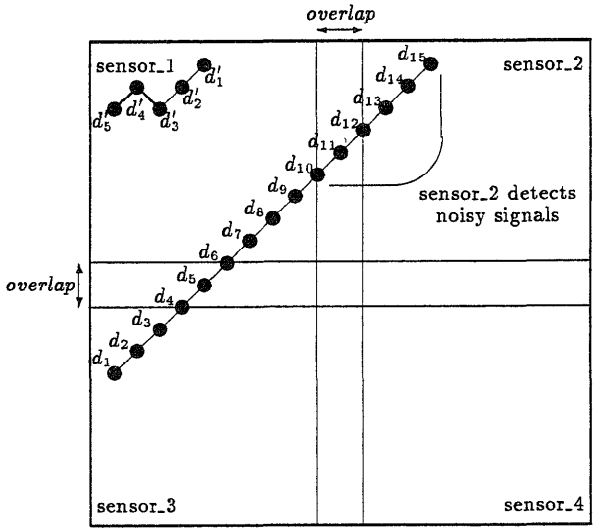
When a plan's deviation from temporal expectations is greater than the time-cushion, the deviation is propagated to the corresponding node-plan, which is transmitted to relevant nodes so that nodes will have consistent views about the plan. Without consistent views, nodes might not only form inconsistent PGPs (which can occur even when they do share their views because of domain dynamics and communication delays), but they might *never* converge on consistent PGPs (as they will eventually if they share their views). When local plan deviations are less than the time-cushion, the node-plan is not changed, and so the model that the nodes have of this particular node remains the same. Similarly, the model this node has of itself with respect to the network remains unchanged. Thus, nodes 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.

3.2 Planning for Deviations.

The PGPlanner also uses the time-cushion to build more robust PGPs. When building the solution-construction-graph, the PGPlanner uses the time-cushion to build more robust (less particular) expectations about where and when the partial results from nodes should be integrated. For example, the PGPlanner might 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 PGPlanner considers the difference between when the nodes could integrate the results to be negligible. The PGPlanner then chooses the least busy of these nodes—the node expected to pursue the fewest activities or complete the results for all of its PGPs soonest—to integrate the results, because this node is most likely to carry out the integration as planned.

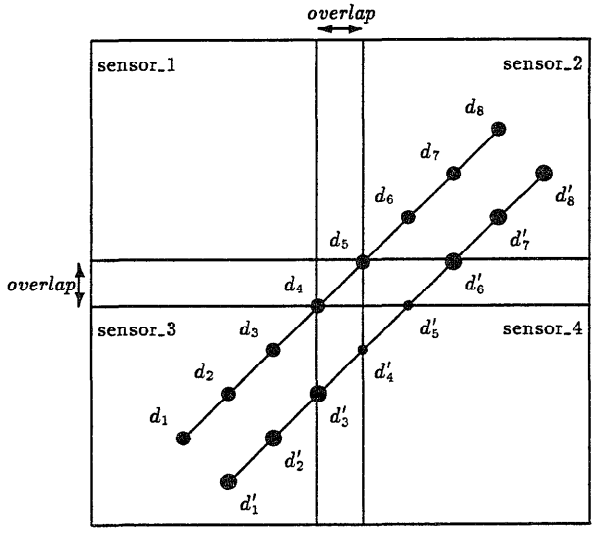
The PGPlanner also uses the time-cushion to build more robust PGPs when it decides to delay acting on one PGP to assist in another. For example, if a node expects to generate a partial result long before the related partial results are available for integration, then the node may choose to delay working on the partial result and instead pursue other PGPs. However, it should return to the original PGP when there is just enough time to form the needed partial result. Because of the uncertainty of predictions, however, the node might add some "cushion" to the ex-

[1] 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.



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).



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).

pected time needs to form the result, just in case. The larger the time-cushion, the more robust and tolerant the PGP is to deviations.

As a final example of how the PGPlanner 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. 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 to be less precise about when they interact so that some nodes may sit idle, waiting for others. Building in redundancy also may cause nodes to unnecessarily duplicate each other's efforts. The PGPlanner 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.

4 Experiments

This section concentrates on experiments showing how different balances between predictability and responsiveness affect network performance in a few situations. These experiments employ environment A (Figure 1) and environment B (Figure 2), which involve four-node networks where node i is connected to sensor i. Environment A tests how well the PGPlanner 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 PGPlanner 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, a time-unit corresponds to the time needed to execute 1 knowledge source (KS). It takes 2 time-units for a message to get from one node to another.

Our experiments use two different meta-level organizations. In the *broadcast* meta-level organization, each node broadcasts its node-plans and develops PGPs based on local and received node-plans. When *centralized*, a single node (the node with the least data) is responsible for forming and distributing PGPs. In environment A, node 4 is the coordinator (nodes 1–3 send their node-plans to 4 which forms PGPs and sends them back to 1–3) while in environment B, node 1 is the coordinator.

For the four combinations of environments and meta-level organizations, we run three experiments: time-cushions of 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. We take four measurements in these experiments

Table 1: Experiment Summary.

	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 (d_1'-d_8'/d_1-d_8).
RT: The actual experimental runtime (in minutes).
H-r: Number of hypotheses communicated.
M-r: Number of meta-level messages (node-plans
and PGPs) communicated.
T-r: Total number of messages communicated.
Store: The total number of structures stored.

[Durfee, 1988]. 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, plans, and PGPs to roughly estimate the storage overhead of the planning mechanisms.

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 and goals.

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 node-plans) 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 and goals, while the coordination storage is essentially the same (since updated node-plans replace earlier versions). The same trends are seen with the centralized organization (E6–E8).

In environment B, similar 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 node-plans 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. Also, when a node does finally react to deviations in its local plans and updates its node-plans and PGPs, the exchange of the changed node-plans causes other nodes to change their plans, and these cause other nodes to further change, and so on. This chain-reaction increases the meta-level communication so that nodes communicate more despite the higher time-cushion (comparing E13 with E12).[2]

[2]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.

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 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.

5 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 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.

We have observed that there is no correct balance between responsiveness and predictability that is independent of the problem situation. Consequently, planning mechanisms for coordinating agents in dynamic domains must have the flexibility to strike different balances, and 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.

More generally, by explicitly representing planned actions and interactions, and by modeling themselves both from a local and more global standpoint, nodes can reason about how responses to dynamic situations can affect predicted network coordination. Partial global plans contain substantial information that can be used in making more complex decisions about different types of deviations and their significance. As nodes become capable of performing more complex reasoning about a variety of types of deviations, however, the overhead of deciding whether to respond to deviations could outweigh the costs of simply responding to all deviations. Meta-level control is needed to determine when various reasoning mechanisms are likely to be cost effective, and our future research will explore such control of control mechanisms. Our preliminary results show the importance of reasoning about deviations to balance predictability and responsiveness, and based on this experience and the possibilities that partial global planning provides us, we expect our future research to lead to even more sophisticated techniques for nodes to reason about the more global ramifications of their local responses in dynamic domains.

References

[Davis and Smith, 1983] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983.

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

[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, 1987] 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.

[Durfee and Lesser, 1988a] 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*, September 1988.

[Durfee and Lesser, 1988b] Edmund H. Durfee and Victor R. Lesser. Negotiation through partial global planning. In *Proceedings of the 1988 Distributed AI Workshop*, May 1988.

[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.