

# The Effects of Agent Topologies on Multiagent Planning Performance

Adam C. Langdon and Michael T. Cox

Department of Computer Science and Engineering

College of Engineering & Computer Science

Wright State University

Dayton, OH 45435-0001

{alangdon, mcox}@cs.wright.edu

## Abstract

In multiagent distributed planning systems, agents can act cooperatively toward a common goal. Using multiagent goal transformations, the planning process can be distributed over an appropriate number of autonomous agents. Each agent will attempt to plan its part of the problem using its share of the knowledge. We will show how the distribution of knowledge affects the planning performance of the system. The factors we will examine in this process include the number of agents used in the system, which agent will initiate the planning process, and what knowledge, if any, will be duplicated among the agents. We will use examples from the logistics transportation domain to demonstrate how these factors affect performance.

## 1 Introduction

The level of control in a multiagent system can vary from complete centralized control to a fully distributed system. With centralized control, knowledge about all of the agents and their actions is known. The planning process as a whole can be monitored, and errors in the system can be detected. For example, if an agent is performing unnecessary steps to solve a problem, a control mechanism can correct the agent's planning. Alternatively, a fully distributed multiagent system relies only on the knowledge of each autonomous agent. While requiring less processing overhead, greater effort must be taken in its design to ensure proper coordination between the agents. This design process includes the development of an agent topology, which determines the number of agents in the system, how knowledge is distributed among them, and which agent will initiate the planning process. Because an agent depends only on what it knows, the structure of this topology may significantly impact its performance.

Cox, Elahi, and Cleerman (2003) first demonstrated a fully distributed multiagent system with multiagent goal transformations as its coordination method. In this system, the planning process is divided among multiple agents using domain-specific topologies. Elahi (2003)

further investigated the performance of this system, showing how the number of agents affected performance. We will expound upon these findings and discuss how topological factors affect overall planning performance. Using experimental results, we will examine the impact of both the initiating agent and an instance of knowledge distribution overlap, in which two identical agents are used. We have used a planning and learning architecture called PRODIGY (Carbonell *et al.*, 1992; Veloso *et al.*, 1995) as the underlying framework, and we have used Prodigy/Agent (Cox *et al.*, 2001) as individual agents.

The paper is organized as follows: Section 2 describes the method used for multiagent coordination. Section 3 defines an agent topology and its important factors. Section 4 continues with an example that illustrates the approach. Section 5 discusses the experimental results. Section 6 focuses on some future works still to be done. Finally, section 7 has concluding remarks.

## 2 Multiagent coordination

Coordination among agents in distributed planning can be performed by goal transformations. If an agent cannot achieve a goal by itself, it can transfer the goal into one that is achievable through co-operation. A multiagent goal transformation is defined in (Cox *et al.*, 2001) as follows: to solve a goal  $G$ , solve instead a goal  $G'$  that generates a sub-solution, and then pass the remainder of the goal (i.e.,  $G$  minus  $G'$ ) to another agent. We distinguish between dynamic preconditions that may be adopted as subgoals because operators exist that modify the state, and applicability conditions (i.e., static preconditions) for which no operator of any agent exists to change the state. For example, no operator may exist to change the weather in a particular domain, yet some operators may require good weather for the operator to be applicable.

To achieve a particular goal state  $G$ , an agent must have an operator that can achieve the state. Now all open dynamic preconditions (those whose state is not true) are split into two sets. Set one contains states for which the agent has an operator, and set two contains states for which

it does not. The agent solves those preconditions in set one, and then it requests other agents to solve each state of set two. For example, a planner of a transportation domain might have two dynamic preconditions for the operator representing the action "Load Airplane P with object O at location L." One precondition requires that the object O be at the location L and the second precondition requires that the airplane P be at the same location. Suppose an agent A only has an operator to move objects but does not have an operator to fly airplanes. In this example, A will solve the first condition, and ask another agent to solve the second.

### 3 Topological Factors

A fully distributed multiagent system sacrifices the benefits of centralized control. Each agent acts independently without knowing the progress of the other agents. Therefore, the performance of this system will be affected by the properties of the agent topology and the properties of the planning problem. We have chosen to focus on the former and investigate how topological factors relate to performance.

The design of an agent topology can be difficult, and can generally be seen as a complex search problem in its own right (Durfee, 1999). To better examine this complexity, we will define an agent topology as the factors that make up its design. These three factors are the number of agents in the system, how knowledge is distributed among these agents, and what agent will initiate the planning process.

#### 3.1 Agent distribution

The first step in designing an agent topology is deciding how many agents will be used. With a smaller number of agents, each one possesses a large amount of knowledge about the planning process. While less communication will be needed between agents, each agent will be responsible for solving a larger part of the plan. When more agents are used, each agent only possesses a small amount of knowledge. Consequently, it will only be required to solve a small part of the problem, but more communication will be needed.

Because multiple agents can work in parallel, it would seem best to favor more agents within the system. However, a larger number of agents requires a larger amount of knowledge distribution. This may negatively impact performance, as we will discuss as part of the next topological factor.

#### 3.2 Knowledge distribution

Each agent possesses operators it can use to plan and must request the appropriate agent when it requires an operator it does not have. Therefore, the distribution of these

operators is the next important topological factor. Using the original problem domain, the operators are divided according to the number of agents in the system. The complexity of this process is heavily domain dependent, affected both by the size and physics of the domain. For example, given a domain with six operators, there would be 5 possible sets of agent topologies, ranging from a 2 agent set to a 6 agent set. Each set, in turn, would have many possible topologies depending on the distribution of the operators. Using a segregated approach, with no knowledge overlap, the number of topologies in a 2 agent set would be the combination of all operator distributions ( ${}_6C_5 + {}_6C_4 + {}_6C_3$ ), or 61 possibilities. The sum of all sets in the domain would then be 483 possible topologies. If a non-segregated approach were taken in which operators were overlapped among agents, the number of possibilities would increase further. It is difficult to determine an optimal topology without enumerating all of these possibilities. However, general heuristics can be developed by initially comparing the performance of various types of topologies.

**Goal loops.** Due to the physics of a domain, certain distributions of operators may negatively impact performance. One such issue is the occurrence of goal loops. These arise in distributed multiagent systems when operators that depend on each other are divided among different agents.

A goal loop occurs in a planning problem when a precondition of a goal produces a subgoal that is already pending. Given a goal G that can be achieved by an operator OP1 with a precondition P1, if P1 produces a subgoal G' that requires an operator OP2 having the precondition P2 = G, then the resulting situation is a goal loop, because the achievement of G is still pending. PRODIGY can detect the goal loop if it exists within an agent, but if the operators exist in different agents, the goal loop will not be detected. Therefore, the first step to avoiding goal loops is to place these codependent operators in the same agent domain. However, if the operators are fully segregated and distributed, goal loops cannot be avoided in this way.

**Agent Clones.** Another similar factor that may hinder performance arises when an agent requests another agent when a request to that agent is already pending. For example, agent A does not have the necessary operator to plan part of a problem. Therefore, it makes a request to agent B, who has the operator, to plan this part. However, during its planning, agent B requires another operator it doesn't have. It must then request agent A to plan part of its sub-problem. Finally, agent A must make a second request back to agent B to achieve this goal it has received. However, because it is still waiting for a response from its original request to agent B, it cannot make a second request. This situation, referred to as a back-to-back request, is disallowed in order to prevent infinite goal loops.

However, a back-to-back request not part of a goal loop may be required to solve certain problems. This can be

addressed by creating a clone of one of the agents. A cloned agent is a specialized type of knowledge overlap in which two agents overlap by 100%, possessing identical sets of operators. When an agent attempts to make a second request to the same agent, it can instead request its clone. Once it receives a plan from the clone, execution can continue as necessary.

The use of cloned agents can be an effective way to increase the performance of a system. However, if there are a large number of agents, creating a clone for each one may be prohibitive. The best approach would be to create a clone dynamically, as it is needed within the system. This, too, may be difficult, depending on the implementation of the system. The final option would then be to create clones for only those agents that have the potential to receive sequential requests. Again, this is dependent on the domain, and may only be determined by examining the performance of the system using various topologies.

### 3.3 Initiating agents

The last important factor in an agent topology is the agent that initiates the planning process. In the original examinations of this system (Cox, Elahi, and Cleereman 2003, Elahi 2003), this factor was not initially taken into account. In this experiment, the starting agent was chosen arbitrarily for each topology. It was later discovered that planning the same problems with different starting agents yielded varying results. The number of problems solved and the amount of search varied depending on which agent initiated the planning. These findings demonstrate the effect the starting agent has on overall planning performance.

To begin the planning process, the problem must be passed to one of the agents in the system. This agent will attempt to plan the entire problem on its own, requesting the help of other agents as needed. Depending on the problem, the initiating agent may be able to begin the planning process, or it may need to immediately request another agent. Furthermore, the starting agent may not even be necessary to solve the problem. The selection of the starting agent will therefore affect problem-solving performance and the number of problems solved.

Before knowing if an agent can plan for a goal, it must choose an operator to solve this goal. If the agent does not possess the operator needed to begin planning, it will request the agent that does. The current state and modified goal are then passed to the next agent. This state and goal will be identical to the initial state and goal of the problem. Although no progress has been made, the planner has already expanded nodes in its search for a solution.

The selection of the starting agent will also affect the number of back-to-back request failures. If the initiating agent can only solve a small portion of the plan, it must

pass the rest to another agent and planning will continue. The initiating agent must then wait until the rest of the plan is returned. Until this happens, the initiating agent cannot make a second request to that agent. Therefore, different starting agents will yield fewer failures than others and ultimately result in more problems solved than others.

## 4 The Logistics Domain

The logistics transportation domain (Veloso 1994) is a well-known benchmark domain in multiagent planning research. The basic elements of the domain are the infrastructure, the transportation units, and the packages. The infrastructure consists of a set of cities, and each city consists of a number of locations (e.g., post offices and airports). Trucks and airplanes are the transportation units: trucks can move freely within a city, but transport between cities can only be achieved by airplanes. The domain involves the movement of packages within cities by truck and between cities by airplane.

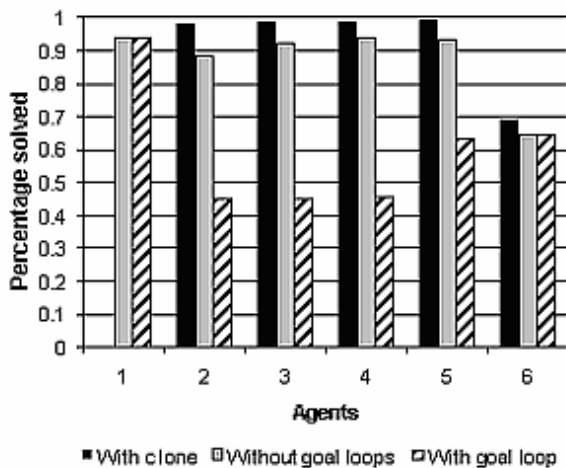
The logistics domain can be split into several subdomains to fit in our distributed planning approach. There are six operators in the logistics domain: LOAD-TRUCK, UNLOAD-TRUCK, LOAD-AIRPLANE, UNLOAD-AIRPLANE, DRIVE-TRUCK, and FLY-AIRPLANE. To split the domain each operator is assigned to the agent one of the agents. If a goal requires operators from different subdomains, multiple planning agents must cooperate. Suppose an agent handles the moving of packages by truck within a city, and another agent handles the moving of packages by airplane between cities. If the goal is to transfer a package from the post-office in city1 to the airport in city2, then neither agent can achieve the goal on its own. The first agent must move the package to the airport in city1 and find a second agent to move it from there to the airport in city2. This intuitive explanation is implemented as a multiagent goal transformation.

## 5 Experimental Results

Experiments were performed in the logistics domain to demonstrate how different agent topologies affect performance in multiagent distributed planning. Several topologies were created by splitting the domain into multiple agent subdomains, varying from 2 to 6 agents. When possible, all of the agents possess an equal number of operators, and except for the use of clone agents, overlapping was not used in our examples. One set of topologies includes operator distributions designed to prevent goal loops, while another set of topologies was designed to allow the occurrence of goal loops. In the case of the logistics domain, the load-truck and unload-truck operators are codependent. Therefore, we placed these operators in separate agents to create goal loops. We then created a third topology by adding a clone agent to the first topology.

In the logistics domain, a problem may require the movement of a package within two cities. While only one plane would be needed to fly between the cities, two trucks would be required. Therefore, we created a clone of the agent or agents possessing the load and unload truck operators. For agent topologies with 2 to 5 agents, these operators were paired together, requiring one clone, while in the 6-agent topology, two clones were required.

We used a set of 100 randomly generated planning problems to test the performance of each topology. All of the problems contained three cities and a maximum of three goals. Performance was first measured by calculating the percentage of problems solved for each agent topology. Figure 1 shows the results our three sets of agents. Each topology was run using each possible initiating agent. The results for each initiating agent were then averaged to obtain a measure for that topology. The single-agent system is also included in our results. Because it is not a distributed topology, back-to-back requests and infinite goal loops are not a factor. The number of problems it solved is listed under both the non-goal loop and goal loop topologies, and the use of a clone would not be necessary for a single agent. The 6-agent topology is also a special case because there is only one way to distribute the operators. Therefore, its results will be the same for both the non-goal loop and goal loop topologies.



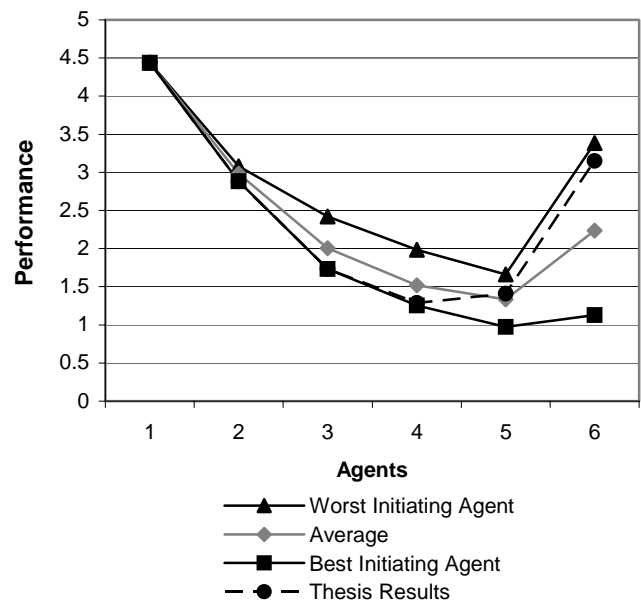
**Figure 1:** Percentage of problems solved by team topology

As we expected, the agent set with goal loops performed the worst, with only the 5 and 6 agent sets solving more than half the problems. The poor performance of the 2, 3, and 4 agent sets is due not only to the presence of goal loops, but also to back-to-back request failures. The further splitting of knowledge in the 5 and 6 agent sets reduces this type of failure. Because

there are more agents, each agent possesses fewer operators. This makes it less likely that multiple requests will be made to the same agent.

The agent set with no goal loops solved the vast majority of problems, with performance peaking for the 4 and 5 agent topologies. This is because certain starting agents resulted in more back-to-back request failures. As more agents were added, these failures were minimized. The addition of a cloned agent to this topology eliminated all back-to-back request failures, resulting in excellent performance for the 2 to 5 agent sets. The use of a clone also improved problem-solving performance for the 6-agent topology, but because operators cannot be paired within the same agent, goal loops cannot be prevented.

Performance was also measured by the amount of search required to solve the problem. The average number of nodes expanded per agent was calculated for each problem, except for any problems that could not be solved by the system. The number of nodes expanded was then divided by the number of steps in the generated solution. Finally, this value was divided by the percentage of problems solved to normalize the results. Fewer nodes expanded means less search, thus a lower score is better. This method of measuring performance gives equal weight to the amount of search and the number of problems solved.

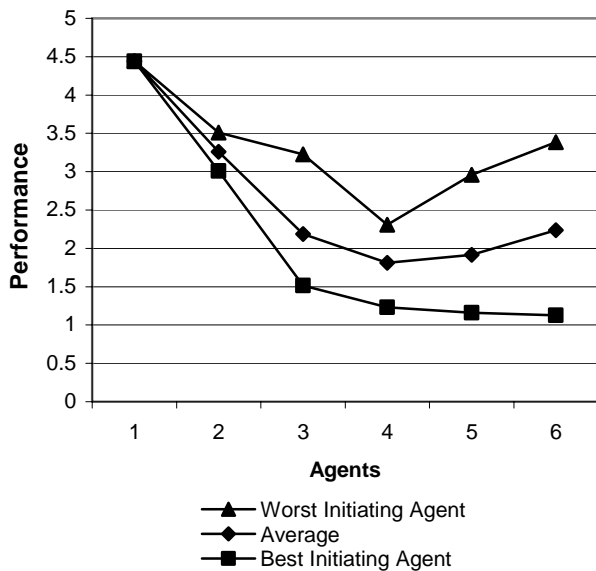


**Figure 2:** Planning performance as a function of team topology, non-goal-loop

Figure 2 shows the performance of topologies from 1 to 6 agents. These topologies were taken from the non-goal loop set, and compare performance using different initiating agents. The best and worst initiating agent for each is shown, as well as the average for all starting agents. We have also included the results compiled by Elahi (2003),

referred to as ‘Thesis Results’ in Figure 2. He employed a similar topology without goal loops, but chose the initiating agent arbitrarily. It can be seen that in all cases, the search performance improves as the number of agents increases, except for the fully distributed 6-agent topology. When the best starting agent is used, the decrease in performance for the 6-agent topology is minimized.

Finally, we compare the performance of the non-goal loop and goal loop topologies. Figure 3 shows the average planning performance when the results of both sets of topologies are combined. It also displays the combined results for the best and worst initiating agent for each topology. When both types of topologies are considered, the 4 and 5 agent sets performed the best over all possible initiating agents. If the best initiating agent is chosen, performance continues to improve slightly as the number of agents increases. The best agents in the goal loop topologies perform worse than those in the non-goal loop topologies, except for the 6-agent set, which is shared by both. Therefore, when the results are averaged, the 6-agent set yields the best performance. While there is an optimal performance that can be approached when using PRODIGY, there is no such upper bound. This is demonstrated in the results of the worst initiating agent. The occurrence of goal loops can yield inconsistent performance, shown most clearly in the 3-agent set.



**Figure 3:** Planning performance as a function of team topology, non-goal-loop and goal loop combined

Both Figure 2 and 3 demonstrate the impact of the initiating agent on planning. The selection of this agent must be considered if optimal planning is to be achieved. By examining the performance of each initiating agent on

a set of test problems, we can determine which agent in general will perform the best in the logistics domain.

## 6 Future Works

Using a relatively small set of topologies, we can see the effects their structures have on performance. To get a more complete idea of these effects, we would need to examine complete planning performance using all possible topologies within the logistics domain. This would require developing an automated method to split a domain into distributed domains. Furthermore, we would like to continue this work on larger domains, e.g. Extended STRIPS.

While topologies can be created that minimize the occurrence of goal loops, techniques need to be developed to better detect and correct for these occurrences. For example, a domain-independent method could be developed to determine what operators have the potential to create goal loops. When goal loops cannot be avoided, they would need to be excised. This would yield more relevant results for topologies in which goal loops prevent problems from being solved, such as a 6-agent topology in the logistics domain.

With more experimental results, we would also like to begin formulating procedures for designing optimal topologies for the domain examined. These procedures could then be generalized in a domain-independent manner. By examining the domain, the best topology could be selected without having to test its performance first.

## 7 Conclusions

In multiagent distributed planning, agents act autonomously according to the knowledge they possess. Structuring these agents and distributing knowledge among them is therefore crucial to the system’s overall performance. We have discussed an approach used to coordinate and carry out distributed planning using multiple agents. Next, we have defined an agent topology to consist of three major factors: the number of agents in the system, the distribution of planning operators among them, and the agent initiating the planning. We then examined each factor and discussed how it affects planning. Finally, we have used an example to demonstrate empirically the impact the agent topology will have on performance.

## Acknowledgements

This paper is supported by a grant from the Information Technology Research Institute (ITRI) at Wright State University. We also acknowledge Sylvia George for her feedback on our work.

## References

Carbonell, J. G.; Blythe, J.; Etzioni, O.; Gil, Y.; Joseph, R.; Kahn, D.; Knoblock, C.; Minton, S.; Perez, A.; Reilly, S.; Veloso, M. M.; and Wang, X. 1992. *PRODIGY4.0: The Manual and Tutorial*, Technical Report, Computer Science Department, Carnegie Mellon University.

Cox, M. T.; Edwin, G.; Balasubramanian, K.; and Elahi, M. M. 2001. Multiagent Goal Transformation and Mixed-Initiative Planning Using Prodigy/Agent. In *Proceedings of the Fifth International Conference on Information, Systems, and Cybernetics*, Florida.

Cox, M. T.; Elahi, M.; and Cleereman, K. 2003. A distributed planning approach using multiagent goal transformations. In A. Ralescu (Ed.), *Proceedings of the 14th Midwest Artificial Intelligence and Cognitive Science Conference*, 18-23. Cincinnati: Omnipress.

Cox, M. T.; and Veloso, M. M. 1998. Goal Transformations in Continuous Planning. In M. desJardins (Ed.), *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning*, 23-30. Menlo Park, Calif.: AAAI Press / The MIT Press.

Durfee, E. H. 1999. Distributed Problem Solving and Planning. In G. Weiss (Ed.), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, 121-164. Cambridge, Mass: The MIT Press.

Elahi, M. 2003. A Distributed Planning Approach Using Multiagent Goal Transformations, M.S. thesis, Department of Computer Science and Engineering, Wright State University