Composing POMDP-based Building Blocks to Analyze Large-scale Multiagent Systems

Hyuckchul Jung, Milind Tambe
Computer Science Department, University of Southern California
941 W. 37th Place, Los Angeles, CA 90089-0781, USA
{jungh,tambe}@usc.edu

Abstract

Given a large group of cooperative agents, selecting the right coordination or conflict resolution strategy can have a significant impact on their performance (e.g., speed of convergence). While performance models of such coordination or conflict resolution strategies could aid in selecting the right strategy for a given domain, such models remain largely uninvestigated in the multiagent literature. This paper takes a step towards applying the recently emerging distributed POMDP (partially observable markov decision process) frameworks, such as the MTDP (markov team decision process) in service of creating such performance models.

To address issues of scale-up, we use small-scale models, called *building blocks* that represent the local interaction among a small group of agents. We discuss several ways to combine building blocks for performance prediction of a larger-scale multiagent system. Our approach is presented in the context of DCSP (distributed constraint satisfaction problem), where we are able to predict the performance of five different DCSP strategies in different domain settings by modeling and combining *building blocks*. Our approach points the way to new tools based on building blocks for performance analysis in multiagent systems.

Introduction

In many large-scale applications such as distributed sensor nets, distributed spacecraft and disaster response simulations, collaborative agents must coordinate their actions or resolve conflicts over shared resources or task assignments (Jung, Tambe, & Kulkarni 2001; Modi *et al.* 2001; Jung *et al.* 2002; Prasad & Lesser 1997). Selecting the right coordination or conflict resolution strategy can have a significant impact on performance of such agent communities. For instance, in distributed sensor nets, tracking targets quickly requires that agents controlling different sensors adopt the right strategy to resolve conflicts regarding shared sensors.

Unfortunately, selecting the right coordination strategy is difficult. First, there are often a wide diversity of strategies available, and they can lead to significant variations in the rate of convergence of the multiagent systems. For instance, when distributed agents must resolve conflicts over shared resources such as shared sensors, they could select among strategies that offer maximum possible resources to most constrained agents, or distribute resources equally among all agents requiring such resources, and so on. Each strategy may create significant variations in the rate of conflict resolution convergence(Jung, Tambe, & Kulkarni 2001; Prasad & Lesser 1997).

Performance modeling of multiagent coordination and conflict resolution could help predict the right strategy to adopt in a given domain. Unfortunately, performance modeling has not received significant attention in mainstream multiagent research community; although within subcommunities such as mobile agents, performance modeling has received significant attention(Rana 2000). Fortunately, recent research in distributed POMDPs (partially observable markov decision processes) and MDPs has begun to provide key tools to aid multiagent researchers in modeling the performance of multiagent systems(Pynadath & Tambe 2002; Bernstein, Zilberstein, & Immerman 2000; Xuan & Lesser 2002). In the context of this paper, we will use the MTDP (Markov Team Decision Process) model(Pynadath & Tambe 2002) for performance modeling, although other distributed POMDP models could be used.

There are at least two major problems in applying such distributed POMDP models. First, while previous work has focused on modeling communication strategies within very small numbers of agents(Pynadath & Tambe 2002), we are interested in strategy performance analysis for a large-scale multiagent systems. Second, techniques to apply such models to investigate other types of coordination strategies and conflict resolution strategies have not been investigated.

We address these limitations in the context of distributed constraint satisfaction problems, which is a major paradigm of research on conflict resolution (Yokoo 2001; Silaghi, Sam-Haroud, & Faltings 2000; Hamadi, Bessière, & Quinqueton 1998; Zhang & Wittenburg 2002). Our contribution in this paper is to illustrate the use of MTDP to model the performance of different DCSP strategies to select the right strategy. First, we illustrate how DCSP strategies can be modeled in MTDP. Second, to address scale-up issues in the MTDP models, we introduce small-scale models called "building blocks" that represent the local interaction among a small group of agents. We discuss several ways of building block composition, and the initial result indicates a promis-

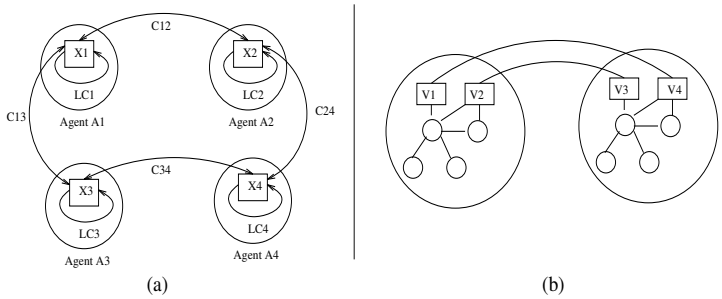


Figure 1: Model of agents in DCSP

ing direction for performance analysis in highly complex multiagent systems.

DCSP Strategies

DCSP (Distributed Constraint Satisfaction Problem) techniques have been used for coordination and conflict resolution in many multiagent applications such as distributed sensor network (Modi *et al.* 2001). In this section, we introduce DCSP and efficient DCSP strategies.

Distributed Constraint Satisfaction Problem (DCSP)

A Constraint Satisfaction Problem (CSP) is commonly defined by a set of n variables, $X = \{x_1, ..., x_n\}$, each element associated with value domains $D_1, ..., D_n$ respectively, and a set of k constraints, $\Gamma = \{C_1, ..., C_k\}$. A solution in CSP is the value assignment for the variables which satisfies all the constraints in Γ. A distributed CSP is a CSP in which variables and constraints are distributed among multiple agents(Yokoo & Hirayama 1998). Formally, there is a set of m agents, $Ag = \{A_1, ..., A_m\}$. Each variable (x_i) belongs to an agent A_j. There are two types of constraints based on whether variables in the constraint belong to a single agent or not:

- For a constraint $C_r \in \Gamma$, if all the variables in C_r belong to a single agent $A_j \in Ag$, it is called a *local constraint*.
- For a constraint $C_r \in \Gamma$, if variables in C_r belong to different agents in Ag, it is called an *external constraint*.

Figure 1-a illustrates an example of a DCSP: each agent A_i (denoted by a big circle) has a local constraint LC_i and there is an external constraint C_{ij} between A_i and A_j. As illustrated in Figure 1-b, each agent can have multiple variables. There is no limitation on the number of local/external constraints for each agent. Solving a DCSP requires that agents not only satisfy their local constraints, but also communicate with other agents to satisfy external constraints.

Asynchronous Weak Commitment Search Algorithm (AWC): AWC is a sound and complete algorithm which shows the best performance among the published DSCP algorithms(Yokoo 1995; Yokoo & Hirayama 1998). In the AWC approach, agents asynchronously assign values to their variables from available domains, and communicating the values to neighboring agents. Each variable has a non-negative integer priority that changes dynamically during search. A variable is consistent if its value does not violate any constraints with higher priority variables. A solution is a value assignment in which every variable is consistent.

For simplification, suppose that each agent has exactly one variable and constraints between variables are binary. When the value of a variable is not consistent with the values of neighboring agents' variables, there can be two cases: (i) *good* case where there exists a consistent value in the variable's domain; (ii) *nogood* case where there is no consistent value in the variable's domain. In the nogood case, an agent increases its priority to *max+1*, where *max* is the highest priority of neighboring agents' variables. This priority increase makes previously higher agents select new values to satisfy the constraint with the new higher agent.

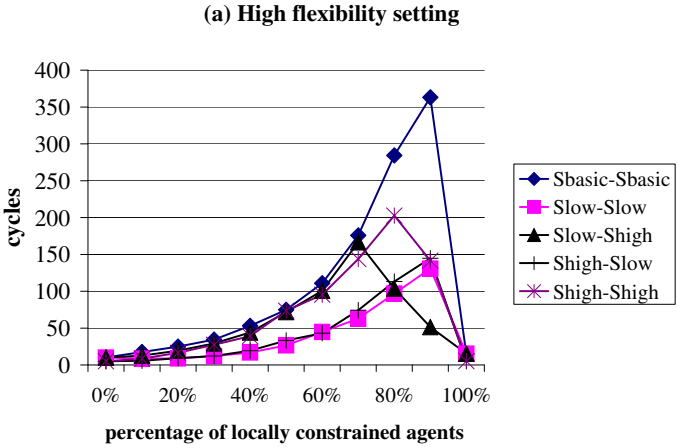
DCSP Value Selection Strategies in AWC Framework: While AWC relies on the *min-conflict* value selection strategy(Minton *et al.* 1990) that minimizes the number of conflicts with other agents, new novel value selection strategies were introduced based on *local cooperativeness*(Jung, Tambe, & Kulkarni 2001) (*local cooperativeness* measures how many value choices are given towards neighbors by a selected value):

- S_{low}: Each agent selects a new value from its consistent values maximizing the sum of compatible values with its *lower* priority neighbor agents.
- S_{high}: Each agent selects a new value from its consistent values maximizing the sum of compatible values with its *higher* priority neighbor agents.
- S_{all}: Each agent selects a new value from its consistent values maximizing the sum of compatible values with *all* neighbor agents.

Note that a value is consistent if the value is compatible with the values of higher priority agents. In the *nogood* case, since an agent increases its priority, every value in its domain is consistent. The three strategies above and the original *min-conflict* strategy (henceforth, referred as S_{basic}) can be applied to the *good* and the *nogood* case. Therefore, there are 16 strategy combinations such as $S_{low} - S_{high}$ (S_{low} is applied in the *good* case and S_{high} is applied to the *nogood* case). Note that, in the nogood case, the higher and lower agents are grouped based on the previous priority before the increase. Since we will consider only strategy combinations, henceforth, they are referred as strategies for short. While agents' strategies can be heterogeneous, for simplification, we assume that every agent applies the same strategy. Performance evaluation with heterogeneous strategies will be considered in our future work.

Experimental Evaluation

To provide the initial evaluation of these strategies, a number of DCSP experiments were done with an abstract problem setting. Here, agents (variables) are in a 2D grid configuration (each agent is externally constrained with four neighbors except for the ones on the grid boundary). All agents share an identical binary constraint by which a value in an agent is not compatible with a set of values in its neighboring



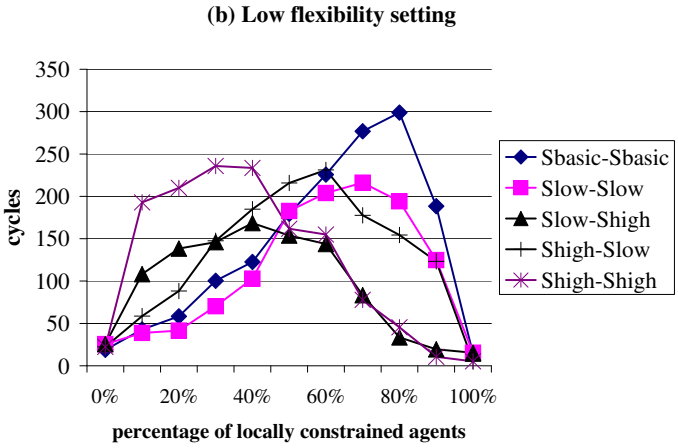


Figure 2: Performance prediction in the case of 90% locally constrained agents

agent. This grid configuration is motivated by the research on distributed sensor network (Modi *et al.* 2001) where multiple agents must collaborate to track targets.

To test the performance of strategies in various problem settings, we make a variation to the problem setting used in (Jung, Tambe, & Kulkarni 2001) by modifying the external constraint: in the new setting, each agent gives less flexibility (choice of values) towards neighboring agents given a value in its domain. Henceforth, the previous setting in (Jung, Tambe, & Kulkarni 2001) is referred as *high flexibility setting*, and the new setting with less choice of values to neighbors is referred as *low flexibility setting*.

Experiments were performed in the two problem settings (*high flexibility setting* and *low flexibility setting*). Other than this external constraint variation, the parameters for the experiments remain same. Each data point in the figures was averaged over 500 test runs. While all the possible strategies for each flexibility base were tried, for expository purpose, only five strategies are presented in Figure 2, which does not change our conclusion. The evaluation followed the method

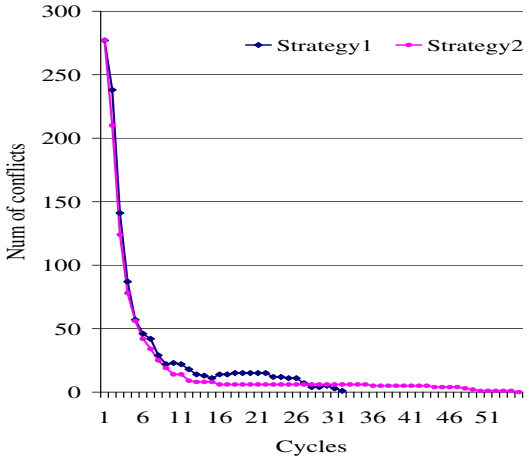


Figure 3: Convergence rate in DCSP with long-tail distribution

used in (Yokoo & Hirayama 1998). In particular, performance evaluation is measured in terms of cycles consumed until a solution is found, and the experiments ranged over all possible strategies. The vertical axis plots the number of cycles and the horizontal axis plots the percentage of locally constrained agents. Having a local constraint, an agent's available domain is reduced. The performance difference between different strategies are proved to be statistically significant by performing t-test.

The experimental results in Figure 2 show that the value selection strategies described above have a great impact on conflict resolution convergence in solving DCSPs. Certainly, choosing $S_{basic} - S_{basic}$ may lead to significantly slower convergence rate while appropriately choosing $S_{low} - S_{low}$ or $S_{low} - S_{high}$ can lead to significant improvements in convergence. However, there was no universal best strategy for different problem domains. To gain maximum efficiency, it would be essential to predict the right strategy to use in a given domain.

Long tail distribution in Convergence

The key factor in determining the performance of strategies is in the long tail where only a small number of agents are in conflicts. Figure 3 shows the number of conflicts at each cycle for two different strategies. In the beginning of conflict resolution, both strategies shows similar performance in resolving conflicts. However, the difference in performance appears in the long tail part. While strategy 1 quickly solves a problem, strategy 2 has a long tail with a small number of conflicts remaining unresolved. This type of long tail distribution has been also reported in many constraint satisfaction problems (Gomes *et al.* 2000).

Performance Analysis

The previous section shows that, given the dynamic nature of multiagent systems, predicting the right strategy to use in a given domain is essential to maximize the speedup of

conflict resolution convergence, and the critical factor for strategy performance is in the long tail part where a small number of conflicts exist. In this section, we provide a formal model for performance analysis and mapping of DCSP onto the model, and present the results of performance prediction.

Distributed POMDP-based Model

As a formal framework for strategy performance analysis, we use a distributed POMDP model called MTDP (Multiagent Team Decision Process) model (Pynadath & Tambe 2002). The MTDP model has been proposed as a framework for teamwork analysis. Distributed POMDP-based model is an appropriate formal framework to model strategy performance in DCSPs since it has distributed agents and the agentView in DCSPs (other agents' values, priorities, etc.) can be modeled as observations. In DCSPs, the exact state of the system is only partially observable to an agent since the received information for the agent is limited to its neighboring agents. Therefore, there is strong correspondence between DCSPs and distributed POMDPs. While we focus on the MTDP model in this paper, other distributed POMDP models such as DEC-POMDP (Bernstein, Zilberstein, & Immerman 2000) could be used.

Here, we illustrate the actual use of MTDP in analyzing DCSP performance. MTDP provides a tool for varying key domain parameters to compare the performance of different DCSP strategies, and thus select the most appropriate strategy in a given situation. We first briefly introduce the MTDP model. Refer to (Pynadath & Tambe 2002) for more details.

MTDP model The MTDP model involves a team of agents operating over a set of world states during a sequence of discrete instances. At each instant, each agent chooses an action to perform and the actions are combined to affect a transition to the next instance's world state. Borrowing from distributed POMDPs, the current state is not fully observed/known and transitions to new world states are probabilistic. Each agent makes its own observations to compute its own beliefs, and the performance of the team is evaluated based on a joint reward function over world states and combined actions.

More formally, an MTDP for a team of agents, α, is a tuple, $< S, A_\alpha, P, \Omega_\alpha, O_\alpha, B_\alpha, R >$. S is a set of world states. $A_\alpha = \prod_{i \in \alpha} A_i$ is a set of combined actions where A_i is the set of agent i's actions. P controls the effect of agents' actions in a dynamic environment: $P(s, a, s') = Pr(S^{t+1} = s' | S^t = s, A^t_\alpha = a)$. $R : S \times A_\alpha \rightarrow \Re$ is a reward function over states and joint actions. Here, S, A_α, P, and R are the most relevant aspects of the model for this paper: while belief states B_α, Observations Ω_α, and observation function O_α (which defines the probability distribution of possible observations for an agent i) are key parts of the model, they are not as relevant here. A policy in the MTDP model maps individual agents' belief states to actions; the combination of individual policies thus forms a joint policy for the MTDP. A DCSP strategy is mapped onto a policy in the model. Thus, we compare strategies by evaluating policies in this model. Our initial results from policy evaluation in this model match the actual experimental strategy performance results shown in Figure 2. Thus, the model could potentially form a basis for predicting strategy performance in a given domain.

Mapping from DCSP to MTDP In a general mapping, the first question is selecting the right state representation for the MTDP. One typical state representation could be a vector whose elements are the values of all the variables in a DCSP. However, this representation leads to a huge state space. For instance, if there are 10 variables (agents) and 10 possible values per variable, the number of states is 10^{10}. To avoid this combinatorial explosion in state space, we use an abstract state representation in the MTDP. In particular, as described in the previous section, each agent can be abstractly characterized as being in a *good* or *nogood* state in the AWC algorithm. We use this abstract characterization in our MTDP model. Henceforth, the *good* and *nogood* state are denoted by G and N respectively. Here, an initial state of the MTDP is a state where all the agents are in G state since, in the AWC, an agent finds no inconsistency for its initial values until it receives the values of its neighboring agents. Note that, for simplicity, the case where agents have no violation is not considered.

In this mapping, the reward function R is considered as a cost function. The joint reward (cost) is proportional to the number of agents in the N state. This reward is used for strategy evaluation based on the fact that the better performing strategy has less chance of forcing neighboring agents into the N state than other strategies: as a DCSP strategy performs worse in a given problem setting, more agents will be in the N states.

In the AWC algorithm (a base DCSP algorithm in this paper), each agent receives observations only about the states of its neighboring agents. Initially, we assume that these observations are perfect (without message loss in communication). This assumption can be released in a future work where agents' communication is not reliable.

Here, S_{low}, S_{high}, S_{all}, and $min - conflict$ in DCSPs are mapped onto the actions for agents in the MTDP model. A DCSP strategy such as $S_{low} - S_{high}$ is akin to a policy in MTDP: S_{low} in the *good* state and S_{high} in the *nogood* state. The state transition in the MTDP model is controlled by an agent's own action as well as its neighboring agents' actions. The transition probabilities can be derived from the simulation on DCSPs.

Building Block While the abstract representation in the mapping above can reduce the problem space, for a large-scale multiagent system, if we were to model belief states of each agent regarding the state of the entire system, the problem space would be enormous even with the abstract representation. For instance, the number of states in the MTDP for the system with 512 agents would be 2^{512}: each agent can be either G or N. To further reduce the combinatorial explosion, we use small-scale models, called *building blocks*. Each building block represents the local situation among five agents in the 2D grid configuration.

In the problem domains for the experiments shown in Figure 2, each agent's local situation depends on whether it has a unary local constraint or not: each agents can be either

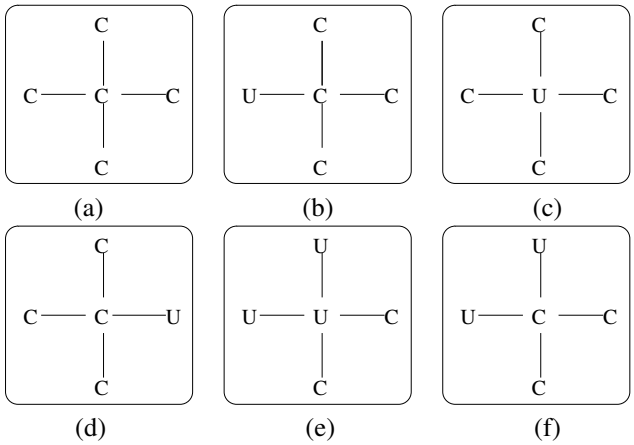


Figure 4: Building block

constrained (a portion of its original domain values are not allowed) under a local constraint (C) or unconstrained (U). Figure 4 illustrates some exemplar building blocks for the domain used in the experiments. For instance, Figure 4-a represents a local situation where all the five agents are constrained (C) while Figure 4-b represents a local situation where an agent in the left side is unconstrained (U) but the other four agents are locally constrained (C).

Note that, when the percentage of locally constrained agents is high, most of building blocks would be the one shown in Figure 4-a and a small portion of building blocks would be like the one shown in Figure 4-b, 4-c, or 4-d. As the percentage of locally constrained agents decreases, more building blocks include unconstrained agents (U) as shown in Figure 4-e and 4-f.

In each building block, as shown in the Figure 5, a middle agent (A_3) is surrounded by the other four neighboring agents (A_1, A_2, A_4, A_5). Thus, the state of a building block can be represented as a tuple of local states $<s_1, s_2, s_3, s_4, s_5>$ (e.g., $<G, G, G, G, G>$ if all the five agents are in *good* (G) state). There are totally 32 states in a building block of the MTDP model (e.g., $<G, G, G, G, G>$, $<G, G, G, G, N>$, $<G, G, G, N, G>$, etc), and the initial state of a building block is $<G, G, G, G, G>$. Here, agents' value selection actions will cause a transition from one state to another. For instance, if agents are in a state $<G, G, G, G, G>$ (Figure 5-a) and all the agents choose the action S_{high}, there is a certain transition probability that the next state will be $<G, G, N, G, G>$ (Figure 5-b) when only the third agent is forced into a *nogood* (N) state. However, the agents may also transition to $<G, G, G, N, G>$ (Figure 5-c) if only the fourth agent enters into N state.

One may argue that these small-scale models are not sufficient for the performance analysis of the whole system since they represent only local situations. However, as seen in Figure 3, the key factor in determining the performance of strategies is in the long tail where only a small number of agents are in conflicts. Therefore, the performance of strategies are strongly related to the local situation where a conflict may or may not be resolved depending on the local agents' actions. That is, without using a model for the whole system, small scale models for local interaction can be sufficient for performance analysis. Furthermore, while this simple model may appear limiting at first glance, it has already shown promising results presented below.

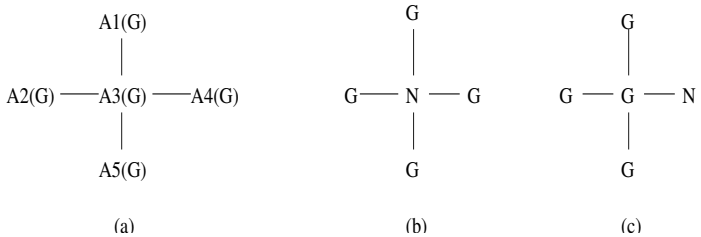


Figure 5: States in a building block

Building Block Composition for Performance Analysis While the building blocks are the basis for performance analysis, we need to deal with multiple building blocks that can exist in a given domain. Different building blocks have an impact on conflict resolution convergence depending on which actions are selected by the agents in the building block. It is expected that the the local interaction in a single building block does not determine the performance of strategies but the interactions which occurs between building blocks have a great impact on the performance of strategies. Here, we propose four methods of combining building blocks to evaluate MTDP policies (to which DCSP strategies are mapped on) as follows:

- **Single block**: for a given domain, a single building block is selected. The selected block has the highest probability of producing *nogood* (N) case within the building block. The performance of strategies are evaluated based on the value of an initial state of the building block ($<G, G, G, G, G>$) given a policy: the lower initial state value, the better policy (strategy).
- **Simple sum**: for a given domain which has multiple building blocks, we compute the value of initial state for each building block. Performance evaluation of a policy is based on the summation of the initial states' values.
- **Weighted sum**: given multiple building blocks, for each building block, we compute the ratio of the building block in the domain and the value of its initial state. Performance evaluation of a policy is based on the weighted sum of the initial states' values where the weight is the ratio of a building block.
- **Interaction**: a sequence of building blocks is considered since the performance difference comes from the long tail part (shown in Figure 3), and their interaction is taken into account: an agent may force its neighboring agents to enter into *nogood* (N) state given an action under a policy to evaluate. For instance, the initial state of a building block is affected by the state of its neighboring blocks.

For the *interaction* method, we don't have arbitrary degrees of freedom: for different domains, there can be commonalities in building blocks. That is, for common building

blocks, the same transition probabilities within a building block are applied. As we change from the first method (*single block*) to the fourth method (*interaction*), we gradually increase the complexity of composition. The accuracy of performance prediction of these methods are presented in the next section.

Here, note that the focus of our building block composition is not on computing the optimal policy but it remains on matching the long-tailed phenomenon shown in Figure 3. Thus, our interactions essentially imply that neighboring blocks affect each other in terms of the values of the policies being evaluated, but we are not computing optimal policies that cross building block boundaries. More details will be available in our forthcoming paper in AAMAS 03(Jung & Tambe 2003).

Performance Prediction

To check whether the MTDP based model can effectively predict the performance of strategies, the performance evaluation results from the MTDP model (based on the four composition methods described above) are compared with the real experimental results presented in the previous section. Among the possible strategies defined in the background section, we focus on the following strategies that were selected for expository purpose: $S_{low} - S_{low}$, $S_{low} - S_{high}$, $S_{high} - S_{low}$, $S_{high} - S_{high}$, and $S_{basic} - S_{basic}$.

The Figure 6-(a) (top) shows the experimental results in the case where 90% agents are locally constrained, and the Figure 6-(b) (bottom) shows the performance evaluation of the case from the MTDP model using the *interaction* composition method. In the performance evaluation, the lower value means the better performance (indicating less number of cycles will be taken until a solution is found). Figure 6-(a) and (b) show that the performance prediction results match to the real experimental results in the problem setting considered here: the patterns of column graphs in the Figure 6-(a) and (b) correspond to each other. That is, the strategies ($S_{low} - S_{high}$ and $S_{high} - S_{high}$) that showed better performance in the experiments are expected to perform better than the others according to the performance prediction. Here, the current goal of performance prediction is mainly to predict the best performing strategy, not to predict the magnitude of speedup difference among the given strategies.

Here, a question remains to be answered for whether the other building block composition methods can provide the same power of performance prediction as the *interaction* method does? Figure 7 shows the performance prediction results with different composition methods for the same problem domain used in Figure 6. The rightmost part is for the *interaction* method shown in Figure 6-(b). It is shown that the result from the *single block* method does not match at all. While the *simple sum* or *weighted sum* method provides a little improvement over the *single block*, they are far from matching to the real experimental results shown in Figure 6-(a). That is, the composition method taking the interaction between building blocks into account shows the best performance prediction results.

Figure 8 also shows the performance analysis results in a different domain using the *interaction* method of building

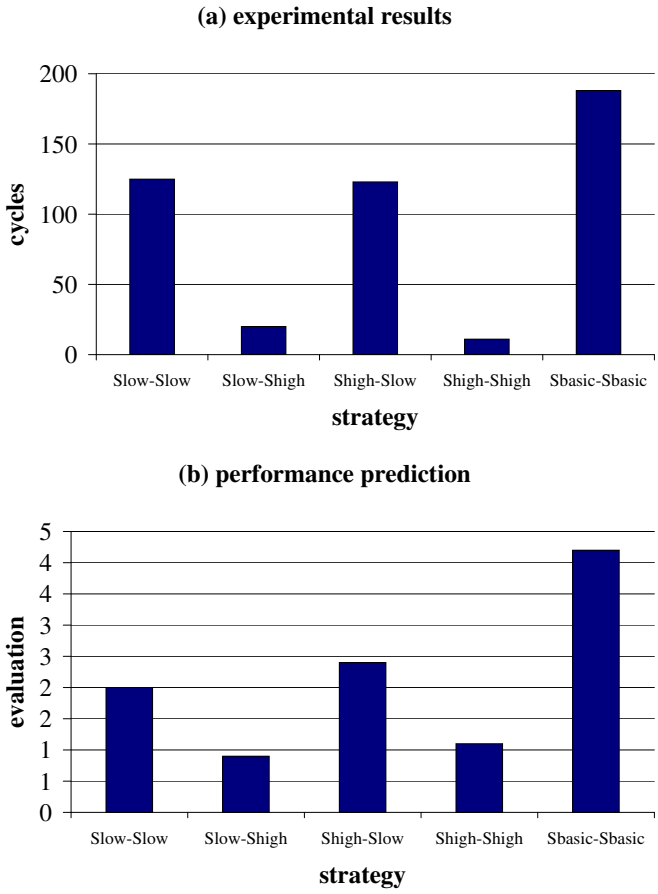


Figure 6: Performance prediction in the low flexibility setting with 90% locally constrained agents

block composition. Note that the composition methods other than the *interaction* method does not provide matching results. Here, while the performance prediction results do not perfectly match to the experimental results, certainly the performance analysis can distinguish better performing strategies from worse performing strategies. This result illustrates that the MTDP model can be used to predict the right strategy to apply in a given situation (possibly with less computation overhead). That is, given a new domain, agents can analyze different strategies with the simple MTDP model, and select the right strategy for the new domain without running a significant number of problem instances for each strategy to evaluate. Furthermore, this approach will enable agents to flexibly adapt their strategies to changing circumstances.

This result indicates a promising direction for performance analysis in DCSP, and potentially other multiagent systems. More generally, the formal model of building blocks and the building block composition methods in this paper begin to provide domain independent techniques which enable automatic abstraction of system modules and bottom up module composition methods for performance analysis in highly complex multiagent systems.

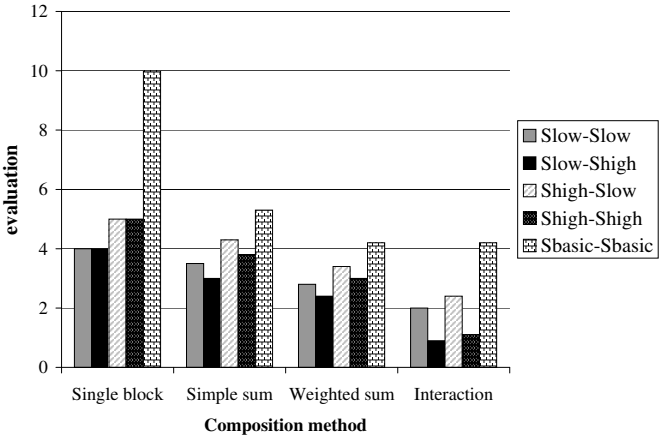


Figure 7: Comparison of composition methods

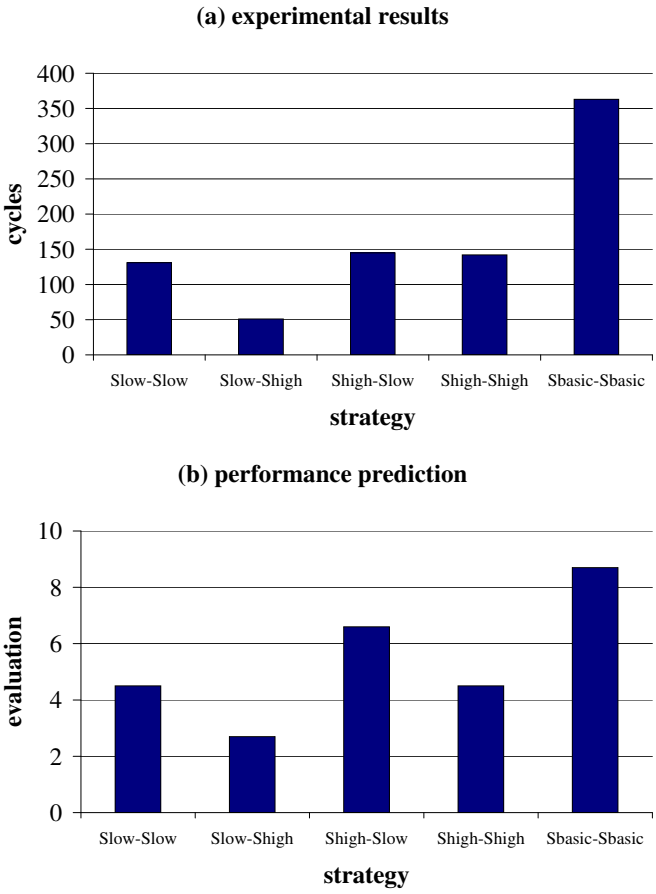


Figure 8: Performance prediction in the high flexibility setting with 90% locally constrained agents

Related Work and Conclusion

In terms of related work, significant work in multiagent learning is focused on learning to select the right coordination strategy(Prasad & Lesser 1997; Excelente-Toledo & Jennings 2002). While this goal is related to our goal of choosing the right strategy, one key difference is that the learning work focuses on enabling each agent to select a strategy. Instead, our focus is on a complementary goal of trying to predict the overall performance of the entire multiagent system assuming homogeneous conflict resolution strategies.

In centralized CSPs, performance prediction for different heuristics has been investigated. However, their method is based on the estimation of nodes to expand for search(Lobjois & Lemaitre 1998). However, this approach is not applicable to DCSP since multiple agents simultaneously investigate search space. Theoretic investigations of heuristic performance were also done in centralized CSPs (Minton *et al.* 1990; Musick & Russell 1992). However, no theoretical investigation has been done for performance prediction in DCSPs.

While there are related works in composition methods of subproblem solutions in MDPs(Dean & Lin 1995; Parr 1998; Hauskrecht *et al.* 1998; Guestrin, Koller, & Parr 2001) and in POMDPs(Pineau, Roy, & Thrun 2001), little have been investigated in distributed POMDPs which our performance models are based on. Furthermore, we are really interested in applying these composition techniques for performance modeling, not computing an optimal policy. For instance, our techniques are heavily influenced by the need to capture the long-tailed phenomena in conflict resolution.

To conclude, in this paper, the recently emerging distributed POMDP frameworks such as the MTDP model were used to create a model for the performance analysis of coordination or conflict resolution strategies. To address issues of scale-up, we used small-scale models, called *building blocks* that represent the local interaction among a small group of agents. We discussed several ways to combine the building blocks for the performance prediction of larger-scale multiagent systems.

These approaches were presented in the context of DCSPs (Distributed Constraint Satisfaction Problems), where we first showed that there is a large bank of conflict resolution strategies and no strategy dominates all others across different domains. By modeling and combining the *building blocks*, we were able to predict the performance of DCSP strategies for different domain settings, for a large-scale multiagent system. Thus, our approach points the way to new tools for the strategy analysis and performance modeling in large scale multiagent systems.

Acknowledgement

The research in this paper was funded by NASA Jet Propulsion Laboratory subcontract "Continual Coherent Team Planning". We thank Tony Barrett for valuable discussions and input.

References

Bernstein, D. S.; Zilberstein, S.; and Immerman, N. 2000. The complexity of decentralized control of mdps. In *Pro-*

ceedings of the International Conference on Uncertainty in Artificial Intelligence.

Dean, T., and Lin, S. 1995. Decomposition techniques for planning in stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Excelente-Toledo, C., and Jennings, N. 2002. Learning to select a coordination mechanism. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*.

Gomes, C.; Selman, B.; Crato, N.; and Kautz, H. 2000. Heavy-tailed phenomenon in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning* 24.

Guestrin, C.; Koller, D.; and Parr, R. 2001. Multiagent planning with factored MDPs. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*.

Hamadi, Y.; Bessière, C.; and Quinqueton, J. 1998. Backtracking in distributed constraint networks. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*.

Hauskrecht, M.; Meuleau, N.; Kaelbling, L. P.; Dean, T.; and Boutilier, C. 1998. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*.

Jung, H., and Tambe, M. 2003. Performance models for large scale multiagent systems: Using pomdp building blocks. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (to appear)*.

Jung, H.; Tambe, M.; Barrett, A.; and Clement, B. 2002. Enabling efficient conflict resolution in multiple spacecraft mission via dcsp. In *Proceedings of the NASA Planning and Scheduling Workshop*.

Jung, H.; Tambe, M.; and Kulkarni, S. 2001. Argumentation as distributed constraint satisfaction: Applications and results. In *Proceedings of the International Conference on Autonomous Agents*.

Lobjois, L., and Lemaitre, M. 1998. Branch and bound algorithm selection by performance prediction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*.

Minton, S.; Johnston, M. D.; Philips, A.; and Laird, P. 1990. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the National Conference on Artificial Intelligence*.

Modi, P.; Jung, H.; Tambe, M.; Shen, W.; and Kulkarni, S. 2001. A dynamic distributed constraint satisfaction approach to resource allocation. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*.

Musick, R., and Russell, S. 1992. How long it will take? In *Proceedings of the National Conference on Artificial Intelligence*.

Parr, R. 1998. Flexibile decomposition algorithms for weakly coupled Markov decision problems. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*.

Pineau, J.; Roy, N.; and Thrun, S. 2001. A hierarchical approach to POMDP planning and execution. In *Proceedings of the ICML Workshop on Hierarchy and Memory in Reinforcement Learning*.

Prasad, N., and Lesser, V. 1997. The use of meta-level information in learning situation-specific coordination. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Pynadath, D., and Tambe, M. 2002. The communicative multiagent team decision problem: analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*.

Rana, O. 2000. Performance management of mobile agent systems. In *Proceedings of the International Conference on Autonomous Agents*.

Silaghi, M.-C.; Sam-Haroud, D.; and Faltings, B. V. 2000. Asynchronous search with aggregations. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*.

Xuan, P., and Lesser, V. 2002. Multi-agent policies: from centralized ones to decentralized ones. In *Proceedings of the International Conference on Autonomous Agents*.

Yokoo, M., and Hirayama, K. 1998. Distributed constraint satisfaction algorithm for complex local problems. In *Proceedings of the International Conference on Multi-Agent Systems*.

Yokoo, M. 1995. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*.

Yokoo, M. 2001. *Distributed Constraint Satisfaction:Foundation of Cooperation in Multi-agent Systems*. Springer.

Zhang, W., and Wittenburg, L. 2002. Distributed breakout revisited. In *Proceedings of the National Conference on Artificial Intelligence*.