Agent Influence as a Predictor of Difficulty for Decentralized Problem-Solving

Martin Allen and **Shlomo Zilberstein**
Computer Science Department
University of Massachusetts
Amherst, MA 01003
{mwallen, shlomo}@cs.umass.edu

Abstract

We study the effect of problem structure on the practical performance of optimal dynamic programming for decentralized decision problems. It is shown that restricting agent influence over problem dynamics can make the problem easier to solve. Experimental results establish that agent influence correlates with problem difficulty: as the gap between the influence of different agents grows, problems tend to become much easier to solve. The measure thus provides a general-purpose, automatic characterization of decentralized problems, identifying those for which optimal methods are more or less likely to work. Such a measure is also of possible use as a heuristic in the design of algorithms that create task decompositions and control hierarchies in order to simplify multiagent problems.

Introduction

Decentralized partially observable Markov decision processes (Dec-POMDPs) extend the well-known POMDP framework to multiple cooperating agents, each basing its actions upon local information, without full knowledge of what others observe or plan. Widely applicable and formally precise, Dec-POMDPs are popular models of multiagent decision making. Unfortunately, theoretical and experimental work shows that solving them optimally can be very difficult. As a step toward alleviating this problem, we provide a precise measure of the degree to which agents in a Dec-POMDP interact, and show experimentally that this measure correlates with the performance of a well-established solution algorithm: as the gap between agent influences grows, problems tend to become much easier to solve. Thus, the identified measure provides an exact general property of Dec-POMDPs, straightforward to calculate for any given problem, and a strong predictor of the difficulty of that problem. As such, it can be used to identify problems for which optimal solutions are likely to be practical. Further, such a measure can be employed in simplification methods for complex multiagent problems.

General Dec-POMDPs are NEXP-complete (Bernstein *et al.* 2002). Optimal solution algorithms face doubly-exponential growth in necessary space and time, rendering even simple problems intractable. The first-known optimal method uses dynamic programming to generate finite-horizon policies, applying iterated pruning techniques to reduce the number considered (Hansen, Bernstein, & Zilberstein 2004). However, such basic pruning still only solves the smallest problems. Similar results have been reported with respect to top-down methods employing heuristic search (Szer, Charpillet, & Zilberstein 2005). A good overview can be found in Seuken and Zilberstein (2005, §3). There are limitations to this previous work, as only a small number of problems have been used to test algorithm performance. We extend these results for dynamic programming in particular, examining performance over a wide range of systematically varied problem instances.

Even ϵ-approximation for Dec-POMDPs is NEXP-hard (Rabinovich, Goldman, & Rosenschein 2003). While locally optimal methods can deal with problem complexity (Nair *et al.* 2003), no sharp guarantees can be given about quality. Attempts to simplify the general problem isolate special sub-classes. Decentralized MDPs for which transitions and observations are independent have reduced complexity (Becker *et al.* 2004). While such problems are still NP-complete, specialized methods solve many reasonably-sized problems. Rather than focus on such special cases, we investigate general application of optimal methods, quantifying problem difficulty in a non-domain-specific manner.

We combine ideas from a number of areas. In game theory, bounding the influence agents have on overall reward can simplify the equilibria calculation (Kearns & Mansour 2002). In decentralized MDPs, it has been suggested that complexity increases with the "degree of interaction" between agents (Shen, Becker, & Lesser 2006). Further, performance of learning methods for single-agent MDPs is inversely proportional to information-theoretic measures of system dynamics (Ratitch & Precup 2002). Combining these ideas, we show how restricting agent effects on system dynamics, measured information-theoretically, correlates with improved solution performance.

Dynamic Programming for Dec-POMDPs

We study the algorithm presented by Hansen, et al. (2004), the first optimal dynamic programming (DP) method for Dec-POMDPs. The algorithm combines DP for single-agent POMDPs, using incremental policy search and pruning, with iterated elimination of dominated strategies.

Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Decentralized POMDPs

A Dec-POMDP is specified formally in terms of a tuple:

$$M = \langle \{\alpha_i\}, S, \{A_i\}, P, \{\Omega_i\}, O, R, T \rangle$$

with individual components:

- Each α_i is an agent.
- S is a finite set of world states.
- A_i is a finite set of actions available to α_i.
- P is a Markovian state-transition probability function.
- Ω_i is a finite set of observations for α_i.
- O is the observation function for state-action transitions.
- R is the global reward function.
- T is the finite time-horizon of the problem.

Existing research has primarily focussed upon the basic feasibility of running exact and approximate algorithms, and success has been measured in terms of the sizes of the sets $\{\alpha_i\}$, S, $\{A_i\}$, and $\{\Omega_i\}$ for the largest problems solved. Nothing systematic has been determined about the relationships between problem solutions and these values, but all existing optimal methods solve only very small problem instances (Seuken & Zilberstein 2005).

The Optimal Dynamic Programming Algorithm

Dec-POMDPs involve multiple agents α_i, taking actions a_i based on their local observations o_i, resulting in a joint reward. With finite time horizon t and n agents, we can represent a solution as a sequence of depth-t *policy trees*, $\delta^t = \langle q_1^t, \ldots, q_n^t \rangle$. Each tree q_i^t consists of nodes, labeled with actions, and edges, labeled with observations. To implement policy-tree q_i^t, agent α_i begins at the root, takes the corresponding action, then follows the branch labeled with the ensuing observation, repeating the process for t steps.

Unfortunately, the process of arriving at some value-maximizing δ^t is made difficult by doubly exponential growth in the set of all depth-t trees, Q^t. Even for a very simple problem, this number quickly grows so large that exhaustive enumeration becomes infeasible.

Since we cannot simply search the entire space of possible depth-t policies, the DP algorithm instead selectively prunes away policies as it goes. (In effect, at each iteration, an agent keeps only those policies that are *best responses* to some possible policy of the other agent.) We generate depth-$t+1$ policy-trees in Q^{t+1} by considering all possible action/observation-transitions into depth-t trees in Q^t. Thus, the elimination of tree $q^\star$ at one time-step in the solution method means that all later trees featuring $q^\star$ as a subtree are never considered, implicitly eliminating them all in advance. Early pruning of trees promises exponential elimination of possibilities later on.

Backup and Value Calculation The DP program is given a set Q_i^t of depth-t policy-trees for each agent α_i. Each agent also possesses a set $\mathcal{V}_i^t$ of value vectors, one for each tree $q_i^t \in Q_i^t$. For a Dec-POMDP problem with a state-set $\mathcal{S}$, and letting $Q_{-i}^t = \{Q_j^t \,|\, j \neq i\}$, each value vector $v_k \in \mathcal{V}_i^t$ is of dimension $\left|\mathcal{S} \times Q_{-i}^t\right|$, and gives the value of following policy-tree $q_k \in Q_i^t$, for each possible start-state $s \in \mathcal{S}$ and sequence of policy-trees for other agents $\langle q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n \rangle \in Q_{-i}^t$. Based on the action-transition, observation, and reward model of the Dec-POMDP, the algorithm then exhaustively generates the next set of trees Q_i^{t+1} for each agent. Once Q_i^{t+1} has been generated for each i, the next-stage value vectors $\mathcal{V}_i^{t+1}$ are also calculated. Note that each such vector will now be of higher dimension than before, rising to size $\left|\mathcal{S} \times Q_{-i}^{t+1}\right|$.

Iterated Pruning To determine the value of a policy-tree for agent α_i, we consider that agent's *belief* about the start-state of the environment and the policies of other agents. This belief is represented as a set of possible probability-distributions, $\Delta(\mathcal{S} \times Q_{-i}^t)$. We then want to prune policy-trees which are *dominated* relative to any possible belief, in the sense that some other policy-tree does just as well or better against any of the other agents' current possible policy-trees. That is, we trim out any policy that is less valuable than any other given any possible belief, eliminating it from the policy-set, and eliminating the associated value vector along with it. The first step in this process uses linear programming to efficiently find a dominated policy-tree. This is carried on iteratively for each agent α_i in turn, looping until no more policies are eliminated.

Output Policy At the end of one step of the algorithm, we have generated the set of possibly valuable depth-$t+1$ policy-trees for each agent. We repeat this process until we reach the time-horizon of the problem. The result is then a collection of policy-trees, each of which is provably optimal for some initial belief-distribution over possible starting states of the Dec-POMDP. For any problem instance in which we are given each agent's starting belief about what that state may be, we can calculate the value of each policy-tree relative to that particular belief, by simple normalization. The final output is then simply a sequence of depth-t policy-trees, $\delta^t = \langle q_1^t, \ldots, q_n^t \rangle$, each of which maximizes the value for the given agent, which can be shown to comprise an optimal policy for the overall Dec-POMDP.

Hypotheses and Experimental Methods

Our work investigates how the structure of Dec-POMDPs affects the performance of optimal dynamic programming. While we study DP in particular, the work is of more general interest, since it may well be that the intrinsic difficulty of various problem sub-classes translates over different solution techniques. Of course, if this turns out not to be so, that is also interesting, since it provides insight into the ability of various algorithms to deal with specific problems; establishing problem classes that are easier for dynamic programming is then a necessary first step in seeing whether these classes are the same across methods.

The complexity of solving a Dec-POMDP is reduced when agent actions affect their own local portion of the state-space independently (Becker *et al.* 2004). That work then applies a specialized algorithm to such problems, showing that it could perform quite well on some cases, although it is not applicable to Dec-POMDPs in general. An open

question is then how existing *general* algorithms fare on transition-independent problems. One hypothesis is that such problems, since they essentially allow agents to ignore some of the others' actions, would cause performance to improve in the policy-comparison and pruning stages of DP. A contrary hypothesis was that independent transitions would make little difference in terms of practical solution ability, since the overall effect would be too small.

Interestingly enough, neither seems to be correct. Our initial tests compared two classes of sample problems, some with transition-independence and some without. All other features of the problem remain the same. Over the few hundred test instances we examined, we were surprised to observe that in fact the performance is on average markedly better for problems in which the transitions are not independent. This result is not definitive, in that we would need to vary other factors controlling the structure of the problems (rewards and observation functions in particular) to determine the exact combinations of features that produce the observed effects. However, it does in fact falsify the preliminary hypothesis, as it shows that making a fixed problem instance transition-independent can actually negatively impact the capacity to solve it using DP. Based on this initial data exploration, our work thus replaces the initial hypotheses with a more general claim: if the transition and reward models *bound influence between agents*, the resulting Dec-POMDP is more often easier to solve optimally.

Problem Domain

Our experimental work utilizes variations on a simple *noisy broadcast channel* problem. In the original version of the problem, two agents access a single channel, attempting to send packets without collision. Packets arrive stochastically, and agents can choose whether or not to send a waiting packet at each time-step, receiving a positive reward only when the packet is sent without collision. Any packets not sent when another arrives are lost; agents observe only their own packet-state, and whether or not there was a collision after a packet is sent. While this problem is extremely simple, it demonstrates the sheer complexity of Dec-POMDPs; as outlined by Seuken and Zilberstein (2005), existing optimal solution algorithms can only solve such problems over a very limited time-horizon (3–4 steps) before the computational burden grows too large. As described below, we systematically vary the reward function, and the sorts of state-action transitions seen, producing a wide range of possible distinct instances of the basic broadcast domain.

Performance Measures

Our primary independent variable measures performance of the DP algorithm in terms of *Policy-Depth*, the number of finite time steps in the policy that the algorithm can produce. As described, this performance is limited by the potential for exponential explosions in problem size as policies extend to longer time-horizons. We measure how many such steps the algorithm can generate before it runs into such unreasonably large policy-sets, cutting things off at a pre-set depth if it appears the problem is trivial. (In such cases, the iterative algorithm can produce policies of effectively arbitrary length, since the problem is so simple.) This provides the primary measure of the algorithm's success.

Dec-POMDP Structure: Influence Gap

Initial data exploration considered many possible dependent variables, attempting to find structural features correlated with problem difficulty, and focussing especially on the degree of influence and interaction between agents. After much investigation, we devised a means for grading the difference between agent effects upon system dynamics, called the *influence gap*. This gap is given by first determining the effect each agent has on either the state transitions or one-step joint rewards, measured in an information-theoretic manner. These measures are defined below for the two agent case; each is easily extended to n-agent problems.

Definition 1 (State-influence). For agent α_i in a 2-agent Dec-POMDP, $\mathcal{D}$, the *state-influence* of α_i, SI_i, is given by the mutual information between that agent's actions (taken as a random variable), and the outcome state variable:

$$SI_i = I(S; A_i) = \sum_{s \in S} \sum_{a_i \in A_i} p(s \mid a_i) p(a_i) \log \frac{p(s \mid a_i)}{p(s)}$$

where we marginalize to get:

$$p(s \mid a_i) = \sum_{s' \in S} \sum_{a_j \in A_j} P(s \mid s', a_i, a_j) p(s') p(a_i) p(a_j)$$

$$p(s) = \sum_{a_i \in A_i} p(s \mid a_i) p(a_i).$$

Definition 2 (Reward-influence). For any agent α_i in a 2-agent Dec-POMDP, $\mathcal{D}$, the *reward-influence* of α_i, RI_i, is given by the mutual information between that agent's actions (taken as a random variable), and the single-step reward:

$$RI_i = I(R; A_i) = \sum_{r \in R} \sum_{a_i \in A_i} p(r \mid a_i) p(a_i) \log \frac{p(r \mid a_i)}{p(r)}$$

where we marginalize to get:

$$p(r \mid a_i) = \sum_{s \in S} \sum_{a_j \in A_j} p(r \mid s, a_i, a_j) p(s) p(a_i) p(a_j)$$

$$p(r) = \sum_{a_i \in A_i} p(s \mid a_i) p(a_i)$$

and use the definition:

$$p(r \mid s, a_i, a_j) = \begin{cases} 1 & \text{if } R(s, a_i, a_j) = r \quad [R(\cdot) \in \mathcal{D}]; \\ 0 & \text{otherwise.} \end{cases}$$

(Note that in both definitions, we treat starting states and actions as occurring uniformly; for instance, $p(s) = \frac{1}{|S|}$.)

These two values thus give an information-theoretic measure of how much an agent's actions tell about the values of the state-transition and reward functions. Note that mutual information is always non-negative, and so both $SI_i \geq 0$ and $RI_i \geq 0$. When $SI_i = 0$, agent α_i has no effect upon

state-transitions; given that agent's action, the outcome state is either purely random (if $S_j = 0$), or is influenced only by agent α_j. A similar point holds for the case $R_i = 0$. We take the sum, $(SI_i + RI_i)$, to measure the *combined influence* that agent α_i can have on the system dynamics. The absolute difference between these quantities measures the difference between how much each agent affects the system.

Definition 3 (Influence Gap). For any 2-agent Dec-POMDP, $\mathcal{D}$, the *influence gap*, $IG(\mathcal{D})$, is the difference between the total influence exerted over the system by each agent:

$$IG(\mathcal{D}) = |(RI_1 + SI_1) - (RI_2 + SI_2)| \quad [\alpha_1, \alpha_2 \in \mathcal{D}].$$

To test how this measure correlates with algorithm performance, we generate a wide range of problem instances, covering the full range of possible influence gaps. Further, we have set up an automated system for problem-generation and dynamic programming. For a basic Dec-POMDP framework determined by a fixed number of agents, actions, local states and reward combinations, the maximum possible value of an individual agent's influence $(RI_i + SI_i)$ can be calculated straightforwardly; furthermore, all such values are bounded below by 0. This provides a range of possible values for each measure, and also a range for the gap between them. Such a range can then be divided into a set of intervals, and the automated system can be set to randomly generate problem instances while also making sure that we consider a sufficiently large number that fall inside each of these intervals. We can then employ statistical model-fitting methods to explore whether or not there is in fact a connection between dependent and independent variables.

The main hypothesis is that as the difference between the maximum and minimum effects of individual agent actions grows, the problems will become easier to solve, and the algorithm will be able to generate policies for longer and longer time horizons. Intuitively, a large influence gap corresponds to a case in which some agent has increasing control of the outcomes in the system, and the resulting problem can be considerably easier to solve.

Experimental Results

We ran a large set of experiments over Dec-POMDPs of varying basic types, to see how influence gap corresponded with algorithm performance. Results suggest a strong correlation between widening influence gap and ease of solution. Our test-bed consisted of 900 distinct two-agent broadcast problems, divided into three main groups (of 300 instances each) based on their transition matrices:

Random non-independent instances: Transitions generated at random (must be valid probability distributions).

Independent instances: Transition probabilities were generated randomly, but constrained so that they could be factored into independent components for each agent.

Single-agent instances: Transition probabilities were generated randomly, but constrained so only the first agent had any influence (i.e., state-influence $SI_2 = 0$).

In each class of 300 instances, there were three subclasses, of 100 instances each, divided based on the reward matrices:

Random non-equal instances: The reward matrix for joint actions was selected uniformly from the set $\{1, 0\}$.

Fixed equal instances: The reward matrix of the original broadcast problem was utilized; each agent had an equal influence upon reward ($RI_1 = RI_2 = 0.0338$).

Fixed single-agent instances: Only the first agent had any influence over rewards ($RI_1 = 0.2158$, $RI_2 = 0$).

Given this range of cases, we calculated state- and reward-influence for each agent, and the influence gap between them. For each problem instance, DP was run as usual. Performance was measured by running until the number of policy-trees for any agent exceeded 1200, at which point further progress was infeasible, given time and memory constraints. If a problem was solved to 100 iterations, the program terminated; such performance indicates a trivial (or nearly trivial) instance, for which continued exploration is no longer interesting, since it is effectively possible to iterate indefinitely. The number of iterations before the limit was hit therefore ranged between 3 and 100.

Initially, we analyzed our results to see whether the number of iterations possible correlated in some direct fashion with the influence gap measure. No such immediate relationship was found, but influence gap does in fact correlate with the *proportion* of problems that are solvable to various degrees. Solutions fell into three main classes:

Hard: Only 3–4 step solutions possible; interestingly, 4 steps is the prior best limit of optimal methods, suggesting that prior research concentrates on the hardest instances.

Medium: More than 4 steps are possible, but the problem is not trivial. 93% of such problems fell in the range $[5, 9]$; (nearly all the remainder fall in the range $[10, 19]$, with just 2 outliers solved to 32 iterations); while not much greater than the hard problems, it does mark a two-fold increase over best prior results.

Easy: A full 100 steps (or more) are possible.

Additionally, the influence gap measure fell into four natural classes, each of which comprised relatively equal proportions of the space of all problem instances:

Zero: Agents exercised the same amount of influence.

Small: A difference in the range $(0, 0.05)$.

Medium: A difference in the range $[0.05, 0.2)$.

Large: A difference up to the maximum, in $[0.2, 0.394]$.

We measure how problems in each class of influence gap fall into the various difficulty classes; results indicate that problems with large influence gaps will be proportionately easier, whereas problems where the difference between agent influences approaches 0 are proportionately harder. Before presenting those results, however, we exclude a certain subclass of problems that otherwise threaten to unfairly skew the numbers in our favor.

In some cases, one agent α_i has no influence at all $(SI_i + RI_i = 0)$. In such instances, analysis shows that the Dec-POMDP is equivalent to a single-agent problem; practically, this has the effect that all such broadcast problem instances are easily solved. At the same time, it tends to make

Variable	Unit	Classes	Range	Prop. (%)
Gap	nats	Zero	0	0.282
		Small	$(0, 0.05)$	0.27
		Medium	$[0.05, 0.2)$	0.22
		Large	$[0.2, 0.394]$	0.228
Steps	Iterations	Hard	3–4	0.407
		Medium	5–32	0.216
		Easy	100+	0.377

Table 1: Classes of the main variables, and their proportions.

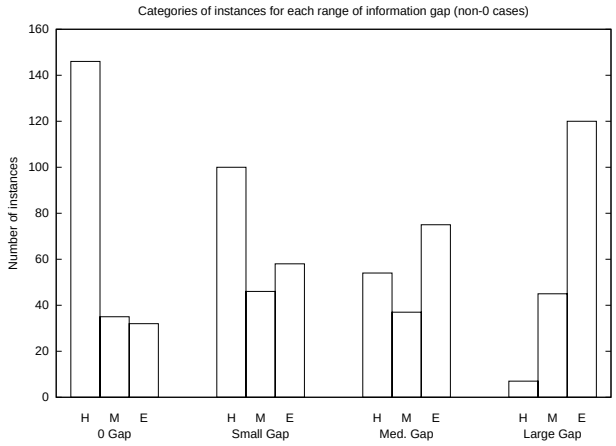

Figure 1: Instances in each range of difficulty, for each gap.

influence gap *larger*, since $IG(\mathcal{D})$ is then equal to the entire influence of the other agent. As we discovered after the fact, if such cases are not excluded, the effect of influence gap on difficulty seems even more pronounced. Such single-agent problems are a special, degenerate case, however, and do not reflect a truly multiagent problem. So as not to favorably bias our results, the final numbers exclude such cases, and are based on 755 remaining instances; Table 1 outlines the categories into which we divided our variables, and their proportion among total cases.

Figure 1 compares the four categories of influence gap, showing the proportion of each set of problems falling into each difficulty class. Evidently, there is a trend towards a higher number of easier problems as the gap between agent influences increases. When the influence gap is 0, the majority (68.5%) of the problems are of the hardest type, solvable only up to at most 4 iterations; the remainder divide almost evenly between medium and easy problems, with 15% falling into the latter class. As the gap increases, the proportions reverse themselves. When the influence gap falls into the large range, most of the problems (69.8%) fall into the easy difficulty class, whereas only a few (4.1%) are hard.

These results confirm that there is a correlation between influence gap and problem difficulty for general Dec-POMDPs. Some research in multiagent systems has suggested that increasing the centralization of control can lead to simplified planning (Georgeff 1984). This intuitive idea, often observed in practice, is given support by our results. In a precise sense, an increasing influence gap reflects a concentration of control more and more in the hands of one agent; as we have shown, this tends to lead to problems that

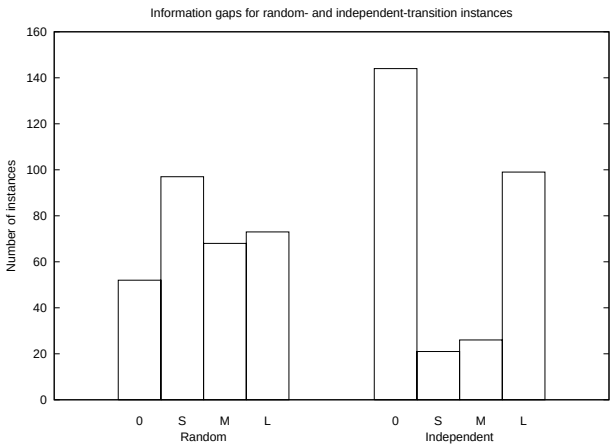

Figure 2: Instances in each range of influence gap for the two types of transitions.

are easier to solve in an optimal fashion.

Furthermore, our results also help account for other observed phenomena that are otherwise somewhat surprising. As already discussed, it had originally been hypothesized that the special subclass of transition-independent Dec-POMDPs, to which specialized algorithms had previously been applied, would turn out to be somewhat easier to solve using the general DP method (especially since they were of a lower complexity class). Contrary to this hypothesis, however, it was found that in fact more problems featuring independent transition matrices were difficult than were those featuring dependencies in the matrices.

We are now in a position to explain this result. If influence gap and proportionate difficulty are indeed correlated, we would expect that the larger proportion of hard problems means that more of the independent problems have small influence gaps. Indeed, this is confirmed, as shown in Figure 2, which compares problems based on type of transition-matrix (excluding the single-agent transition problems, so we compare just 600 instances here). More independent problems (49.7%) feature a 0-level influence gap than do non-independent problems (18%). More of the independent problems (34.1%) also exhibit large gaps than do the non-independent cases (25.2%), but that difference is less pronounced. While transition-independent problems do fall into a lower worst-case complexity class, a large proportion of them still fall into the hardest class. The fact that this proportion accords with prediction is further evidence for the correlation between influence gap and difficulty.

Applicability and Limitations

Two main questions remain with this work. First, we need to consider other aspects of problem dynamics and agent interaction that might influence problem difficulty. In particular, we need to examine how the *observation model* affects the performance of solution algorithms. In this work, we concentrated solely upon transitions and reward, in order to try to isolate their effect on problem difficulty. We plan to continue our work in this area, varying the other main dimension of problem dynamics. It is hoped that we can gain even

greater predictive power when agent influence over observations is added into our measure.

We also need to extend our tests to larger, more complicated problem instances. While the several hundred versions of the basic broadcast domain, with distinct transition and reward models, assures us that our results are not tied to a single test-problem, there is still the issue of size. On this point, we can only say that we expect difficulty will continue to correlate with influence gap: it may just be that more problems lie in the strictly infeasible range. Extending our work to problems with more than two agents is somewhat more complicated, as there are a number of possibilities as to how influence gap should be defined. Preliminary results suggest that this is best handled as the *pairwise minimum* of influence gaps, but more work is needed.

Conclusions and Future Work

Our experimental work demonstrates a connection between the information-theoretic influence gap measure and the ease with which a Dec-POMDP can be solved using dynamic programming. As the gap between agent influences widens, more problems become easier to solve, although some still remain of the hardest difficulty. As the gap tightens, so that agents exert the same (non-zero) amount of influence, more problems become difficult, although some are still easily solved. This suggests some further factor in the dynamics of system performance that interacts with algorithm performance. Ongoing work pursues such further factors, and extends current results to problems with more interesting structure than the relatively simple instances considered here.

At the same time, our measure is of real interest. Prior research on exact algorithms for Dec-POMDPs has tended to generate largely negative results, as even simple problems are found to be difficult to solve optimally. The usual response has been to turn to approximate methods, for which exact performance guarantees are not readily available, or to special algorithms only applicable to particular restricted subcases. Influence gap can be easily and automatically calculated in advance, for any Dec-POMDP whatsoever. Since a large gap correlates with problems that tend to be easier to solve, it can be used as a means of sorting out instances to which optimal methods are more likely to be applicable.

In addition, such a measure could serve as a heuristic in algorithms that seek to simplify Dec-POMDP problems. Work in multiagent systems has explored various organizational structures, including hierarchical control or task decomposition, investigating how well they produce good solutions. In practice, however, finding the best such structures and decompositions is often as difficult as solving the original problem—if not more so—given the sheer number of ways in which a problem can be decomposed, or the profusion of possible distinct control hierarchies. A measure like influence gap suggests itself here. When considering different control hierarchies, for instance, one might first examine the solutions generated by task distributions that maximize influence gap, since it will tend to be easier to determine the actual value of the associated policy.

Finally, while some preliminary work has sought to associate the general structure of agent interactions with the complexity of multiagent tasks (Shen, Becker, & Lesser 2006), there has not yet been much precise analysis of these interactions. This work is a first step in just that direction.

Acknowledgments

Work contained here was supported in part by National Science Foundation under grant number IIS-0535061 and by the Air Force Office of Scientific Research under grant number FA9550-05-1-0254. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not reflect the views of the granting agencies.

References

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2004. Solving transition independent decentralized MDPs. *Journal of Artificial Intelligence Research* 22:423–455.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27(4):819–840.

Georgeff, M. 1984. A theory of action for multi-agent planning. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, 121–125.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 709–715.

Kearns, M., and Mansour, Y. 2002. Efficient Nash computation in large population games with bounded influence. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 259–266.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 705–711.

Rabinovich, Z.; Goldman, C. V.; and Rosenschein, J. S. 2003. The complexity of multiagent systems: The price of silence. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 1102–1103.

Ratitch, B., and Precup, D. 2002. Characterizing Markov decision processes. In *Proceedings of the Thirteenth European Conference on Machine Learning*, 391–404.

Seuken, S., and Zilberstein, S. 2005. Formal models and algorithms for decentralized control of multiple agents. Technical Report UM-CS-2005-68, University of Massachusetts, Amherst, Department of Computer Science.

Shen, J.; Becker, R.; and Lesser, V. 2006. Agent Interaction in Distributed MDPs and its Implications on Complexity. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 529–536.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 576–583.