Multi-Agent Policies: From Centralized Ones to Decentralized Ones *

Ping Xuan and Victor Lesser

Department of Computer Science
University of Massachusetts at Amherst
Amherst, MA 01003

University of Massachusetts Technical Report 2001-48

Abstract. Since most multi-agent systems consist of decentralized agents, it is necessary to provide the agents with decentralized policies instead of centralized ones. While the centralized policies specify the decision of the agents according to the global state of the multi-agent system, the decentralized policies must assume only a partial knowledge of the system, and have to specify a communication policy to obtain additional information. This does not mean that centralized policies are invalid, but we need to be able to provide ways of implementing a centralized policy in a decentralized system. Also, it is important to derive decentralized policies from centralized ones because of the complexity involved in designing decentralized policies. In this paper, we present a systematic transformation method based on the decentralized multi-agent Markov decision process framework, provide a representation for discussing decentralized communication decisions, and introduce a set of strategies. Also, we connect and compare this method to traditional multi-agent planning work, and provide some insights from a decision-theoretic perspective.

1 Introduction

When dealing with cooperative multi-agent problem solving, there are generally two classes of views, and as a result, two types of solutions. One is the *centralized* view, such as the Multi-Agent Markov decision process (MMDP) model proposed in [2], where the focus is on the behavior of the system as a whole. The starting point of the the centralized model is the *global state*, which consists of local states of the agents in

* Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreement number F30602-99-2-0525 and by the National Science Foundation under Grant number IIS-9812755. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory, National Science Foundation, or the U.S. Government.

the system. A solution, or policy, for a problem under this model, describes the *joint action* to take, i.e., the local action each agent should take, when the system is in any global state S. After the joint action finishes, the system would be in one of all possible next states, since the outcome of the joint action may not be deterministic. Typically, a global utility function defines the reward the system will receive when the system is in state s, taking joint action a, and results in state s'. Thus, the policy can be evaluated based on the expected total utility the system would receive. Figure 1 shows how the problem solving is carried out under such a policy (in a two-agent system with agents X and Y).

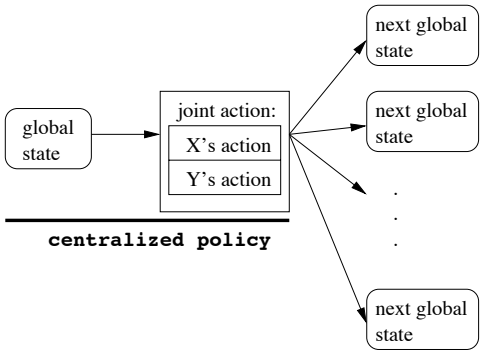

Fig. 1. Centralized View and Policy

A different class of view, the *decentralized* view, focuses on the behavior of each agent, taking the view of a situated agent in the system [9]. In this view, an agent only sees its own local states, and has to decide the next local action on its own. Here, the agent each has only a partial view of the system's global state. Furthermore, an agent can choose to expand its partial view, by communicating with each other to share information. A theoretical model must represent each agent's local knowledge, which would consist of local state and other information such as past communications. A policy under this model would then have to define, for each agent, the local action to take when the agent has local knowledge K, including whether to communicate or not. Note that each time when a local action finishes (and hence local action outcome is observed), or when communication happens, the local knowledge is updated. The global state is not automatically observed by the agents, but can be uniquely identified given the set of local knowledge in all agents. Furthermore, the global state can be uniquely identified by the agents when the agents communicate. The collection of individual policies define the total system policy, which can be used to determine how the actual global state may evolve under this decentralized policy. Then, the policy can also be evaluated by calculating the expected total utility based on the same utility function mentioned earlier. Figure 2 shows how the problem solving progresses under a decentralized policy.

Decentralized view is certainly much more complex, but since most multi-agent systems are distributed in nature, and agents are generally autonomous — meaning that each agent is a decision maker on its own, taking a centralized view in such a system would often oversimplify the problem (e.g. assuming the agents see the global

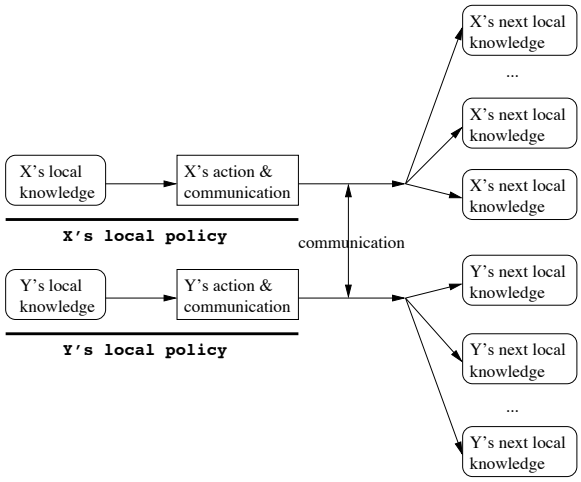

Fig. 2. Decentralized View and Policy

view instead of the partial view), and a centralized policy would not be implementable by situated agents without imposing some strong assumptions, constraints, or special mechanisms. Therefore it is very important to develop decentralized policies so that it is possible for agents with partial views to effectively perform cooperative problem-solving strategies.

Clearly, decentralized policies are quite different from centralized ones. The key distinction is that the centralize policy (CP) is working on the global level, with all agents knowing the global system status at all times. However, for decentralized policy (DP), each agent is limited as its own (limited) view of the global system, and has to rely on communication in order to obtain non-local information. Each agent will have its own local policy, which decides its actions and communications. Thus, each local policy has to be based on its local information set, which is different from each other. Obviously, if all the agents choose to communication at every stage of the problem solving, they can maintain the same global state at all time, which is equivalent to the CP case. But, when the cost of communication, or the availability of communication, is an issue to be considered in the problem solving, agents may have to consider that if the communication is worthwhile. If, an agent chooses not to communicate at a certain stage, then the local decision making at that stage has to rely on previous information, which include previous communications and local state/action history. Therefore, it is easy to see that DP is, in general, history dependent. In other words, while in the centralized view, the system can be modeled by a standard, "memoryless" Markov decision process (MDP) — and the CP is simply a standard MDP policy, in the decentralized view, the system has to be modeled by a decentralized, history-dependent decision process, and the DP is a decentralized, history-dependent policy. This casts a serious impact on the complexity of obtaining an optimal DP for a multi-agent problems: the complexity becomes nondeterministic exponential-time (NEXP) [1], whereas obtaining an optimal

CP, which involves solving a standard MDP optimally, is polynomial-space (PSPACE) [5], a lower complexity class.

Thus, heuristic approaches and approximation methods for developing DPs are extremely important. However, due to the infancy state of this research subject, only a number of simple heuristic approaches have been studied so far [9]. Needless to say, these approaches are often quite domain-specific and therefore cannot be easily applied to develop DPs for multi-agent problem solving. On the other hand, significant progresses have been recently made in the area of developing heuristic and approximate CPs for multi-agent problem solving. Thus, it is quite desirable that we can benefit from there by finding ways to deduct DPs directly from CPs. This way we can also provide insight and feedback to the research on CPs, and be able to tell how feasible or effective a CP is when it is to be implemented in decentralized systems, which we believe the bulk of multi-agent systems must be.

This transform is illustrated in Figure 3. Note that the decentralized policy is consisted of several per-agent parts (known as *local policies*) - in our case, two local policies, one for agent X (DP^X) and one for agent Y (DP^Y). Together they form a single DP.

$$\mathrm{CP} \longrightarrow \begin{pmatrix} \mathrm{DP}^X \\ \mathrm{DP}^Y \end{pmatrix}$$

Fig. 3. Centralized Policy to Decentralized Policy

There are several challenges in this transforming process. First of all, there are many DPs that may be derived from a single CP, as the agents can choose a number of communication policies. The simplest one is the DP that let all agents communicate at all stages. This DP would often result in excessive communication, so we are interested at finding DPs that uses a little communication as possible, yet still follows the exact CP (and therefore the same expected utility value as the expect utility of the CP, provided that the communication cost is zero), or allows only limited degradation of expected utility compared to the CP. Ideally, we want to be able to provide a spectrum of DPs that have different amount of communication and expected utility, and therefore enable us to choose a DP that best reflects the tradeoff between communication costs and expected utility values. However, the choice of communication policies cannot be arbitrary: it has to follow the fact that the agents only have local observations and past communication before making the communication decision. In this paper, we discuss our approach which is based on the use of common belief states. Note this is not the same belief states often used in the POMDP context, since our theoretic model is the decentralized multi-agent MDP model, which is not standard MDP or POMDP. In fact, communication introduces another type of observability: the agents can decide whether to observe the nonlocal information or not by deciding whether to communicate or not.

2 Transform CP to DP

Now we discuss our approach of transforming a centralized multi-agent policy (CP) to decentralized ones (DP). Suppose we are given a CP for a particular multi-agent cooperation problem.

As mentioned before, a CP is often described in terms of a policy for a standard Markov decision process (MDP). The policy consists of a mapping from the set of states (global system states) in the MDP to the actions (joint agent actions) set, telling the joint action $a(s)$ to be performed at each global state s. We use $a_X(s)$ and $a_Y(s)$ to represent the X's and Y's local action part in the joint action, respectively. Note that the mapping only need to cover the *reachable* states. For unreachable states, the mapping is totally irrelevant. This can reduce the size of a CP considerably.

For DP, however, the policy is not to be described by a standard MDP policy, nor any of its local policies. For each agent, its local policy cannot be based on the global state because it is not known to the agent. To describe a DP, we introduce three concepts: (a) common belief state, (b) local belief set, and (c) local history set. Roughly speaking, an agent's local policy can be described through a mapping between local belief sets to local actions, and a mapping between local history sets to communication decisions. However, the common belief state, which describes the consensus of the agents, is the key to the computation of local belief sets and local history sets.

Based only on local information (include the history as it sees), an agent only knows that the current global state belongs to a set of possible global states. We call this set the agent's *local belief*, or LB set. Obviously, if an outsider sees current local beliefs of all agents, it can uniquely decide the current global state of the system (which is the one and only one state appearing in the local beliefs of all agents), i.e., the current local beliefs of all agents contains enough information to identify the current global state (by calculating the joint set of all local beliefs).

Typically, one agent's local belief is different from the other's, since the agent sees some local information (namely the outcome of local methods) that other agents cannot see. Also, local belief changes from one episode to another. Therefore, local beliefs cannot be the consensus of the agents, which is the set of possible global states that each agent can derive without using any information the other agents do not know. Note that once the agents communicate, they will observe the global state, and thus the local beliefs in each agent are the same – containing just the global state observed – for all agents. In this case the new consensus equals the local belief. In other words, communication synchronizes the agents with consensus global states. However, if the agents do not perform communication, the new consensus is going to cover all non-communicated next states of any state in the current consensus.

2.1 Common Belief States

The consensus of the agents is what we call the *common belief states*, or simply *belief states*, noted as the B set. Common belief states are sets containing the global states. They are called common because they are common knowledge to all agents, i.e., they can be independently calculated in each agent. The definition is an iterative one: at the

beginning of the problem solving, the common belief state contains only the starting global state, which is known to all agents (i.e., synchronized).

Given a CP, each agent's local action is uniquely defined (and also known to all agents). Thus, each agent knows the possible local outcomes of the local actions in any agent, although it only observes its own local outcome.

For any common belief state, there are a set of possible next belief states, each corresponds to sets of next local beliefs. Basically, given a CP, each agent's local action is uniquely defined (and also known to all agents). Thus, each agent knows the possible local outcomes of the local actions in any agent, although it only observes its own local outcome. Each agent then decides if it needs communication to decide the next local action. The results of *all* agents' communication actions will decide how the next common belief states are formed: when agents communicate and the global states is discovered, the common belief state contains the global state only. Otherwise, the common belief state of the next stage would be the set that contains all the non-communicated next states of every global state in the previous common belief states.

In other words, suppose the current common belief state is B, and for any state $s \in B$, the set of next states (according to the centralized policy CP) is $N(s)$, and let $N = \cup N(s)$, then if K is the subset of N that are discovered based on the communication among the agents, then there will be $|K| + 1$ common belief states (if $K \neq N$) at the next stage, and they are:

- the set $N - K$ (containing all states that incurs no communication), and
- for each state n in K, the set contains n only (n is the global state synchronized via communication, so local beliefs are updated).

Now it is easy to see the relationship between common belief state B and local belief (LB) sets. Common belief state is either the local belief itself (when the global state is synchronized), or the union of possible local belief sets in each agent (for the states not discovered via communication). Finding the mapping from local belief (LB) sets to local actions is trivial since local actions for each global state is already defined by the CP. However, the necessary condition for such mapping to be meaningful is to make sure all states in a local belief set must share the same local action. This is the key for deciding communication, which will be discussed shortly.

As an example, let's assume that agent X performs action a and agent Y performs action b, and resulting in four possible states (distinguished by the outcome of a and b), which forms a common belief state, depicted in Figure 4, assuming both agents choose not to communicate regardless the local outcomes:

This figure illustrates how the common belief state B is partitioned into local belief sets for each agent: for some episodes, X would have local belief LB_1^X (for outcome a=1), for others episodes, LB_2^X. Similarly, Y's local belief would be either LB_1^Y or LB_2^Y. Clearly, in this belief state, each agent knows the next local action without the need of communication: X chooses c for both LB_1^X and LB_2^X, and Y chooses d if in LB_2^Y and e if in LB_2^Y. This is a valid mapping from LB sets to actions in each agent.

To add some intuition, common belief states are the beliefs of an outside observer who sniffs all communication messages among the agents, but does not have the local information of any agent in the system. Such an observer will discover the global state if the agents communicate (and hence the singleton belief states), but otherwise can only

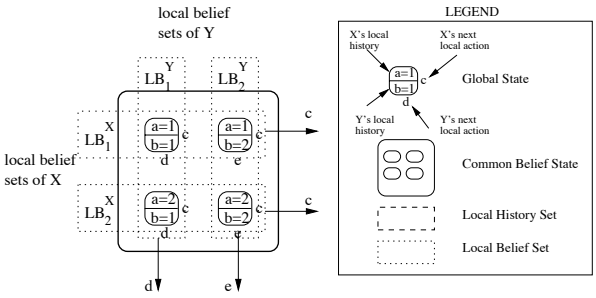

Fig. 4. Common Belief State

conclude that the global state is one of the non-communicating next states ($N - K$). It naturally follows that the common belief is the union of possible local belief sets, since the only additional information an agent has is its local outcomes, which uniques decides the local belief set of the agent.

Since the belief states are common and can be calculated by all agents independently, the common belief states form the basis of mutual understanding among the agents. In decision theory terms, that means that common belief state summarizes the history information that is known to all agents. The agents therefore need not keep records of any historic event happened before the last time the common belief state was updated. For a situated agent, its current local belief is a subset of the common belief state, which uniquely defines the whole local history of the agent, since information known to this agent only is used to calculate the local belief, i.e., the difference between the local belief and the common belief state symbolizes the information known to this agent only.

An immediate result from this is that when the common belief state is a singleton state, the local belief must equals the common belief state. In this case the system is synchronized, since all agents now understand what the current global state is. Also, the belief state does not contain any historic information in this case, therefore the agents need not keep history information (memory) other than the current state.

2.2 The Calculation of Belief State

In the above discussion we introduced the concept of the (common) belief states. However, we have not elaborate exactly how the belief states is calculated from current stage to the next stage. We stated before that this process depends on the particular communication policy to be used by the agents. The decision of communication is not arbitrary, however. As mentioned before, for each state in a local belief set LB of an agent, its next local action must be the same for the agent. In other words, either

1. the local belief contains only one (global) state, and hence the next local action is the local part of the joint action specified by the CP, or,
2. the local belief state contains several global states, but the local actions for all of them (according to the centralized policy CP) are the same.

Since the common belief state is the union of several local beliefs in each agent, this no-ambiguity condition must be satisfied for each of the local belief sets in the common belief state. This becomes a constraint for common belief states:

1. the common belief contains only one (global) state, or,
2. the belief state contains several global states, but for each agent, the local actions for any two of them are different only when they belong to different local belief sets of the agent.

This tells us that knowing only the belief states and not the exact global state does not bother the agent as long as the agent has no ambiguity toward the choice of its next local action. Thus, if choosing not to communicate does not introduce ambiguity toward the local action, the agent can decide not to communicate at the current stage (after the previous action finishes).

Thus, we need to make sure that this no-ambiguity rule holds during the process of deriving the set of common belief states in the next stage. In turn, this means that we need to apply communication decisions so that the $N-K$ set satisfies this no-ambiguity rule (the states in K are synchronized, so no-ambiguity rule holds automatically). Note that the communication decisions are applied to the next states N (which has not begin yet), not the states in B. This means that we need to consider all possible outcome of current actions.

To do this, for each agent, we divide the set of next states (the set N – calculated according to the CP) of the current common belief state into sets of states that share the same local history. We call these sets LH_i^X and LH_j^Y, for agents X and Y, respectively. The states in each one of these sets are indistinguishable by the agent's local information (including local action and outcome history, plus all previous communications). Let $LH^X = \{LH_i^X\}$ and $LH^Y = \{LH_j^Y\}$. Thus, for each state $s \in N$, we can obtain the i and j index for s. As a result, we obtain a matrix M — a $|LH^X| \times |LH^Y|$ matrix whose elements are $s_{i,j}$ — that lists the state that is in both LH_i^X and LH_j^Y. Note that some $s_{i,j}$ states may be non-existent, in this case we will write ϵ as the matrix elements (see the Appendix for explanations). This is illustrated in Figure 5.

$$
\begin{array}{c} \\ LH_1^X \\ LH_2^X \\ \vdots \\ LH_m^X \end{array}
\begin{array}{c}
\begin{array}{cccc} LH_1^Y & LH_2^Y & \dots & LH_n^Y \end{array} \\
\left(\begin{array}{cccc}
s_{1,1} & s_{1,2} & \dots & s_{1,n} \\
s_{2,1} & s_{2,2} & \dots & s_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
s_{m,1} & s_{m,2} & \dots & s_{m,n}
\end{array} \right)
\end{array}
$$

Fig. 5. Local History Set Matrix

For each LH_i^X (and LH_j^Y), we can check the ambiguity of local actions for agent X (and agent Y), and accordingly decide the communication policy, which simply decides if agent X (or agent Y) wants to communicate when the local history set is LH_i^X (or LH_j^Y).

Based on X and Y's communication policies, the calculation of the K set is straightforward: for each $s \in N$, if its matrix index is (i, j), then if agent X chooses to communicate for LH_i^X or agent Y chooses to communicate for LH_j^Y, then s belongs to K (synchronized). Thus, we obtained the $|K| + 1$ next common belief states.

Figure 6 (using the same legend as in Figure 4) shows an example of this process, where the common belief state B is illustrated in Figure 4. The N set has 9 states, and is partitioned into 3 LH sets for each agent. LH_3^X and LH_3^Y have ambiguous next local actions, so one communication strategy is to communicate when agent X sees itself in LH_3^X or when agent X sees itself in LH_3^Y, otherwise not to communicate. This results in a K set containing 5 states, and therefore 6 common belief states in the next stage.

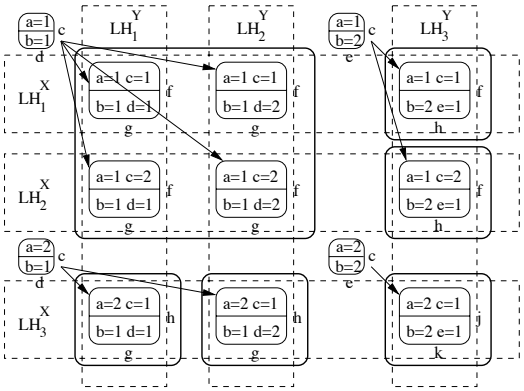

Fig. 6. Next Common Belief States

Clearly, we can see that the key here is to decide the communication policies, which is simply a mapping from LH^X (and LH^Y) to a yes or no. The constraint is to make sure that the no-ambiguity rule holds for all next belief states. Clearly, for the $|K|$ singleton states the constraint holds automatically. So the task is to make sure the set $N - K$ meets the constraint.

Intuitively, choosing to communicate on LH_i^X (or LH_j^Y) can be symbolized by crossing out all matrix elements on row i (or column j). So, after applying all communicating LH_i^X's and LH_j^Y's, the remaining matrix is the $N - K$ set. The no-ambiguity constraint for the belief node $N - K$, therefore, is to make sure that for each row (and column) of the *remaining* matrix, there is no ambiguity about the next local action for X (or for Y).

To summarize the above discussion, we list the steps for calculating the next belief states from a given belief state (in a two-agent system). Since the initial belief state – the starting global state — is known to both agent, this calculation allows us to iteratively enumerate all belief states that might occur during the process solving. Figure 7 summarizes the process of calculating next belief states. Essentially, this is a 6-step process:

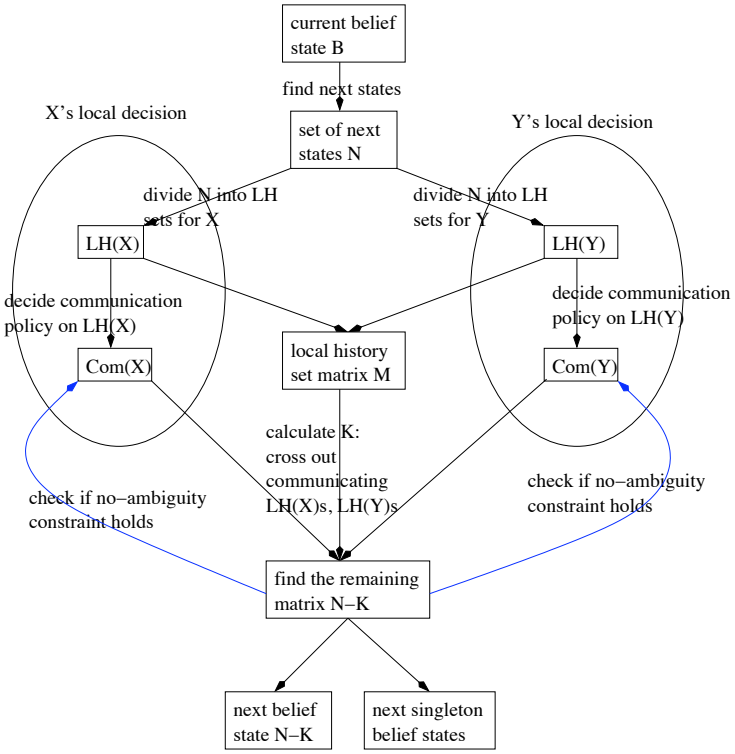

Fig. 7. Centralized Policy to Decentralized Policy

1. Given a current belief state B, the first step is to find the next global states: for each global state $s \in B$, each agent can calculate the next possible global states next(s) according to the centralized policy (CP), which is known to both agents. The set of next states N is simply $\cup_s$next(s).
2. Once N is computed, each agent can then divide the N set into local history sets LH_i^X (and also LH_j^Y). This is done by analyzing the different local history path of each state $s \in N$, and putting the states having the exact same local history path (e.g. taking the same local action sequence and having the same local outcome sequence) into the same history set. In other words, LH_i^X (or LH_j^Y) sets group the global states that are indistinguishable by agent X (and Y).
3. Now we can construct the local history matrix M: the matrix element $s_{i,j} = LH_i^X \cap LH_j^Y$ ($1 \leq i \leq |LH^X|$ and $1 \leq j \leq |LH^Y|$). Clearly, $s_{i,j}$ is either the global state that is uniquely identified by X's and Y's local histories, or a non-existent state ϵ that is impossible to reach given X's and Y's local histories.
4. Decide a mapping Com^X from LH^X to $\{yes,no\}$ and also a mapping Com^Y from LH^Y to $\{yes,no\}$, i.e., the communication policy of each agent at belief node B.
5. Based on Com^X and Com^Y, do the following operations on M: if Com$^X(LH_i^X)$ is *yes* (meaning to communicate), cross out the i'th row (the LH_i^X set) of M;

and similarly, if $\text{Com}^X(LH_j^Y)$ is *yes*, cross out the j'th column (the LH_j^Y set) of M. The crossed-out elements form the set K and the remaining matrix is the set $N-K$. The no-ambiguity rule requires that the remaining $N-K$ matrix maintains the following property: X's actions (a_X, according to the CP) for all elements in each row must be the same; and Y's actions (a_Y) for all elements in each column must be the same. If this constraint is not met, a different Com^X and Com^Y must be chosen.

6. If Com^X and Com^Y are legal and the remaining $N-K$ matrix meets the no-ambiguity constraint, the next belief states for B is simply the set $N-K$ (if non-empty) and $|K|$ singleton belief states, one for each $s \in K$. The LB sets for $N-K$ is simply $(N-K) \cap LH_i^X$ (for X) and $(N-K) \cap LH_j^Y$ (for Y.)

This process is the core of constructing a DP from a CP. By using belief states, a DP can be described in two parts: the local action part and communication decision part. In the above process we already see how belief state B is used in constructing the communication decision part: from B, we get N, and then the LH sets, and the Com^X and Com^Y based on the LH sets. The local action part is also derived from B, but not using the LH set but the LB sets. The set B is directly divided into local belief LB^X (LB^Y) sets. For singleton belief states (i.e., synchronized), LB equals B, and the local action is directly specified by CP. For other belief states, the above process also determines the LB states in the next belief states: they are simply $(N-K) \cap LH_i^X$ (or $(N-K) \cap LH_j^Y$), e.g, the rows and columns of the remaining matrix $N-K$. Due to the no-ambiguity constraint, there is only one unique next local action for any global state in a local belief set in $N-K$. In other words, the no-ambiguity constraint requires that $\left|\{a_X(s) | s \in (N-K) \cap LH_i^X\}\right| = 1$ (if $(N-K) \cap LH_i^X$ is not empty), and $\left|\{a_Y(s) | s \in (N-K) \cap LH_j^Y\}\right| = 1$ (if $(N-K) \cap LH_j^Y$ is not empty). Hence, the mapping from LB sets to local actions is also determined by the CP, with the help of the above process of calculating next belief states. Figure 8 shows how the DP is calculated:

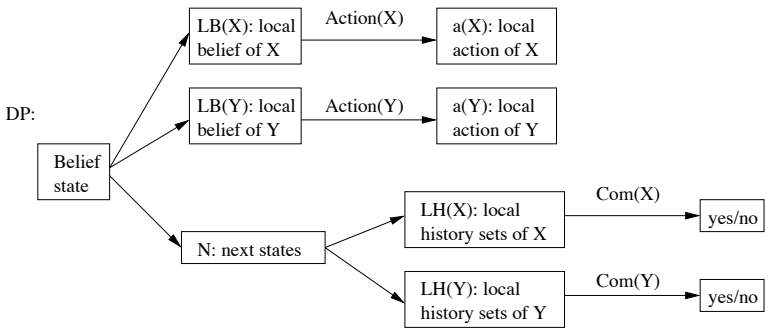

Fig. 8. Constructing a DP

Once the DP is determined, the multi-agent problem solving process is quite simple. Since all the above calculation happens within an agent, every agent in the system can independently perform the same reasoning and obtain the same DP. Thus, during the actual problem solving process, each agent first decides the current common belief state (B), then determines its local belief (LB) based on the actual local history, thus find the local action to perform (the no-ambiguity rule guarantees the same local action for all states in local belief). Once the local action finishes, the agent calculates the next states (N), and decides the local history set (LH), and according to the communication policy, decide whether to communicate or not. After the communication, the agent decides if it is in one of the singleton next belief state, or the non-communicated next belief state ($N - K$), and the process repeats from then on. Clearly, this process reflects the exact nature of decentralized problem solving — information sharing through communication and decentralized decision making.

2.3 Choosing the Communication Policies

It is easy to see that in the process of constructing a DP based on a CP, the only place that involves some freedom of choice is when deciding the communication policy Com^X and Com^Y. The rest of calculation is quite mechanical and does not offer any choice. Furthermore, the choice of Com^X and Com^Y is not arbitrary. It has to adhere to the no-ambiguity constraint. As explained before, the choice can be illustrated as selecting and deleting rows and columns of the history set matrix M, and the constraint then translates to the uniqueness of local actions for each row and column (X's actions for each row and Y's actions for each column) of the remaining matrix ($N - K$). In the following we denote this remaining matrix as M'.

Obviously, if the local actions are unique for a row (or column) of M, the same must be true for the same row (or column) of M', no matter what the selections are. Thus, one strategy is to simply select (and delete) the rows not having unique X's actions, and the columns not having unique Y's actions. The resulting M' must satisfy the no-ambiguity rule. We call this strategy the *default* strategy:

The Default Strategy:

If X's actions of LH_i^X are not unique (i.e. $\left|\{a_X(s)|s \in LH_i^X\}\right| > 1$), then we have $\text{Com}^X(LH_i^X) = yes$. Similarly, $\text{Com}^Y(LH_j^Y) = yes$ if Y's actions of LH_j^Y are not unique. Otherwise, $\text{Com}^X(LH_i^X)$ and $\text{Com}^Y(LH_j^Y) = no$.

Figure 9 shows an example of the default strategy. For simplicity, the matrix elements shown are the joint actions for the states rather than the states. The selected row and column in the matrix is shown as boldface. It is easy to see that the remaining matrix M' satisfies the no-ambiguity constraint.

The default strategy is clearly not the only strategy. For example, selecting extra rows (and/or columns) in addition to the ones selected in the default strategy would also generate legal strategies. But since we are mostly interested at reducing communications, we focus on the strategies that might use less communication than the default one. It is quite easy to see that this problem belongs to the family of constraint satisfaction problems, which are NP-hard and often solved by approximation methods. In the

$$\begin{array}{c} \\ LH_1^X \\ LH_2^X \\ \mathbf{LH_3^X} \end{array} \begin{array}{c} \begin{array}{ccc} LH_1^Y & LH_2^Y & \mathbf{LH_3^Y} \end{array} \\ \begin{pmatrix} (a1|b1) & (a1|b1) & (\mathbf{a1|b2}) \\ (a1|b1) & (a1|b1) & (\mathbf{a1|b2}) \\ (\mathbf{a2|b1}) & (\mathbf{a2|b1}) & (\mathbf{a3|b3}) \end{pmatrix} \end{array}$$

Fig. 9. Example: the Default Strategy

following we also describe a hill-climbing algorithm that is based on heuristic search methods. We call this the hill-climbing strategy: it evaluates a matrix by heuristic value — the sum of the number of ambiguous rows and the number of ambiguous columns. Given a matrix, it selects the row or column that would best reduce the heuristic value (hence the hill-climbing algorithm), then applies the best selection and produces the next matrix, and applies the same method on the next matrix. When the heuristic value is zero (no-ambiguity constraint satisfied), the search stops and the current matrix is the result:

The Hill-Climbing Strategy:

1. Let $\mathcal{M} = M$.

2. For each un-selected column or row r of $\mathcal{M}$, calculate the heuristic value of $\mathcal{M}'(r)$, which is $\mathcal{M}$ with r selected.

3. Choose the r such that $\mathcal{M}'(r)$ produces the smallest heuristic value h.

4. If $h = 0$, $M' = \mathcal{M}'(r)$; otherwise, let $\mathcal{M} = \mathcal{M}'(r)$, and go back to step 2.

5. If row i (or column j) of M' is selected, $\mathrm{Com}^X(LH_i^X)$ (or $\mathrm{Com}^Y(LH_j^Y)$) is *yes*, otherwise *no*.

Figure 10 shows an example of the hill-climbing strategy, using the same matrix in Figure 9. Instead of selecting both LH_3^X and LH_3^Y in Figure 9, here only LH_3^X needs to be selected (alternatively we can also just select LH_3^Y, depending on the implementation of the hill-climbing algorithm).

$$\begin{array}{c} \\ LH_1^X \\ LH_2^X \\ \mathbf{LH_3^X} \end{array} \begin{array}{c} \begin{array}{ccc} LH_1^Y & LH_2^Y & LH_3^Y \end{array} \\ \begin{pmatrix} (a1|b1) & (a1|b1) & (a1|b2) \\ (a1|b1) & (a1|b1) & (a1|b2) \\ (\mathbf{a2|b1}) & (\mathbf{a2|b1}) & (\mathbf{a3|b3}) \end{pmatrix} \end{array}$$

Fig. 10. Example: the Hill Climbing Strategy

3 Evaluation of DP

Once a DP is constructed, the next question is about its performance. Since the DP exactly follows the centralized policy CP, i.e., if an observer can observe the global state and the joint action, it would find that the problem solving behaves exactly as the

CP prescribed. Thus, the expected utility (EU) of the system of such a DP shall not be different from the EU of the CP. However, since in DP the reasoning is not based on global states but on belief states, it is necessary for us to examine how the expected utility is calculated for a DP.

3.1 Expected Utility

Let us first recall how the EU of a CP is calculated. In typical dynamic programming, the calculation is based on the computation of the value of each state [6]. The value of a state is simply the expected utility of the system when the problem solving reaches that state. For a terminal state, its value equals the terminal reward. For other states, the value of a state is to be calculated through its next states:

$$V(s) = \sum_{s'} p(s'|s)V(s'). \tag{1}$$

Here s' is a next state of s, and $p(s'|s)$ is the conditional probability of reaching the s' state from s. (Typically the transition probability is written as $p(s'|s, a)$, where a is the action, but since the CP specifies a for each s, we do not need to include a).

If we use $p(s)$ to note the probability of reaching state s given the CP, then, if s' is one of the next states of s,

$$p(s'|s) = p(s')/p(s). \tag{2}$$

Thus, combined with Equation 1, we have,

$$p(s)V(s) = \sum_{s' \in \text{next}(s)} p(s')V(s'). \tag{3}$$

For the starting state s_0, $p(s_0) = 1$. Thus the EU of the CP is simply $V(s_0)$, and $p(s)V(s)$ can be viewed as the *contribution* of the state s toward total EU.

For a DP, we would like to similarly define the value function $V(B)$ and probability $p(B)$ of a belief state B. Clearly,

$$p(B) = \sum_{s \in B} p(s). \tag{4}$$

Since the contribution of B is the sum of contributions of all states in B, in the style of Equation 3, we can define

$$p(B)V(B) = \sum_{s \in B} p(s)V(s), \text{ and thus,} \tag{5}$$

$$V(B) = \sum_{s \in B} (p(s)/p(B))V(s). \tag{6}$$

This tells us that, given a DP, the value of a belief state is simply the weighted value of its member states. And, because the next states of B are the states of next belief states,

$$p(B)V(B) = \sum_{B' \in \text{next}(B)} p(B')V(B'). \tag{7}$$

Equation 7 is identical to Equation 3 except that belief states are used in place of global states. Thus the dynamic programming method use in CPs can be applied to DPs as well.

3.2 The Amount of Communication

The above formula also give us hints about how to calculate the total amount of communication (AoC for short) for a given policy. For a CP, although communication is not specified, the agents need to be able to observe global states, which means synchronization among the decentralized agents. Thus, this means that the agents need to communicate at all states. Assuming that each synchronization count as 1, then for each reachable state s, its contribution toward total amount of communication is the probability of reaching that state, i.e., $p(s)$. Thus, we can define $c(s)$, the total expected amount of communication at and after state s (i.e. how much more communication is needed when the system reaches state s):

$$c(s) = p(s) + \sum_{s' \in \text{next}(s)} c(s') \tag{8}$$

However, for the starting state s_0, since no synchronization is needed at that state, the above definition is changed: $c(s_0)$ is simply $\sum c(s')$. If s is a terminal state, $c(s) = 0$, since the agents do not need to communicate to realize the current state is a terminal state.

For a DP, we can define the $c(B)$, the total expected amount of communication at and after belief state B. Note that some belief states are synchronized via communication (the singleton belief states computed from K), and others are not communicated ($N - K$ belief states and starting state s_0):

$$c(B) = \begin{cases} p(B) + \sum_{B' \in \text{next}(B)} c(B'), & \text{if B is communicated;} \\ \sum_{B' \in \text{next}(B)} c(B'), & \text{otherwise.} \end{cases} \tag{9}$$

The total expected amount of communication of a DP is simply $c(B_0)$, where B_0 is the starting belief state, which just contains the starting global state s_0. Obviously B_0 is common knowledge to the agents and hence not communicated. Also, if every state $s \in B$ is a terminal state, $c(B) = 0$, since the agents would realize that the global state is a terminal state without communication. Such a belief state B is called a terminal belief state.

Thus, we have defined the value function and amount of communication of a belief state, and the way to calculate the EU and expected total amount of communication of a DP. This gives us the way to evaluate a DP.

4 Non-Conforming DPs

Up to this point we are only describing *conforming* DPs derived from a given CP, i.e., the DPs that follow the CP exactly (as seen by an outside observer that observes the

global state and joint action). As a result the EU of such DPs equals the EU of the CP, but the expect total amount of communication could be much less than the AoC of the CP. In other words, the conforming DPs offers no degradations of the EU but reduces AoC. This property of the conforming DPs is highly desirable, but it may also be quite limiting.

The problem is that it limits the kinds of DPs that might be derived from a given CP. Later we will show some ways to create non-conforming DPs from a CP. These non-conforming DPs may not follow the exact CP as seen by an observer. As a result, the EU of these DPs may change. In other words, they may degrades the EU of the CP. However, they also have the potential of further reducing AoC to the extent not possible by conforming DPs. This offers a tradeoff between EU and AoC when selecting DPs, which the conforming DPs lack. This is why it is interesting to study non-conforming DPs: it offers a wider selection of DPs and offers tradeoff choices.

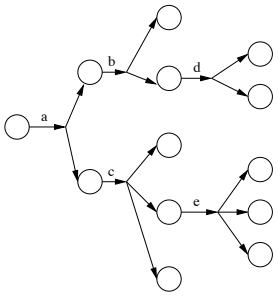

Fig. 11. The Tree View of a Centralized Policy

In order to derive non-conforming DPs from a given CP, we first examine how to derive new CPs from a given one. We are mostly interested at domain-independent techniques, which modifies the given CP based on its structure, not on domain knowledge. Specifically, since a policy can be viewed as a directed tree with each node representing a state, and the outgoing edges representing the next states, as in Figure 11. Note that the label of the outgoing edge represents the action to be taken at that state, and the action result may be non-deterministic, resulting in multiple resulting states from a single action.

One domain-independent technique for derive a new CP from an existing one is *terminating*: to mark one or more non-terminal states in the original policy to be terminal states in the new policy. This technique is illustrated in Figure 12. The resulting policy would then terminate when reaching the marked states. Clearly, the new policy may not receive any reward beyond the marked states, therefore the expected utility of the new policy is different from the original policy's. For sequential decision-making processes, marking a state terminal simply means not to perform any further tasks, so any state can be marked terminal.

Another domain-independent technique is *merging*. Illustrated in Figure 13, this technique marks one state to be merged: grafting the subtree beyond another state (tar-

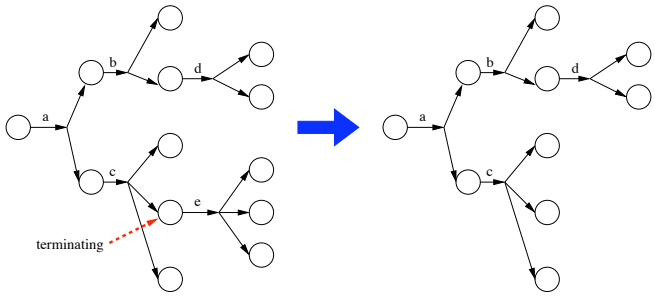

Fig. 12. Terminating a State

get state) onto this state, replacing the original subtree of the merged state. Such merging operation would require that the merged state is compatible to the target states, so that the subtree grafted is both complete and conflict-free. Here, being complete means that the grafted subtree covers all reachable states based on the series of the actions prescribed in the subtree; and conflict-free means that each state represented by the grafted subtree is valid (i.e. exists in the state space).

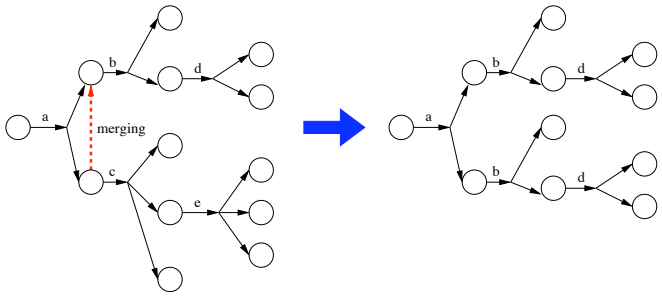

Fig. 13. Merging a State

Typically, the part of state space beyond the merged state should be topologically identical to the part of state space beyond the target state, so that the complete and conflict-free constraints are met. Usually, merging happens between sibling states: two states correponds to the different outcomes of an action. The merged state is identical to the target state except the minor difference that differentiate them.

For example, an action produces two possible outcomes o_1 and o_2 (and ends in state s_1 and s_2), but the only difference between them is that o_1 produces utility 10 and o_2 produces utility 5, and otherwise they are exactly the same in the problem solving. This means that for any episode containing s_1, we can replace s_1 with s_2 (and vice versa), and the resulting episode is also valid. Thus, the structures of the state spaces beyond s_1 and s_2 must be the same.

Since utility value is typically a function of the outcomes, the calculation of the value of the merged state is straightforward: simply use the same process of calculating the utility value of the targeted state, but replace any occurrance of outcome o_2 with o_1 (assuming s_1 is merged to s_2). This technique is used in our calculation when dealing with merged states. The calculation of AoC is even simpler: it is simply the AoC of s_2 multiplied by the factor $p(s_1)/p(s_2)$ — to account for the fact that the probabilities of getting o_1 and o_2 are different.

4.1 Generating Non-Conforming DPs

Given the above discussion, one method of generate non-conforming DPs would be to generate new CPs by applying various combinations of the aforementioned techniques, and to create conforming DPs for the new CPs. However, this method is not tied to the communication issue, and therefore may not be efficient in our search for DPs that reduce communication. Therefore, we introduce a method that ties the choice of techniques with communication, by putting the process of selecting the non-conforming technique into the process of generating next belief states. Specifically, during the process of deciding the mapping of Com^X and Com^Y according to the local history matrix M, we can create alternative strategies of generating next belief states. Previously we have defined the *default* strategy, and an alternative, the *hill-climbing* strategy. Now we would like to perform the *terminating* and *merging* operations on these alternatives and thus producing new alternatives. Similarly, these operations are translated into the matrix representation used in previous discussions.

First we discuss the terminating operations. We identify two strategies for applying this operations:

Terminating 1 (T1) : For a given belief state B, one strategy is to mark all of its next states (i.e. any state in N) terminal. In this case all next belief states are terminal, and as a result no more communication is needed. This corresponds to marking all states in M terminal and therefore the next joint action (in the form of $(a_X|a_Y)$) for each state in M is $(\lambda|\lambda)$, where λ means no action. In this case M automatically satisfies the no-ambiguity condition, thus K is empty (by default).

Terminating 2 (T2) : We notice that the above method may result in drastic changes — it eliminates all further communications, but also eliminates all further activities. As a result, the expected utility may suffer. So an alternative method is to keep the $N - K$ belief state intact (based on the conforming alternatives) and mark only a subset of the rest of the next states. Thus, the communication decision remains the same as the conforming alternatives, but some of the synchronized next belief states would be terminal. This may reduces communication if communication is needed in the further stages for those belief states according to the conforming strategies. And since it only marks a subset of the next states, the degradation of utility is more limited compared to T1.

Now we study the merging operations. Since communication decisions are based on local history sets (LH^X and LH^Y) rather than individual states, we focus on merging

between *LH* sets. The goal is to merge communicating *LH* sets to non-communicating *LH* sets, so that the communication on the merged sets can be saved. Thus, the merging happens in the local history level rather than the state level: it replaces one local history with another one.

Merging : for a given belief state B, examine the *LH* sets (the rows and columns of M). For example, if a row is ambiguous (not counting elements already marked or merged), we try to find another row which is compatible to it but is unambiguous. If such a row exists, we can then merge each element in the original row to the same column element in the target row. The same can be applied to columns as well.

4.2 Putting Things Together

So far we introduced the default communication strategy, the hill-climbing strategy, and 3 methods (T1, T2, Merging) for generating non-conforming alternatives. Thus, for a given belief state B, we can generate several alternatives, which lead to different sets of next belief states. By choosing different alternatives at these belief nodes, we can obtain a great number of DPs. Of course, except for the default strategy, all strategy and methods are not applicable to each belief state. Furthermore, we can add criteria to limit the number of alternatives generated by these methods. These criteria may involve the calculation of EU and AoC, and therefore the selection of DPs represents a reasoning process — a criteria based constraint optimization process — for deciding what alternatives to take. Since the value of a policy is generally unknown unless it is fully explored, these criteria have to rely on currently available information and estimations, not the exactly value. One of the available data comes from the default policy, i.e. the policy that chooses the default strategy at all belief states. Hence we define the *default value* of a belief state to be the value of the belief state when the default strategy is applied at the beyond this point, and similarly the *default amount of communication*.

Clearly, there might be many alternatives for a given belief state, but it is only feasible for us to consider a few of them due to the computational complexity involved. In our experiments, we choose the following algorithm for finding the alternatives for a given belief state B:

1. Apply the default strategy, which generates the default alternative. If no communication is needed under the default alternative, stop here.
2. Apply the hill-climbing strategy. If the result is not identical to the default, add this hill-climbing alternative. In our experiments these are the only conforming alternatives.
3. The first non-conforming alternative is obtained by applying the T1 operation, i.e., let the problem solving stop at the current belief state, *if* the difference between the current stopping reward and the default value is small then the default AoC times a constant (the cost of communication). Intuitively, this criteria means that it is not worthwhile to continue the problem solving when the cost of communication outweighs the potential utility gain by further actions.
4. Next, apply the T2 operations on the conforming nodes generated so far — marking a communicating next belief state terminal *if* its default AoC is greater than the

probability of reaching it, and its default value is below the default value of the current belief state.

5. Finally, apply the merging operations on the conforming nodes. In our case, for each communicating *LH* row (or column), merge the row (column) to the compatible row (column) with the best default value, if such a compatible row (column) exists.

Thus, our algorithm of generating DPs from a given CP is a search-and-explore process that constructs a bipartite graph consisting of belief states and alternatives (choices of generating next belief states), illustrated in Figure 14. The edges from a belief state to alternatives reflect the strategy used, and the edges from an alternative to belief states reflect the next belief states after applying the alternative. The process finishes when all belief nodes are explored. Using such a representation, a DP is then simply a mapping from each (non-terminal) belief state to one of its alternatives. Since not all belief states are reachable under such a mapping, the size of DP can be limited to reachable belief states only.

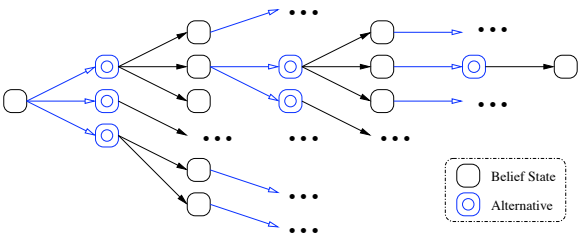

Fig. 14. A Bipartite Tree of Belief States and Alternatives

Furthermore, we notice that for any non-terminal belief state, the default strategy is always applicable, but other strategies such as hill-climbing and non-conforming strategies, may not be always applicable. Thus, to describe a DP, we just need to list the non-default alternatives for the reachable belief states. Thus, an algorithm that tries the valid combinations of the the non-default alternatives in the bipartite tree can enumerate all possible DPs in the tree.

The DPs generated in this process would then have different EU and AoC characteristics, and therefore help us understand the tradeoff between them, which is very important for the design of coordination policies. In the next section we shall study how this decision-theoretic approach relates to mainstream multi-agent planning methodologies and how this work complements them.

5 Multi-Agent Planning

Although tools such as MDPs are commonly used in decision-theoretic planning research, they are yet to be used extensively in multi-agent system research. Typically, in the area of multi-agent planning, a hierarchical task model is often used, which divides

tasks into subtasks and specifies the relationships of the tasks (subtasks). A search process that involves constraint optimization or satisfaction, is often used to find solutions. This is different from the state-based models used in decision-theoretic planning, for example the decentralized MDP model of multi-agent decision process framework used in this work. In order to connect the multi-agent decision process approach discussed in this work to those types of multi-agent planning work, we need to construct a state space representation for the hierarchical task models. In this paper we used the TAEMS model [4] as an example to show how our approach relates to multi-agent planning.

5.1 The TAEMS Task Model

TAEMS is a typical example of a hierarchical task model. A graphical representation of a TAEMS task structure for a PC build-to-order process is illustrated in Figure 15. TAEMS describes high-level tasks (task nodes) and lower-level tasks (method nodes), the outcomes of methods (the q,c,d distributions), and the composition of utility from subtasks to higher-level tasks (the *quality accumulation functions* such as **q_max**, **q_sum**, **q_min**, etc.) It also allows interrelationships (such as *enables*, which is a precedence relationship) between tasks. Typically, there is also a deadline associated with a TAEMS task, meaning that the problem solving must finish before that time.

The ability of representing uncertainty - in terms of the uncertain method outcomes described by the (q, c, d) distributions - and interrelationships allows TAEMS to specify complex task structures. Another important feature of TAEMS is its ability to represent multi-agent tasks, by specifying separate TAEMS tasks of each agent, and combine the root tasks of the agents into a global root node via a **sum** quality accumulation function.

Figure 16 shows a multi-agent task for agents A and B, with nonlocal interrelationships (such as A2 *enables* B2) between the tasks in different agents. Thus, TAEMS gives us the ability to model non-local effects (NLEs) in the agent planning process. In this paper we will be using this task as our example in our experiments.

5.2 Agent Task Scheduling and Coordination

Before we try to make a mapping from a TAEMS multi-agent task to an MDP (per the centralized view), we would like to examine how agent task scheduling and coordination is done, and how it is done in a decentralized fashion, so that we can correspond

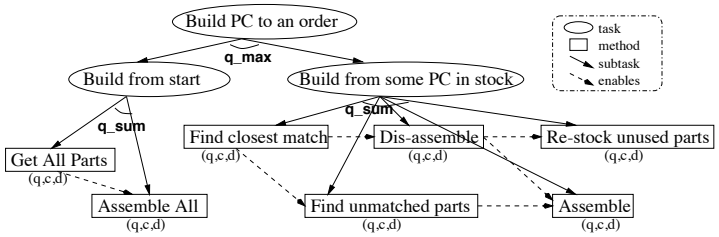

Fig. 15. A TAEMS Task Structure for Building PC to an Order

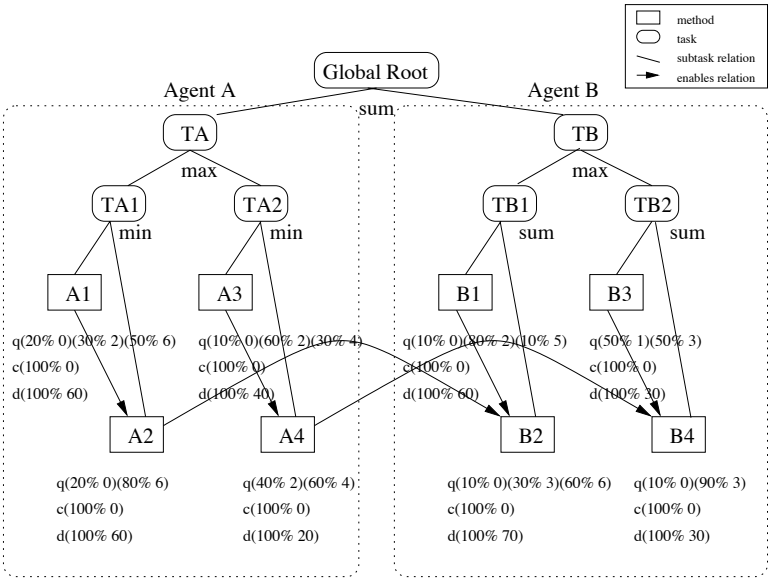

Fig. 16. A Multi-Agent Task

that with the problem solving in a decentralized decision model, and therefore make the mapping a natural one.

The TAEMS agent (i.e. the agents adopting the TAEMS model of task based problem solving) typically has a scheduler module, which formulates a local plan for the agent, and a coordination module, which interacts with other agents' coordination module to coordinate the agent activities. The scheduler and the coordination module also interacts within the agent.

From a communication perspective, the coordination module sends and receives *control* messages to and from other agents, thus performs the communication actions in the decision-theoretic view. This corresponds to the communication decision part of the local policy of the agent. On the other hand, the scheduler does not communicate with the counterparts in other agents. Although the scheduling approach used in the scheduler could be very complex, from a decision-theoretic view the function of the scheduler is simply to decide what next local action to perform. Thus, this corresponds to the action part of the local policy in our framework.

To illustrate our points, let us study some details of the scheduling and coordination activities using the problem in Figure 16. For example, under the design-to-criteria (DTC) [7] scheduling framework, each agent tries its best to construct a local schedule that meets the criteria (actually, DTC produces not one, but several alternative schedules, and evaluates them according to the criteria, and suggests the best one). A schedule tells the local action sequence of the agent, therefore it usually has a *linear* structure. However, when unexpected events happens, the agent may re-invoke the scheduler to produce a new schedule during the course of problem solving. These unexpected events may include task failures, changes of agent commitments, etc., that make the current

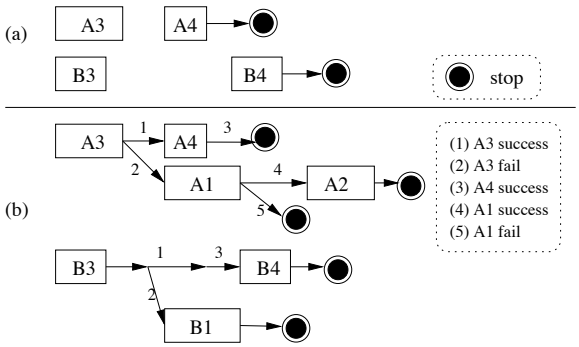

Fig. 17. Schedules

schedule unfeasible or undesirable. Thus, a complete schedule may have branching structures that reflects the effects of rescheduling. Figure 17 shows an example of agent schedules (using the deadline of 160), with the top part showing the linear schedule in each agent (i.e. the initial schedule of each agent, without any rescheduling), and the bottom part showing the actual schedule taking into account the effects of rescheduling, (i.e. how each agent's schedule responds to the events in the system.)

Figure 17 also shows the events need to be communicated between the agents in order to understand which branch of the schedule to take in actual problem solving. In the bottom part of the figure, we see that B's schedule has a branching structure after the outcome of A3 (event 1 and 2). Since A3 is a non-local task, its outcome is not observable to B and therefore communication is needed. Under the framework of the Generalized Partial Global Planning (GPGP) [3] coordination, a coordination framework is often used for TAEMS agents, agent communication is tied to the dynamics of the commitments between the agents [8], which in turn is tied to the outcome of tasks. In our case, agent A has a commitment to complete task A4 before time 60. This commitment can be viewed as mutual knowledge at the beginning of the episode. However, when task A3 fails (in TAEMS, that means the quality outcome is 0), the commitment is not feasible any more, since task A4 has to be enabled by A3 (i.e. need a positive quality outcome from A3.) Thus, a message would be sent to agent B indicating that the commitment could not be kept. This is the only communication needed for B to follow the schedule, so the amount of communication needed here is the probability A3 fails, which is 0.1. Also it is easy to calculate the expected total utility of the agents, which is 6.948: for A, the expected utility of its schedule is 2.448, and for B, 4.5.

From the above example we notice that the branching structures of the schedules clearly resemble a MDP policy representation. The action sequence indicates the order of tasks, and the branching structure indicates the need for synchronization (where non-local outcomes are required), so that there is no ambiguity toward the choice of the next action. Thus, our model of a decentralized policy does correspond very well to the actual problem solving in typical planning agents. This important characteristic allows us to interpret the a DP via the planning language, and vice versa.

5.3 Constructing Centralized MDP from TAEMS

The first step of constructing a centralized MDP from a TAEMS multi-agent task structure is to decide how to define stages, i.e. what signal the transition to new states. It is easy to see that the completion of a task (as depicted in Figure 17) leads to the execution of the next task, but since we are dealing with *concurrent* schedules, the completion of a task in either agent signals a change in the state of the system. Thus, to reduce the size of the resulting MDP, the stages are set according to the completion of any task in both agents, not based on time (the discrete time model in TAEMS, which may cause a very large state space when the time frame is large).

Under such a stage model, the agent actions lead to the state transitions are as the following:

- when agent X's task finishes and agent Y's task is still running, X's possible next local actions include to start an eligible task, or to wait until Y's action completes, and Y's next local action is to continue running the unfinished task.
- when Y's task finishes and X's task is still running, X's next local action is to continue running the unfinished task, and Y's possible next local actions include to start an eligible task, or to wait until X's action completes.
- otherwise — e.g. at the starting state, when both X's and Y's task finish at the same time, or when one agent's task finishes while the other agent is waiting — X may start any of its eligible task (or idle), and so may Y, however they cannot both be idle.

In other words, each agent's local actions now include *wait* and *continue* (note that we treat idle the same as wait), but as defined above, only a subset of the joint actions is legal.

For each legal joint action, there might be a number of resulting states that reflect the uncertain task outcomes in TAEMS. Thus, a state represents a complete history of the problem solving (according to the path leading to the state), which defines the actions in each agent and the outcomes of the actions. The transition probabilities are calculated based on the probabilities of the outcomes, which is specified in TAEMS.

Since the quality calculation in TAEMS is complex, we define the reward of the states to be 0 for all non-terminal states, and define the reward of the terminal states by applying the quality calculation process according to the task outcomes indicated in the states. In addition, we can apply the same quality evaluation process on the non-terminal states and the results are called the *stopping rewards* of the states, i.e. the reward if the problem solving stops at a state.

The interrelationships specified in TAEMS are incorporated in the construction process through the calculation of the *eligible* tasks. For example, if task a *enables* b, then b is an eligible task only when the state contains the history of a positive quality outcome of task a.

Since there is a deadline for the TAEMS task, a state becomes a terminal state if there is no time for performing any additional task, or if there is no further action available.

This completes our TAEMS to MDP mapping and we now obtain a MDP representing the global view of the problem solving. The next step is to obtain a centralized

policy for the MDP. This can be done through many ways, including using dynamic programming for solving the optimal policy, standard value iteration or policy iteration methods, learning techniques, or other approximation methods.

In actual implementation, the state space may still be too large to fit in the computer memory. However, since not all states are reachable given a policy, and the most of the states in the state space are only visited during the process of obtaining a policy, we can save the space (at the cost of additional computation time) by saving only the reachable part of state space for the policy, and generate the other states on-the-fly when computing a policy (for example, when using dynamic programming to compute the optimal policy.)

In our experiments we use the TAEMS task structure depicted in Figure 16 and construct the MDP using the above method. We obtain the optimal policy for the MDP by using dynamic programming. The optimal policy has 278 states, with an EU of 7.27425 and AoC of 5.5425, according to the evaluation method described in section 3.

In Figure 18 we show the sequences of joint actions that might occur in an episode when this policy is used, by analyzing the policy graph. As listed, there are five possible sequences. Sequences 2-4 differ from sequence 1 after the end of the fourth stage (the first 4 stages of sequences 2-4 are the same as those of sequence 1, therefore are omitted in the figure). Similarly, sequence 5 differs from sequence 1 after the end of the second stage. Note sequence 3 is identical to sequence 2 except the last stage.

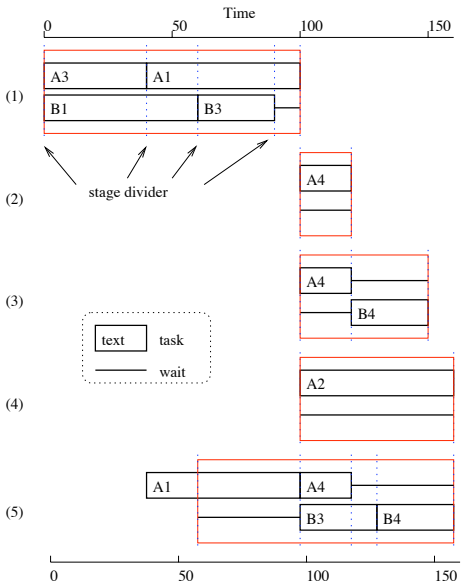

Fig. 18. Joint Action Sequences of the Optimal CP

It is easy to see that each sequence represents a concurrent schedule that consists of the local schedule in each agent, without any branching structure. The difference among

the sequences represents the branching of the CP graph. Thus, the key for deriving DPs from a CP is to understand when and how to switch from one sequence to another. In between the switchings, the local schedule of each agent is linear, meaning that no communication is needed in order to decide the local actions to be performed in each agent (i.e. no ambiguity). In other words, one possible communication policy is to communicate at the end of the stages that are followed by branching structures. This policy can be further refined by looking at the branching structures of local action sequences rather than joint action sequences (and hence make local decisions of switching to a different sequence), with the goal of identifying unambiguous local action sequences and thus lead to reduction of communication.

Translating to the languages of our decentralized multi-agent decision process, the linear part of an action sequence means that the common belief state simply "grows" from one stage to the next, without the need of communication; while the switching among the sequences indicates ambiguity toward next action and thus lead to multiple belief states in the next stage. The switchings are based on local beliefs, and communication is needed in order to obtain the information for deciding which branch to take. In our model, this is formalized by the the choice of communication and the process of deciding the formation of the next beliefs states based on communication decisions.

6 Experimental Results

To evaluate our approach, we implemented our CP to DP method, applied the method on the optimal CP generated above, and evaluated the expected utility and average total communication of each DP. Again, the TAEMS task structure is illustrated in Figure 16, with a deadline of 160, and the CP is the optimal policy illustrated in Figure 18. We obtained 160 different DPs from the CP and Figure 19 shows how their EU and AoC values are scattered.

To give a comparison and also some details about the results, the following table lists the EU and AoC of various policies:

	EU	AoC	comment
Centralized Policy (CP)	7.27425	5.5425	
Heuristic Planning DP (DTC/GPGP)	6.948	0.1	see section 5.1
Default DP	7.27425	1.07	
Conforming DPs	7.27425	0.996 – 1.07	total 12 DPs
All DPs	2.7 – 7.27425	0 – 1.07	total 160 DPs

Immediately, we notice that even the default DP (EU=7.27425, AoC=1.07) reduces communication greatly compared to the AoC of the centralized policy (5.5425). This is done without any loss to EU. But the AoC can be further reduced, even to 0, while still having good EU (the rightmost data point on the EU axis has a EU of 6.228). However, the conforming DPs offer only a limited very range of AoC compared to nonconforming DPs, which reduce AoC but degrade the EU.

Compared to the DP obtained through traditional multi-agent planning, e.g. the heuristic planning approach using DTC and GPGP, we observe that the heuristic DP

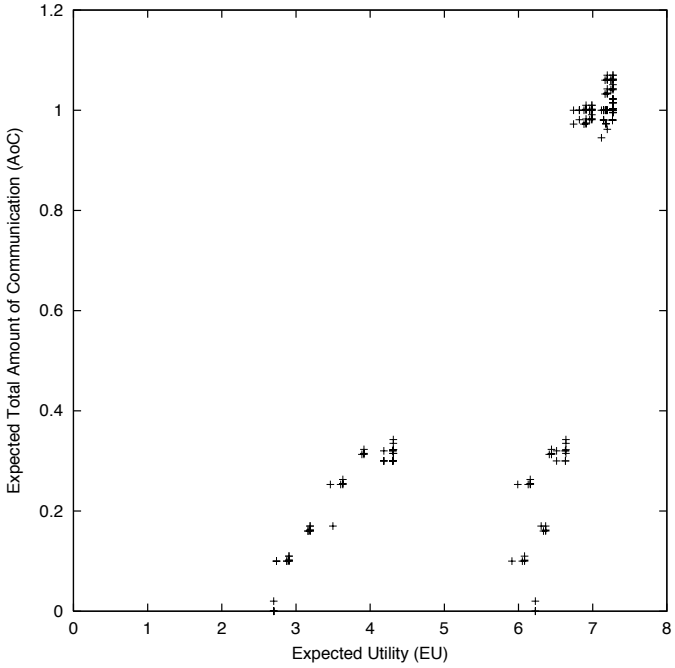

Fig. 19. Utility/Communication of Some Generated Policies

is a very good one, with only 0.1 AoC but EU of 6.948, very close to the optimal policy. In comparison, with the same or less AoC, our best generated DP has a EU of 6.228. This tells us that in general, our method cannot replace the role of traditional multi-agent planning, which typically generates good performance, and are easy to understand. Even though we used the optimal CP as our input, there is no guarantee that we obtain a DP that dominates heuristic DPs (i.e. having better EU and less AoC at the same time).

However, the good performance of the heuristic policy can give us hints about how to reduce communication. For example, typically DTC/GPGP has the notion of *failure*, i.e., task quality outcome equals 0. A simple form of GPGP will communicate when commitment fails (due to the failure of the task), and not communicate otherwise (positive quality). Since failure is often a low probability event, and the dichotomy of failure/success in fact groups all positive quality outcomes into one outcome category (hence to the effect of merging in our model), the heuristic DP is well positioned to reduce AoC. Similar techniques can also be applied in our method to generate low AoC DPs.

Another interesting pattern shown in Figure 19 is that the data points are somewhat "clustered". The points can be roughly divided into 3 clusters: top-right, lower-right, and lower-left. Note the AoC gap from 0.4 to 0.9, and EU gap from 4.5 to 5.5. By looking at the details of the DPs we notice that gaps are largely due to the effect of some non-conforming operations, such as marking some belief states terminal, i.e., to stop the problem solving at an earlier stage and therefore abandon all further communications, or the merging of one local outcome to another, therefore causes a gapping change of AoC and EU. Since conforming alternatives typically have less variance of AoC (and no EU variance at all), it is natural that these points are clustered.

As an example, let us examine the DP corresponding to the data point (EU=6.228, AoC=0).

- At the beginning of the first stage (time 0), A starts task A3 and B starts task B1. The next stage will begin at time 40, and will have 3 states. By default strategy there is no communication needed, so the next belief state contains these 3 states.
- A3 finishes (at time 40), and the second stage begins. A starts task A1 and B continue B1. Next stage will have 9 states. Merging is applied to B1's q=0 outcome (which contains 3 states). The rest 6 states — the next belief state — need no communication.
- B1 finishes (at time 60), and stage 3 begins. A continue A1 and B starts B3. From the current belief state there are 12 next states, and by default strategy they become the next belief state. The next stage begins at time 90, when B3 finishes.
- When B3 finishes, the current belief state contains 12 states and 36 next states. During this stage (stage 4), A will continue A1 and B waits. The merging operation is applied to 8 of the next states, and the rest of 28 states become the next belief state. The next stage will start at time 100, when A1 finishes.
- During this stage (stage 5), agent A would choose either A2 to A4, depending on the previous outcome of A3 and A1: if the q(A1) = 6 or if q(A1) = 2 and q(A3) = 0, — in other words, if q(A1) > q(A3) — choose A2, otherwise A4. Clearly this is a local decision. B simply waits (idle). The next stage will have 56 states, and 50 of

them are merged, so the next belief state contains only 6 states, all of them terminal states. So the problem solving finishes at the end of the stage.
- No matter what outcome A's action in the last stage is, the problem solving ends, since all other states (the 50 next states of the previous belief state) are merged to the terminal states.

Essentially, this DP contains 3 merging nodes, and as a result the agents choose not to report any unexpected local outcomes, by merging these outcomes to other ones. As mentioned before, merging requires compatible states, but in our case it is not satisfied because of the interrelationships, i.e., the state space following the q=0 outcome of an enabler will not be the same as that of the q>0 outcome. But since we calculate the reward of states based on the whole history of action outcomes, we can artificially create the same state space follows a q>0 outcome for a q=0 outcome, and ignore the outcome of the enablees (i.e., pretending perform the enablee task but performing no-op instead) when calculating the rewards. Thus, we can safely apply merging operations for q=0 outcomes.

7 Summary

In this paper we propose a method for deriving decentralized multi-agent policies from centralized ones. In the past, the design of decentralized policies has been limited to the use of *ad hoc* heuristic methods, and the results are often domain-specific. By using this method, we now have a domain-independent, systematic way of developing decentralized policies. Furthermore, this method provides a bridge between centralized policies and decentralized policies, thus allow us to connect the research in these areas and often more insights.

Another use of this method is to provide ways of implementing a centralized multi-agent policy to a decentralized system. Since most multi-agent systems are inherently decentralized, with each agent sees only a partial view of the system and has to communicate with other agents to exchange information, this method allows a policy which is based on a centralized (i.e. global) view to be adopted by the decentralized agents. The use of non-conforming policies also can be viewed as changes to the centralized policy, therefore provides insights and feedback for the design of centralized policies.

Also, this method explores the possibility of making tradeoffs between the expected total utility and the amount of communication to be used in multi-agent cooperation. The underlying research question is how to balance the cost of communication and the amount of uncertainty in the system. Communication in multi-agent systems can be viewed as the dynamic process of obtaining (exchange) information, which reduces uncertainty in the system but may incur a cost. Thus, it belongs to one of the fundamental domain in artificial intelligence — the question of the value of information. The DPs generated by this method allows the agents to select from several alternatives each with different EU and AoC, therefore can respond to different situations where the amount of uncertainty and the cost of communication may differ.

By connecting the decision-theoretic approach to the traditional multi-agent planning research, this method also offers insights toward the problem of uncertainty handling when designing multi-agent plans. There, communication can be viewed as the

way of monitoring and acting to the dynamics of the commitments. Thus, the communication decisions correspond to the agents' decisions regarding the commitments, and therefore the study of communication policies gives important hints on how to deal with commitments in multi-agent planning.

The problem of designing good decentralized policies is a very complex problem, in terms of both its computational complexity and the complexity of the model itself, since we have to understand exactly what information is available to an agent and what is not. However, we believe that, when the focus of computation shifts from the centralized perspective to the decentralized perspective (due to the advancement and the ever-increasing popularity of networked computing and multi-agent systems), to understand reasoning, planning, and decision-making in a decentralized, situated agent would be more and more important. One important direction of our future work to further enforce the assumption of decentralization. Specifically, in this paper we assume that the knowledge of the global state space is available to every agent in the system. In the future, we would relax this assumption and examine how to generate decentralized policies when each agent only has its own, partial view of the global state space - apparently a much more realistic reflection of actual systems.

8 Appendix

Although decentralized agents decide their local actions independently, it should not be taken for granted that the distribution of their joint outcomes is simply the product of their local outcome distributions. There is a different between the independency of the decision making in each agent and the independency of the local outcomes (a factual distribution regardless of the decisions). A joint outcome is an *interpretation* of the local outcomes in each agent, not an *equivalence*. In fact, the outcome distribution of a local action may be affected by the actions in other agents, or the outcome of the actions in other agents. Thus, the observation of local outcome may allow an agent to infer the outcome of tasks in some other agents. In decision theoretic planning terms, this means that the local outcome is a partial observation of the global state.

For example, let us examine the following scenario: two robots, A and B, roam in a grid world, and their current positions are depicted in Figure 20. Suppose each robot takes a random move to a neighboring block. If the joint goal of the robots is to meet (to be in the same block), then in some cases (A moves to the right and B up, or A moves down and B to the left) the robots meet — therefore enter a terminal configuration. Here we assume that each agent is able to observe the meeting event, i.e., recognize the meeting locally. Thus, the decision of A moving to the left block may have different observed outcomes depending on B's move. It is easy to see that the combination of local outcomes have different interpretations.

Using a Markov decision process representation, consider the policy trees illustrated in Figure 21. There, the joint action is (a|b) and the outcome distribution for action a is 60% a=1 and 40% a=2. Similarly 60% b=1 and 40% b=2 for action b. So when the outcomes of a and b are independent, there are 4 joint outcomes (see part (1) of Figure 21.) The same local outcome distributions are to be observed for the policy tree depicted in part (2), however there are only 2 joint outcomes. In this case the local outcomes in

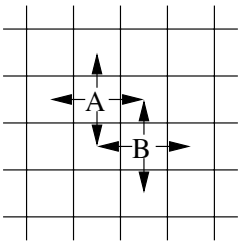

Fig. 20. An Example of Joint Outcome Interpretation

each agent are not independent any more. Each agent can infer the outcome of the other agent's action.

Furthermore, in (3), the policy tree lists a belief state containing two states, one with joint outcome (a|b) and on with (a|c). Now the same local action a produces different outcomes when belongs to different joint actions.

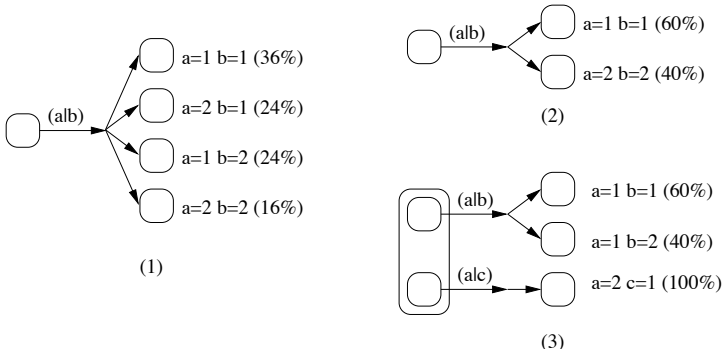

Fig. 21. Various Types of Joint Outcomes

Using the method mentioned earlier, let us compute the local history matrix M for the starting node in each policy tree lists in Figure 21. Again, the rows reflect local history sets for agent X, and columns for Y:

$$(1) \quad \begin{pmatrix} \text{a=1 b=1} & \text{a=1 b=2} \\ \text{a=1 b=2} & \text{a=2 b=2} \end{pmatrix}$$

$$(2) \quad \begin{pmatrix} \text{a=1 b=1} & \epsilon \\ \epsilon & \text{a=2 b=2} \end{pmatrix}$$

$$(3) \quad \begin{pmatrix} \text{a=1 b=1} & \text{a=1 b=2} & \epsilon \\ \epsilon & \epsilon & \text{a=2 c=1} \end{pmatrix}$$

In some task models, such as TAEMS, local outcomes distributions are independent to the actions in other agents. Under such models, the state space typically has some

symmetrical structures: the subspace after one outcome is identical to that of another outcome, hence allows operations such as *merging* to be easily applicable to a MDP policy. But we should be aware that these models are not the only models, and therefore these operations may not be applicable under other models. However, the treatment of communication decisions, and the transformation method discussed in this paper is applicable to all MDP policies, no matter how the outcomes are correlated.

References

1. Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, 2000.
2. C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI-99)*, July 1999.
3. Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1992.
4. Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 214–217, 1993.
5. C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.
6. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.
7. Thomas A. Wagner, Alan J. Garvey, and V. R. Lesser. Criteria directed task scheduling. *Journal for Approximate Reasoning: Special Scheduling Issue*, 19:91–118, 1998.
8. Ping Xuan and Victor Lesser. Incorporating uncertainty in agent commitments. In *Intelligent Agents VI: Agents, Theories, Architectures and Languages (ATAL), Proceedings of The Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99), Lecture Notes in Artificial Intelligence 1757*. Springer-Verlag, 1999.
9. Ping Xuan, Victor Lesser, and Shlomo Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the Fifth International Conference on Autonomous Agent (AGENTS 01)*, 2001.