Submitted to journal on Autonomous Agents and Multiagent Systems

Modeling Cooperative Multiagent Problem Solving as Decentralized Decision Processes

Ping Xuan, Victor Lesser, and Shlomo Zilberstein

Department of Computer Science
University of Massachusetts at Amherst
Amherst, MA 01003
pxuan@cs.umass.edu, lesser@cs.umass.edu, shlomo@cs.umass.edu

Abstract

We present a formal framework for modeling and control of cooperative multiagent problem solving. Such a framework is crucial for understanding and evaluating various design choices based on first principles. This framework allows us to define precisely the notion of optimal cooperation and analyze its complexity. Communication decisions are introduced to allow agents to reason explicitly about coordination, capturing effectively the decentralized nature of the problem solving process and uncertainty in the system. Because finding optimal control is intractable, we develop several approximations and evaluate their effectiveness. The result is an integrated decision-theoretic framework for both agent planning and coordination.

1 Introduction

Cooperative multiagent problem solving is an important and active field of research in distributed AI. However, the design of cooperative multiagent systems is frequently based on *ad hoc* principles. The lack of a mathematical, quantitative, and decision-theoretic foundation for studying cooperative multiagent system has made the design process difficult. Without such a framework, system designers do not know what defines the optimal approach, how to evaluate the system, and what improvements should be pursued. Decision theoretic planning has been a powerful technique in traditional AI (single agent problem solving) [5], but applying the same techniques to multiagent systems remains a challenging problem.

A good theoretical model must be able to capture the characteristics of a cooperative multiagent system, such as:

- Based on first principles - the system must reflect rational problem solving principles in terms of maximizing the expected utility [18] (in this case the global utility of the system).
- Decentralization - the system is inherently decentralized, meaning that each agent only observes part of the global state of the system.
- Autonomy - the agents must be able to make individual decisions based on the information available, requiring a local decision process in each agent.
- Uncertainty - the system must be able to represent the uncertainty in the system and also in each agent's reasoning.
- Communication and Coordination - agents may exchange additional nonlocal knowledge and manage the interdependency constraints of their activities.

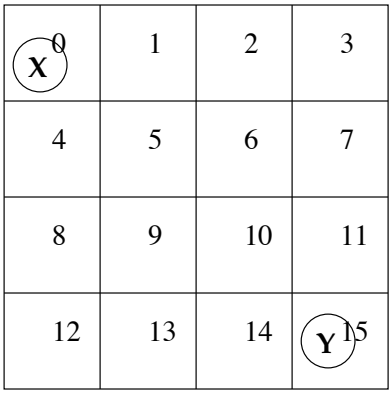

Figure 1: A Grid World Meeting Problem

In addition, it is also desired that the model would take a domain-independent, quantified approach and serve as an evaluation tool, be able to define optimality and thus have an understanding of the complexity of the problem solving, and be able to formally derive and represent approximation methods, easily map and evaluate existing approaches such as heuristics and design new ones.

To better understand the nature of multiagent cooperation problems, let us consider a simple two-agent meeting problem in a grid world. Figure 1 illustrates such a problem: there are two agents (robots), X and Y, in a 4×4 grid. Initially, X is at cell 0 and Y is at cell 15. The goal is for the agents to meet each other (to be in a same grid cell) before a certain time to receive a global reward. If they meet in t steps, the terminal reward is $\beta^t R$, where R is a constant (set to 100 in our experiments) and $\beta \leq 1$ is a time discount factor. If at time T (deadline) the robots still have not met, the reward is 0.

Each agent can choose an action at each stage of problem solving, including to move left, right, up, down, or to stay where it is. However, there is uncertainty regarding the outcome of these actions: if an agent chooses to move, there is a probability q (called the success rate) that it moves to the neighbouring cell in the direction of the move, $(1-q)/4$ chance resulting in any of the other neighbouring cells (note that a cell in the middle has four neighbouring cells, but a cell on a border has three, and a cell at a corner has only two) and with the remaining probability $(1-q-k(1-q)/4$, where k is the number of other neighbouring cells) it stays in the current position (i.e., it gets stuck in the current cell).

The assumption here is that both agents know their own positions in the grid at any time, but they do not know the current position of the other agent unless they communicate. Furthermore, there is a fixed cost c for each communication between them. The problem is to design the agents so that they can get the highest expected utility, which is the final reward minus the communication costs incurred.

This simple problem illustrates the important elements and some challenges for building a theoretical model. For the agent local reasoning part, the uncertainty an agent faces is coming from three sources. First, the agent (say X) does not know where the other agent (Y) is, although it may know historic positions of Y through past communications. Second, X does not know Y's plan, which in turn may depend on Y's knowledge about X's position. Third, even if the position and plan is known, the agent cannot predict how well the plan will be carried out because the moves have nondeterministic outcomes. For the coordination part, there are numerous communication and coordination mechanisms that the agents may adopt. To model such an open-ended process directly would be impossible. To address these challenges, we propose a framework that does not directly model the reasoning and coordination mechanism, but rather focuses on the end-to-end behavior model. Namely, if two agents would generate the same actions for all problem solving episodes, they are considered identical even if their internal reasoning processes may be totally different. In other words, if given the same information, two agents produce same actions, they can be considered as identical. Thus, an agent can be described through a mapping from its possible information states to its actions.

Under this framework, if an agent knows the other agent's mapping *a priori* or can compute it online, then there is consensus about the agent's plan and therefore no need to transmit plan information. Such consensus often can be established through organizational context or implied in the communication process. For example, a commitment between two agents establishes the implicit common knowledge about their expected behaviors. In this context, communication can be viewed as resolving uncertainty about each other's information states, and coordination is therefore implied through the mutual understanding of the ensuing actions. Using this approach, we can now model communication as obtaining non-local information (and hence the change of information states), therefore allowing us to abstract away the unnecessary details of communication activities.

In this paper we shall elaborate our decision theoretical framework for formally modeling cooperative multiagent problem solving using decentralized Markov decision processes (DEC-MDPs), which were introduced in our earlier works [28, 27]. Recently, there has been a significant amount of research activities in this field, for example the work by Becker *et al* [2] in which an optimal algorithm has been recently designed for a special class of *transition-independent* DEC-MDPs, without allowing run-time communication between the agents. Also, Goldman and Zilberstein [10] have developed a myopic-greedy approach to communication in a special class of DEC-MDPs. The approach optimizes the choice of when to send a message assuming no more communication will be possible after that point. Thus far, no optimal algorithm has been introduced for the general case and, given the complexity of the problem, it is unlikely that an efficient and practical approach will be found. Therefore, it is important to explore the merits of different heuristic approaches such as the ones presented in this paper. Pynadath and Tambe's recent work [22] also compared this approach with other teamwork models and discussed the use of partially observable MDPs (POMDPs) formalism as a generalization.

In the next section we will first discuss several decision models and then describe our model and how it relates to existing ones. We will then define the decision process and show how to represent and evaluate approximation problem solving methods, especially in terms of coordination strategies. We will then discuss some communication policies and their implications, and use our example to show how to use our decision-theoretic framework to help us better understand and design problem solving strategies in a quantitative way.

2 The Decentralized Multiagent Markov Decision Process

Standard decision theoretic planning approach is considered the *de facto* way of quantitatively formulating single agent decision making under uncertainty. The model of agent problem solving is very simple and general: it is simply a sequential, repeated process of making an observation of the system and then choosing and performing an action. The system consists of the agent and the "environment". The outside world is usually modeled through some random process, and the agent's action will cause the system state (actually, it is really the agent state – the perceived system state in the agent, since the real system may be partially observable instead of fully observable) to change (according to the random process) into another state, and the cycle repeats until the problem solving is finished, which is noted through a terminal state. This decision theoretic model for single agent problem solving is illustrated in part (a) of Figure 2, where an action a may produce two possible outcomes: noted as a=1 or a=2.

In other words, a problem solving episode can be described through a sequence of agent states and actions, starting from the initial state and ending at the terminal state:

$$s_0, a_0, s_1, a_1, ..., s_i, a_i, ..., s_{T-1}, a_{T-1}, s_T \tag{1}$$

where s_i (i=0,1,...,T) are the states and a_i (i=0,1,...,T-1) are the actions, and T is the number of cycles in this episode.

The standard tool for modeling this type of problem solving is the Markov decision process (MDP) model. It assumes the *Markov* property of the system: the next state depends *stochastically* only on the current state and current action. A standard MDP is defined as the tuple $\langle S, A, p(s_j|s_i, a), r(s, a)\rangle$, where S is the

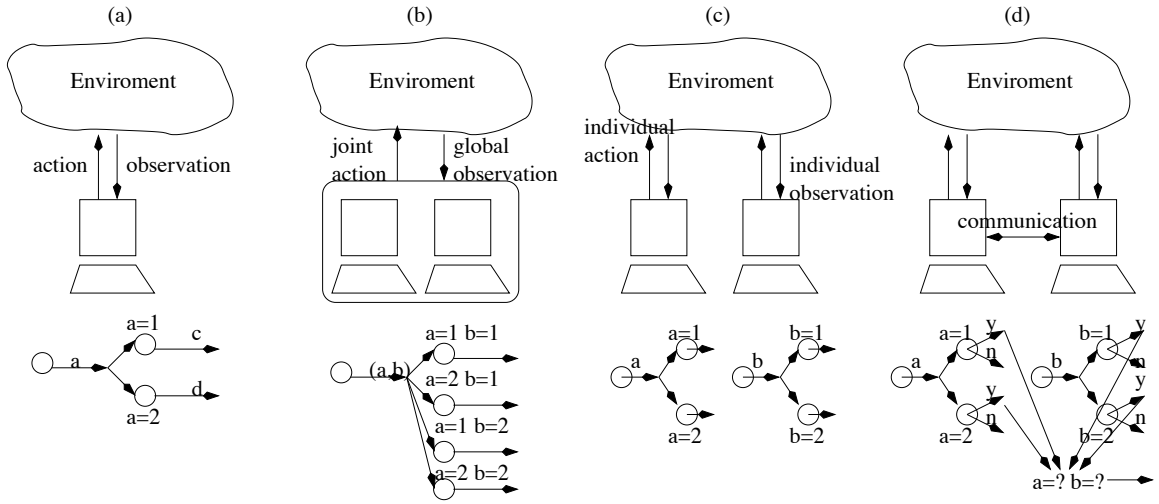

Figure 2: Decision Theoretic Models

state space, A is the set of actions, the reward function $r()$ defines the reward received when the agent is in state s and taking action a (it should be noted the reward function is often a function of s only, and that does not lead to any loss of generality, because it is easy to write a new $r(s)$ function to replace the action-dependent $r(s,a)$ function and have an equivalent MDP), and the local state transition probability $p()$ defines the probability of ending in state s_j when taking action a in state s_i. The time interval between two consecutive state transitions is called a *stage* (stages not necessarily have equal durations). In this paper we are dealing with fixed horizon problems, which means that the MDP only has T (called the horizon) stages and all states at stage T are terminal states.

Solving an MDP means finding a *policy* π, which is a mapping from state s to action a. A policy is a natural generalization of a schedule: it extends the linear structure of a schedule into a directed graph structure of a policy (by viewing the possible next states of a state as the child nodes of a graph node). A linear schedule thus can be viewed as one path from the root node to a leaf node, and a contingency schedule can be viewed as a subset graph.

An MDP policy reflects the Markov property of the system since the policy depends only on the current state s, not the whole history. The episode can then be written as

$$s_0, \pi(s_0), s_1, \pi(s_1), ..., s_i, \pi(s_i), ..., s_T \tag{2}$$

or simply,

$$s_0, s_1, ..., s_i, ..., s_T \tag{3}$$

when the policy is known. The total reward of this episode is $\sum r(s_i, \pi(s_i))$ and the probability of this episode occurring is $\prod p(s_{i+1}|s_i, \pi(s_i))$. The expected reward at state s, according to the policy is simply

$$v(s|\pi) = \begin{cases} r(s,\pi(s)) + \sum_{s'} p(s'|s,\pi(s))v(s'|\pi)), & \text{if s is not a terminal state;} \\ r(s), & \text{otherwise.} \end{cases} \tag{4}$$

The above equation allows us to calculate the expected total reward of policy π, which is $v(s_0|\pi)$. The algorithm is based on a simple graph traversing process (starting from terminal states then moving backwards toward the starting state).

To compute the optimal policy for a standard MDP, a standard dynamic programming (also called *backward induction*) algorithm is used: it involves the computation of the value function $v(s)$ for each state s in the MDP, which the *optimal* expected reward to be accumulated *at and beyond* this state. For terminal

states $v(s)$ is simply the terminal reward. Otherwise, $v(s)$ is the expected reward of taking the optimal action in state s, i.e.,

$$v(s) = \max_{a \in A}(r(s,a) + \sum_{s'} p(s'|s,a)v(s')) \tag{5}$$

The optimal action a at s is then the action chosen during the computation of $v(s)$. The algorithm for calculating the $v(s)$ can be described as:

CalculateValue(s):

1. If s is a terminal state, return $r(s)$.
2. For each possible action a at s, find $\mathbf{next}(s,a)$, the set of next states of s.
3. Calculate $v_a = r(s,a) + \sum_{s' \in \mathbf{next}(s,a)} p(s'|s,a) \times \mathbf{CalculateValue}(s')$.
4. Pick the a with the max v_a, this is the optimal action at s.
5. return v_a.

It is tempting to simply replace a_i by joint actions (i.e., (a_i^x, a_i^y) for two agents x and y) when extending this model to multiagent systems. The episode for multiagent problem solving will be,

$$s_0, (a_0^x, a_0^y), s_1, (a_1^x, a_1^y), ..., s_i, (a_i^x, a_i^y)..., s_{T-1}, (a_{T-1}^x, a_{T-1}^y), s_T \tag{6}$$

In fact, this is exactly the approach taken by Boutilier [4]. Such a model is what we call a *centralized* multiagent MDP. Solving it involves the same algorithm above except replacing single agent action a_i with joint action (a_i^x, a_i^y). A policy for such a MDP is called a *centralized* policy, which maps global states to joint actions, i.e., $\pi : s \rightarrow (a^x, a^y)$. This is illustrated in the part (b) of Figure 2. Comparing to (a), (b) simply replaces single agent states to joint states (based on global observations) and single agent actions to joint actions.

This is what the system designer or an outside observer may use to represent the episode, however it is not what the decentralized agent would see, because of two reasons: (1) the agent only sees its local state (s_i^x or s_i^y) rather than the global state s_i, and (2) the agent's decisions are not only based on current local state: it also may communicate to obtain nonlocal information.

The former point may suggest a *local* episode (in agent X, for instance) of:

$$s_0^x, a_0^x, s_1^x, a_1^x, ..., s_i^x, a_i^x, ..., s_T^x \tag{7}$$

and suggest a local decision process to decide a_i^x from previous s_i^x, i.e., a local policy $\pi^x : s^x \rightarrow a^x$. This is the model used in the study of decentralized control of finite state Markov processes [1, 15, 23]. There, both decentralized states (as results of individual observations) and partitioned (individual) actions are assumed, and each agent's decision is based on its local information (note this could include history information in addition to the current state s_i^x. Part (c) of Figure 2 illustrates this model. As we can see, in this model there is no communication between the agents, so an agent does not have a way to know the state of the other agent. It is as if the other agent is a part of the environment – a more complex environment that cannot be modeled as a simple stochastic process.

The latter point is the key to understand the decentralized nature of the system. In part (d) of Figure 2, we illustrate the computation model that our framework will represent. This different between (c) and (d) is that the agents in (d) can explicitly communicate with each other and therefore share information and coordinate plans. At the end of local action execution, the agents may decide if more information is needed. If yes, the agents would perform communication actions and obtain nonlocal information. In this case the agents would know both outcomes of a and b. If the agents choose not to communicate, then the decision will be based on existing local information only, just like the decision in (c).

Generally speaking, communication certainly should have an impact on local decisions. Information about other agents' current and past information certainly affects an agent's decisions. This means that in general, an agent's local decisions are history-dependent rather than based on current information only, even though the state transitions still follow the Markov property, namely the next state stochastically depends on the current state only, because communication itself does not change the global state of the system — it only changes the agents' beliefs about the current global state.

2.1 The Communication Issue

Communication is crucial for the agents to coordinate properly, since an agent usually only has a partial view of the system and is uncertain about the state of other agents. Here by communication we mean the process of obtaining nonlocal information rather than the actual conversation itself. We use the notion of *local states* to represent an agent's partial observation of the system, such as the agent's current position in our grid world example. We further assume that each agent can take local actions to change its local states (i.e., change the environment through the local actions and take another partial observation of the system), and the choices of actions in each agent are independent of the choices in other agents. Under the Markov assumption, the system state in the next stage depends statistically only on the current system state and the joint actions of the agents, i.e., the resulting positions of the agents depend only on the agents' current positions and the joint moves. In many cases, the Markov property holds at the local state transition level, i.e., an agent's next position depends on only on the agent's current position and the local move. As such, we can define a local Markov process for each agent. In this case, the social effect of the local actions are reflected by the definition of a global utility (or reward) function, in our example the reward as a function of the positions of the two agents.

Suppose that each agent can solve the *centralized MDP* independently to obtain the same policy (for instance, the optimal policy – here for the purpose of clarity we assume that the optimal policy is unique. When there are more than one optimal policies, the agent may use simple rules such as the lexical order to decide which policy to choose. A more general discussion of this issue is presented by Boutilier [4]). Then, if the agents know the global state at all times, they do not need any communication at all, since both the state and the policy is common knowledge and it is easy to find what local action to take according to the policy, which maps global states to joint actions. But if the agents only have partial information, communication of nonlocal state information (in order for the agents to observe the global state) would be needed if the agents want to behave exactly as specified in the centralized policy [26]. Likewise, when the decision problem is not a centralized MDP but a decentralized one, and the common decentralized policy is the implied consensus among the agents, all that may need to be communicated is the state information. In our example, if the two agents know each other's action if they know each other's information state, once the agents communicate and get that information, they know what action each would take. For cooperative agents, sharing such a consensus is essential for coordination. For example, the use of commitments is a way of ensuring compatible behaviors of the involved agents. Note that for self-interest agents this may not be always true, and the agents may need to maintain its models of the other agents' strategies, which in turn are based on their models of this agent's strategy. This creates a recursive representation, as discussed in the work of recursive modeling method (RMM) by Gmytrasiewicz and Durfee [9].

Using this communication model, the theoretic model of cooperative multiagent problem solving can be qualitatively described as the following: each agent has a plan (or *policy*) which maps agent beliefs (i.e. information states) to local actions. During the problem solving, the agent updates its belief and chooses the corresponding action according to the plan. The belief of the agent can be updated through observing local states and/or exchange nonlocal states with other agents via communication. We will define the exact information structure that constitutes the agent belief later in this paper, although we note here that it may include historic information as well as current information. On the other hand, an *action* may include both executing a method and performing communication. The execution of a local method does not change the beliefs of other agents (since they do not observe the outcome), but communication may cause other agents to update their beliefs. On the other hand, the execution of a method changes the system state, while communication does not. This introduces a new kind of observability, where an agent can decide whether the global state (or part of the global state) needs to be observed. In contrast, the agents in decentralized control only have partial observation of the system state, and the agents in centralized multiagent MDP are assumed to have collective observation.

In many cases, communication may carry a cost. In the single agent case, it is studied in [11, 12, 13], where communication takes the special form of an agent sensing the environment. Here, communication can also be viewed as a special kind of sensing that changes the beliefs of multiple agents. A rational policy for

each agent thus must balance the amount of communication such that the information is sufficient for proper coordination but the cost for communication does not outweigh the expected gain. In order to understand agent coordination in a quantitative way rather than qualitatively, we need to have a framework that deals with communication decisions in addition to decisions about agent actions. Such a framework would also be very instrumental for the design of heuristic policies. Many coordination policies have been studied [19] but the issue of communication cost is largely overlooked. This is partly due to the complexity involved in introducing communication decisions, and also due to the lack of a clear, quantified model for integrating communication aspects in agent problem solving. In our framework, however, communication decisions and costs are directly represented.

Decentralized decision problems are inherently different from centralized ones [14, 25], which are typically modeled through MDPs or POMDPs. It has been recently shown that solving a decentralized MDP with global utility functions optimally is NEXP-complete (i.e. nondeterministic exponential time) [3]. Thus, even without communication, multiagent decision problems belong to a higher computational complexity class than standard MDP (P-complete) and POMDP (in the PSPACE-complete class) [20], and therefore cannot be reduced to them.

2.2 The Decision Process

Now let us formally define our decision process. As mentioned above, our definition is based on decentralized decision processes. Each agent has its own Markov process. For clarity, we will assume that the system consists of two agents X and Y in the following notations. The same notations can be easily extended to systems with 3 or more agents.

We define the set of agents $\alpha = \{X, Y\}$, and the tuple $M^x = \langle S^x, A^x, p^x(s_j^x | s_i^x, a^x) \rangle$ defines the Markov process (not a Markov *decision* process because there is no local reward function) in X: its local state space is S^x, local action space is A^x, and the local state transition probability $p^x(s_j^x | s_i^x, a^x)$ defines the probability of resulting in state s_j^x when taking action a^x in state s_i^x. Similarly we can define Y's process $M^y = \langle S^y, A^y, p^y(s_j^y | s_i^y, a^y) \rangle$, and the global state space is $S^x \times S^y$ and joint action space is $A^x \times A^y$.

The global reward function $r_t(s_i^x, s_j^y, a_k^x, a_l^y)$ defines the reward the system gets when the global state is (s_i^x, s_j^y) and the joint action is (a_k^x, a_l^y). For simplicity we focus on finite-horizon problems only, and thus we define the reward at terminal time T is $r_T(s_i^x, s_j^y)$. Also, if (s_i^x, s_j^y) represents a terminal state (i.e., we allow the process to finish when a certain relationship between s_i^x and s_j^y are met even when the current time is less than T), we also define terminal reward for those terminal states as $r_t(s_i^x, s_j^y)$, i.e., there are no further actions after t.

To represent communication in our model, we assume that each stage of agent problem solving consists of two substages. The first substage is the communication substage, which begins with the observation of the agent's local state and ends with the completion of all communication actions (send and receive). The agent makes the decision about whether to communicate or not after the observation of the local state. The second substage is the action substage, where the agent makes the decision regarding which local action is the best action to choose after the communication substage finishes. The action substage finishes when the action completes, and the agent moves to a new local state and a new stage. Figure 3 shows the sequence of the substages and the events occurring in one stage.

To model communication, we use *messages* to represent the information exchange between agents. Local state information is the content of a message, and in particular, if an agent chooses not to communicate, its message will be *null*. Each agent can initiate communication independently, we assume that the message format is mutually understood and that no message is lost/changed during communication.

Let m_t^x and m_t^y denote the message(s) sent by x and y during the communication phase. In general, however, when there are more than two agents in the system, we should use a *tensor* to represent all communication messages among the agents, i.e., use the notation $m_t^{u,v}$ to denote the message sent by agent u to agent v. Obviously, broadcast can be viewed as the transmission of multiple messages, therefore this model is not restricted to one-to-one communication.

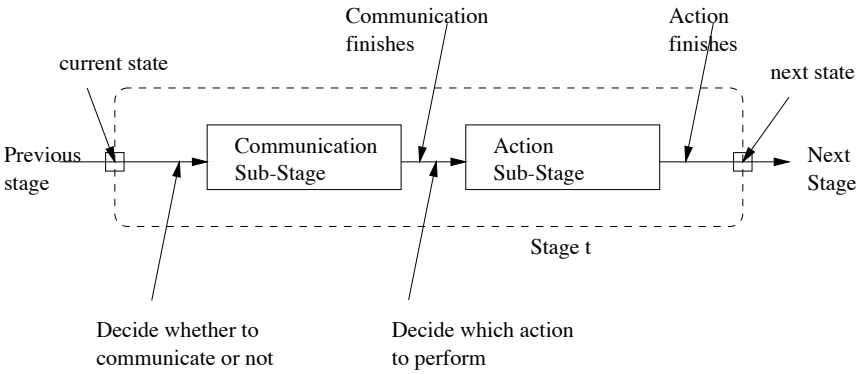

Figure 3: Communication substage

Exactly how the information is shared after the communication clearly depends on the nature of communication. There are many communication types, to name a few:

- *tell*: in this type of communication, one agent simply tells its current local state to the other agent, i.e., information going outward. The sender will not know the receiver's local state as a result of this communication. In this type of communication, an agent knows the other agent's local state only when the other agent voluntarily decides to tell.
- *query*: here, the result of the query is that the query message sender agent receives the local state information of the other agent, i.e., information going inward. In reality, this usually means that the receiver sends back a tell message, but such detail is not necessary in our abstract model of communication. Here, the sender agent does not reveal its local state information to the other agent. In other words, in this type of communication, an agent can know the other agent's local stage whenever it wants to do so, but there is no way to voluntarily tell other agent about its current local state.
- *sync*: this is the combination of the above two, in that when an agent performs a sync communication, it reveals its own state to the other agent, and at the same time obtains the other agent's local state. As a result of sync (regardless of which agent initiates the communication), both agents now know the each other's local state, and also the knowledge that the other agent knows the same. Again, in actual implementation more than one message may be needed, but in our model it is sufficient to symbolize the process into one message communication.

Obviously, the choice of which communication type to choose usually is constrained by the actual communication ability of the agent. For example, if the agent's only communication means is to broadcast, then only the *tell* type is possible, and the agent must tell all other agents. However, it is very important to know that each type has different complexity. For example, with the sync type, the agents know that whenever they communicate, they know the global state, and as a result the previous history often becomes less important because the agents do not need the history information to reason about the uncertainty in the other agent's state and belief.

Let $c_t^x(s^x, m^x)$ and $c_t^y(s^y, m^y)$ denote the cost of communication in each agent given a particular time and current state, using our two-agent system notation. In the simple case that a communication action has a fixed cost regardless of the time and state, a single function $c(m^x)$ (or $c(m^y)$) will suffice, where if m^x is null, the cost is zero, and otherwise, a fixed value c.

In summary, in the above we defined two local Markov processes M^x and M^y, global reward function $r()$, communication messages m^x and m^y, and cost of communication $c()$. Together, they define our model of decentralized cooperative multiagent problem solving with explicit communication and cost of communication. It is Markov because the *global* state depends stochastically only on current state and current actions,

although now actions include communication. Now we can develop an extension to the traditional Markov decision process based on this model. This means that we need to define the decision problem for this model.

2.3 The Decision Problem

We start by defining a global problem solving episode. At each stage, each agent first observes its current state, then makes a decision about communication, and then chooses an action. Thus, we can use (s_t^x, m_t^x, a_t^x) and (s_t^y, m_t^y, a_t^y) to represent all events occurring at this stage, and thus $((s_t^x, s_t^y), (m_t^x, m_t^y), (a_t^x, a_t^y))$ for the global events. Thus, a *global* episode for this process can be described as:

$$\begin{aligned} I &= (s_0^x, s_0^y), (m_0^x, m_0^y), (a_0^x, a_0^y), \cdots, \\ &\quad (s_t^x, s_t^y), (m_t^x, m_t^y), (a_t^x, a_t^y), \cdots, (s_{t'}^x, s_{t'}^y) \end{aligned} \tag{8}$$

and a local episode, for example as seen by agent X, is,

$$\begin{aligned} I^x &= s_0^x, (m_0^x, m_0^y), a_0^x, \cdots, \\ &\quad s_t^x, (m_t^x, m_t^y), a_t^x, \cdots, s_{t'}^x \end{aligned} \tag{9}$$

because the messages are known to both agents.

Here, $(s_{t'}^x, s_{t'}^y)$ satisfies the terminal conditions (including the case when $t' = T$). Note that since we assume that initially both agents know each other's initial state, (m_0^x, m_0^y) would always be (*null, null*).

The probability for that episode to happen (i.e., the probability of having the state sequences $(s_0^x, s_1^x, ...s_{t'}^x)$ and $(s_0^y, s_1^y, ...s_{t'}^y)$), considering that the communication decisions are deterministic and communication does not change an agent's past observations, i.e., local state values such as s_i^x and s_j^y stay the same during communication, and each agent's action is independent of the other agent's action, is:

$$p(I) = \prod_{t=0}^{t'-1} p^x(s_{t+1}^x | s_t^x, a_t^x) \cdot p^y(s_{t+1}^y | s_t^y, a_t^y) \tag{10}$$

For such an episode, its total reward is the terminal reward plus rewards collected at intermediate steps, and minus all of the communication costs at both agents,

$$R(I) = r_{t'}(s_{t'}^x, s_{t'}^y) + \sum_{t=0}^{t'-1} (r_t(s_t^x, s_t^y, a_t^x, a_t^y) - c_t^x(s_t^x, m_t^x) - c_t^y(s_t^y, m_t^y)) \tag{11}$$

Since agents can choose not to communicate and thus they may not always sync themselves, the same past information set in one agent can correspond to many different paths in other agents. In general, each agent's decision about communication/action could be based on all locally available information, including the history and the current information. This means that the decision problem in general is history dependent, not Markovian. Let $H_t^{x,m}$ be the information available to agent x before it makes the communication decision m, and $H_t^{x,a}$ the information before the local action decision a, then,

$$H_t^{x,m} = s_0^x, (m_0^x, m_0^y), a_0^x, \cdots, s_k^x, (m_k^x, m_k^y), a_k^x, \cdots, s_t^x \tag{12}$$

$$\begin{aligned} H_t^{x,a} &= s_0^x, (m_0^x, m_0^y), a_0^x, \cdots, s_k^x, (m_k^x, m_k^y), a_k^x, \cdots, s_t^x, (m_t^x, m_t^y) \\ &= H_t^{x,m}, (m_t^x, m_t^y). \end{aligned} \tag{13}$$

Thus, the local decision problem for agent x is to find out a policy π^x that consists of two parts:

$$\begin{aligned} \pi^{x,m} : \quad & H_t^{x,m} \quad \rightarrow m_t^x \\ \pi^{x,a} : \quad & H_t^{x,a} \quad \rightarrow a_t^x \end{aligned} \tag{14}$$

Here, $\pi^{x,m}$ defines a mapping from all local information to a communication decision, and $\pi^{x,a}$ defines a mapping from all local information to a decision about the next action. Together, π^x encodes all decisions x needs to make. $H_t^{y,m}$, $H_t^{y,m}$, π^y, $\pi^{y,m}$, $\pi^{y,a}$ can be defined similarly so we omit them here.

Based on a pair of local policies: $\langle \pi^x, \pi^y \rangle$, all possible episodes are defined by the set $\{I^{(\pi^x,\pi^y)}\}$, where

$$\begin{aligned} I^{(\pi^x,\pi^y)} &= (s_0^x, s_0^y), (\pi^{x,m}(H_0^{x,m}), \pi^{y,m}(H_0^{y,m})), \\ &\quad (\pi^{x,a}(H_0^{x,a}), \pi^{y,a}(H_0^{y,a})), \cdots, \\ &\quad (s_t^x, s_t^y), (\pi^{x,m}(H_t^{x,m}), \pi^{y,m}(H_t^{y,m})), \\ &\quad (\pi^{x,a}(H_t^{x,a}), \pi^{y,a}(H_t^{y,a})), \cdots, \\ &\quad (s_{t'}^x, s_{t'}^y) \end{aligned} \tag{15}$$

$$p(I^{(\pi^x,\pi^y)}) = \prod_{t=0}^{t'-1} p^x(s_{t+1}^x | s_t^x, \pi^{x,a}(H_t^{x,a})) \cdot p^y(s_{t+1}^y | s_t^y, \pi^{y,a}(H_t^{y,a})). \tag{16}$$

Thus the total global expected reward for the policy pair $\pi = \langle \pi^x, \pi^y \rangle$ is,

$$E(\pi) = E(\pi^x, \pi^y) = \sum_{I \in \{I^{(\pi^x,\pi^y)}\}} p(I) \cdot R(I). \tag{17}$$

The decision problem is, therefore, to find the optimal pair of (π^x, π^y) such that it maximizes $E(\pi^x, \pi^y)$.

Note that although the formula above may seem daunting, the calculation of $E(\pi)$ given a policy π is quite simple and straightforward by way of graph traversing if we use the induction relations between time t and $t+1$ (due to the Markov property of this decision process). For any global state (s_t^x, s_t^y), (which implies $H_t^{x,m}$ and $H_t^{y,m}$, and hence we can find out $H_t^{x,a}$ and $H_t^{y,a}$ according to $\pi^{x,m}(H_t^{x,m})$ and $\pi^{y,m}(H_t^{y,m})$), we can find out the joint action under π which is simply $(a_t^x, a_t^y) = (\pi^{x,a}(H_t^{x,a}), \pi^{y,a}(H_t^{y,a}))$ and then calculate all possible next states $\mathbf{next}(s_t^x, s_t^y)$.

We can define the value of global state (s_t^x, s_t^y) *before communication* under policy π as $v(s_t^x, s_t^y | \pi)$, and we can see that, if the state is terminal, $v(s_t^x, s_t^y | \pi) = r_t(s_t^x, s_t^y)$, otherwise,

$$\begin{aligned} v(s_t^x, s_t^y | \pi) &= r_t(s_t^x, s_t^y, a_t^x, a_t^y) + \\ &\quad \sum p^x(s_{t+1}^x | s_t^x, a_t^x) \cdot p^y(s_{t+1}^y | s_t^y, a_t^y) \cdot v(s_{t+1}^x, s_{t+1}^y | \pi). \end{aligned} \tag{18}$$

where the summation is over all $(s_{t+1}^x, s_{t+1}^y) \in \mathbf{next}(s_t^x, s_t^y)$. The calculation is then similar to the calculation of $v(s|\pi)$ for standard MDP and $E(\pi)$ is simply $v(s_0^x, s_0^y | \pi)$. This tells us that even though our decision problem itself is not a standard MDP (and harder to solve), once we have a policy (no matter how we obtain it) the evaluation of the policy is the same as a standard MDP. As such, this framework can be effectively used for evaluating policies.

Note that in the above discussion we use the notation of standard Markov decision processes (MDPs), not partially observable Markov decision processes (POMDPs). This should not be viewed as a limitation of the framework, since POMDPs can always be mapped into equivalent MDPs by introducing belief states. It can be easily extended to POMDP notations, for example in Pynadath and Tambe's work [22]. In fact, what we refer as states (s) are essentially the representation of the observations to the environment: global states for the global observations and local states for local observations. Also, local states are precisely the result of partial observations of global states. Using an MDP notation and modeling local states as components (s^x and s^y) of a global state (s) help us easily formulate local decision processes (deciding a^x and m^x) in each agent.

3 Issues in Solving Decentralized Multiagent MDP

In the above we defined our multiagent extension to (partially observable) Markov decision processes. This extension deals with decentralized cooperative multiagent problem solving, where the agents can decide

on the use of communication to obtain nonlocal information. By using this framework, we can provide a decision-theoretical foundation for complex multiagent problem solving.

Calculating an optimal policy is not going to be computationally feasible in most cases. Decentralized decision problems are NP hard in general [24]. Furthermore, since optimal policy is history dependent, the size of a policy (i.e., all possible histories) is too large to handle even for small problems. Even with our simple grid world problem problem, finding the best policy based on all local information, i.e., optimal (π^x, π^y) is computationally infeasible. Since in each stage, each agent can take 5 actions, each local action can have up to 5 resulting positions, and 2 communication choices, while the other agent may have 16 different possibilities in its message content (assuming a 4×4 grid), that means an up to $(5 \times 5 \times 2 \times 16) = 800$ fold increase of local information history in each stage (since $H^{x,m}_{t+1} = H^{x,m}_t, (m^x_t, m^y_t), a^x_t, s^x_{t+1}$). Obviously, this means an explosion of the size of the local policy, and therefore is infeasible to compute a truly optimal policy. In a simplified case, when the agents do not have communication ability, our model degenerates to the decentralized MDP with global reward function, and as mentioned earlier, the problem is NEXP-complete. Therefore, given such computational complexity, to solve our decentralized multiagent MDP with communication exactly is often infeasible.

Approximation methods and heuristics solutions, on the other hand, exist and can be very efficient. One direction for approximation is to notice that although an optimal policy is history dependent, we can develop policies that uses not all local history but only a part of it (presumably only some most recent information). This way we can reduce the size of the policy drastically. Another approximation method is to try to introduce some form of local utility measures, so that the optimization of global utility can be decomposed into individual optimization of local utilities. This way, the methods for dealing with traditional Markov decision processes (such as dynamic programming) can then be used to solve these approximation problems.

Also, in the past years many coordination mechanisms have been proposed, and they correspond to many heuristic methods for multiagent problem solving. We can also learn from the way coordination and communication is performed in human society. Indeed, as we show in this research, heuristic solutions are abundant and are often easy to compute, and by examining a family of heuristic solutions we may gain insight for designing good policies for agent coordination.

Note that if the communication cost is 0, which means that both agents can communicate to obtain global information at all times (and hence no communication issue anymore), this problem reduces to a centralized problem (a typical MMDP problem mentioned earlier): we can construct a standard MDP based on global state, joint action, and global utility, and then solve the optimal global policy. We will use this as a baseline to compare to decentralized policies in our experiments. Obviously, the expected utility of this optimal global policy gives us an upper-bound for our decentralized policies.

4 Quantitative Evaluation

In the previous section we defined a rigorous and quantitative framework for modeling multiagent cooperation. The definition of optimality in this model is clear: it is essentially a multiagent version of the Maximizing Expected Utility principle. A more important task is to apply this framework in actual design of multiagent systems, including the development of better multiagent problem solving strategies. However, although this framework offers a model based on first-principles, solving the optimal policy is computationally infeasible. Therefore, we focus the use of this framework on the evaluation of strategies to gain insights into the choice of strategies, and also on the development of approximate solutions. In this paper, we discuss the issue of representing and evaluating heuristic problem solving strategies. Our focus is on the issue of communication strategies. We will instantiate our approach on the grid world example problem, and provide quantitative evaluation results.

In the following we will first introduce the centralized policy and use that as a baseline comparison. Then we will study how to represent the heuristic strategies, discuss some specific communication strategies, and show their performance results.

4.1 The Baseline: the Centralized Policy

Under the centralized multiagent MDP model, each agent observes the global state directly and each agent knows the centralized policy before hand (theoretically this can be achieved by solving the multiagent MDP independently). Thus, the problem solving is simply the repeating process of observing the state and then choosing the local action according to the joint action specified by the policy, which maps global states to joint actions. No communication is involved here, and therefore there are no communication cost issues here either.

The optimal policy for the centralized multiagent MDP model, i.e., the baseline, defines an upperbound of expected reward for decentralized policies. If the communication cost is zero, the optimal decentralized policy should have the same expected reward as the baseline since one of the strategy is to communicate with each other at all times. The baseline can also be used to provide a lowerbound by counting all communication costs involved, assuming the agents communicate in each step.

Thus, it is expected that good decentralized policies should have expected utility values between the upperbound and lowerbound. The upperbound is impossible to surpass while being lower than the lowerbound suggests that the decentralized policy does not have very good performance. Of course, this is based on the assumption that the (optimal) centralized policy is easy to compute. Also, if the communication cost is really small, the gap between the upperbound and lowerbound will be small. In this case it is obvious that the centralized policy would be good enough since it is much easier to compute than the decentralized policies.

For the grid world problem under study (see Figure 1), the environment variables are listed in Table 1:

Parameter	Meaning	Range
T	the deadline	3 — 6
β	the time discount factor of reward	0.6 — 1.0
q	success rate	0.70 — 1
c	communication cost	0 — 5

Table 1: Environment Parameters for the Example Problem

For the centralized baseline, communication cost c is not used, so the variables involved are just T, β, and q. However, we note that the centralized policy (i.e., which actions are optimal given their current positions) is not changed by different β or q values (as long as q is close to 1). Clearly, changes in β or q values do not change the state structure of the MDP – only the reward values are affected. Since q is close to 1, the intended outcome of an action always dominates other outcomes and therefore the reward for the intended meeting path has a dominating share in the expected reward, and thus the calculation of the optimal actions is roughly equivalent to finding the shortest paths, and hence not affected by different β or q values. However, the policy does change with regard to the deadline T, since the MDP is changed due to a different horizon.

We assume that if the agents meet at time 1 (the starting time is 0), the reward is 100. Otherwise, the reward is discounted each step by a factor of β. If the agents do not meet by time T, the reward is 0. Figure 4 shows how the expected reward changes with regard to the β, where q is fixed at 0.80. In this and the following figures, the Y axis is always the expected reward.

It is clear from the above figure that when the deadline T ranges from tight ($T = 3$) to loose ($T = 6$), there are more chances that the agents can eventually meet in time, therefore the expected reward increases when T increases.

Figure 5 shows how the expected reward changes with regard to q while fixing β at 0.95. It is obvious that a higher q leads to better chances of meeting, and even so when the deadline is loose because the agents then are better able to recover from previous failures. It can be seen from the figure that when q is good enough the deadline does not matter much, because most of the meetings would occur before the deadline,

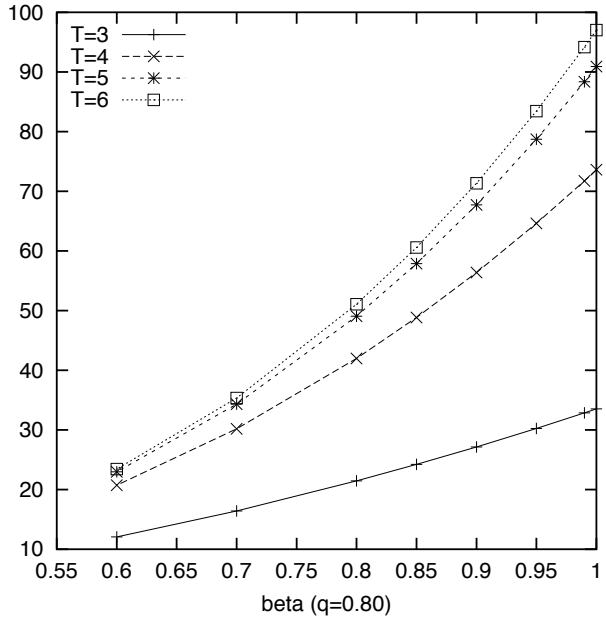

Figure 4: The Baseline Policy: Discount Factor

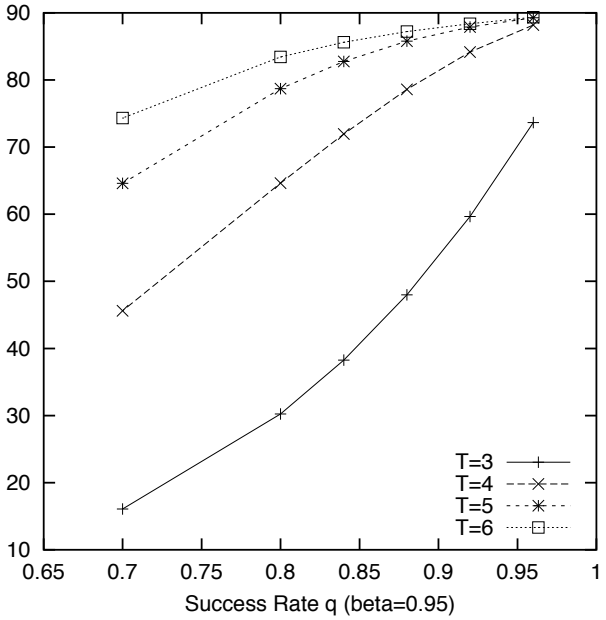

Figure 5: The Baseline Policy: Success Rate

which is why the reward value is nearly the same when $q = 0.96$ and T =4, 5, or 6. When $q = 1$ then there is no uncertainty in the system and the agents are guaranteed to meet at time 3.

4.2 Heuristic Approaches

Now we study the issue of using heuristic solutions for our decentralized multiagent MDPs. We have discussed earlier that the evaluation of heuristic policies (i.e., computing $E(\pi)$) is quite straightforward. Thus, given a predefined policy, to evaluate its performance, i.e., expected total reward, is not a hard problem. However, traditionally the research on multiagent problem solving strategies often uses a task representation, not a low-level state-based representation. The reason is that task level representations are more intuitive and convenient for describing agent goals, intentions, and utilities than a state representation, which is based on states, actions, and rewards. Therefore, in order to gain insight into the design of multiagent problem-solving strategies, it is very important that we have a way of translating task-level concepts into state-level representations.

A task level representation is more coarse-grained than a state level representation, and this implies some approximation and simplification of the model of problem solving. The translation from task models to state models is fairly simple: each task is an action, and for each task a we can use a vector to represent all possible outcomes of the task. Then the local state space is simply a subset of all the combinations of the task outcome sequences. Task interrelationships are represented via the structure of the state space itself: for example, if A enables B, then there is no path in the state space that would allow B to be executed without the existence of A in its preceding path. This is essentially a way of encoding history information (or at least a subset of history) in the state. The construction of reward model is also straightforward: we can simply assign a reward of 0 to all intermediate states and calculate the reward on the terminal states based on task utility model. This transformation also clearly defines the local states of the agents by listing the local execution history of the agents.

The key to translating heuristic strategies is to recognize how to represent coordination. The concept of *commitments* [16] are the cornerstones of coordination, and we will show that it can be interpreted in our decision-theoretic framework. Commitments are social constructs that reflect the mutual agreement of the agents with regard to some desired outcome. As such, commitments can be seen as defining some joint goals of the agents. A joint goal is then interpreted as a local goal to the involved agent, and therefore separates the multiagent problem solving into local problem solving in each agent. Thus, problem solving can be viewed as *intermittent periods of local problem solving* in each agent: the agents adopt a joint goal and then work on their own for a while before a new joint goal is set up.

Setting up joint goals certainly involves communication. After all, the goal of communication is to reduce uncertainty in the agent beliefs *so that better actions can be decided*. In our model, the agent's problem solving can also be viewed as *intermittent periods of problem solving without communication*. This leads to the following interpretations:

1. Communication implies the setting up of a joint goal. Communication leads to the observation of nonlocal states, which in turn decides the joint goal.

2. The agent can interpret the joint goal as a local commitment, and set up a local process for problem solving toward that commitment.

3. During a period of problem solving without communication, the agent performs local problem solving in lieu of the above commitment, until the commitment needs to be changed. This implies that the agent has a way of deciding if the commitment is optimal (within its bounded reasoning capability) and monitoring the progress of the local problem solving.

4. When the need of change is detected, the agent communicates and as a result a new goal may be decided. The process then repeats from step 1 above.

Specifically, a commitment can be reflected as the agent's promise to be in a certain set of states (having a proper value for the outcome of a local task) at some future time. The strategy for choosing commitments can be characterized as a function F based on the agent's information. Similarly, the strategy for achieving the commitment can also be characterized as a function G that defines which actions (communication actions or task actions) are needed in order to fulfill the commitment. F and G roughly correspond to $\pi^{x,m}$ ($\pi^{y,m}$) and $\pi^{x,a}$ ($\pi^{y,a}$) for agent X (Y). A non-communicating period of problem solving, therefore, can be represented as the following:

- The period begins (at t_0) immediately following a communication by applying F function on the common information in $H_{t_0}^{x,a}$ and $H_{t_0}^{y,a}$ to decide a joint goal, which specifies a future goal state (global state), denoted as $F(t_0)$.
- The G function then decides the local actions during this period, i.e., the optimal local action in order to reach the local state specified by the joint goal. During this period,

$$\begin{aligned} \pi(H_t^{x,m}) &= \text{no (e.g. not to communicate)}, \\ \pi(H_t^{x,a}) &= G(H_t^{x,a}, F(t_0)). \end{aligned} \tag{19}$$

- At the same time, the agent continues to monitor if there exists a better joint goal based on $H_t^{x,m}$: it calculates $F(t)$ and compares it to $F(t_0)$. If $F(t)$ appears to be a better goal, then the period finishes at time t and $\pi(H_t^{x,m}) = yes$, and $F(t)$ is the new goal.

Obviously, the F and G functions are simply a different way of expressing the agent policy π, and to solve the optimal F and G functions is the same as solving the optimal policy π. However, the use of F and G functions can allow us to effectively describe heuristic methods.

First, the G function is based on the result of F. Intuitively, the result of F represents a *goal* of the agent. When a new goal is established, the way to achieve the goal based on current situation typically can be represented through a search process, and does not depend on history. Also, since the time-frame for the G function is limited to the time-frame indicated in the result of the F function, G often has a short time-frame. This means that G often represents a local Markov decision process with a short horizon, and therefore is fairly easy to calculate. More importantly, since the goal indicated by F remains the same for a period of time, the G function simply reflects the progress of the same Markov decision process, and therefore requires little additional reasoning.

Second, the F function represents commitments, and in turn the goals of the agent. Naturally, although commitments and goals could be dynamic, in typical problem solving they are not changed very often [8, 7]. It is often convenient, then, to introduce an additional function f to represent the monitoring process, where f decides if F needs to be evaluated again based on the new information since the last time F is evaluated. In other words, f is a boolean function that checks whether or not new commitments need to be established, i.e, it represents the commitment monitoring process. This way, F needs not to be fully evaluated all the time. Since the evaluation of F, i.e., deciding what commitments to make, is often a complex reasoning process, the use of f can significantly reduce the complexity and computational costs.

The approximation methods suggested earlier can also easily be used to help define the F, G, and f function, for example, F functions can be defined on only part of the history information, and G, f functions can use the result of the F function as one of their parameters. Similarly, having a local reward measure (even a temporary, short-term one) can reduce the complexity in the reasoning process represented by these functions. Using these approximation methods can lead to succinct and efficient descriptions of the functions, which help us understand the implications of the policies.

An important note is that while the F function requires mutual agreement, the G and f functions do not. In fact, they can be completely local policies that other agents do not need to worry about. This means that our assumption about the complete knowledge of the multiagent policy is relaxed: we only need to have a mutual understanding of the joint goal function. This is a much more realistic assumption in multiagent problem solving, and therefore makes our framework much more applicable.

Clearly, central to the coordination is the F function which corresponds to the communication decisions. Once F is defined, G can be viewed as a local optimization process based on F. This allows us to focus on the issue of coordination strategies since we can now apply the optimal G so that the effects due to a biased planner can be minimized. Next we introduce some general heuristic coordination strategies, each of which has social analogies to popular protocols of cooperation in human interactions. These can be applied to many domains besides the grid world domain in our example.

4.3 The NN Policy: "No News is Good News"

The first heuristic coordination strategy we introduce here is what we call an NN policy. In this policy, agents select an optimal plan based on their last observed global state (i.e., the state where they last performed a sync communication), and they communicate (sync) whenever their current plan cannot be achieved (so that a new plan can be selected), but do not communicate if the plan is still achievable. This corresponds to the so-called "No News is Good News" type of social convention, where if both parties are making progress as intended, they do not communicate (no news), however they will negotiate a new plan if the progress is not as intended. An example in this grid world problem is that assuming both agents first choose to meet at position 3 (top-right corner) in 3 steps, and they will not communicate if in each step they are getting closer to block 3. However, if X slips into block 4 when it tries to move to block 1, X will sync with Y and the agents will reelect a best position to meet, possibly block 6.

In our representation of heuristic policies, the details of NN are the following (using the example problem):

- The history information structure is reduced: in NN, X's local policy uses only part of the history information, namely the time they last communicated, and the global state they discovered at that time (using the sync type of communication). This reduces $H_t^{x,m}$ and $H_t^{x,a}$ to l, s_l^x, s_l^y (and of course current information t, s_t^x), where l is the last time that $m_l^x \neq$ null or $m_l^y \neq$ null, and s_l^x is X's local state at time l, and s_l^y is Y's local state at time l (transmitted as part of the content of m_l^x or m_l^y). In other words, agents only remember the global state at the beginning of the current local problem solving period.

- The F function $F(s_l^x, s_l^y)$ decides a best short-term goal: a global state $(\hat{s}^x, \hat{s}^y)$. And the progress function for current state $f_l^x(s_t^x, \hat{s}^x, t)$ tells if X (or Y) has made *sufficient progress* at current t toward the the goal state $\hat{s}^x$ $(\hat{s}^y)$. For our example, F simply tells the mid-point of a shortest path between the two agents, and f tells if the *distance* (which is *Manhattan distance*, not straight line distance) from the current local position to the mid-point has been shortened as planned (i.e., reduced by $t-l$). In other words, if the distance between two positions is $d(p_1, p_2)$,

$$f^x = \begin{cases} true & \text{if } d(s_t^x, \hat{s}^x) \leq d(s_l^x, \hat{s}^x) - (t-l); \\ false & \text{otherwise.} \end{cases} \tag{20}$$

  and,

$$\pi^{x,m}(t, s_t^x, l, s_l^x, s_l^y) = \begin{cases} \text{sync} & \text{if } f_l^x(s_t^x, \hat{s}^x, t) \text{ is false;} \\ \text{null} & \text{otherwise.} \end{cases} \tag{21}$$

- The G function for X is simply the best local action so that the short term goal $\hat{s}^x$ is mostly likely to be reached. In other words, we can view the local processing as solving the local MDP of one agent trying to move from s_l^x to $\hat{s}^x$ (with the total probability of ending in $\hat{s}^x$ as the utility), thus during the non-communicating period the $\pi^{x,a}$ would choose the optimal action according to the optimal policy for the local MDP. In our case we can use a very simple heuristic that has almost identical performance as the optimal local policy, which chooses the move intended at getting closer to $\hat{s}^x$ (when multiple moves are possible, choose the direction that reduces the straight line distance most as the tie breaker).

We study the performance of NN by varying all 4 environment parameters. We will make a comparison of the policies later in this paper so we only address some interesting observations about this policy here.

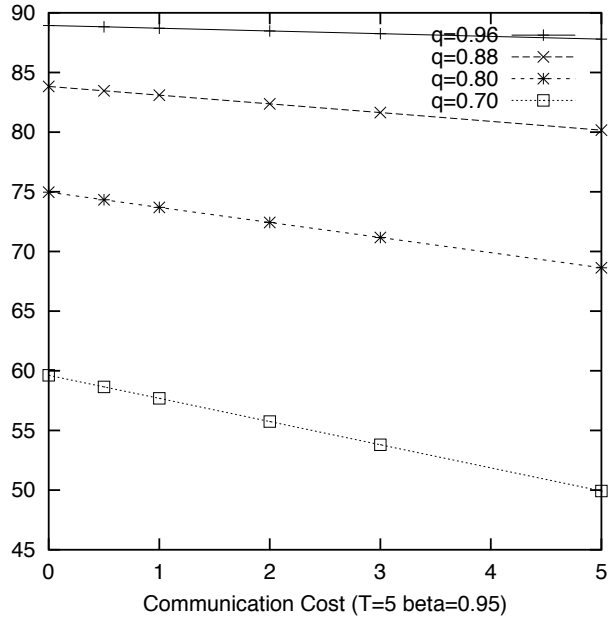

Figure 6: The NN Policy: Communication Cost

Figure 6 shows the expected reward of NN with regard to the communication cost. Note that since the heuristic functions do not directly take into account the communication cost c, the policy itself does not change according to c. As a result, the expected reward is linear to c, and the slopes of the lines reflect the average amount of communication used in the policy. As we can see, the amount of communication increases when the system has more uncertainty, i.e., a small success rate q, and this explains the increase in the slope of the lines in reaction to the decrease of q.

Figure 7 shows how NN handles uncertainty by varying the success rate q. The shape of the curves is similar to that of the baseline policy (Figure 5), although the curves are all below the T=5 curve in Figure 5, which is the upperbound. Clearly, when there is less uncertainty, the chance of communication goes down, and therefore the difference in the expected reward due to different communication cost values is small, and that explains the convergence of the curves when q goes to 1.

4.4 The Silent Commitment (SC) Policy

The second policy, in which no communication is needed, basically divides the problem into two independent parts and then each agent is committed to perform their part. In this case, this division of work may have high probability of success (i.e., in some cases agents may be able to recover from adverse outcomes), however they cannot change their plan dynamically, partly because they choose not to communicate at all. Of course, this approach depends on both agents knowing their initial global state so that they can choose the best division. We call this the "silent commitment" (SC) approach. This approach also has its social counterpart, where when two parties decide to coordinate, they divide the work, set up a deadline when each party's work has to be completed, and then work on their own. Normally the deadline should be far enough in the future so that both parties feel comfortable. In our grid world problem, the agents agree to be at block 3 by time T (the deadline). Thus, even if X's first move to the left resulted in block 4, X will try to correct that and possibly still be able to enter block 3 by time T.

The SC heuristic chooses a completely different subset from local history compared to NN: only the initial global state! It uses a heuristic function $F(s_0^x, s_0^y)$ – note the initial global state here – which also decides

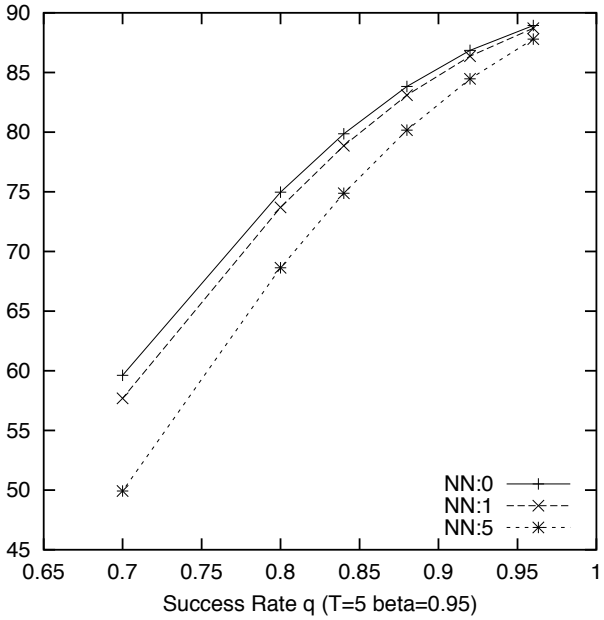

Figure 7: The NN Policy: Success Rate

a goal state $(\hat{s}^x, \hat{s}^y)$, in our case the mid-point of a shortest path between X and Y's initial states. The F function is the same function in NN but it is only used once at the starting state.

The difference between NN and SC is that in SC the agent has T time to reach its own goal state, but in NN a progress function imposes a stronger constraint and thus becomes a dynamic plan. In other words, SC does not have de-commitment but NN allow dynamic commitments.

SC never communicates, thus, $\pi^{x,m}(s^x)$ is always null and there is no need for the f function. The same G function in NN is used here, too.

We study the SC strategy by varying T, β, and q, since c is irrelevant to SC due to no communication. As such, the expected reward will not change according to c. Figure 8 shows the change of expected reward according to q, with β fixed at 0.95. Compared to Figure 5 (the upperbound baseline), we can see that the shape of the curves are similar. The T=3 lines in both figures are almost the same, suggesting that with a very tight deadline, agents will not have enough time to recover from failures and thus the simple silent commitment is close to optimal. However, when the deadline loosens, although the agents have a chance to recover, it may be better to switch to a new goal, which explains the big differences when T is greater. Also, we note that when q is close to one, the same converging effect applies here too, simply because the need for recovery decreases when failure occurs less frequently.

4.5 Allowing Slack Time

In both NN and SC, the size of the policy is significantly reduced so that it is computationally feasible. Also, we note that the calculation of $\pi^{x,a}$ involves optimization, but in both cases the optimization is completely local, i.e., both try to maximize the probability that $\hat{s}^x$ (or $\hat{s}^y$) to be reached. In other words, a local utility measure is introduced. In NN the utility measure is a short-term, dynamic one, and in SC it is a fixed one. As a result, SC and NN react differently to communication costs: SC is immune to the increase of communicate costs, but NN may incur a large number of communications.

This leads to our third heuristic strategy: we try to reduce communication by relaxing the condition for keeping the current commitment. The same F and G functions in NN are used, but the f function has

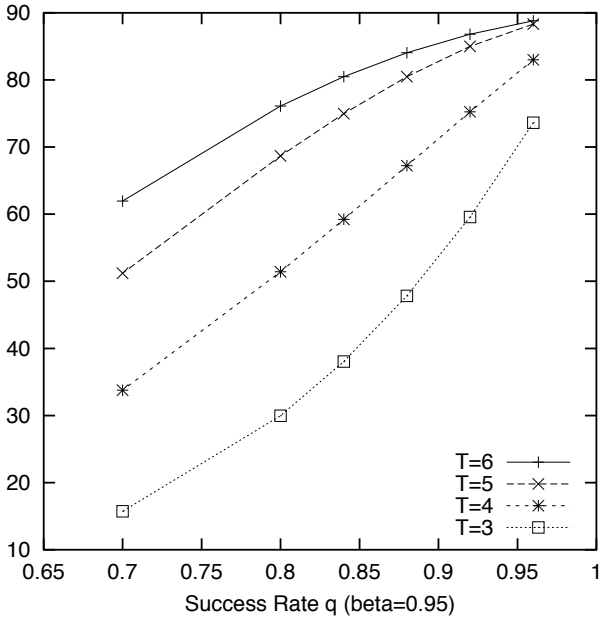

Figure 8: The SC Policy: Success Rate

a slight change: we introduce a slack distance $\delta > 0$ and the *sufficient progress* is redefined as having the distance between s_t^x and $\hat{s}^x$ shortened by at least $t - l - \delta$. This allows the agent to have failures and not to reduce the distance between s_t^x and $\hat{s}^x$ in each step, which is required in NN. Thus,

$$f^x = \begin{cases} true & \text{if } d(s_t^x, \hat{s}^x) \leq d(s_l^x, \hat{s}^x) - (t - l) + \delta; \\ false & \text{otherwise.} \end{cases} \tag{22}$$

For example, $\delta = 1$ means that if X tries to move to position 4 (the $\hat{s}^x$) and its current position is 1, if the current move (to the right) results in staying at 1, the agent would still think it has made a sufficient progress and thus does not need to establish a new goal yet, while in NN this would not be sufficient progress since it does not shorten the distance between the goal position and its previous position.

We use SLn to represent a slack strategy with $\delta = n$. SL0 is the same as NN. Here we consider 3 choices: SL1, SL2, and SL3. Clearly, the larger δ is, the more fault-tolerant the current local process becomes. A δ of 3 (SL3) in our example problem virtually coincides with SC since the distance between the initial position and initial goal is 3. Thus, unless the agent has made at least 2 failures, f remains true.

Figure 9 shows the expected rewards of the SLn strategies. Note that the SL3 curve almost completely overlaps the SC curve, as discussed above. The slopes of the SLn curves clearly show the reduction of communication when n increases.

4.6 The Hybrid Policy

Intuitively, NN pays communication costs to reduce uncertainty in coordination, but SC avoids communication at the cost of increased uncertainty. However, we note that both NN and SC do not respond to communication costs — the calculation of communication decisions does not involve communication costs. Thus, there might be cases that less uncertainty does not offset the cost paid for communication (when using NN), or that a communication could have resulted in a large increase in expected utility (when using SC). This prompts us to try to find a hybrid policy that has the best of both worlds: it communicates to reduce

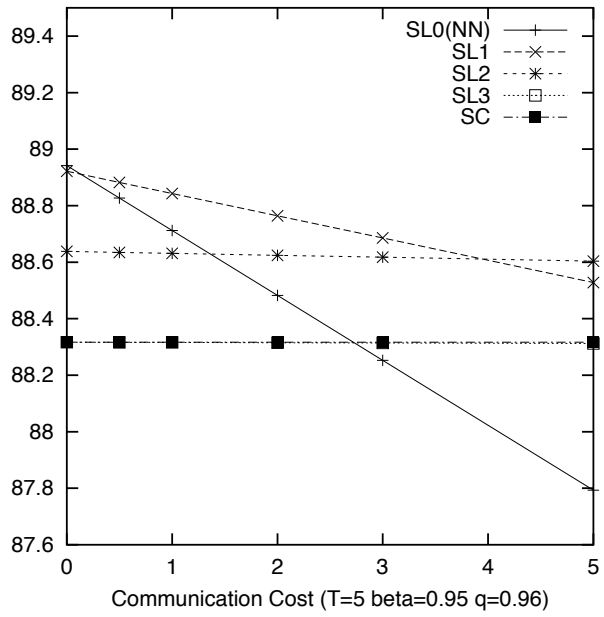

Figure 9: The Slack Policies: Communication Cost

uncertainty when the cost is less than the expected gain, and avoids communication when the cost would be greater than the expected gain.

Just like SC or NN, the hybrid heuristic starts by introducing local goals. Like NN, this goal could be a dynamic goal: the agents have no long-term goals. Also, an agent assumes that the other agent is making progress toward its local goal if it does not receive communication from the other agent.

But unlike NN, where goals are changed only when the progress is not as expected, in this hybrid heuristic, the agent is *always deciding* if there is a better goal and hence a greater reward (assuming the other agent is making good progress). However, even when a better goal exists, the agent will first calculate the expected gain, and *compare it to the communication cost*. If the latter is greater, the agent remains silent, otherwise, it communicates, and a new goal will be established. Thus, the goal could potentially be a long term goal (like in SC), when communication cost is high enough.

In other words, while the F and G functions remain the same, the f function is much more complex: it involves the reasoning of *possible* better goals. If the current local processing is modeled as a local MDP for the agent, the evaluation of possible better goals requires building alternative local MDPs and evaluation of those alternative MDPs. Thus, local processing can be viewed as *competing local MDPs* and the f function reflects the result of the competition.

We need to note, though, that it may be quite difficult to precisely decide if there exists a better goal, and how much the expected gain is. This is also where heuristic functions may be applied. More importantly, to calculate the utility difference by choosing a competing goal would involve the estimation of the other agent's position. In our case, each agent assumes that the other agent is making sufficient progress in each step if there is no communication, i.e., just the same as defined by the NN policy. We use this simple heuristic: try the adjacent positions of the current goal and check if the agents may meet sooner in any of these positions (assuming the other agent has never failed since last synchronization). If so, a better goal is found, and the potential gain is the utility difference due to meeting earlier, times the probability that both agents always make good progress toward the new goal. For example, if the current time is 2 and the agents have a chance of meeting at time 4 instead of 5, the difference is $\beta^3 R - \beta^4 R$, and since both agents have 2 more steps to go, the chance that they are always "on track" is $q^2 \times q^2$, and therefore the expected gain (heuristic value)

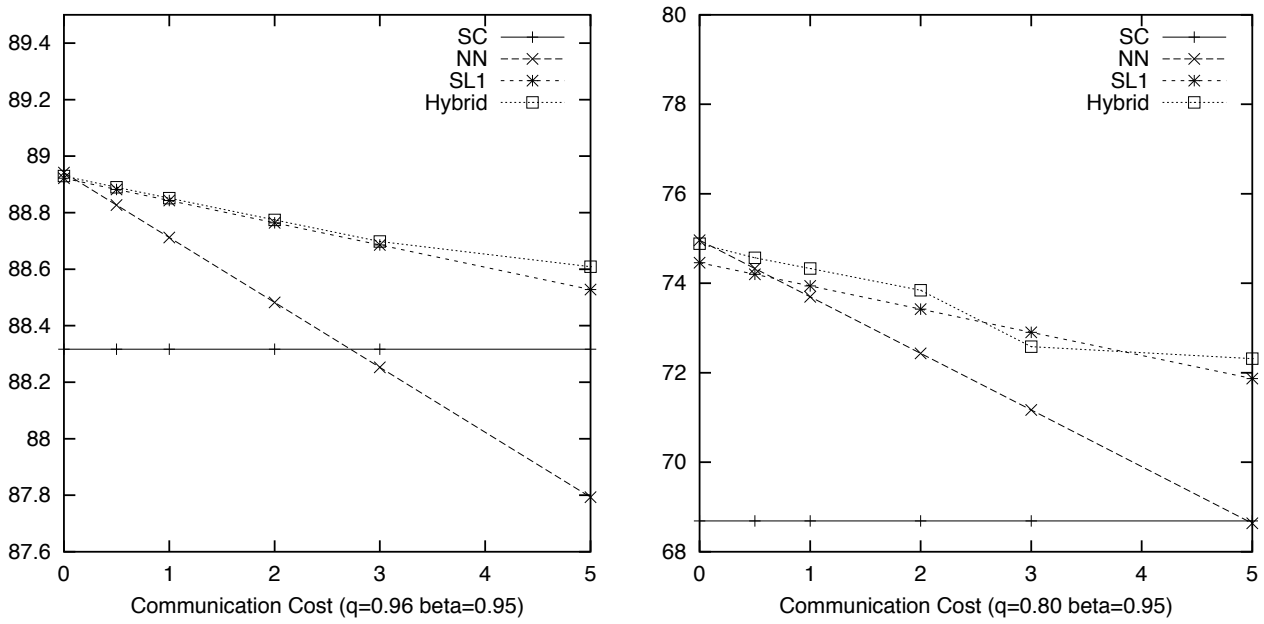

Figure 10: Communication Cost

is $q^4R(\beta^3 - \beta^4)$. We can now specify f as the following:

$$f = \begin{cases} true & \text{if a better goal } p \text{ exists } \textit{and} \text{ expected reward gain} > c; \\ false & \text{otherwise.} \end{cases} \tag{23}$$

Note that such heuristic calculation is certainly *myopic* in nature, because it does not take into consideration of further changes of goals, which may result in additional gains in later stages. In other words, this calculation is assuming that the new goal would be a long-term goal, but in fact it may be changed later.

We evaluate the performance of the hybrid strategy under different T, c, q, and β. We will discuss its performance in comparison to those of the other strategies in the next section.

5 Results and Discussions

In the following we evaluate the example problem and try to discuss the implications of these heuristics with regard to multiagent coordination. We will compare the baseline, the SC policy, the NN policy, the Slack policy, and the Hybrid policy.

First we study the expected rewards of NN, SC, SL1, and Hybrid with respect to the communication costs, as in Figure 10. Here $q = 0.96$, $T = 5$, and $\beta = 0.95$. The baseline value is 89.28, which gives an upper bound. With the SC policy, the expected reward (y-axis) does not change at all, because this policy never utilizes communication. The NN policy has better performance when communication is free, but it does not scale with communication cost, thus we see a crossing point between NN and SC when the communication cost increases. This illustrates the general intuition: communication is a rational thing (will achieve better performance) unless the cost of communication is too high. In our case, communication in NN indicates a change of short-term goal (de-commitment or goal modification in typical multiagent coordination language). This is rational as long as the communication cost is low. Otherwise, SC (where commitment cannot be changed and the agent always tries to honor the commitment despite local failures) would be a better solution. The SL1 policy is somewhat in between SC and NN. The expected reward of

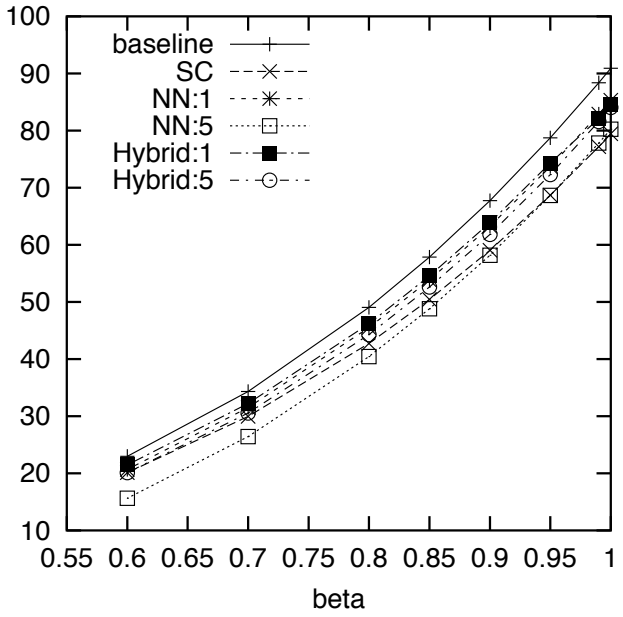

Figure 11: Beta: Discount Factor

SL1 is still linear to the communication costs although it uses less communication than NN. In our example it shows that SL1 is a good combination of SC and NN because it reduces communication costs while at the same time does not allow problem solving to deviate too much from the intended course. That's why SL1 outperforms SC and NN in most cases, and has similar performance compared to the Hybrid, which also combines the good features of SC and NN. The Hybrid line clearly reinforces our points, and it shows that such a hybrid heuristic indeed gets the best of both worlds, and performs better than any of the other two heuristics when communication cost is present. It performs as good as NN when cost is 0, and it scales like SC when cost is very high.

How soon the cost of communication outweighs the benefit of more information depends on the uncertainty in the system. Clearly, with a higher q, meaning the agents' movements are more reliable, the amount of uncertainty in the system is not high, and hence the increase in performance due to the reduction of uncertainty via communication is not much. Therefore, the crossing point in the figure would come earlier when q is greater. The benefit of the Hybrid heuristic is that it eliminates the guess work when choosing over SC or NN, since it adjusts to communication cost automatically.

Next, in Figure 11 we vary the time discount factor β and see how these heuristics react. The smaller β is, the quicker the reward decreases, thus the agents have an incentive to achieve the goal as quickly as possible (if β=1 then the reward is the same as long as they meet before the deadline.) First we note that not surprisingly, Hybrid is the best. NN is very close to Hybrid, and in general is better than SC, since by resolving the uncertainty via communication the agents can adapt quicker. Also, it is interesting to note that when β decreases, performance of SC decreases slower than the NN policy, and depends on the cost of communication, the SC line can meet with NN:c lines (although it is always under Hybrid:c lines), where c is the communication cost. The reason here is that, when β decreases, the cost of communication becomes more and more comparable with the reward, since the communication cost is fixed. In the extreme case, the reward can be discounted so much that it is smaller than the cost of communication. Obviously in this case the rational decision is not to communicate (Hybrid will reach the same conclusion via its calculation, so it performs consistently better than SC). The implication is that, in a time critical system, the agents should choose to communicate earlier than later, since the weight of communication may become greater when time

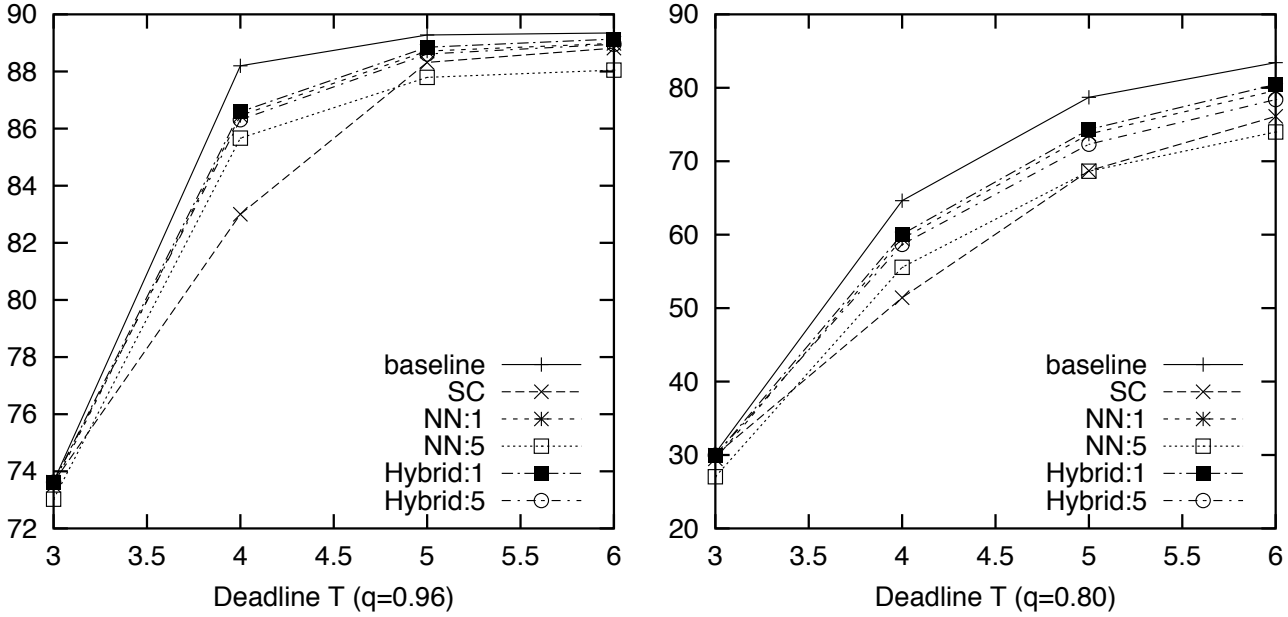

Figure 12: Deadline

passes.

Next, in Figure 12 we vary deadline T and see how they perform from very time-constrained (3) to having plenty slack time (6). Here β is fixed at 0.95. We notice that when the deadline is tight, Hybrid and SC are slightly better than NN since the agents do not have time for an alternative plan when their initial plan fails. In these cases all three heuristics are quite close to optimal (baseline upper bound). On the other hand, when the deadline is far away, both NN and SC would allow agents to reach their eventual goals (in the case of SC, agents have enough time to recover from earlier failures), thus in this case their performance again becomes close. (Of course, β close to 1 is still needed). In this case, all three heuristics are also quite close to the optimal. The most interesting case is in the middle of the lines, when the deadline is not so tight, the reduction of uncertainty and the use of dynamic goal adaptation can certainly help agents achieve their group goals in a timely fashion, and hence Hybrid and NN performs better, i.e., sophisticated planning pays off. This result confirms some of Decker's results on when to use sophisticated control [6]. However, NN communicates whenever there is uncertainty about the current commitment, so it may communicate too much in high cost situations, and results in a lower expected utility than SC.

We also observe that with higher uncertainty (lower q) the agents need to communicate much more often in NN policy, thus causing the performance difference between SC and NN to be smaller. Again, Hybrid is the dominating one among the three heuristics, suggesting the need for explicitly reasoning about communication costs.

Finally, it is interesting to see how SC, NN, and Hybrid differ with the success rate q – the indicator of how reliable the agent's actions are. In Figure 13, we again see that when the uncertainty is low, all policies achieve about the same performance (possibly close to the optimum). The Hybrid policy is again the dominating policy, outperforming both NN and SC solidly. Between NN and SC, though, we can see that if communication cost is zero or low, when uncertainty increases NN is much better than SC. We need to note that the time constraints play a very importance role here: when T becomes greater (longer deadline) the SC line can become better than some of the NN lines, in particular, the ones with high communication costs (not shown here due to the space constraint). The underlying reason is that when agents have enough time to perform local recovery (as in SC) without any communication, the loss of performance due to not being able

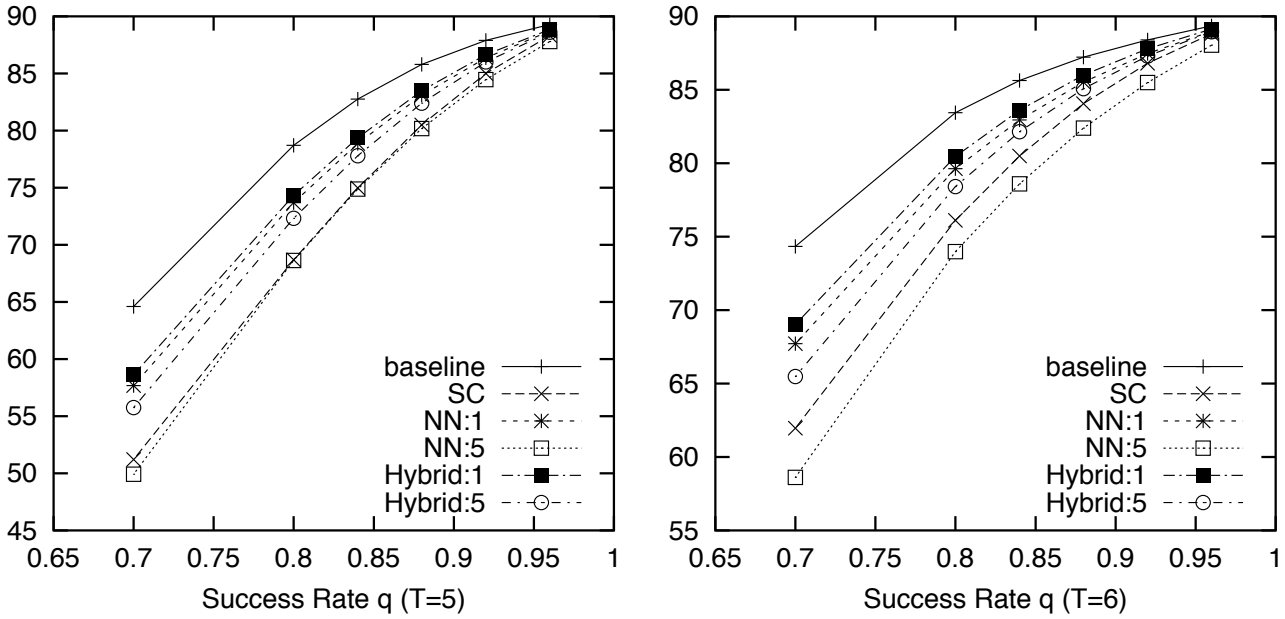

Figure 13: Success Rate

to de-commit can be offset by not spending on communication, especially when the cost of communication is quite high and the amount of communication needed could be quite large when uncertainty is high.

Overall, these three policies give us some intuition about when to use a policy that relies heavily on communication, and when to use a policy that relies little on communication. In general, frequent communication (such as NN) often means short-term/dynamic commitments, while low communication policies (such as SC) often use long-term, unchangeable, commitments. The optimum may be somewhere in the middle, although the computation demand is prohibitive. The Hybrid policy demonstrates the need for following these intuitions. More importantly, however, the success of the Hybrid policy indicates that it is necessary that we deal with communication costs directly and explicitly when designing a policy. The result of this integration is that the new policy can adapt better to different situations, and also is more effective in terms of correctly reasoning about expected utilities and decision making. In other words, we should begin to treat communication decisions the same way we do for agent action decisions.

6 Summary

In this paper we proposed a decision theoretic framework for cooperative multiagent problem solving. We showed that by modeling communication and communication decisions, we can formalize a decentralized model that characterizes the decentralized nature of decision making in agents with partial view of the global state of the system. This formal model combines both the planning and coordination aspects of multiagent problem solving and clearly defines the meaning of solution optimality. This model defines a new type of observability: observable via communication with other agents. This work introduces the model of decentralized multiagent Markov decision processes as the basis for decision theoretic study of cooperative multiagent systems, allows quantitative evaluation of planning and coordination strategies. Although the model is rather complex in terms of its computational complexity, we proposes some approximation solution methods for obtaining solutions and demonstrates that this model can effectively model heuristic solutions and provides intuitive interpretations that may help us design better heuristics.

We studied the impact of communication decisions on the construction of control policies in a multiagent setup. We argued that communication decisions are a fundamental aspect of the agent decision problem, and that the problem solving model should integrate these decisions explicitly. In our decentralized framework of a multiagent MDP, we described how communication and the cost of communication should be modeled in such a framework. In particular, we tied the issue of communication with the commitment dynamics of the agents, and we were able to gain insights into some of the most important problems of multiagent coordination, for example, when the agents should use dynamic commitments. This also leads to one particularly promising approximation method: modeling an agent's decision process as dynamically evolving Markov decision processes, and use communication to guide the evolution of the decision processes, as shown in our examples.

The study of the foundation of coordination in multiagent system has become more and more important, and we believe that a decentralized approach provides a formal foundation and captures the complexity of the problem of coordination. A lot of work remains to be done. First, since the optimal policy is history-dependent, it would be very interesting to see under what situations an approximation still maintains the optimality, i.e., under what conditions it is safe to ignore a large part of the history information.

We are still in search of efficient algorithms for approximation approaches. Since in general dynamic programming (the standard value iteration and policy iteration algorithms) cannot be used [29] in decentralized decision problems, we need to know if there are special cases where dynamic programming is possible, and if there are other efficient computational techniques that are suitable for decentralized multiagent MDPs. Besides efficient approximation methods, learning techniques such as [17] may also contribute to decentralized decision-making. However, this remains to be a very hard problem. For example, although it is possible to find policies for coordination by search in policy space, as in the work by Peshkin *et al*[21], the search can converge on local maxima and there is no way to guarantee that the approximation is close to optimal within some desired bound.

Finally, decentralized MDPs may be extended so that they cover infinite-horizon processes and also are able to deal with the case where the agents do not have the same static global understanding (for example, the agents do not have the complete map). Also, it would be very interesting to study communication when agents are clustered into sub-groups in a multiagent system. This would be very important as the system scales up.

Acknowledgments

The authors would like to thank Andy Barto and Dan Bernstein for fruitful discussions of this problem.

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.

References

[1] M. Aicardi, F. Davoli, and R. Minciardi. Decentralized optimal control of markov chains with a common past information set. *IEEE Transactions on Automatic Control*, AC-32:1028–1031, 1987.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized markov decision processes. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems*, Melbourne, Australia, 2003.

[3] 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.

[4] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI-99)*, July 1999.

[5] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 1999.

[6] Keith Decker and Victor Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, San Francisco, June 1995.

[7] Edmund H. Durfee and Victor R. Lesser. Predictability versus responsiveness: Coordinating problem solvers in dynamic domains. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 66–71, 1988.

[8] S. Fujita and V. Lesser. Centralized task distribution in the presence of uncertainty and time deadlines. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 87–94, 1996.

[9] Piotr J. Gmytrasiewicz and Edmond H. Durfee. Rational interaction in multiagent environments: Coordination. *Autonomous Agents and Multi-Agent Systems Journal*, 1999.

[10] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems*, Melbourne, Australia, 2003.

[11] E. Hansen. Cost-effective sensing during plan execution. In *Proceedings of the Twelth National Conference on Artificial Intelligence*, 1994.

[12] E. Hansen, A. Barto, and S. Zilberstein. Reinforcement learning for mixed open-loop and closed-loop control. In *Proceedings of the Ninth Neural Information Processing Systems Conference*, December 1996.

[13] E. A. Hansen and S. Zilberstein. Monitoring the progress of anytime problem-solving. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1229–1234, 1996.

[14] Y. C. Ho and T. S. Chang. Another look at the nonclassical information problem. *IEEE Transactions on Automatic Control*, AC-25:537–540, 1980.

[15] K. Hsu and S. I. Marcus. Decentralized control of finite state markov processes. *IEEE Transactions on Automatic Control*, AC-27:426–431, 1982.

[16] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 1993.

[17] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. 11th International Conf. on Machine Learning*, pages 157–163, 1994.

[18] R. Duncan Luce and Howard Raiffa. *Games and Decisions: Introduction and Critical Survey.* John Wiley & Sons, New York, NY, 1958.

[19] G.M.P. O'Hare and N.R. Jennings, editors. *Foundations of Distributed Artificial Intelligence.* John Wiley, 1996.

[20] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[21] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *Proceedings of the the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 489–496, Stanford, CA, 2000.

[22] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 16:389–423, 2002.

[23] N. R. Sandell, P. Varaiya, M. Athans, and M. Safonov. Survey of decentralized control methods for large scale systems. *IEEE Transactions on Automatic Control*, AC-23:108–128, 1978.

[24] J. N. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, AC-30:440–446, 1985.

[25] H. S. Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal on Control*, 6(1):138–147, 1968.

[26] Ping Xuan and Victor Lesser. Multi-agent policies: from centralized ones to decentralized ones. *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, 2002.

[27] Ping Xuan, Victor Lesser, and Shlomo Zilberstein. Communication in multi-agent markov decision processes. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, 2000. Poster paper.

[28] 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)*, pages 616–623, Montreal, Canada, 2001.

[29] T. Yoshikawa. Decomposition of dynamic team decision problems. *IEEE Transactions on Automatic Control*, AC-23:443–445, 1978.