In *AAMAS 2009 workshop on Agent Mediated Electronic Commerce* (AMEC 2009), May 2009.

Leading a Best-Response Teammate in an Ad Hoc Team

Peter Stone[1], Gal A. Kaminka[2], and Jeffrey S. Rosenschein[3]

[1]Dept. of Comp. Sci.
U. of Texas at Austin
pstone@cs.utexas.edu

[2]Dept. of Comp. Sci.
Bar Ilan U.
galk@cs.biu.ac.il

[3]School of Eng. and Comp. Sci.
Hebrew U.
jeff@cs.huji.ac.il

Abstract. Teams of agents may not always be developed in a planned, coordinated fashion. Rather, as deployed agents become more common in e-commerce and other settings, there are increasing opportunities for previously unacquainted agents to cooperate in ad hoc team settings. In such scenarios, it is useful for individual agents to be able to collaborate with a wide variety of possible teammates under the philosophy that not all agents are fully rational. This paper considers an agent that is to interact repeatedly with a teammate that will adapt to this interaction in a particular suboptimal, but natural way. We formalize this setting in game-theoretic terms, provide and analyze a fully-implemented algorithm for finding optimal action sequences, prove some theoretical results pertaining to the lengths of these action sequences, and provide empirical results pertaining to the prevalence of our problem of interest in random interaction settings.

1 Introduction

As agents proliferate in the world, both in software and robotic settings, they will increasingly need to band together for cooperative activities with previously unknown or unfamiliar teammates. For example, consider a disaster rescue scenario in which robots developed by many different people in different parts of the world converge to work together to locate and extract victims from places that are yet too dangerous for human rescue teams to enter. These robots can be thought of as forming a *team*: they are fully cooperative with no notion whatsoever of individual self-interest separate from the team's interest. They all aim to act so as to maximize the likelihood of finding survivors, even if it means risking their own safety.

However, unlike most team settings considered so far (e.g., [15]), the robots are not all programmed by the same people, and may not all have the same communication protocols or world models. Furthermore, they are likely to have heterogeneous sensing and acting capabilities that may not be fully known to each other. As a result, team strategies cannot be developed a priori. Rather, a robot that is to succeed in such an *ad hoc team* setting must be prepared to cooperate with many types of teammates: those with which it can communicate and those with which it cannot; those that are more mobile and those that are less mobile; those with better sensing capabilities and those with worse capabilities. A good team player's best actions are likely to differ significantly depending on the characteristics of its teammates.

In this paper, we consider the case of such a good team player, *Agent A* that is interacting with a teammate, *Agent B*, with whom it cannot communicate directly, but that is capable of adapting to its teammate's behavior. Specifically, *Agent B* observes its teammate's actions and acts according to the best response to some fixed history window of *Agent A*'s past moves. *Agent A*'s goal is to find the sequence of moves that will lead to the highest (expected) payoff in a fully cooperative setting. In addition to the robot disaster rescue scenario mentioned above, such a situation may arise in an e-commerce setting, for example, if *Agent B* is a legacy trading agent within a firm that is essential to daily operations but difficult or impossible to modify, whereas *Agent A* is a new, more intelligent agent that needs to take *Agent B*'s behavior into account when selecting its own actions. In this paper, we abstract this setting to a game-theoretic formalism in which the agents interact in a fully cooperative iterative normal form game.

The remainder of the paper is organized as follows. First, in Section 2, we provide an example game-theoretic setting and formalize the situation of study. Then, in Section 3, we present some analytical results, followed by some empirical results in Section 4. Section 5 situates our problem of interest within both the game theory and agent modeling literature, and Section 6 concludes.

2 Formalism and Example

In this paper, we represent the multiagent interaction of interest as a fully cooperative iterative normal-form game between two agents, *Agent A* and *Agent B*. Throughout the paper, we will consider *Agent A* to be the agent that is within our control; *Agent B*, which reacts in a fixed way, is given by the environment.

Let the x actions available to *Agent A* be $a_0, a_1, \ldots, a_{x-1}$ and the y actions available to its teammate, *Agent B*, be $b_0, b_1, \ldots, b_{y-1}$. The immediate payoff when *A* and *B* select actions a_i and b_j, $m_{i,j}$ is stored in row i and column j of the payoff matrix M: $M[i, j] = m_{i,j}$. In addition we define the value of the highest payoff in the matrix, which could be realized by multiple entries, to be m^*. Without loss of generality, throughout this paper, we assume that $m_{x-1,y-1} = m^*$.

For example, consider the payoff matrix $M1$ for a scenario in which agents *A* and *B* each have three possible actions. If both agents select action 0 (i.e., their *joint action* is (a_0, b_0)), then the joint team payoff is $m_{0,0} = 25$. Similarly if their joint action is (a_2, b_0) their joint payoff is 0. In this case, there is a unique joint action that leads to m^*: $m_{2,2} = m^* = 40$.

$M1$	b_0	b_1	b_2
a_0	25	1	0
a_1	10	30	10
a_2	0	33	40

Assume that b_0 is *Agent B*'s default action or that, for whatever reason, the agents have been playing (a_0, b_0) in the past. This could be, for example, because *Agent B* is not fully aware of *Agent A*'s payoffs so that it cannot unilaterally identify the best joint action, or because *B* does not fully trust that *A* will play its part of the best joint action. The question we examine is what sequence of actions should *Agent A* take so as to maximize the team's undiscounted long-term payoff over iterative interactions using the identical payoff matrix? The answer to this question depends on *Agent B*'s strategy. For example, if *Agent B* is non-adaptive and always selects b_0, then the best *Agent A* can do is always select a_0.

However, if *Agent B* is adaptive, *Agent A* can lead it towards the optimal joint action by taking a sequence of actions the responses to which will cause *Agent B* to abandon

b_0 and choose other actions. In order to do so, it may need to accept short-term losses with respect to the current payoffs (e.g., immediate payoffs of less than 25); however in the long run these losses will be offset by the repeated advantageous payoff of (a_2, b_2).[1]

In this paper, we consider a particular class of strategies that *Agent B* could be using. Though they may not be the most sophisticated imaginable strategies, they are reasonable and often studied in the literature. The fact that they are possibly suboptimal represents the philosophy that *Agent A* must be able to adapt to its teammates as they are, not as they should be. That is, we assume that we have control only over *Agent A*, not over *Agent B*.

In particular, we specify *Agent B* as being a bounded-memory best response agent with an ϵ-greedy action strategy. That is, the agent's behavior is determined by two parameters: a memory size *mem*; and a random action rate ϵ. The agent considers the most recent *mem* actions taken by its teammate (*Agent A*), and assumes that they have been generated by the maximum likelihood policy that assigns fixed probabilities to each action. For example, if *mem* $= 4$ and *Agent A*'s last four actions were a_1, a_0, a_1, a_1, then *Agent B* assumes that *Agent A*'s next action will be a_0 with probability 0.25 and a_1 with probability 0.75. It then selects the action that is the best response to this assumed policy with probability $1-\epsilon$; with probability ϵ it chooses a random action. For example, for payoff matrix $M1$ in this situation, it would select b_1 with probability $1 - \epsilon$. We denote this *best response* action as *BR*$(a_1, a_0, a_1, a_1) = b_1$. Note that when $\epsilon = 1$, the agent acts completely randomly.

To illustrate, we begin by considering the case of *mem* $= 1$ and $\epsilon = 0$. For the remainder of this section, we consider the same case, in which *Agent B* always selects the action that is the best response to *Agent A*'s previous action: b_0, b_1, or b_2 depending on whether *A*'s last action was a_0, a_1, or a_2 respectively.

Now consider *Agent A*'s possible action sequences starting from the joint action (a_0, b_0) with payoff $m_{0,0} = 25$. Because its last action was a_0, it knows that *B* will select b_0 on the next play. It could immediately jump to action a_2, leading to the joint action (a_2, b_0). This action will lead to an immediate payoff of $m_{2,0} = 0$, but then will cause *Agent B* to select b_2 next, enabling a payoff of 40 on the next turn and thereafter (assuming *A* continues to select a_2 as it should). The resulting sequence of joint actions would be $S_0 = [(a_0, b_0), (a_2, b_0), (a_2, b_2), (a_2, b_2), \ldots]$ leading to payoffs $[25, 0, 40, 40, \ldots]$.

Alternatively, *Agent A* could move more gradually through the matrix, first selecting a_1 for a joint payoff of 10 and leading *B* to select b_1 on its next turn. It could then shift to a_2 for a payoff of 33, followed by 40 thereafter. The resulting sequence of joint actions would be $S_1 = [(a_0, b_0), (a_1, b_0), (a_2, b_1), (a_2, b_2), (a_2, b_2), \ldots]$ leading to payoffs $[25, 10, 33, 40, 40, \ldots]$.

We define the *cost* $C(S)$ of a joint action sequence S to be the loss from playing S when compared to always playing the joint action (a_{x-1}, b_{y-1}), which leads to payoff

[1] In principle, it is possible that the game will not continue long enough to offset these losses. In this paper, we assume that the game will be repeated a large enough number of times that it will not terminate before the agents reach the best joint action in the way that we specify. In a setting where this is not the case, one would need to include the number of iterations left as a part of the state.

m^* — in the case of $M1$, 40. Thus

$$C(S_0) = (40-25)+(40-0)+(40-40)+(40-40)+\cdots = 15+40+0+0+\cdots = 55$$

and

$$C(S_1) = (40-25)+(40-10)+(40-33)+(40-40)+\cdots = 15+30+7+0+0+\cdots = 52$$

In this case, S_1 is preferable to S_0, and is in fact the optimal (lowest cost) sequence starting from (a_0, b_0).

We define the *length* $L(S)$ of a joint action sequence S to be the number of joint actions prior to the first instance of the infinite sequence of joint actions that yield m^*. Thus $L(S_0) = 2$ and $L(S_1) = 3$. Note that S_1 has lower cost even though it is longer. Note also that sequences that begin with a joint action (a_i, b_j) such that $m_{i,j} = m^*$ have both length 0 and cost 0.

For a given payoff matrix, we define $S_n^*(a_i, b_j)$ to be the lowest cost sequence of length n or less starting from joint action (a_i, b_j). $S^*(a_i, b_j)$ is the lowest cost such sequence of any length. Thus, for matrix $M1$, $S_2^*(a_0, b_0) = S_0$ and $S_3^*(a_0, b_0) = S^*(a_0, b_0) = S_1$.

For the special case that no sequence of a given length exists (e.g., if $n = 0$ or $n = 1$), we define $S^*(a_i, b_j) = \omega$ and $C(\omega) = \infty$. Thus, for $M1$, $C(S_0^*(a_0, b_0)) = C(S_1^*(a_0, b_0)) = \infty$, but $C(S_1^*(a_2, b_1)) = 7$ and $C(S_0^*(a_2, b_2)) = 0$.

Finally, for a given payoff matrix M, we are interested in the length of the longest optimal sequence over all the possible starting points. We define this value as $\overline{L}(M) = \max_{i,j} L(S^*(a_i, b_j))$. For example, in matrix $M1$, $L(S^*(a_0, b_0)) = L(S_1) = 3$, and there is no optimal sequence longer than 3 starting from any other cell of the matrix (as we will prove below). Thus $\overline{L}(M1) = 3$.

3 Finding Optimal Sequences and Analysis

In this section, we develop algorithms for finding $S^*(a_i, b_j)$ given a payoff matrix M, and we examine the question of how long these S^*'s can be. We divide the analysis based on *Agent B*'s strategy. First, in Section 3.1 we assume that *Agent B* has *mem* $= 1$ and $\epsilon = 0$ as in Section 2. Next in Section 3.2 we consider the more difficult case of *mem* > 1. Then, in Section 3.3 we allow *Agent B*'s actions to be non-deterministic by considering $\epsilon > 0$.

3.1 Deterministic Teammate with 1-Step Memory

We begin by presenting an efficient algorithm for finding all of the S^*'s for a matrix M when interacting with a deterministic teammate ($\epsilon = 0$) that always selects the best response to our most recent action (*mem* $= 1$). Detailed in pseudocode as Algorithm 1, it uses dynamic programming, using the S_{n-1}^*'s to compute the S_n^*'s.

The algorithm takes as input an $x \times y$ dimensional payoff matrix M and begins by initializing the optimal sequence of length 0 for every cell in the matrix according to the definition (lines 1–5). It then enters the main loop (7–21) that successively finds the best sequences of increasingly longer lengths (as indicated by the variable *len*).

A key insight that aids efficiency is that for a given a_i, the optimal sequences for b_1–b_y are the same as the optimal sequence starting from (a_i, b_0), other than the first joint action. The reason is that a_i determines *Agent B*'s next action independently from *Agent B*'s current action: in all cases, its next action will be $b_{BR(a_i)}$. Thus, *Agent A*'s task is to select its action, a_{act}, that leads to the best possible joint action of the form $(a_{act}, b_{BR(a_i)})$.

Algorithm 1 Find S^*'s (M)

1: **for** $i = 0$ to $x - 1$ **do**
2:   **for** $j = 0$ to $y - 1$ **do**
3:     $S_0^*(a_i, b_i) = \begin{cases} [(a_i, b_i), (a_i, b_i), \ldots] & \text{if } m_{i,j} = m^* \\ \omega & \text{if } m_{i,j} < m^* \end{cases}$
4:   **end for**
5: **end for**
6: *len* $= 0$
7: **repeat**
8:   *len* $=$ *len* $+ 1$
9:   **for** $i = 0$ to $x - 1$ **do**
10:     $S^*_{len}(a_i, b_0) = S^*_{len-1}(a_i, b_0)$
11:     **for** $act = 0$ to $x - 1$ **do**
12:       $S' = S^*_{len-1}(a_{act}, b_{BR(a_i)})$
13:       **if** $m^* - m_{i,0} + C(S') < C(S^*_{len}(a_i, b_0))$ **then**
14:         $S^*_{len}(a_i, b_0) =$ PREPEND$((a_i, b_0), S')$
15:       **end if**
16:     **end for**
17:     **for** $j = 1$ to $y - 1$ **do**
18:       $S^*_{len}(a_i, b_j) =$ REPLACEHEAD$(S^*_{len}(a_i, b_0), (a_i, b_j))$
19:     **end for**
20:   **end for**
21: **until** *len* $=$ UPPERBOUND$(\overline{L}(M))$

This very computation is carried out in lines 10–16, specifically for *Agent B*'s action b_0. First, it is possible that the optimal sequence of length *len*, $S^*_{len}(a_i, b_0)$ is the same as that of length *len*-1. Thus it is initialized as such (line 10). Then for each possible next action on the part of *Agent A*, denoted a_{act}, the cost of the resulting sequence is simply the cost of the current joint action (a_i, b_0), which is $m^* - m_{i,0}$, plus the cost of the best possible sequence of length *len* $- 1$ that starts from $(a_{act}, b_{BR(a_i)})$. If that cost is less than the cost of the best sequence of length *len* found so far, then the running best sequence is updated accordingly by prepending joint action (a_i, b_0) to the sequence $S^*_{len-1}(a_{act}, b_{BR(a_i)})$ (lines 14–16).

The resulting optimal sequence is then used to determine the optimal sequence starting from all other values of (a_i, b_j) for $1 \le j < y$ by simply replacing the first joint action in the sequence $S^*_{len}(a_i, b_0)$ with the joint action (a_i, b_j) (lines 17–19). At the end of this loop, the optimal sequence of length *len* starting from any joint action (a_i, b_j) $(S^*_{len}(a_i, b_j))$ is known and stored.

The computational complexity of the main loop of Algorithm 1 (lines 7–21) is quadratic in x and linear in y. Assuming x and y are of similar dimension (Agents *A* and *B* have roughly the same number of possible actions), we can call the dimensionality of M to be $d = \max(x, y)$. In that case, the main loop has complexity $O(d^2)$.

Note that sequence costs $C(S)$ can be calculated incrementally in constant time as the sequences are constructed.

The only thing left to determine is how many times this main loop needs to be run. In particular, for what value of *len* is it no longer possible to find a better sequence than the best of length *len* -1. We denote this value UPPERBOUND($\overline{L}(M)$). The following two theorems prove that this value is exactly $\min(x, y)$. Thus the overall computational complexity of algorithm 1 is $O(d^3)$.

First, in Theorem 1, we prove that there is no need to consider sequences of length greater than $\min(x, y)$: UPPERBOUND($\overline{L}(M)$) $\leq \min(x, y)$. Then, in Theorem 1, we show that it *is* necessary to to consider sequences up to length $\min(x, y)$: UPPERBOUND($\overline{L}(M)$) $\geq \min(x, y)$.

Theorem 1. *When interacting with a teammate with* $\text{mem} = 1$ *and* $\epsilon = 0$ *based on an* $x \times y$ *dimensional payoff matrix* M, $\overline{L}(M) \leq \min(x, y)$

Proof. We argue that $\forall M, \overline{L}(M) \leq \min(x, y)$ by first showing that $\overline{L}(M) \leq x$ and then showing that $\overline{L}(M) \leq y$. Intuitively, both cases hold because an optimal sequence can visit every row and column in the matrix at most once. If there were multiple visits to the same row or column, any steps in between could be excised from the sequence to get a lower-cost sequence. The formal arguments for the two cases are quite similar, though with a couple of subtle differences.

Case 1: $\overline{L}(M) \leq x$**.** This is equivalent to proving $\forall n \geq x$, and $\forall i, j, S^*_{n+1}(a_i, b_j) = S^*_n(a_i, b_j)$. Suppose not. Then $\exists k$ and a corresponding sequence S' such that $S' = S^*_{n+1}(a_i, b_j) =$ PREPEND$((a_i, b_j), S^*_n(a_k, b_{BR}(i)))$ with $C(S') < C(S^*_n(a_i, b_j))$. Since $S^*_n(a_i, b_j)$ is the optimal sequence of length n or less, $L(S') = n + 1$. $n + 1 > x$, so by the pigeonhole principle, $\exists q$ such that *Agent A* selects a_q more than once in S' prior to the first instance of the terminal joint action with value m^*. Assume that (a_q, b_r) appears earlier in the sequence than $(a_q, b_{r'})$. In both cases, *Agent B*'s next action in the sequence must be *BR*(a_q). Thus after joint action (a_q, b_r), *Agent A* could have continued as it actually did after $(a_q, b_{r'})$. This revised sequence would have cost less than S', violating the assumption that $S' = S^*_{n+1}(a_i, b_j)$. Therefore $\overline{L}(M) \leq x$.

Case 2: $\overline{L}(M) \leq y$**.** Similarly, this case is equivalent to proving that $\forall n \geq y$, and $\forall i, j, S^*_{n+1}(a_i, b_j) = S^*_n(a_i, b_j)$. Suppose not. Then $\exists k$ and a corresponding sequence S' such that $S' = S^*_{n+1}(a_i, b_j) =$ PREPEND$((a_i, b_j), S^*_n(a_k, b_{BR}(i)))$ with $C(S') < C(S^*_n(a_i, b_j))$. Since $S^*_n(a_i, b_j)$ is the optimal sequence of length n or less, $L(S') = n + 1$. $n + 1 > y$, so by the pigeonhole principle, $\exists r$ such that *Agent B* selects b_r more than once in S' after the first entry (a_i, b_j) and up to and including the first instance of the terminal joint action with value m^*.[2] Assume that (a_q, b_r) appears earlier in the sequence than $(a_{q'}, b_r)$. Then at the point when *Agent A* selected a_q leading to (a_q, b_r), it could have instead selected $a_{q'}$, and then finished the sequence as from $(a_{q'}, b_r)$ in S'. Again, this revised sequence would have cost less than S', violating the assumption that $S' = S^*_{n+1}(a_i, b_j)$. Therefore $\overline{L}(M) \leq y$.

Therefore $\forall M, \overline{L}(M) \leq \min(x, y)$. □

[2] This portion of the sequence still includes $n + 1$ elements, since we are ignoring the first element (a_i, b_j), but then including the first instance of the terminal joint action.

Theorem 2. *$\forall x, y, \exists\ x \times y$ dimensional matrix M such that, when interacting with a teammate with* $\text{mem} = 1$ *and* $\epsilon = 0$, $\overline{L}(M) = \min(x, y)$.

Proof. To prove existence, we construct such a matrix.

Case 1: $x = y$**.** Consider the matrix $M2$ where $\delta = 10/x$. All cells on the diagonal are $100 - \delta$ except for the bottom right corner, $m_{x-1,y-1} = m^* = 100$. All cells below this diagonal are $100 - 2\delta$, and all other cells are 0. We show that for $M2$, $L(S^*(a_0, b_0)) = x$. Specifically,

$$S^*(a_0, b_0) = [(a_0, b_0), (a_1, b_0), (a_2, b_1), \ldots, (a_{x-2}, b_{y-3}), (a_{x-1}, b_{y-2}), (a_{x-1}, b_{y-1})].$$

$M2$	b_0	b_1	b_2	$\cdots$	b_{y-3}	b_{y-2}	b_{y-1}
a_0	$100-\delta$	0	0	$\cdots$	0	0	0
a_1	$100-2\delta$	$100-\delta$	0		$\vdots$	0	0
a_2	0	$100-2\delta$	$100-\delta$			$\vdots$	0
$\vdots$	$\vdots$		$\ddots$	$\ddots$			$\vdots$
a_{x-3}	0	$\vdots$		$\ddots$	$100-\delta$	0	0
a_{x-2}	0	0	$\vdots$		$100-2\delta$	$100-\delta$	0
a_{x-1}	0	0	0	$\cdots$	0	$100-2\delta$	100

To see that this sequence is optimal, note that its cost is $\delta + (x-1) * 2\delta < 2x\delta = 20$. Note further, that $\forall i, \mathit{BR}(a_i) = b_i$. Now working backwards, in order to reach the optimal joint action (a_{x-1}, b_{y-1}), *Agent A* must have selected action a_{x-1} in the iteration prior to the first appearance of (a_{x-1}, b_{y-1}) in the sequence. When that happened, if *Agent B* had selected anything other than b_{y-2} (b_{y-1} is not an option since we are considering the iteration prior to the *first* appearance of b_{y-1} in the sequence), then there would have been a payoff of 0, leading to a sequence cost of ≥ 100. Thus joint action (a_{x-1}, b_{y-2}) must appear in the optimal sequence. Similarly, considering the first appearance of this joint action, for *Agent B* to have selected b_{y-2}, *Agent A* must have selected a_{x-2} on the prior iteration. Again, any joint action other than (a_{x-2}, b_{y-3}) (here b_{y-2} is not an option for the same reason as above) leads to a payoff of 0 and a sequence cost of ≥ 100. Continuing this line of reasoning, we can see that all the cells under the diagonal must appear in the optimal sequence starting from joint action (a_0, b_0). Furthermore, adding any additional joint actions (including those on the diagonal) only raise the cost. Therefore the sequence presented above, of length x, is indeed $S^*(a_0, b_0)$. It is easy to see that no optimal sequence from any other cell is longer.[3] Thus $\forall x, \exists x \times x$ dimension matrix M such that $\overline{L}(M) = x = \min(x, y)$.

Case 2: $x < y$**.** If $x < y$ we can construct a matrix $M2'$ that includes the $x \times x$ dimensional version of $M2$ as a submatrix and contains an additional $y - x$ columns of all 0's. By the same argument as above, $S^*(a_0, b_0)$ is the same sequence as above, which is of length x: $\overline{L}(M2') = x = \min(x, y)$.

Case 3: $x > y$**.** In this case, we can construct a matrix $M2'$ based on the $y \times y$ dimensional version of $M2$ that adds $x - y$ rows of all 0's. Again, $S^*(a_0, b_0)$ is the same as above and $\overline{L}(M2') = y = \min(x, y)$.

[3] To be precise, $\forall i, j, L(S^*(a_i, b_j)) = x - i$ with one exception: $L(S^*(a_{x-1}, b_{y-1})) = 0$.

Therefore, $\forall x, y, \exists$ an $x \times y$ dimensional matrix M such that $\overline{L}(M) = \min(x, y)$. $\square$

Theorems 1 and 2 establish that the value of the call to the function UPPERBOUND in line 21 of Algorithm 1 is $\min(x, y)$.

Note that in our analysis of this case in which *Agent B* has *mem* $= 1$ and $\epsilon = 0$, all of the arguments hold even if there are multiple cells in the payoff matrix M with value m^*. Furthermore, Algorithm 1 computes the optimal sequence of joint actions from *all* starting points, not just a particular starting point, all in polynomial time in the dimensionality of the matrix.

3.2 Longer Teammate Memory

In this section we extend our analysis from the previous section to consider interacting with teammates with *mem* > 1. This case presents considerably more difficulty than the previous one in two ways. First, though the algorithm can be naturally extended, it is no longer polynomial, but rather exponential in *mem*. Second, it is no longer straightforward to compute UPPERBOUND($\overline{L}(M)$), the maximum value of $L(S^*(a_i, b_j))$. We identify a lower bound on this maximum value, but can only conjecture that it is a tight bound.

Since the algorithm and analysis is so similar to that in Section 3.1, rather than presenting them fully formally, we discuss how they differ from the previous case.

To begin with, we need an added bit of notation for indicating sequences. Because *Agent B*'s actions are now no longer determined by just *Agent A*'s previous action, but rather by *Agent A*'s history of previous *mem* actions, we keep track of these actions in the sequence, indicating a step as $(a_i, b_j)[h_0; h_1; \ldots; h_{mem-1}]$ where $h_0 = a_i$ is *Agent A*'s most recent action, h_1 is its prior action, etc. Then *Agent B*'s next action in the sequence must be $b_r = BR(h_0, h_1, \ldots, h_{mem-1})$ and if *Agent A*'s next action is a_q, then the next element in the sequence is $(a_q, b_r)[a_q; a_i; h_1; \ldots; h_{mem-2}]$.

For example, returning to matrix $M1$ from Section 2, consider the case in which *Agent B* has *mem* $= 3$ (and still $\epsilon = 0$ throughout this section). A valid sequence starting from $(a_0, b_0)[a_0; a_0; a_0]$ is

$$S_2 = [(a_0, b_0)[a_0; a_0; a_0], (a_2, b_0)[a_2; a_0; a_0], (a_2, b_0)[a_2; a_2; a_0], (a_2, b_2)[a_2; a_2; a_2]]$$

Note that because $BR(a_2, a_0, a_0) = b_0$, *Agent A* needs to select a_2 twice before *Agent B* will shift to b_2. $C(S_2) = 15 + 40 + 40 = 95$. As in Section 2, there is another valid sequence S_3 in which *Agent A* leads *Agent B* through joint actions (a_1, b_0) and (a_2, b_1) on the way to (a_2, b_2). But now, *Agent A* must select a_1 twice before *B* will switch to b_1 and then a_2 three times before *B* will switch to b_2. Thus $C(S_3) = 25 + 2 * 30 + 3 * 7 = 106$. Hence, unlike in Section 2, when *Agent B* has *mem* $= 3$, *Agent A* is best off jumping straight to a_2.

The first necessary alteration to Algorithm 1 in this case is that it is no longer sufficient to simply calculate S^*_{len} for every joint action (a_i, b_j) on each loop of the algorithm. Rather, we must now calculate such values for each joint action-history $(a_i, b_j)[h_0; \ldots; h_{mem\text{-}1}]$. Since h_0 is constrained to be the same as a_i, there are x^{mem-1} such histories for each joint action, leading to a total of $x^{mem}y$ optimal sequences computed on each main loop of the algorithm. To accommodate this alteration, we simply need to nest additional **for** loops after lines 2 and 10 of Algorithm 1 that iterate over the (exponential number of) possible histories.

The second necessary alteration to Algorithm 1 in this case is that it is no longer sufficient to simply arrive at a joint action (a_i, b_j) such that $m_i, j = m^*$. Rather, the agents must arrive at such an action with a history of *Agent A*'s actions such that if it keeps playing a_i, *Agent B* will keep selecting b_j. We define such a joint action-history to be *stable*.

To see why the concept of stability is necessary, consider matrix $M3$. A valid sequence starting from $(a_2, b_2)[a_2; a_1; a_0]$ proceeds to $(a_2, b_2)[a_2; a_2; a_1]$ if *Agent A* selects a_2. However from there, *Agent B*'s best response is b_0, not b_2. Thus the agents do not remain stably at joint action (a_2, b_2).

$M3$	b_0	b_1	b_2
a_0	0	30	50
a_1	41	20	0
a_2	99	20	100

To accommodate this situation, the only change to Algorithm 1 that is needed is that in line 3, only stable joint-action histories such that $m_{i,j} = m^*$ should be initialized to the sequence of repeated terminal joint actions. Unstable ones should be initialized to ω (along with all instances such that $m_{i,j} < m^*$, no matter what the history). We can check stability by computing the best response to all histories that result from repeating action a_i until the entire history window is full of action a_i. If any of these best responses is not b_j, then the joint action-history is not stable.

Third, the main loop of Algorithm 1 needs to be altered to accommodate the inclusion of histories. In particular, in line 12, care needs to be taken to compute S' correctly, with *Agent B*'s action being based on the best response to the current history, and the history being the result of taking action a_i from the current history. Furthermore the PREPEND and REPLACEHEAD operators must manipulate the histories (and incremental cost computations) in the appropriate, obvious ways.

Finally, and most significantly, the value of UPPERBOUND in line 21 of Algorithm 1 must be altered. Unfortunately, we only can prove a lower bound of this value and a loose upper bound ($\min(x, y) * x^{mem-1}$). We conjecture, but have not proven, that the lower bound is tight as it is in Section 3.1.

Theorem 3. *$\forall x, y, \exists\ x \times y$ dimensional matrix M such that, when interacting with a teammate with* mem > 1 *and* $\epsilon = 0$, $\overline{L}(M) = (\min(x, y) - 1) * mem + 1$.

Proof. (*sketch*) This theorem, which is the analog of Theorem 2, can be proven using a similar construction. In particular, redefining δ as $\delta = 10/((x - 1) * mem + 1)$, the same matrix $M2$ serves as our existence proof. Consider the optimal sequence starting from (a_0, b_0) with history full of a_0's. In that case, *Agent A* needs to select action a_1 *mem* times before *Agent B* will switch to b_1. Similarly, it then needs to select a_2 *mem* times before *B* will switch to b_2, and so on until *A* has selected each of the actions a_1–a_{x-1} *mem* times. The additional one is for the initial action (a_0, b_0) which appears only once in the sequence. As before, any joint actions with payoff 0 will lead to a higher sequence cost than this entire sequence, and any additional joint actions also increase the cost.

Also as before, the cases of $x \neq y$ are covered by simply adding extra rows or columns of 0's to $M2$ as needed. □

Conjecture 1. When interacting with a teammate with *mem* > 1 and $\epsilon = 0$ based on an $x \times y$ dimensional payoff matrix M, $\overline{L}(M) \leq (\min(x, y) - 1) * mem + 1$.

Proving or disproving this conjecture is left as an important direction for future work. An additional important direction for future work is developing heuristics for more efficiently finding the S^*'s when *mem* > 1.[4] The exponential runtime in *mem* of the algorithm for finding the S^*'s is of practical significance. Our algorithm finds all the best sequences for a 60×60 matrix in less than 30 seconds of user time on a 1GHz laptop (calculated by the Unix `time` command) when *mem* $= 1$, but it can only handle an 18×18 matrix in that time when *mem* $= 2$, a 9×9 matrix when *mem* $= 3$, 6×6 when *mem* $= 4$, and 4×4 when *mem* $= 5$. For larger matrices than those listed, java ran out of heap space with the default settings, often after running for more than 10 minutes.

3.3 Teammate Non-Determinism

Until this point, we have assumed that *Agent B* acts deterministically: *Agent A* could predict *Agent B*'s next action with certainty based on its own previous actions. In this section we relax that assumption by allowing *B*'s ϵ to be greater than 0.

Once again, Algorithm 1 needs to be changed minimally to accommodate this case, so we just describe the changes. In fact, here, the only change necessary is that costs of joint actions be computed as expected values in comparison to the expected value of the optimal joint action.

The expected value of a joint action $EV(a_i, b_j) = (1 - \epsilon)m_{i,j} + \frac{\epsilon}{y}(\sum_{k=0}^{y-1} m_{i,k})$. m^* is then defined to be the maximum expected value of a joint action in M. The cost of a sequence $C(S)$ is then the sum of the differences between m^* and the expected values of the joint actions in the sequence. After these changes in notation, which simply generalize our previous notation (all prior definitions hold for the case when $\epsilon = 0$), the only change necessary to Algorithm 1 is in line 13: the term $m_{i,0}$ must be replaced by $EV(a_i, b_0)$. The notion of stable joint action-histories remains unchanged from Section 3.2.

$M4$	b_0	b_1	b_2	b_3
a_0	25	0	0	0
a_1	88	90	99	80
a_2	70	98	99	80
a_3	70	70	98	100

Note that as ϵ changes, both the optimal sequence of joint actions and the "target" joint actions (the ones that lead to expected value of m^*) can change. For example, consider the 4×4 matrix, $M4$. If *Agent B*'s *mem* $= 3$, then if its $\epsilon = 0$, the optimal sequence from (a_0, b_0) starting with history $[a_0; a_0; a_0]$ ends at (a_3, b_3) and has length 10: $L(S^*(a_0, b_0)[0; 0; 0]) = 10$. When $\epsilon = 0.1$, and $\epsilon = 0.3$ the optimal lengths are 8 and 3 respectively, still ending at (a_3, b_3). When $\epsilon = 0.4$, the optimal sequence is of length 3, but now ends at (a_2, b_2). All of these sequences have different costs.

The intuitive reason for these changes is that as ϵ increases, it is no longer sufficient to reach a good cell in the matrix, but rather *Agent A* must aim for a good row: any value in the row is possible to be the payoff of the joint action. For this reason, with high ϵ, the row corresponding to a_2 is preferable to that corresponding to a_3 (the sum of the values is higher).

[4] Unfortunately, the problem is NP hard. Thanks to Michael Littman for helping work out a reduction from Hamiltonian cycle.

The analysis of the algorithmic runtime remains mostly unchanged. For efficiency, the expected values of joint actions can be cached so that they only need to be computed once. However ϵ does have some effects on the value of UPPERBOUND in line 21 of the algorithm. For $\epsilon < 1$, Theorems 1–3 all hold, though δ in the example matrix $M2$ needs to be generalized to $\delta = \frac{20(1-\epsilon)}{((x+1)*mem)(2-2\epsilon+\frac{\epsilon}{y})}$. However when $\epsilon = 1$, UPPERBOUND$(\overline{L}(M)) = 1$: *Agent A* can always jump immediately to the action that leads to the row with the highest expected value, which will be attained by all joint actions in that row. It is not clear whether ϵ has any effect on Conjecture 1.

4 Empirical Results

All variations of the algorithm presented in Section 3 are fully implemented. In this section, we present some brief empirical results from running them in various settings that shed some light on the nature and prevalence of our problem of interest.

In particular, we consider how frequently action sequences of various lengths appear in random matrices. At first blush, it may seem that when interacting with an agent with *mem* $= 1$, matrices for which there $\exists(a_i, b_j)$ such that $L(S^*(a_i, b_j)) > 2$ (such as $M1$ in Section 2) would be relatively rare in practice.

To test this hypothesis, we generated random $x \times y$ matrices such that $m_{x-1,y-1} = 100$ and all other values $m_{i,j}$ are generated uniformly randomly from $[0, 100]$. Table 1 shows the distribution of $\overline{L}(M)$ for $x \times x$ matrices when *Agent B*'s *mem* $= 1$ or 3. For matrices larger than 7×7, the *mem* $= 3$ case takes more than a day to run on a modern laptop, so we stop at that point. Matrices such that $x \neq y$ did not show any interestingly different patterns.

mem=1	1	2	3	4	5	6	7	8	9	10
3×3	104	852	44							
4×4	12	825	158	5						
5×5	3	662	316	19	0					
6×6	0	465	489	45	1	0				
7×7	0	349	565	81	5	0	0			
8×8	0	236	596	159	8	1	0	0		
9×9	0	145	640	193	20	2	0	0	0	
10×10	0	72	636	263	29	0	0	0	0	0

mem=3	1	2	3	4	5	6	7	8	9	10	11
3×3	98	178	344	340	28	8	4	0	0	0	0
4×4	15	76	266	428	134	60	21	0	0	0	0
5×5	1	19	115	408	234	145	71	7	0	0	0
6×6	0	0	22	282	272	222	164	27	11	0	0
7×7	0	0	5	116	293	282	220	55	17	10	1

Table 1. Distribution of $\overline{L}(M)$ for 1000 randomly generated matrices of various sizes. *left*: *Agent B*'s *mem* $= 1$. No entries are shown for values that we know to be impossible from Theorem 1. *right*: *mem* $= 3$. No values greater than 11 were found.

From these results we see that even in 3×3 matrices with *mem* $= 1$, it is not uncommon for *Agent A* to need to reason about the cost of various sequence lengths: In 44 of 1000 cases, there is at least one joint action from which *Agent A* is best off not jumping immediately to action a_2. In 104 of the cases, all optimal sequences are of length 1, which occurs exactly when b_2 is the best response to all of *A*'s actions: $\forall 0 \leq i < x, BR(a_i) = b_{y-1}$ (as expected, this occurrence becomes less common as the matrix size increases). In the other 852 cases, *Agent A* is best off switching to a_2 immediately, leading to longest sequences of length 2.

Though matrices such that $\overline{L}(M) > 2$ are not uncommon, it is also noticeable that matrices with optimal sequences of lengths close to the theoretical maximum do not occur naturally as the matrix size increases. A carefully selected construct such as $M2$ in Section 3 is required to find such sequences.

5 Related Work

Our work builds on existing research in game theory and in opponent modeling. Game theory [8] provides a theoretical foundation for multiagent interaction, and though originally intended as a model for human encounters (or those of human institutions or governments) has become much more broadly applied over the last several decades.

There is a vast research literature covering iterated play on normal form game matrices, the overall framework that we explore in this paper. Many of these papers have examined the specific questions of what, and how, agents can *learn* when repeatedly playing a matrix game; special emphasis has been given to developing learning algorithms that guarantee convergence to an equilibrium in self-play, or that converge to playing best response against another player that is using one of a fixed set of known strategies.

For example, Powers and Shoham [13] considered multiagent learning when an agent plays against bounded-memory opponents that can themselves adapt to the actions taken by the first agent. They presented an algorithm that achieved an ϵ-best response against that type of opponent, and guaranteed a minimum payoff against any opponent. A small selection of other research on multiagent learning includes Conitzer and Sandholm's work [5] on a learning algorithm that converges in self-play, Littman's multiagent reinforcement learning algorithm [9], and Chakraborty and Stone's [3] presentation of an algorithm that aims for optimality against any learning opponent that can be modeled as a memory-bounded adversary.

There are also a large number of articles in the economics and game theory literature on repeated matrix games, also often focused on issues related to reaching equilibria. Hart and Mas-Colell [7] presented an adaptive procedure that leads to a correlated equilibrium among agents playing a repeated game, while Neyman and Okada [11] considered two-player repeated games in which one agent, with a restricted set of strategies, plays against an unrestricted player (and considered the asymptotic behavior of the set of equilibrium payoffs).

A popular game theoretic model that may lead agents to converge to an equilibrium is that of fictitious play [1], in which agents play best response under the assumption that their opponents have a unchanging (though possibly mixed) strategy. At each step, each agent imagines that others will play as they have played up to this point, and responds according to the empirical frequency of those opponents' past play. Young [17, 18] explored a related concept called "adaptive play", which similarly models a dynamic process whereby agents, each employing bounded-memory best-response algorithms based upon a random sample of past plays of the game, may gradually move towards an equilibrium (the specific choice of equilibrium by a population of agents may be affected by small amounts of noise, which are part of the adaptive play model).

Much of the research above focused specifically on automated agent repeated play; similar questions have been taken up by researchers who have considered repeated play among humans. For example, a seminal paper by Nyarko and Schotter [12] investigated the beliefs that humans have as they repeatedly play a constant-sum two-person game; the authors elicited the players' beliefs during play, and factored those beliefs into the model of how players chose their moves.

All of the research mentioned above differs in fundamental ways from the work presented in this paper. First, our model assumes that the agents are cooperative; we are not considering general payoff matrices that model opponent rewards, nor zero sum games. Second, we are not examining the learning behavior of our agent (or agents), but rather are assuming that one agent is playing some variant on a best-response strategy, and its partner is fashioning its play accordingly, for their mutual benefit. This lack of symmetry between agents' algorithms distinguishes our model from that of, for example, the fictitious play model as well as Young's adaptive play model. In addition, we are exploring different aspects of the interaction than do those models.

More closely related to our current work is research by Claus and Boutilier [4] that, first of all, considers cooperative agents with identical payoffs, and then considers how (using reinforcement learning) these agents can converge to the maximal payoff. That research considers the *dynamics* of the convergence (e.g., speed of convergence), and the sliding average rewards that agents accrue as they explore their payoffs. What distinguishes our work is its emphasis on the path through matrix payoffs imposed by a reasoning *Agent A*, faced with a best-response *Agent B* as its partner. The process of movement through the matrix is deliberate and optimal, the path "searched-for," based on knowledge of partner behavior.

Indeed, the algorithms in this paper make an explicit assumption that the teammate observing the agent is playing a best-response policy to the observed actions of the agent. In doing so, the agent is actually planning its actions *intending* for them to be observed and interpreted. *Intended plan recognition* (in contrast to *keyhole recognition*) is the term used when the observed agent knows that it is being observed, and is acting under the constraints imposed by this knowledge [2].

Much of the work on planning for intended recognition settings has focused on natural language dialogue systems. Here, one agent plans its utterances or speech acts intending for them to be interpreted and understood in specific ways. Seminal work in this area was carried out by Sidner [14] and later Lochbaum [10], who have focused on collaborative dialogue settings. However, unlike our work, their focus is on the interpretation (the recognition), rather than on the planning of observed actions.

The SharedPlans framework for collaboration is concerned with choosing actions in collaborative settings [6]. However, while SharedPlans provides a logical framework which provides guidelines informing agent design, it does not provide detailed algorithms for specific cases, such as the cases covered in this paper.

Because our algorithm is—to a limited extent—reasoning about the teammate reasoning about itself, it is in fact engaged in a special case of *recursive modeling* [16]. Indeed, one question that remains open is what happens when the teammate is also trying to select actions that would cause the agent to shift policies. In this case, our agent would have to address 3-level recursive modeling.

6 Conclusion and Future Work

In this paper, we have introduced a novel game theoretic formulation of an important problem in multiagent teamwork. Specifically, we focus on the case in which an intelligent agent interacts repeatedly in a fully cooperative setting with a teammate that responds by selecting its best response to a fixed history of actions, possibly with some randomness. Based on its teammate's behavior, the intelligent agent can lead it to take a series of joint actions that is optimal for their joint long-term payoff.

The main contributions of this paper are a precise formulation of the problem (Section 2); an algorithm for finding optimal sequences of actions and a set of theoretical results regarding the maximal lengths of optimal action sequences (Section 3); and the results of some empirical results based on our fully-implemented algorithm (Section 4).

A few directions for future work have been mentioned throughout the paper. In particular, our proposed algorithm is exponential in the teammate's memory size, making solutions to interaction scenarios with more than a few possible actions per agent intractable. Heuristics enabling a streamlining of this algorithm would be very useful. Similarly, Conjecture 1 regarding the maximal possible value of $\overline{L}(M)$ is left open, as is the effect of ϵ on this bound.

One limiting assumption of the work presented in this paper is that *Agent A* knows *Agent B*'s action policy with certainty. Looking forward, this work sets the stage for developing strategies for interacting with teammates that have unknown values of *mem* and/or ϵ. In this more complex setting, it will be necessary to reason about the costs of action sequences as a function of teammate strategy in order to develop strategies that are robust to various possible teammate responses. Ultimately, we view this continuing work as a step towards the large, multi-faceted challenge of developing agents that are capable of interacting with a wide variety of possible teammates in *ad hoc team* settings.

Acknowledgments

Thanks to Michael Littman and Jeremy Stober for helpful comments and suggestions. This work is partially supported by grants from the Fulbright and Guggenheim Foundations, as well as Israel Science Foundation grant #898/05.

References

1. G. W. Brown. Iterative solutions of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.
2. S. Carrbery. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11:31–48, 2001.
3. Doran Chakraborty and Peter Stone. Online multiagent learning against memory bounded adversaries. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 211–226, September 2008.
4. Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, 1998.

5. Vincent Conitzer and Tuomas Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the 20th International Conference on Machine Learning*, pages 83–90, 2003.
6. Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.
7. Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, September 2000.
8. Kevin Leyton-Brown and Yoav Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2008.
9. Michael L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–28, 2001.
10. Karen E. Lochbaum. An algorithm for plan recognition in collaborative discourse. In *ACL*, pages 33–38, 1991.
11. Abraham Neyman and D. Okada. Two-person repeated games with finite automata. *International Journal of Game Theory*, 29:309–325, 2000.
12. Yaw Nyarko and Andrew Schotter. An experimental study of belief learning using elicited beliefs. *Econometrica*, 70(3):971–1005, 2002.
13. Rob Powers and Yoav Shoham. Learning against opponents with bounded memory. In *IJCAI'05*, pages 817–822, 2005.
14. Candace L. Sidner. Plan parsing for intended response recognition in discourse. *Computational intelligence*, 1(1), 1985.
15. Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.
16. José M. Vidal and Edmund H. Durfee. Recursive agent modeling using limited rationality. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 125–132. AAAI/MIT press, 1995.
17. H. Peyton Young. The evolution of conventions. *Econometrica*, 61:57–84, 1993.
18. H. Peyton Young. *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, Princeton, New Jersey, 1998.