Optimal Rewards in Multiagent Teams

Bingyao Liu
Automation Science & Engin.
Beihang University
Beijing, China
Email: bingliu@umich.edu

Satinder Singh
Computer Science & Engin.
University of Michigan
Ann Arbor, USA
Email: baveja@umich.edu

Richard L. Lewis
Psychology
University of Michigan
Ann Arbor, USA
Email: rickl@umich.edu

Shiyin Qin
Automation Science & Engin.
Beihang University
Beijing, China
Email: qsy@buaa.edu.cn

***Abstract*—Following work on designing optimal rewards for single agents, we define a multiagent optimal rewards problem (ORP) in common-payoff (or team) settings. This new problem solves for individual agent reward functions that guide agents to better overall team performance relative to teams in which all agents guide their behavior with the same given team-reward function. We present a multiagent architecture in which each agent learns good reward functions from experience using a gradient-based algorithm in addition to performing the usual task of planning good policies (except in this case with respect to the learned rather than the given reward function). Multiagency introduces the challenge of nonstationarity: because the agents learn simultaneously, each agent's learning problem is nonstationary and interdependent on the other agents. We demonstrate on two simple domains that the proposed architecture outperforms the conventional approach in which all the agents use the same given team-reward function (even when accounting for the resource overhead of the reward learning); that the learning algorithm performs stably despite the nonstationarity; and that learning individual reward functions can lead to better specialization of roles than is possible with shared reward, whether learned or given.**

I. Introduction

We consider the problem of designing reward functions for individual agents in multiagent sequential decision-making problems in the common-payoff, or *team*, setting. Throughout we will be focused on decentralized planning/learning approaches and will assume no direct communication among the agents. Furthermore, the shared reward function that defines the objective in the team problem is assumed known. The usual approach to such problems would, of course, make the given joint reward function the reward function for each agent. Why should a designer of multiagent systems do anything different? As we discuss below, in single agent settings it has been shown that designing reward functions can help mitigate agent limitations, e.g., help overcome computational limitations that prevent perfect planning. In this paper, directly following work [16], [17] on designing optimal rewards in single agent settings, we present an algorithm for learning individual reward functions in multiagent team problems and investigate three questions: 1) whether multiagent optimal rewards are capable of overcoming agent limitations that include individual agents not knowing exactly how other agents would behave; 2) whether our proposed multiagent optimal reward function learning algorithm is able to successfully handle the nonstationarity that comes from multiple agents learning simultaneously when their actions collectively impact the experience of all agents; and 3) whether in some domains learning individually-customized reward functions leads to more effective specialization of roles for the agents, and thus more effective coordination among the agents.

Background on Designing Rewards in Single-Agent Settings. The problem of designing rewards for single autonomous agents is typically studied in cases where a reward function is *unavailable* to begin with[1]. Nevertheless, multiple approaches have been developed for the seemingly counterintuitive problem of designing reward functions when already given a reward function. For example, reward-shaping of the variety developed by Ng et.al. [8] considered the question of what space of reward functions yields the same optimal policy as the given reward function with the hope that such a space contains an alternative reward function that makes the learning task faced by the agent easier. More recently, intrinsic rewards based on psychological motivations such as curiosity and exploration that are added to the given reward as bonuses have been shown to improve the performance of computational agents [10], [11], [13], [14], [20].

In this paper, we exploit the recent optimal rewards framework of Singh et.al. [16] that stems from the observation that reward functions play two distinct roles in autonomous agents: that of *evaluation* because cumulative expected reward determines a preference-ordering over policies, and that of *guidance* because approximate value functions computed/learned using the reward function determine the actual behavior of the autonomous agent. Separating these two distinct roles into two separate reward functions, with the given reward function used for evaluation, sets up the formal optimal rewards problem (ORP) whose solution is the reward function used for guidance (the ORP can also be seen to be an optimization approach to learning of intrinsic reward functions [16]). If the agent is limited in some form (constraints on computation

[1]For example, the field of preference elicitation develops methods for eliciting an approximate reward function from some human expert through queries (e.g., [4]), while the field of inverse reinforcement learning infers an approximate reward function from data observed by having a human expert teleoperate the autonomous agent (e.g., [5], [9]). Of course, other approaches to dealing with unknown reward functions in constructing autonomous agents eschew the problem of designing rewards altogether and instead learn agent-policies directly from observed expert behavior using some form of imitation learning (e.g., [6]) or learning by demonstration (e.g., [1]) or supervised learning (e.g., [19]).

or knowledge), then solving the ORP for a guidance-reward function can improve the agent's performance relative to using the evaluation-reward function for guidance [17]; in other words, optimal rewards can mitigate agent limitations. In addition, algorithms and architectures to solve the ORP have demonstrated empirical success [18] and we will build upon those in this paper.

Our departure from prior work on ORP is in extending it to multiagent teams. Our contributions include: 1) defining the optimal rewards problem in the multiagent team setting; 2) extending a single-agent gradient-based approach to learning optimal rewards from experience to multiagent teams; 3) empirically demonstrating that despite the non-stationarity introduced by the multiagency, learning optimal rewards can mitigate agent limitations; and 4) learning optimal rewards in teams can lead to more effective specialization in that each agent is able to learn its own guidance-reward whilst sharing the team's evaluation-reward.

II. Multiagent Optimal Rewards Problem

Let the set of agents be $\{\mathcal{A}^i\}_{i=1}^n$; our mathematical formulation is for arbitrary n though our empirical evaluation will be restricted to $n = 2$. At time step t agent $\mathcal{A}^i$ gets an observation $o_t^i \in O^i$ from the environment M and takes an action $a_t^i \in A^i$. Agent i's history at time t is $h_t^i = o_1^i a_1^i \cdots o_{t-1}^i a_{t-1}^i o_t^i$. The joint observation, action and history at time t are denoted o_t, a_t and h_t respectively. The join action a_t causes a stochastic change in the underlying state of the environment which in turn influences the next joint observation o_{t+1}. The given joint reward function, hereafter the *objective reward function*, is denoted $R^{\mathcal{O}}$ and will be used to evaluate joint histories, i.e., the objective utility achieved by the agents after history h is $U^{\mathcal{O}}(h) = \frac{1}{|h|}\sum_{t=1}^{|h|} R^{\mathcal{O}}(h_t)$. In the conventional formulation the reward to each agent at history h would be $R^{\mathcal{O}}(h)$. Here we allow an individual *guidance reward function* R^i for agent $\mathcal{A}^i$. Thus the guidance utility for agent i with joint history h is $U^i(h) = \frac{1}{|h|}\sum_{t=1}^{|h|} R^i(h_t)$. Of course, in practice reward functions are defined as mappings from some abstraction of history to scalar rewards but for complete generality we present our formulation with no abstractions. In single-agent settings, the agent's guidance reward function determines its behavior. In multiagent settings, the joint setting of the multiple individual guidance reward functions will collectively determine the team's behavior. Any coordination among the agents has to be the result of independent learning/planning guided by the individual reward functions.

Agent $\mathcal{A}^i$ using guidance reward function R^i is denoted $\mathcal{A}^{R^i}$ for ease of notation.

Definition of multiagent optimal rewards problem. Given a set of agents $\{\mathcal{A}^i\}_{i=1}^n$, a search space of reward functions $\{\mathcal{R}^i\}_{i=1}^n$, an environment M, and the objective reward function $R^{\mathcal{O}}$, the set of reward functions $\{R^{i,*}\}_{i=1}^n$ is jointly optimal for the team of agents, if

$$\{R^{i,*}\}_{i=1}^n = \underset{\{R^i\}_{i=1}^n \in \{\mathcal{R}^i\}_{i=1}^n}{\arg\max} \mathbb{E}_{h\sim\langle\{\mathcal{A}^{R^i}\}_{i=1}^n, M\rangle}\left[U^{\mathcal{O}}(h)\right], \quad (1)$$

where $h \sim \langle\{\mathcal{A}^{R^i}\}_{i=1}^n, M\rangle$ is a random history sampled by the interaction of the set of agents $\{\mathcal{A}^{R^i}\}_{i=1}^n$ in M.

In words, the optimal reward functions for the team of agents are the choice of individual reward functions that guide the team of agents to joint-behavior that in expectation maximizes the objective utility as measured by the given joint reward function. This paper is about approaches to solving this new multiagent ORP.

Multiagent optimal rewards versus other approaches to multiagent rewards. Crucially, the optimal set of guidance reward functions are optimal not just with respect to the environment and given objective reward function, but also with respect to the details of the architectures and algorithms of the various agents. Specifically, the agents and their limitations (including those from the multiagency) help determine the distribution over h which in turn determines the effectiveness of choices of guidance reward functions. This sets the multiagent optimal rewards problem and its solutions apart from other approaches to designing rewards that also frame the problem of finding good rewards or incentives in multiagent settings as an optimization problem, e.g., from the collective intelligence approach [21] that ignores the limitations of the agents and focuses on dealing with strategic settings, and from approaches based on mechanism design [12] that typically assume the availability of a central auctioneer and more importantly assume that the agents know and can communicate their correct utility function to the central auctioneer. This paper is also different from a recent and related approach to learning social awareness via intrinsic rewards for multiagent systems [15] because it demonstrates weak-mitigation, while our emphasis is on the greater challenge of strong-mitigation (see Section III for this important distinction).

The potential for specialization. By definition, in team problems the agents have to learn to cooperate and coordinate to achieve high shared reward. In some problems such coordination requires different members of the team to adopt different roles, i.e., to specialize. Thus one of the questions of interest is whether the ability to learn different guidance reward functions by the different agents can lead to increased specialization of roles, and thereby an improvement in the coordination among the team. We address this question empirically in Section V-B.

III. Multiagent Strong-Mitigation Architecture

As in the single-agent setting, it is straightforward (see Equation 1) that as long as the search space of reward functions for each agent contains the objective reward function the objective utility obtained by using the optimal guidance reward functions would be at least as high as that obtained by the conventional use of the objective reward function as the guidance reward function for all agents. This was called the *weak-mitigation* property [3] in single-agent settings and it holds by construction for the multiagent setting as well. Note that the existence of guidance reward functions better than the objective reward function in the search space, which is all weak-mitigation demands, does not mean that such reward functions can be found cheaply or at all. A more interesting result would be to show that it is worth it for an agent to devote some of its limited computational resources to learning a good guidance reward function at the cost of having fewer computational resources to plan good behavior with respect to the guidance reward function. Such a favorable property was called *strong-mitigation* [3] in single agent settings and it accounts for the cost of finding good reward functions, whilst weak-mitigation ignores it.

Bratman et.al. [3] showed that strong-mitigation implicitly suggests a **N**ested **O**ptimal **R**eward and **C**ontrol (NORC) architecture in which an overall agent has two components, an *actor-agent* which is the usual agent that takes actions so as to optimize reward, and a novel *critic-agent* that learns guidance reward functions. They demonstrated strong-mitigation in a setting where the single-agent repeatedly plans from the current state to select a current action, and where the computational constraint is the CPU-time available per action-decision. For different constraints on the amount of CPU-time per decision available, they compared the performance of two agents: 1) an NORC agent that splits the available CPU-time between the critic-agent updating the guidance reward function and the actor-agent doing limited planning with respect to the guidance reward function learned by the critic-agent, and 2) a conventional agent that spends all available CPU-time per decision on planning more deeply. They showed that with limited CPU-time per decision, the NORC agent does better in terms of objective utility than the conventional agent. In this paper, we will follow the same empirical approach by considering limited CPU-time per decision constraints and comparing the performance of our multiagent NORC architecture (described next) and a conventional multiagent architecture.

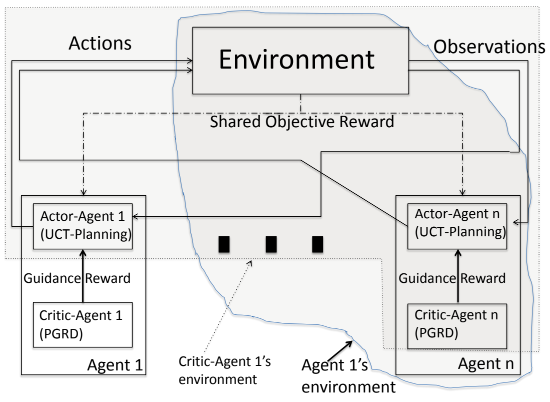


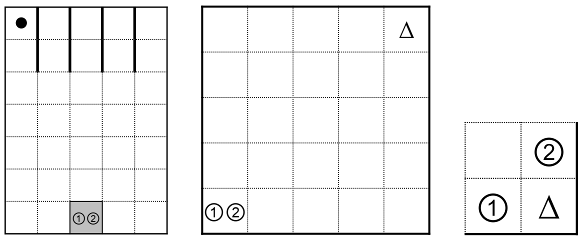
Fig. 1. The MultiAgent Nested Optimal Reward and Control (MNORC) Architecture. Each agent is composed of a conventional actor-agent and a novel critic-agent. The critic-agent learns a guidance reward function that guides the associated actor-agent's behavior. The joint action of all the agents influences the shared objective reward received by all the agents. Note the large irregular shape showing that the actor-agent-1's environment includes the external environment as well as all the other agents, and the wider and more-rectangular shape showing that the critic-agent-1's environment includes its own actor-agent-1 as well as the external environment and the other agents.

Multiagent NORC Architecture. Figure 1 illustrates our multiagent NORC architecture. Each agent has within it an actor-agent and a critic-agent. In our empirical work the actor-agent is a UCT-based planner [7] and the critic-agent uses the **P**olicy **G**radient for **R**eward **D**esign or PGRD algorithm of Sorg et.al. ([18]; we describe this algorithm briefly below). The significant departure from the single-agent setting is that while each agent gets its own observation and produces its own action, it is the joint action of the agents that determines the shared objective reward.

Another way of understanding the additional challenge posed by multiagency is to consider the effective environment of a particular actor-agent. As shown in Figure 1, the environment of actor-agent-1 includes not only the external environment but also all the other agents including their critic-agents. Even if all the other actor-agents are fixed planners, the fact that the critic-agents are learners and thus nonstationary make each actor-agent's environment nonstationary. Figure 1 also shows a critic-agent's environment, and again because of multiagency it includes other learning components. The big open questions are whether multiple NORC agents simultaneously learning guidance reward functions and acting in the world will converge reliably and stably to good guidance functions for each agent, and whether multiagent strong-mitigation will be achieved, i.e., whether given CPU-time per decision constraints the multiagent NORC architecture will achieve more expected objective utility than the conventional multiagent architecture. Before we turn to our empirical results that are focused on these questions, we describe the actor-agent and critic-agent briefly.

Actor-Agents and the search for good policies. We assume that the actor-agents have a perfect model of the dynamics of the environment and use UCT to plan. UCT has two parameters T and D; it builds a search tree by simulating (from the current history) T trajectories of depth D where action-choices in the trajectory generation are treated as a bandit problem solved by the UCB1 algorithm [2]. The guidance reward function is used to compute from the resulting search-tree an estimated long-term utility for each action at the root node of the tree, and finally the best-estimated-utility action at the root node is selected. This process is repeated afresh at each time step for the current history as root node. The CPU-time per decision is heavily dependent on the two parameters T and D.

Because this is a distributed multiagent setting and the agents can not communicate with each other, there is an interesting challenge, however. When it builds a UCT tree, each agent must decide what actions the other agents would do at each time step. A further complication is that if there is any learning by the agents, as will be the case if the guidance reward functions are learned through experience, then the behavior of the other agents will be nonstationary and thus difficult to predict. In all our experiments, agents learn the other agents stochastic policy using the empirical probabilities observed in historical data, and use this learned behavior model to sample the other agents actions during the tree-building process in UCT.



(a) Food-Shelter (b) Pursuit (c) Capture

Fig. 2. Circled numbers indicate different agents. a) In Food-Shelter, thick black lines indicate impassable walls, the large solid circle represents one possible location for the consumable food, and the shaded grid square in the middle of the bottom row is the shelter for the agents. b) In Pursuit, the triangle represents the prey. c) In Pursuit, agents capture prey by pinning it in a corner; illustrated for the lower right corner.

Critic-Agents and the search for good guidance rewards. We assume that the search space of reward functions for each agent is defined as a continuously-parameterized function of reward-features (specific examples are in the empirical results below). The PGRD algorithm views the actor-agent as a procedure for translating the continuous parameters of the reward search space to a policy via planning. For certain classes of actor-agents, including the UCT agents used here, PGRD adapts standard policy-gradient approaches to use experience to update the reward space parameters in the direction of the gradient of the expected objective utility. We refer the reader to Sorg et al. [18] for the details of PGRD. In our multiagent NORC architecture each critic-agent implements the PGRD algorithm for the UCT-planning actor-agents. The main challenge is the fact that multiple critic-agents are learning guidance reward functions simultaneously and the resulting additional nonstationarity may lead to interference across the agents and prevent convergence or create additional low-quality local minima in reward space.

IV. Experiment Objectives and Structure

We now describe a set of experiments in which we field two multiagent NORC architecture in two different environments, one which clearly has intuitively separate roles for the two agents (called "Food-Shelter"), and one which demands cooperation between the two agents but does not have intuitively separate roles (called "Pursuit"). We provide descriptions of the environments below, but first lay out what we intend to demonstrate with these experiments via a set of key comparisons among different agent types.

A. Overall Objectives and Key Comparisons

There are three main objectives:

1) Demonstrate *strong-mitigation* in team problems using our multiagent NORC architecture.
2) Demonstrate that learning individual guidance reward functions affords *better specialization* than using objective reward functions for shared guidance.
3) Demonstrate that independent PGRD learning by the critic-agents can lead to *relatively stable and effective learning* of guidance reward functions.

These demonstrations take the form of comparisons of the performance of three different multiagent systems (MASs) realized as multiagent NORC architectures that vary in the critic-agent:

- *UCT-ObjRe.* In this MAS, both UCT-based actor agents use the objective reward function as their guidance reward functions; i.e. the critic agent supplies a single stationary guidance reward that is fixed to be the objective reward.
- *UCT-PGRD.* In this MAS, the actor-agents plan with guidance reward functions learned independently by the critic-agents via PGRD using gradients of the objective utility with respect to the coefficients on the reward-features[2]. We describe the search space of reward functions below.
- *UCT-PGRDSame.* In this MAS, both agents use PGRD to update guidance reward functions, but are constrained to use the same learned guidance reward function. We implemented this constraint by averaging the gradients computed independently for each agent and updating each reward function with the same average gradient.

To demonstrate strong mitigation (Objective 1), we must show that *UCT-PGRD* outperforms *UCT-ObjRe* with equivalent resource consumption. To demonstrate better specialization via learned individual guidance rewards (Objective 2), we must show that *UCT-PGRD* outperforms *UCT-PGRDSame* (the MAS constrained to learn a single guidance reward for both agents) and does so via increased specialization of reward and behavior. To demonstrate stable and effective learning of guidance reward functions (Objective 3), we must show that *UCT-PGRD* both outperforms *UCT-ObjRe* and does so while settling on stable guidance rewards despite the non-stationarity of the learning problem.

B. Two Environments: Food-Shelter and Pursuit

Food-Shelter. The environment has two clear and separate roles for the two agents: one should gather food and the other should keep the shelter repaired. As illustrated in Figure 2(a), it is a 7×5 grid world with a shelter for the agents at the bottom and 5 corridors separated by impassable walls at the top. The agents objective utility is the expected objective reward per time step over a lifetime of 5000 time steps.

Dynamics. The shelter breaks down (perhaps by invaders) with probability 0.3 at each time step; once broken it remains so until repaired. Food appears at the top of one of the corridors; when food is gathered new food reappears at the top of a different (randomly chosen) corridor. The agents have four deterministic actions corresponding to each cardinal direction; if the resulting direction is blocked by a wall or the boundary, the action results in no movement. The agents have a *gather* action when at a food location and a *repair* action when at the shelter location that repairs a broken shelter immediately.

[2]Throughout all the experiments, we used the following PGRD parameters without tuning, discount rate $\gamma = 0.95$, temperature $\tau = 100$, learning rate of $\alpha = 1e - 4$, decaying-average parameter $\beta = 0.9$, and regularization parameter $\lambda = 0$.

The agents act simultaneously at each time step. The starting location of both agents is at the shelter.

Objective reward. Gathering food contributes a joint objective reward of 1.0 while if the shelter is broken it contributes an objective reward of -1.0; the objective reward is 0 else. Thus the task for the two agents is to gather as much food as possible while keeping the shelter repaired; ideally one agent should focus on keeping the shelter repaired and the other agent should focus on gathering food.

Observations and prior-knowledge. Both agents observe the full state of the environment, i.e., observe the location of both agents, the location of the food, and whether the shelter is broken or not. The agents have a perfect model of the dynamics of the environment. They don't know how the other agent will act, of course, but they can observe how the other agent has acted in the past.

Pursuit. This environment also demands cooperation between two agents to catch a prey but does not have intuitively separate roles for the two agents. As shown in Figure 2(b), it is a 5×5 discrete, square grid-like world in which two agents (circles) need to coordinate to capture a prey (triangle). The prey's movements are fixed and not learned (see details below). The agents capture prey by pinning it any one of the 4 corners, as illustrated in Figure 2(c). The agents objective utility is the expected objective reward per time step over a lifetime of $10,000$ time steps.

Dynamics. Both agents and the prey have four deterministic actions corresponding to each cardinal direction; if the resulting direction is blocked by the boundary, the action results in no movement. At each time step, the two agents move simultaneously while the prey observes the agents new locations before moving. The prey moves so as to maximize distance to the closest agent[3]. The initial positions of the agents and prey are at opposite corners as shown in Figure 2(b); the predators and prey are reset to this position once the prey is captured.

Objective reward. The objective reward function $R^{\mathcal{O}}$ provides a reward of 1.0 when the prey is captured and a reward of 0 otherwise.

Observations and prior-knowledge. Both agents observe the full state, i.e., the locations of the two agents and the prey. The actor-agents have a perfect model of environment dynamics. Each agent can observe the other agent's past actions.

C. Search Space of Guidance Reward Functions

For both domains and both UCT-PGRD and UCT-PGRDSame, we define a reward function search space as linear combinations of two features of history, the objective reward and an inverse-recency feature. The inverse-recency feature for agent $\mathcal{A}^i$ in history h^i, is defined as $\phi^i_{\text{inv.rec}}(h^i) = 1 - \frac{1}{c(h^i)}$, where $c(h^i)$ is the number of time steps since the agent $\mathcal{A}^i$ was in the same location as the location at the end of history h^i. Low values of this feature (close to 0) indicate that the agent is visiting a location it visited recently while large values of this feature (close to 1) indicate that the agent is visiting a location it has not visited recently. Formally, guidance reward as a function of history h are of the form,

$$R^i(h) = R^{\mathcal{O}}(h) + \theta^i_{\text{inv.rec}}\phi^i_{\text{inv.rec}}(h^i) \tag{2}$$

where $\theta^i_{\text{inv.rec}}$ is the one continuous scalar parameter and its range provides the search space of reward functions[4]. Note that a positive value of $\theta^i_{\text{inv.rec}}$ will mean that the agent should want to visit locations not visited recently while a negative value of $\theta^i_{\text{inv.rec}}$ will mean that the agent should avoid locations not visited recently. This form of reward function space was used in previous work with single-agent ORP [3], [18] and was found to be effective in overcoming limited planning by encouraging a more systematic approach to exploration. We use this same reward function space because it is not task-specific and because seeking or avoiding exploration is an interesting abstract form of specialization that might manifest itself in interesting ways (and we will see such a phenomenon in Food-Shelter).

Equation 2 defines a one dimensional continuous search space for PGRD in each critic-agent in the UCT-PGRD MAS for a 2-dimensional joint search space. Recall that for UCT-PGRDSame the two critic-agents are required to use the same reward function and thus the joint search space is one dimensional. In both MASs, for $i = 1, 2$, the initial value of $\theta^i_{\text{inv.rec}} = 0$, i.e., the critic-agents start with the objective reward function as the guidance reward function.

V. Experiment Results

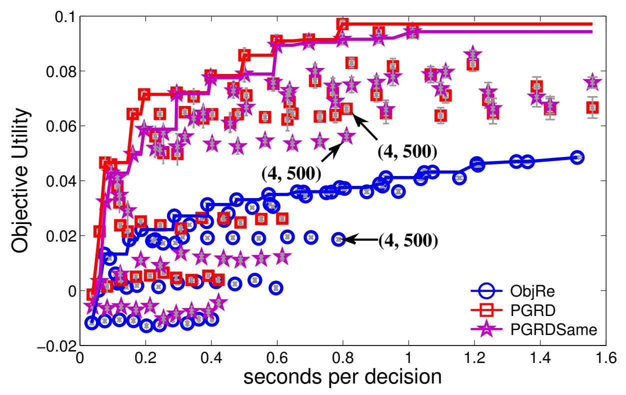
The results below are organized around the three main empirical objectives set out above and use the following methodology. For demonstrating strong mitigation we focus on computational time as the resource of interest. But instead of imposing a priori CPU-time-per-decision constraints, for both domains we evaluate all three MASs with UCT planning depths $D \in \{2, 3, 4, 5, 6, 7, 8\}$ and number of Monte Carlo (MC) trajectories $T \in \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$; together these parameters determine in large part the CPU-time per decision. The aspects of MAS performance collected will include CPU-time per decision, average objective utility, the guidance reward functions learned, and the behavior of the agents. In particular, this means that we present the results in a way that does not commit to a particular combination of D and T as being best for some given time-per-decision limit.

A. Demonstrations of Strong Mitigation

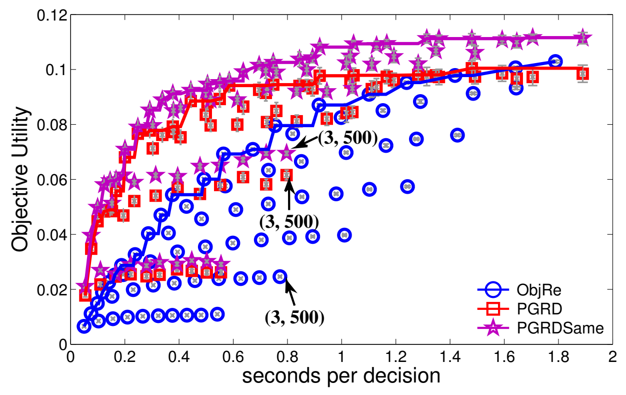
Figure 3(a) shows the per-time-step objective utility in Food-Shelter over $5,000$ time-steps plotted against CPU-time per decision for the three different MASs; the results are averages over a 100 trials and the error-bars are shown. The points in the

[3]Formally, the prey moves according to the action: $a = \arg\max_{a \in A}\{\min[dist(l_1, \Delta'), dist(l_2, \Delta')]\} \cap \arg\max_{a \in A}\{\text{sum}[dist(l_1, \Delta'), dist(l_2, \Delta')]\}$, where l_1, l_2 represent the agents' locations and Δ' is the prey's next location after taking its action, and $dist$ measures the Manhattan distance between the agents and prey.

[4]Note that the inverse-recency reward function feature is only a function of h^i and not h and so does not require any communication among the agents. The objective reward is already assumed to be shared among the agents.



(a) Food-Shelter: Envelope View



(b) Pursuit: Envelope View

Fig. 3. The points in each of the two figures correspond to different architectures (shown as circles for ObjRe, squares for PGRD, and stars for PGRDSame; different points of the same shape correspond to different UCT parameters (depth, trajectories). See for example the three points labeled with $(4, 500)$ in panel (a) that correspond to PGRD (square), PGRDSame (star), and ObjRe (circle). The x-value and y-value of each point is the CPU time resource consumption and the average objective utility respectively for the corresponding architecture and UCT parameters. For example, the leftmost points for each MAS is $(2, 50)$ and rightmost points for each MAS is $(8, 500)$ (because these take the smallest and largest CPU-time per decision respectively). Each point also has error-bars shown though in most cases the error-bars are too small to be visible. The three curves in each figure correspond to the upper envelopes of performance for the three MAS architectures. See text for further discussion.

figure correspond to different architectures (shown as circles for ObjRe, squares for PGRD, and stars for PGRDSame); different points of the same shape correspond to different UCT parameters (depth, trajectories). For all constraints on CPU-time per decision, using PGRD to learn reward functions is better than simply using the given objective reward. This is seen most clearly via the upper envelope curves in which UCT-PGRD that learns separate guidance reward functions for both agents does the best, UCT-PGRDSame with the constraint that both agents use the same learned guidance reward functions is slightly worse, and finally UCT-ObjRe that employs the objective reward function for guidance is far worse.

This is clear evidence of strong mitigation in this domain. Consider any point on the x-axis—for example 0.8 seconds per decision—and if that were to be the computational bound on the MAS, spending some of those resources on learning guidance reward functions is far better than spending all those resources on action-planning. This improvement is the vertical gap between the corresponding upper-envelope curves at the vertical line $x = 0.8$. In other words, using PGRD in the critic-agent to learn guidance reward functions helps overcome the bounded depth/trajectories in UCT planning as well as the limited knowledge of the other agent's actions.

Figure 3(b) shows the results for Pursuit in the same format as above. As was the case for the Food-Shelter, for all constraints on time per decision, using PGRD to learn guidance reward functions is better than simply using the given objective reward. The main difference from the Food-Shelter results is that the performance of the UCT-PGRDSame MAS is better than the performance of the UCT-PGRD MAS, though as for Food-Shelter both PGRD-based MASs are significantly better than the UCT-ObjRe MAS. Strong-mitigation is observed again in that for all constraints on the available seconds per decision, i.e., for all points on the x-axis, the vertical gap between the upper-envelope curves for UCT-PGRDSame and UCT-ObjRe is the performance advantage of spending some of the fixed computational resource on updating the guidance reward function.

B. *Demonstrations of Specialization*

As expected intuitively from the specifics of the two domains, we did not find specialization in Pursuit and so focus here primarily on analyzing the results of Food-Shelter. Our first positive result concerning specialization is in the dominance of UCT-PGRD over UCT-PGRDSame in Food-Shelter as described above (see Figure 3(a)): learning individual guidance rewards does produce greater to slightly greater performance, depending on the CPU-time per decision constraint, than learning a shared reward. But is this dominance in performance associated with clearly specialized *behavior* and clearly specialized *individual rewards?* We address these two questions here.

The expression of specialization in behavior. By design, in order to achieve high objective utility in Food-Shelter, one agent needs to focus on shelter-repairing work while the other focuses on food-gathering work. We defined a measure of this specialization as follows. For each agent we compute the following fraction: the number of times it gathers food divided by the number of times it gathers food and repairs shelter. For each run, the absolute value of the difference between this fraction for the two agents is a measure of the specialization for that run. This difference in specialization ratio is between 0 and 1. The more specialized the agents are the closer the specialization ratio will be to one, and the less specialized they are the closer the specialization ratio will be to zero. Figure 4 shows the average amount of specialization as a function of CPU-time-per-decision in curves that correspond to different number of trajectories (the UCT parameter). The points along the curve from left to right correspond to increasing depth. It is clear that the UCT-PGRD MAS specializes far more than the UCT-ObjRe MAS. Finally, observe that in each pair of

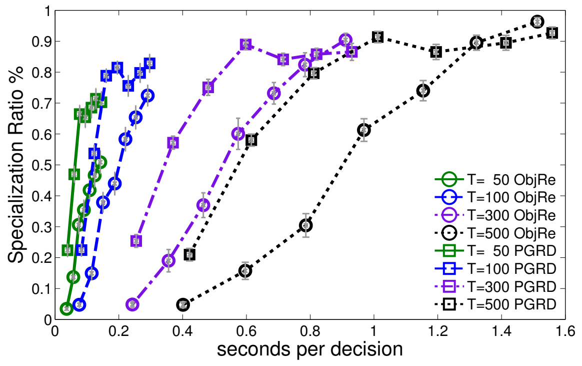


Fig. 4. Specialization Ratio Curves for Food-Shelter. Each point is for a distinct agent, i.e., a MAS (circles for ObjRe and squares for PGRD) using particular UCT parameters. The x and y values of each point are the seconds per decision and the specialization ratio respectively for the corresponding agent. Curves connect points for agents using the same MAS and the same number of trajectories in UCT. The main result is that PGRD agents achieve higher specialization ratios relative to ObjRe agents when both are constrained to similar CPU-time per decision. See text for further discussion.

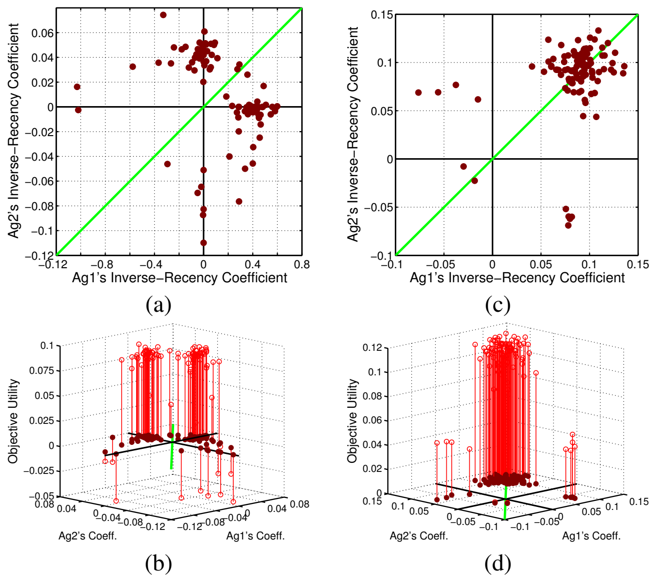


Fig. 5. The two panels (a) and (b) correspond to Food-Shelter with UCT-PGRD MAS $(D = 8, T = 500)$ agent. Each of 100 data points in panel (a) corresponds to a separate run and plots the coefficient for the inverse-recency reward feature for one agent against the same coefficient for the other agent. Panel (b) shows the objective utility along the vertical-axis for each of the same 100 runs. Panels (c) and (d) are similarly plotted results for Pursuit (with UCT-PGRD MAS $(D = 5, T = 500)$ agent). The striking specialization of reward functions in Food-Shelter is apparent from most of the points being off-diagonal in panel (a). This is in contrast to Pursuit in which the lack of reward function specialization is apparent in most of the points being close to the diagonal in panel (c). See text for further discussion.

curves corresponding to the same number of trajectories, the rightmost points that correspond to the highest depths show that as the consequential computational limitation that small depth imposes on UCT is removed from the agents, the UCT-ObjRe MAS specializes just as well or even better than the UCT-PGRD MAS. This is because learning guidance reward functions can really only help in bounded agents and at the largest depth the agents have no planning bounds.

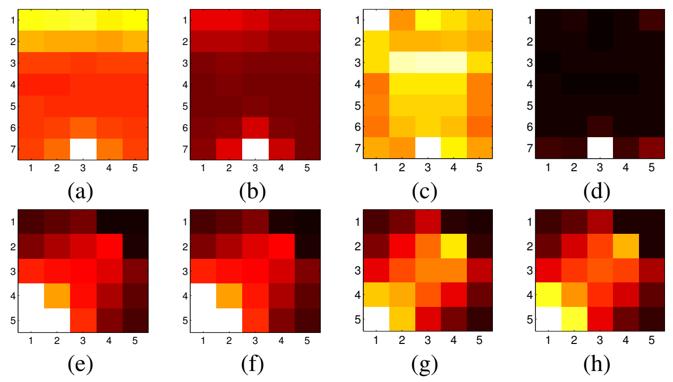


Fig. 6. The heat-maps above show the number of time steps an agent spends in each location on average over a 100 runs; the lighter the color in a grid-square the more the time spent in that location. The top row is for Food-Shelter with UCT-parameters $D = 3$, $T = 500$ and the panels correspond to: (a) UCT-ObjRe Food Agent. (b) UCT-ObjRe Shelter Agent (c) UCT-PGRD Food Agent. (d) UCT-PGRD Shelter Agent. In the bottom row for Pursuit, the UCT-parameters are $D = 5, T = 500$, and the panels are: (e) UCT-ObjRe Agent 1, (f) UCT-ObjRe Agent 2, (g) UCT-PGRD larger coefficient for inverse-recency reward feature, and (h) UCT-PGRD smaller coefficient for inverse-recency reward feature. The top-row provides evidence for greater specialization in behavior for Food-Shelter relative to the undifferentiated heat-maps in the lower-row for Pursuit. See text for further discussion.

The expression of specialization in the learned reward functions. What form did the learned guidance reward functions actually take in the two domains? Figures 5(a-b) present the results for the UCT-PGRD MAS (with $D = 8$ and $T = 500$) on Food-Shelter. Each point in panel (a) is for a separate run; it plots the learned coefficients on the inverse-recency reward feature for the two agents. As is visually clear, one agent tends to learn a near zero coefficient while the other agent tends to learn a slightly positive coefficient. Recall that a positive coefficient for the inverse-recency reward features encourages the agent to explore, to visit places it hasn't visited recently, and this helps this agent become better at finding food which moves from place to place. The agent with a near-zero or slightly negative coefficient for the inverse-recency reward feature is discouraged from exploring and this helps it stay put at the shelter location. Thus, it is clear that the critic-agents for the two agents learn quite different guidance reward functions, i.e., there is specialization in the guidance reward functions. Panel (b) shows the objective utility on the vertical-axis for each point plotted in panel (a); it is clear that the runs that don't end up with one agent having a near-zero coefficient and the other agent a positive coefficient do worse in terms of objective utility. Figure 6(a-d) show that there is specialization also in behavior as a result. Each panel shows the number of time steps spent in each location of the grid where lighter color means more time spent (a kind of heat-map). Panels (a) and (b) are for the UCT-ObjRe MAS and correspond respectively to the agent that gathers food more often and the agent that repairs the shelter more often. Similarly, panels (c) and (d) are for the UCT-PGRD MAS food and shelter agents respectively. From visual inspection it is clear that there is increased behavioral specialization in the

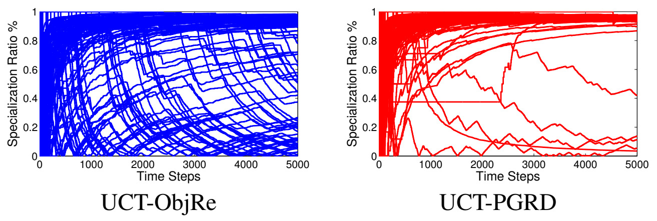


Fig. 7. Stability of learning guidance reward functions. The graphs show the specialization ratio as a function of time for 100 independent runs for UCT-ObjRe and UCT-PGRD MASs (with $D = 5, T = 500$).

MAS that learns guidance reward functions; the shelter agent concentrates more heavily on the shelter location in the UCT-PGRD MAS compared to the shelter agent in UCT-ObjRe MAS.

Figures 5(c-d) plot the coefficients for the inverse-recency reward features for the two agents in UCT-PGRD MAS for Pursuit. Visually it is clear that for most of the 100 runs, agents learn roughly the same coefficients. The lower panel (d) shows that the runs that don't end up with the both agents having similar positive coefficients do worse in terms of objective utility. There is little specialization of the guidance reward functions in Pursuit. This lack of behavioral specialization is apparent as well from panels (e-h) in Figure 6 in which both UCT-ObjRe agent's heat-maps (panels e & f) look similar to each other as well as similar to the heat-maps for the two agents in the UCT-PGRD MAS (panels g & h).

C. Demonstration of Stability in the Face of Non-stationarity

Figure 7 illustrates how adapting the guidance reward functions can lead to more stable agent behavior. The curves in the two panels show the specialization ratio as a function of time for 100 different independent runs in Food-Shelter. The left & right panels are for the UCT-ObjRe and UCT-PGRD respectively. As is visually apparent for a significantly greater fraction of runs in UCT-PGRD the specialization ratio converges to nearly 1. This is evidence of behavioral stability in that relative to UCT-ObjRe in UCT-PGRD it is more often the case across runs that one agent mostly focuses on food whilst the other focuses mostly on shelter.

VI. Conclusion

In this paper, we defined a new multiagent optimal rewards problem whose solution defines the guidance reward for each agent in a team such that the resulting behavior of the team ends up maximizing the objective utility of the team as measured by a given shared objective reward function. We provided a multiagent nested optimal reward and control architecture and empirically demonstrated on two domains that given limited CPU-time per decision, it is better to spend some of the limited computational resources in learning good guidance reward functions instead of spending them all on planning good actions. This is the first evidence of strong-mitigation in multiagent settings. We also showed that our specific adaptation to multiagents of existing single-agent methods for learning guidance reward functions and for planning did reliably and stably learn good guidance reward functions. In Food-Shelter, both guidance reward function and increased behavioral specialization were seen in our results. More theoretical analysis and extensive empirical results await future work.

Acknowledgment

This work was supported by NSF grants IIS-0905146 and IIS-1148668. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Machine Learning: Proceedings of the fourteenth international conference*, pages 12–20, 1997.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] J. Bratman, S. Singh, R.L. Lewis, and J. Sorg. Strong mitigation: Nesting search for good policies within search for good reward. In *11th International Conference on Autonomous Agents and Multiagent Systems*, 2012.

[4] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *In Proceedings of the National Conference on Artificial Intelligence*, 2000.

[5] A. Coates, P. Abbeel, and A.Y. Ng. Apprenticeship learning for helicopter control. *Communications of the ACM*, 52(7):97–105, 2009.

[6] J. Kober and J. Peters. Imitation and reinforcement learning. *IEEE Robotics and Automation Magazine*, 17(2):55–62, 2010.

[7] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning*, 2006.

[8] A.Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*, 1999.

[9] A.Y. Ng and S.J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2000.

[10] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 2007.

[11] P.-Y. Oudeyer, F. Kaplan, and V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11, 2007.

[12] D.C. Parkes and Satinder Singh. An MDP-Based approach to Online Mechanism Design. In *Proc. 17th Annual Conf. on Neural Information Processing Systems*, 2003.

[13] J. Schmidhuber. Curious model-building control systems. In *Proceedings fo the International Joint Conference on Neural Networks*, 1991.

[14] J. Schmidhuber. Artificial curiosity based on discovering novel algorithmic predictability through coevolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 1999.

[15] P. Sequeira, F.S. Melo, R. Prada, and A. Paiva. Emerging social awareness: Exploring intrinsic motivation in multiagent learning. In *Proceedings of the IEEE International Conference on Developmental Learning*. 2011.

[16] S. Singh, R.L. Lewis, A.G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2010.

[17] J. Sorg, S. Singh, and R.L. Lewis. Internal rewards mitigate agent boundedness. In *Proceedings of the International Conference on Machine Learning*, 2010.

[18] J. Sorg, S. Singh, and R.L. Lewis. Optimal rewards versus leaf-evaluation heuristics in planning agents. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[19] G. Tesauro and T.J. Sejnowski. A 'neural' network that learns to play backgammon. In *Neural Information Processing Systems*, 1987.

[20] E. Uchibe and K. Doya. Constrained reinforcement learning from intrinsic and extrinsic rewards. In *Proceedings of the IEEE International Conference on Developmental Learning*. London, UK, 2007.

[21] D. H. Wolpert and K. Tumer. Collective intelligence. In *Fourth Workshop on Economics with Heterogeneous Interacting Agents*, 1999.