Exploiting Structure and Agent-Centric Rewards to Promote Coordination in Large Multiagent Systems

Chris HomesParker, Parflux LLC, chris@parflux.com
Matthew E. Taylor, Washington State University, taylorm@eecs.wsu.edu
Yusen Zhan, Washington State University, yzhan@eecs.wsu.edu
Kagan Tumer, Oregon State University, kagan.tumer@oregonstate.edu

ABSTRACT

When scaling systems to hundreds or thousands of agents, the ability of agents to observe their environment and to coordinate during decision making becomes increasingly difficult. This increased complexity frequently results in significantly reduced system performance as the number of agents in the system increases. We address this by introducing the *Factored action Reinforcement learning Agents in Cooperative tasKs Exploiting Difference rewards* (FRACKED) algorithm, which conglomerates three existing multiagent techniques to significantly improve coordination and scalability within large multiagent systems. In particular, we combine i) a Factored-Action Markov Decision Processes (FA-FMDP) framework which exploits problem structure to reduce observation and coordination requirements; ii) reinforcement learning which enables agents to learn from experience, and iii) agent-centric reward shaping (i.e., difference rewards) which provide an implicit coordination mechanism for agents during learning. We show that the FRACKED algorithm outperforms reinforcement learning with global rewards and a standard learning automata method in two domains containing up to 10,000 agents.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Factored MDPs, Reward Shaping, Scalability

1. INTRODUCTION

Coordinating the actions of multiple agents is a very difficult problem, particularly as scaling increases to hundreds or thousands of agents — techniques which worked for small- or medium-scale systems break down. Effective solutions to the large-scale coordination problem must address two key issues that are prevalent in these systems. First, the solution must provide agents with access to the right information or observations: too much information can be both impractical and overwhelming, but too little information can lead to inaccurate conclusions [2, 8, 19, 27]. Second, the solution must enable agents to make local decisions that are coordinated with the system-level objective — in massive multiagent systems, it is frequently impractical for all agents to directly coordinate [27]. We address these issues with the *Factored action Reinforcement learning Agents in Cooperative tasKs Exploiting Difference rewards* (FRACKED) algorithm, which uses a Factored Action Markov Decision Process (FA-FMDP) framework to exploit problem structure and reduce the per-agent observation and coordination requirements, and reinforcement learning with shaped difference rewards to enable coordination and decision making within the context of the FA-FMDP.

As scaling increases within large multiagent systems, it is typically impractical to assume that agents will maintain full observability and complete system knowledge [27]. To address this, it is common to exploit the underlying problem structure in order to decompose the system and reduce the observation and coordination requirements for individual agents[14, 27]. One of the most popular approaches involves decomposing a large Markov Decision Process (MDP) into several smaller joint MDPs with localized rewards (a so-called Factored Markov Decision Process (FMDP)) [14, 22, 33]. Although factorization reduces the coordination complexity, solving an FMDP is still much more complex than solving a disjoint set of MDPs in parallel. This is because there is significant coupling between the local MDPs in an FMDP framework. Thus, solving an FMDP still represents a formidable coordination task for agents. In this work, we use a particular type of FMDP known as a Factored-Action FMDP to exploit the problem structure and reduce agent observations and coordination requirements.

In addition to reducing the observations and coordination requirements, we propose the use of reinforcement learning coupled with agent-centric shaped rewards (i.e., difference rewards) to address coordination and decision making for agents within the context of the FA-FMDP. Reinforcement learning addresses the temporal credit assignment problem present within learning (how to assign credit for actions with time-delayed rewards) and difference rewards address the structural credit assignment problem (how should credit be assigned to agents based upon their individual contributions to the collective goal). Difference rewards have shown previous success in many domains, but they typically require agents to have full observability. We reduce these observability requirements by establishing the problem within an FA-FMDP framework which reduces the coordination and observation requirements for agents within the system.

In this work, we introduce the FRACKED algorithm that combines an FA-FMDP framework, reinforcement learning, and difference rewards to promote coordination and scaling in large multiagent systems. The FRACKED algorithm ad-

dresses three key issues present within these systems: i) it reduces the per-agent observation and coordination requirements by exploiting problem structure, ii) it enables agent decision making by using reinforcement learning, and iii) it addresses the structural credit assignment problem by using difference rewards. The remainder of this paper is structured as follows. Section 2 provides background information on FA-FMDPs, reinforcement learning, difference rewards, and learning automata. Section 3 introduces the two domains used in this work. Section 4 outlines the FRACKED algorithm. Section 5 contains empirical results demonstrating the performance benefits of using our algorithm in a congestion domain and a combinatorial optimization domain with up to 10,000 agents. Finally, Section 6 provides discussion and conclusions of this work.

2. BACKGROUND

Learning in large multiagent systems with large state-action spaces can be extremely challenging, if not infeasible, without some form of exploitation of the problem structure to reduce the agent-to-agent coordination requirements [12, 32, 33, 38]. Exploiting the underlying structure of a problem has long been recognized as an important aspect in representing sequential decision making tasks [37, 3]. One popular application of exploiting structure involves factorization within Markov Decision Processes (MDPs) that are extremely large or intractable [7, 37]. An MDP can be broken down with respect to states (FS-FMDPs), actions (FA-FMDPS), or both (FMDPs) [14, 21, 31]. These techniques exploit system knowledge and problem structure in order to simplify the problem's representation [4, 7, 13, 22, 37]. In this work, we use FA-FMDPs to exploit system structure and reduce coordination complexity for agents [21, 24, 28].

FMDPs have been shown to work well in a number of situations including controlling teams of rovers, robots playing robo-cup soccer, and vehicle traffic within a traffic grid [15, 22, 23]. There have been several methods proposed for solving FMDPs such as *Coordinated Reinforcement Learning* or a *coordination graph*, which uses the dynamic Bayesian network (DBN) structure, message passing schemes, and action selection mechanisms (e.g., variable elimination and max-plus) [15, 22, 23]. These techniques have typically relied upon iterative message passing schemes between agents in order to make decisions, which come with high communication, overheads and are frequently slow due to the message passing process. In this work, we use techniques (i.e., reinforcement learning with difference rewards) which rely only upon the agents having local observability and do not require explicit communication between agents.

2.1 Factored Action - Factored Markov Decision Processes

Factored Markov Decision Processes (FMDPs) represent one approach to generalization, which exploits the underlying structure of the problem in multiagent systems [31]. The idea of representing a large MDP using a factored model was first proposed by Boutilier et al. [4]. A factored MDP is a mathematical framework for sequential decision problems in stochastic domains [14]. It thus provides an underlying semantics for the task of planning under uncertainty [14]. FMDPs are a representation that allows exploitation of the problem structure, allowing compact representations of large multiagent systems [14]. We focus on Factored-Action Factored Markov Decision Processes (FA-FMDPs), but our algorithm could be extended into Factored-State Factored Markov Decision Processes, representations that are factored with respect to both states and actions.

Dean *et al.* [13, 21] included a description of the Factored-Action FMDP framework, which we call FA-FMDP hereafter. An FA-FMDP is a tuple $M = \{\mathbf{X}, \mathbf{A}, T, R\}$. $\mathbf{X} = [X_1, ..., X_n]$ is a state vector of variables which collectively define the states [21]. Similarly, $\mathbf{A} = [A_1, ...A_m]$ is a vector of agent actions, which jointly define the system action [21]. The action space for M is $\Pi_m[A_i]$, where $[A_i]$ denotes the set of values for A_i [21]. T is the set of transition probabilities and R is the system reward. The system structure in a FA-FMDP is defined so that the system reward, R, is decomposed into a set of localized rewards, R_i, such that $R = \sum_i R_i$. Each local reward, R_i, depends upon a subset of the agents in the system, $\Gamma(i)$.

2.2 Finite Action Learning Automata

Learning automata are a method commonly used to address decision making in both stateful and state-less environments [36]. The finite action learning automata (FALA) can be defined as a quadruple $(A, B, \mathcal{T}, \mathbf{p}(k))$, where $A = \{a_1, \ldots, a_n\}$ is a finite set of actions, B is a set of rewards or reinforcements from the environment, and $\mathcal{T} : B \times \mathbf{p}(k) \rightarrow \mathbf{p}(k+1)$ is the learning algorithm for updating the probabilities vector $\mathbf{p}(k)$. $\mathbf{p}(k)$ is the probability vector at timestep k, defined by $\mathbf{p}(k) = (p_1(k), \ldots, p_n(k))^T$ such that $p_i(k) \in [0, 1]$, $i = 1, \ldots, n$ and $\sum_{i=1}^{k} p_i(k) = 1$.

This work uses FALA as a second learning mechanism and adopts the linear reward-inaction(L_{R-I}) algorithm as our update mechanism [36]. When an agent takes action a_i at step k, we have following updating equations:

$$\begin{aligned} p_i(k+1) &= p_i(k) + \lambda\beta(k)(1 - p_i(k)) \\ p_j(k+1) &= p_j(k) - \lambda\beta(k)p_j(k), \text{ for } j \neq i. \end{aligned} \tag{1}$$

where $\lambda \in (0, 1)$ is the learning rate and $\beta(k) \in \{0, 1\}$ is the reward or reinforcement from environments at step k. We adopt a naïve method that if the reward that an agent received after taking an action is greater than the average rewards among all agents, the environmental feedback is positive, otherwise negative. In other words, if the agent achieves better than the average reward, it receives a positive feedback such that $\beta(k) = 1$. Otherwise, the reward is $\beta(k) = 0$.

2.3 Reinforcement Learning

Reinforcement learning is frequently been used throughout the multiagent literature to address the decision making problem for agents [5, 23, 34, 35]. Reinforcement learning addresses the temporal credit assignment problem present within learning. In this paper, each agent is an ϵ-greedy reinforcement learner using Q-learning updates [34]. The domains considered in this work are stateless, similar to a multi-agent, multi-armed bandit. This simplifies the reinforcement learning framework, eliminating the concept of state. Agents in this work use table-based immediate reward reinforcement learning, equivalent to a ϵ-greedy Q-learners with a discount rate of 0 [34]. On every timestep, an agent executes an action and then receives a reward evaluating that action. After taking an action and receiving a reward,

an agent updates its Q table as:

$$Q(a) \leftarrow Q(a) + \alpha(r - Q(a)) \quad (2)$$

where a is the agents' action selection, r is the reward received for taking action a, α is the learning rate, and Q is the value associated with taking action a. At every timestep the agent chooses the action with the highest table value with probability $1 - \epsilon$ and chooses a random action with probability ϵ.

2.4 Difference Rewards

Difference rewards are a type of shaping reward in the form [6, 9, 16, 18, 17, 40]:

$$D_j \equiv R(z) - R(z_{-j} + c_j) \quad (3)$$

where R is the system objective, z is the complete system state vector, and z_{-j} contains all the variables not affected by agent j. Because this work focuses on stateless domains, the vector z is a vector of actions. To construct z_{-j}, all actions in z that are affected by agent j are replaced with the fixed constant c_j (a counterfactual action). Difference rewards are directly aligned with the system objective, R, regardless of the choice of c_j because the second term does not depend on j's actions [18]. Furthermore, they provide a learning signal that often leads to faster learning — the second term of D removes much of the effect of other agents (i.e., noise) from j's reward.

In many situations it is possible to use a c_j that is equivalent to taking agent j out of the system, which causes the second term to be independent of j (i.e. the system performance without agent j), and therefore D_j evaluates the agent's contribution to the global performance. There are two key advantages to using D_j: First, because the second term removes a significant portion of the impact of other agents in the system, it provides an agent with a "cleaner" signal than the global reward [18]. Second, because the second term does not depend on the actions of agent j, any action by agent j that improves D, also improves R (the derivatives of D and R with respect to agent j are the same) [18].

The *expected difference reward* (EDR) is given by:

$$EDR_j \equiv R(z) - E_{z_j}[R(z)|z_{-j}] \quad (4)$$

where $E_{z_j}[R(z)|z_{-j}]$ gives the expected value of R over the possible actions of agent j. Because this term does not depend on the immediate actions of j, this utility is still aligned with R [1]. Furthermore, because it removes noise from an agent's private utility, EDR yields far better learnability than does R [18]. This noise reduction is due to the subtraction which, to a first approximation, eliminates the impact of other agents on the learning signal of agent j. The major difference between EDR and D is how they handle z_j. EDR provides an estimate of agent j's impact by sampling all possible actions of agent j, whereas D removes agent j from the system.

3. DOMAINS

Our framework is empirically validated in two domains: the first is a congestion domain based upon previous work done by Kok et al. [22] and the second is a combinatorial optimization problem based upon the Defect Combination Problem introduced elsewhere [18].

3.1 Gaussian Squeeze Domain

Often agents need to allocate resources in such a way that enough resources are allocated to accomplish a goal, but not too many resources are allocated as to be wasteful. This problem is also similar to congestion domains, where we do not want too much traffic on a particular route, but we still want that route to be sufficiently used. In this section we describe an abstract version of this problem, where the action of agent j is a choice allocation quantity x_j, and the goal of the system is to have a sum of allocations $\sum_j x_j$ that is neither too high or too low. The penalty for miss-allocation is defined by a scaled Gaussian. Given the sum of actions $x = \sum_j x_j$, the reward is as follows:

$$x e^{\frac{-(x-\mu)^2}{\sigma^2}}, \quad (5)$$

where μ and σ are mean and sigma parameters describing approximately how much total allocation we want and how important it is for the allocation to be close to its desired value.

In the above problem there is only one target allocation value and all agents are participating in the problem. In this paper, we will use a more complex version of this problem in which there are multiple allocation targets and a subset of agents is participating in each target. Each target i has its own mean μ_i and its own sigma, σ_i. For each target, a subset of agents participates, C_i, such that the sum of allocated resources x_i for each target i is $x_i = \sum_{j \in C_i} x_j$. Note that these are overlapping sets, and that a single agent can participate in multiple targets. The full reward is a sum of target rewards:

$$R = \sum_i x_i e^{\frac{-(x_i-\mu_i)^2}{\sigma_i^2}} \quad (6)$$

3.2 Multi-Target Defect Combination Problem

Many real world sensing applications require large sets of disparate sensing devices to coordinate their actions in order to collectively optimize their network attenuation, coverage areas, and sensing schedules [10, 30, 39]. In this domain, a set of sensing devices must coordinate their schedules to optimize their aggregated attenuation over a set of targets within an environment. Sensing agents are deployed so that each can observe a subset of the targets — the agents must work together to collectively optimize the accuracy of their sensor readings over these targets. This domain is a type of Defect Combination Problem (DCP) introduced in [18] (although similar domains aiming to minimize Bayesian uncertainty in sensor networks for multi-target tracking have also been proposed [25]). This problem assumes that there is a set of imperfect sensors, $\mathbf{X}$, which have constant attenuations due to manufacturing defects or imperfections. Each of the sensors, x_j, has an associated attenuation, ζ_j, which can be positive or negative in its reading. If agent j takes tries to measure the actual value, A, it instead produces a value of $A + \zeta_j$. The agents must learn how to best choose a

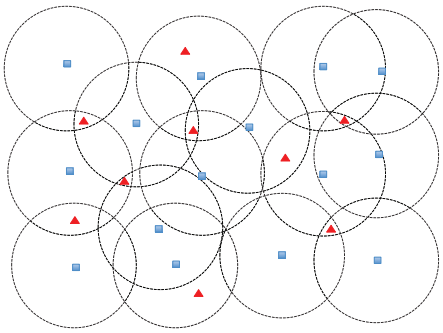


Figure 1: In the Multi-Target Defect Combination Problem, a set of imperfect sensors (blue squares) are in an environment with multiple observation targets (red triangles). The sensors each have different attenuations and each sensor can only observe a subset of the targets. Here, we represent the coverage region of each sensor node by a circle of a given radius. Each sensor can observe any targets that fall within its radius. The goal of the sensors is to optimize their aggregated accuracy of the observations across all of the targets (Equations 7).

subset of sensors that minimizes the aggregated attenuation of the combined readings for the set of targets within the domain:

$$R = \sum_{i=1}^{M} \frac{\left| \sum_{j=1}^{N} n_{j,i} \zeta_j \right|}{\sum_{j=1}^{N} n_{j,i}} \tag{7}$$

where R is the aggregated attenuation (system reward) of the combined sensor readings over the set of M targets, ζ_j is the attenuation of a particular sensor j, N is the number of sensors, and $n_j \in \{0, 1\}$ is an indicator based upon whether sensor j chooses to be on or off.

This is an NP-complete optimization problem [18] and simply choosing the single sensor with the best attenuation for each target is an inadequate solution, as is choosing the best K sensors ($1 \leq K \leq N$). To illustrate this, consider the case where there is a single target ($M = 1$) and there are 6 sensing devices whose attenuations are $\zeta_1 = -0.19$, $\zeta_2 = 0.54$, $\zeta_3 = 0.1$, $\zeta_4 = -0.14$, $\zeta_5 = -0.05$, and $\zeta_6 = 0.21$. Choosing only the best sensor ζ_5 would yield an aggregated attenuation of $|0.05|$, while choosing sensors ζ_3, ζ_4, and ζ_5 would yield an aggregated attenuation of $|0.03|$, which is better than the single best sensing device ζ_5 alone. This is still not the optimal solution in this 6 sensor case however, as combining sensors ζ_1 and ζ_6 results in an aggregated attenuation of $|0.01|$. In this problem, individual sensors acting independently without coordinating their actions can drastically decrease the system performance. Consider the case where sensors ζ_1 and ζ_6 are turned on in conjunction with sensor ζ_2, the aggregated attenuation jumps to from $|0.01|$ to $|0.18|$. Finding good solutions requires a great deal of coordination between sensors, as any one sensor can heavily impact the system performance.[1]

[1]The DCOP [26] framework is a popular, and important, framework for solving coordination problems. However, because the rewards for different agents actions are not known *a priori*, a learning, rather than planning, method is required.

Algorithm 1 – FRACKED**:** Given a large joint multiagent system, exploit symmetries within the problem by framing as a Factored MDP framework and use reinforcement learning and difference rewards.

Given a set of N sensing agents and M sub-objectives
Frame as an FA-FMDP framework
for $Run = 1 \rightarrow Run_{Max}$ **do**
  for $Episode = 1 \rightarrow Episode_{Max}$ **do**
    Agents Select Actions (ϵ-greedy)
    Calculate System Performance:
    $$R = \sum_{i=1}^{M} R_i$$
    Calculate Agent Rewards:
    $$D_{j'} = \sum_{i \in C_{j'}} R_i - \sum_{i \in C_{j'}} R_{i,-j'}$$
    or
    $$EDR_{j'} = \sum_{i \in C_{j'}} R_i - E[\sum_{i \in C_{j'}} R_i | a_{j'}]$$
    Perform Value Updates:
    $$Q(a) \leftarrow Q(a) + \alpha(r - Q(a))$$
  end for
end for

4. THE FRACKED ALGORITHM

This section introduces the FRACKED algorithm, allowing effective learning in large multiagent systems by reducing observability and coordination requirements for individual agents and addressing the structural credit assignment problem through shaped rewards. The two domains used in this paper are stateless, making the agents action-value learners (i.e., the limiting case of the single-state FA-FMDP). The reward decomposition from R to local rewards, R_i, is based upon localized joint-actions and localized rewards R_i only depend upon a subset of the agents $\Gamma(i)$ (see Section 2.1). This decomposition means that agents are only required to coordinate with other agents in their own localized subset. Unfortunately, even with this decomposition these localized joint-actions quickly become intractable as the number of agents increases. Our algorithm avoids the computational costs associated with requiring agents to enumerate their local joint-actions by only requiring agents to enumerate *only their own actions*, exploiting the reward decomposition defined by the FA-FMDP structure.

Exploiting structure within a problem to reduce both the observability and coordination requirements for agents is critical within large multiagent systems. In the domains of this work, it is possible to decompose the system reward R into a set of M localized rewards R_i, each of which only depend upon a subset of the agents in the system, $\Gamma(i)$, such that $R = \sum_{i=1}^{M} R_i$ [14]. Here, each agent j is involved with a subset C_j of the M localized rewards, and therefore each agent only needs to coordinate directly with the subsets of agents $\Gamma(i)$ involved in each of its localized rewards $i \in C_j$. In this setting, each agent's individual reward function r_j is comprised upon the localized rewards R_i to which the agent is contributing, such that $r_j = \sum_{i \in C_j} R_i$. Agents optimizing their individual rewards, r_j within the FA-FMDP structure, will simultaneously be acting to optimize the system-level objective R [4, 7, 22].

Although the FA-FMDP structure reduces coordination

requirements and establishes local rewards for agents, it does not address decision making. Reinforcement learning enables agent decision making and addresses the temporal credit assignment problem. In this work, agents use reinforcement learning to iteratively improve their decisions based upon their individual rewards, r_j. Although reinforcement learning addresses the temporal credit assignment problem, it does not directly address the structural credit assignment problem. The problem of how agents should be assigned reward becomes increasingly complex as the number of agents, and groups of agents, increases. We address this shortcoming by using agent-specific difference rewards which were designed specifically to address the structural credit assignment problem within large multiagent systems.

Difference rewards are agent-specific shaped rewards that were designed to address the structural credit assignment problem in multiagent systems. These rewards have been shown to work well in a number of domains including coordinating networks of satellites, controlling swarms of UAVs, and managing air traffic [1, 18]. Although these rewards have been shown to work well within a number of problem settings they have high observability requirements. We address this shortcoming by casting the problems into an FA-FMDP framework, which defines localized coordination requirements and rewards for agents. Agents using difference rewards only require local observability within the FA-FMDP structure. We will now derive difference rewards for agents within the GSD and the MTDCP. Expected Difference Rewards can be similarly derived, but are excluded for brevity.

4.1 Agent-Centric Rewards for GSD

The system reward R in the GSD can be represented by a combination of M local rewards R_i as follows:

$$R = \sum_{i=1}^{M} R_i = \sum_{i=1}^{M} x_i e^{\frac{-(x_i-\mu_i)^2}{\sigma_i^2}} \tag{8}$$

where M is the total number of local rewards in the system (each R_i is a congestion problem, and each agent simultaneously participates in C_j localized rewards), x_i is the sum of the actions of the agents associated with local reward i ($x_i = \sum_{\forall j \in \Gamma(i)} x_j$, where $\Gamma(i)$ is the subset of agents involved with local reward i), μ_i is the mean of Gaussian i, and σ_i^2 is the variance of Gaussian i. In this setting, the goal of each individual agent is to optimize its own set of localized reward functions , $C_{j'}$, as follows:

$$r_{j'} = \sum_{i \in C_{j'}} R_i = \sum_{i \in C_{j'}} x_i e^{\frac{-(x_i-\mu_i)^2}{\sigma_i^2}} \tag{9}$$

where r_j is the individual reward for agent j', which is the sum of all of its $C_{j'}$ local rewards, M is the total number of targets in the system, x_i is the sum of the actions of the agents associated with R_i ($x_i = \sum_{\forall j \in \Gamma(i)} x_j$, where $\Gamma(i)$ is the subset of agents involved with R_i), μ_i is the mean of Gaussian i, and σ_i^2 is the variance of Gaussian i.

Given each agent's individual reward $r_{j'}$, their difference reward can be derived by combining Equations 3 and 9, where the system reward R in Equation 3 is replaced with the individual reward $r_{j'}$ that agent j' is attempting to optimize as follows:

$$\begin{aligned} D_{j'} &= \sum_{i \in C_{j'}} R_i - \sum_{i \in C_{j'}} R_{i,-j'} \\ D_{j'} &= \sum_{i \in C_{j'}} x_i e^{\frac{-(x_i-\mu_i)^2}{\sigma_i^2}} - \sum_{i \in C_{j'}, j \neq j'} x_i e^{\frac{-(x_i-\mu_i)^2}{\sigma_i^2}} \end{aligned} \tag{10}$$

where $D_{j'}$ is the difference reward for agent j', $r_{j'}$ is the reward of agent j' (which depends upon all agents involved in each local reward $R_i \forall i \in C_{j'}$), $R_{i,-j'}$ is the performance of local reward i without the contribution of agent j', and the remaining variables are as defined in Equation 9. The difference rewards in this domain are modified from their traditional form in that they depend upon each agent's individual reward $r_{j'}$ instead of a system reward R. This replacement is because of the FA-FMDP reward decomposition: each agent attempting to optimize its individual reward $r_{j'} = \sum_{i \in C_{j'}} R_i$ is simultaneously attempting to optimize the system reward R. This is a key requirement which enables us to use difference rewards in this localized capacity.

4.2 Agent-Centric Rewards for MTDCP

In the MTDCP, the system objective can be decomposed into a set of sub-objectives, where each sub-objective is the aggregated attenuation of an individual target. In this setting, each sensing agent, j', is able to observe $C_{j'}$ of the M targets in the system simultaneously. Each individual agent's reward function becomes the sum of the attenuations of the targets it senses as follows:

$$r_{j'} = \sum_{i \in C_{j'}} \frac{\left| \sum_{j \in \Gamma(i)} n_{j,i} \zeta_j \right|}{\sum_{j \in \Gamma(i)} n_{j,i}} \tag{11}$$

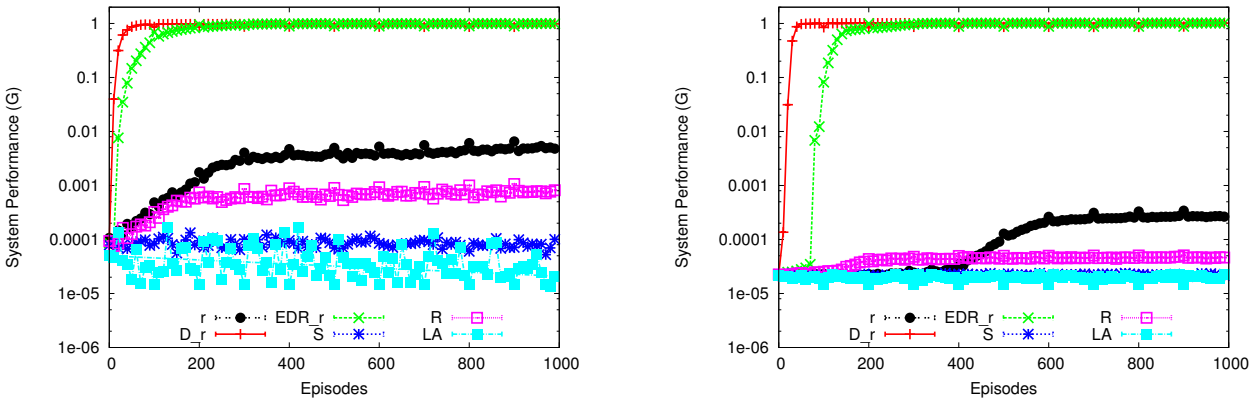
where $r_{j'}$ is the reward of agent j', $i \in C_{j'}$ is the set of $R_i \in C_{j'}$ that agent j' effects, $j \in \Gamma(i)$ is the subset of agents that are impacting reward R_i, ζ_j is the attenuation of agent j's sensor, and n_j is an indicator function that takes the value $\{0, 1\}$ that determines whether sensor j chose to be on or off. By combining Equations 3 and 11, we derive the difference rewards for agents within the MTDCP domain:

$$D_{j'} = \sum_{i \in C_{j'}} \frac{\left| \sum_{j \in \Gamma(i)} n_{j,i} \zeta_j \right|}{\sum_{j \in \Gamma(i)} n_{j,i}} - \sum_{i \in C_{j'}} \frac{\left| \sum_{j \in \Gamma(i), j \neq j'} n_{j,i} \zeta_j \right|}{\sum_{j \in \Gamma(i), j \neq j'} n_{j,i}} \tag{12}$$

where $D_{j'}$ is the difference reward for agent j', $r_{j'}$ is the base reward for agent j' which depends upon all agents involved in $R_i \forall i \in C_{j'}$, and the remaining variables are as defined in Equation 11. Here, each agent is attempting to optimize the coverage over the subset of targets it is sensing.

5. RESULTS

In this work we compared the performance of six types of agents:



(a) 1000 Agents

(b) 10000 Agents

Figure 2: Scaling the number of agents between N =1,000 and N =10,000 in the Gaussian Squeeze Domain with $\mu = 0.10N$ and $\sigma = 0.10N$. Agents using traditional global rewards R perform worse than other learning agents. This is due to problems with structural credit assignment: individual agents cannot determine how their own actions impacted the reward they received. Agents using rewards r_j, which are defined by the FA-FMDP structure perform better than standard global rewards as they eliminate the impact of many agents by focusing on local areas, however they still fail to achieve good coordination. The FRACKED algorithm overcomes these shortcomings by first establishing local coordination requirements using an FA-FMDP framework, and using reinforcement learning with difference rewards to address the local coordination problems.

- Learning agents using FALA (LA)
- Learning agents using traditional global rewards (R)
- Learning agents using localized rewards r_j defined by the FA-FMDP framework (r)
- Learning agents using localized difference rewards based upon r_j (D_r)
- Learning agents using localized expected difference rewards based upon r_j ($EDRr$)
- Agents behaving randomly (S)

All agents used ϵ-greedy Reinforcement learning with $\epsilon = 0.10$ and $\alpha = 0.10$. All experiments were run for $r = 30$ statistical runs and $e = 5000$ learning episodes, unless otherwise specified (all results were statistically significant as verified by a t-test with $p = 0.05$). In each set of experiments, the overall system performance R was plotted as the performance metric. Unless otherwise stated, each agent was connected to $C_j = 10$ of the M localized rewards R_i, resulting in approximately 90% less observability than an equivalent system without the FA-FMDP reward decomposition. LA Agents set $\Lambda = \epsilon$ and randomly initialize the probability vector.

5.1 Normalized Gaussian Payoff Domain

The first set of experiments is conducted in the GSD. Here, there were $M = 100$ independent Gaussian congestion problems and each agent is randomly connected to $C_j = 10$ of these Gaussians. As seen here, agents using random policies (S) performed poorly, which could be expected in this domain. Here, agents are sporadic in nature and are unable to optimize the congestion within each Gaussian due to a lack of coordination and intelligent decision making. Agents using system level rewards struggle due to their rewards being directly coupled to the actions of all other agents within the system, making it difficult to determine how their individual actions impacted the reward they received. These results suggest that providing agents with a system reward within a large multiagent system is an inadequate solution.

In GSD, learning automata (LA) with reward-inaction algorithm shows one weakness of LA — LA perform poorly. [2] The primary reason for poor performance of the LA algorithm is that the GSD has a maximum and minimum value but the mean value does not contain enough information to lead allow agent find a gradient and obtain higher rewards.

An FA-FMDP framework attempts to address this problem by exploiting the system structure in order to decompose the system into 'localized regions', which focuses the observability and coordination requirements of individual agents. In this case, the individual agents attempt to coordinate within the localized regions defined by the FA-FMDP in order to optimize their local rewards r_j. Unfortunately, even using these individual rewards r_j defined by the FA-FMDP structure is insufficient and only marginally improves learning as seen in Figure 3. This is because even these individual rewards r_j still depend upon many other agents. In this setting, standard global rewards R or even localized team-based rewards such as the r_j do not adequately address the structural credit assignment problem (how to assign credit to agents for their individual actions).

The FRACKED algorithm addresses this problem by using the reduced observability and individualized agent rewards r_j for agents defined by the FA-FMDP structure, while simultaneously addressing the structural credit assignment for agents using difference rewards. These rewards provide each agent with specific feedback on how its individual action impacted both its own reward r_j, as well as how it impacted the system reward. As seen in Figures 2a and 2b, agents using difference rewards within this setting Dr and $EDRr$ significantly outperform other methods. This is because as

[2]Multiple LA algorithms and formulations of β were tested — this paper reports the highest-performing combinations.

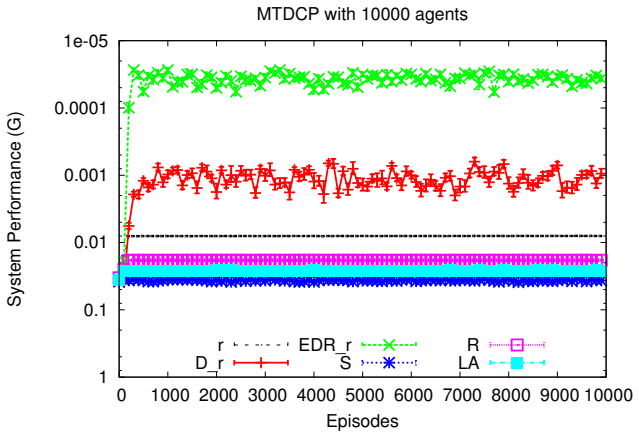


Figure 3: 10,000 agents with 100 targets and each agent is assigned to 10 targets. The FRACKED algorithm outperforms all other methods by significantly reducing the coordination requirements for agents in the system.

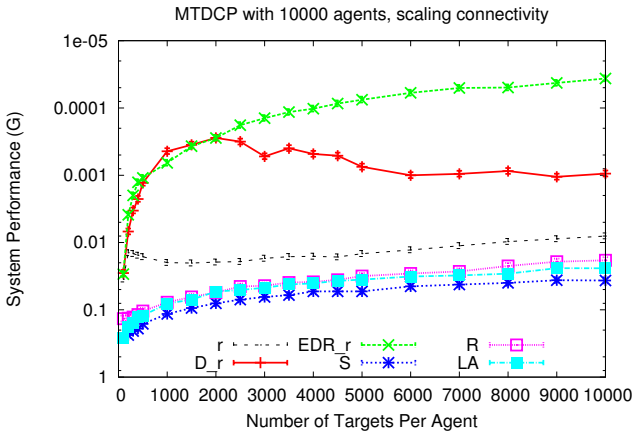


Figure 4: 10,000 agents with 100 targets and each agent is assigned to 10 targets. As seen, agents using the FRACKED algorithm (Dr and $EDRr$) significantly outperform all other methods with scaling up to 10,000 agents.

multiagent systems become increasingly large, the structural credit assignment problem becomes increasingly debilitating to system performance. As seen, the FRACKED algorithm results in significantly improved learning performance.[3]

5.2 Multi-Target Defect Combination Problem

The first set of experiments in the MTDCP domain uses N = 10,000 sensing agents, T = 100 targets, and each agent is connected to C_j = 10 targets, randomly. Agents using a random policy S perform poorly, as they fail to coordinate their actions intelligently within this combinatorial optimization problem. Learning automata agents use the reward-inaction algorithm and do indeed improve learning over time, but perform poorly relative to Q-learning agents.[4] Agents using traditional global rewards R also perform poorly within this setting. This is because agents struggle to coordinate their actions due to noise on their learning signals caused by the actions of other agents. Again, the FA-FMDP framework benefits performance in that it reduces this noise by restricting agents to communicating directly with a subset of the system. As seen, performance is improved when the agents receive individual rewards r_j defined by the FA-FMDP framework (Equation 11) instead of system rewards. However, agents still have difficulty learning due to the structural credit assignment problem. Again, our algorithm uses the strengths associated with this reward decomposition, while at the same time addressing its shortcomings by addressing the structural credit assignment problem using difference rewards which results in significantly improved performance (Figure 3). To show the robustness of this approach to scaling, we included a plot of scaling the number of agents in the system from 100 to 10,000. These results show that our algorithm is robust to scaling and is beneficial for systems of ranging sizes (Figure 4).

[3]Performance tradeoffs between D and EDR rewards are discussed in [18].

[4]All other reward formulations discussed in this paper were tested with LA, but they (surprisingly) performed worse than when using the global reward. Our experiments thus focus on Q-learning agents in this domain.

6. DISCUSSION

We demonstrated that in many large multiagent systems, the coordination requirements for agents are overwhelming, resulting in poor coordination and performance. To address this, we have introduced the FRACKED algorithm which reduced agent-to-agent coordination requirements, increased performance, and offered improved scalability over many existing techniques. The FRACKED algorithm used an FA-FMDP framework to exploit the problem structure that exists in many large scale multiagent systems in order to decompose the system and establish localized coordination requirements for agents. It then leveraged reinforcement learning with difference rewards in order to promote learning and coordination within the FA-FMDP structure. We demonstrated that the FRACKED algorithm significantly outperformed agents using traditional global rewards R, localized "global" rewards within an FA-FMDP structure r, and finite action learning automata LA in two domains with scaling up to 10,000 agents.

This algorithm shows significant promise for scaling to large multiagent systems. However, this work assumed that the system designer understood the system's structure and was able to decompose the system within an FA-FMDP framework accordingly. As Degris *et al.* pointed out, such system information is not always available. Our algorithm can be extended to the setting that the agents must collectively learn the underlying FA-FMDP framework for the system without significant prior knowledge. This would extend the benefits of our algorithm to complicated domains where casting the problem into an FA-FMDP framework is non-trivial. Finally, to improve the results for LA, we could use advanced techniques for determining $\beta(k)$ from the environment to improve performance.

Acknowledgements This work was partially supported by the National Energy Technology Laboratory under grant DE-FE0011403 and by the National Science Foundation under grant IIS-1149917.

7. REFERENCES

[1] A. Agogino, C. HolmesParker, and K. Tumer.

Evolving large scale UAV communication system. In *GECCO*, Philadelphia, PA, July 2012.

[2] S. Barrett, P. Stone, S. Kraus, and A. Rosenfeld. Teamwork with limited knowledge of teammates. In *AAAI*, 2013.

[3] J. Berry, L. Kaelbling, and T. Lozano-Perez. Hierarchical solution of large markov decision processes. In *ICAPS Workshop on Planning and Scheduling Under Uncertainty*, 2010.

[4] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning - structural assumptions and computational leverage. *JAIR*, 1999.

[5] T. Brys, A. Harutyunyan, P. Vrancx, M. Taylor, D. Kudenko, and A. Nowe. Multi-objectivization of reinforcement learning problems by reward shaping. *IEEE IJCNN*, 2014.

[6] M. Colby and K. Tumer. Multiagent reinforcement learning in a distributed sensor network with indirect feedback. In *AAMAS*, 2013.

[7] T. Degris, O. Sigaud, and P. Wuillemin. Learning the structure of factored markov decision processes in reinforcement learning problems. In *ICML*, 2006.

[8] S. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *AAMAS*, 2012.

[9] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *AAMAS*, 2014.

[10] A. Farinelli, A. Rogers, and N. Jennings. Maximising sensor network efficiency through agent-based coordination of sense/sleep schedules. *In Proceedings of the International Workshop on Energy in Wireless Sensor Networks*, 2008.

[11] J. Fossel, D. Hennes, S. Alers, D. Claes, and K. Tuyls. Octoslam - a 3D mapping approach to situational awareness of unmanned aerial vehicles. *AAMAS*, 2013.

[12] M. Grzes and D. Kudenko. Learning shaping rewards in model-based reinforcement learning. In *AAMAS (ALA)*, Budapest, Hungary, 2009.

[13] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps. In *NIPS*, 2014.

[14] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *JAIR*, 2003.

[15] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *ICML*, 2002.

[16] C. HolmesParker, A. Agigino, and K. Tumer. Evolving distributed resource sharing for cubesat constellations. *GECCO*, 2012.

[17] C. HolmesParker, A. Agogino, and K. Tumer. Clean rewards for improving multiagent coordination in the presence of exploration. In *AAMAS*, 2013.

[18] C. HolmesParker, A. Agogino, and K. Tumer. Combining reward shaping and hierarchies for scaling to large multiagent systems. *Knowledge Engineering Review*, 2013.

[19] E. Howley and J. Duggan. Investing in the commons - a study of openness and the emergence of cooperation. In *Advances in Complex Systems*, 2011.

[20] A. Iscen, V. SunSpiral, and K. Tumer. Controlling tensegrity robots through evolution. In *GECCO*, 2013.

[21] K. Kim and T. Dean. Solving factored MDPs with large action space using algebraic decision diagrams. In *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*, 2002.

[22] J. Kok. Coordination and learning in cooperative multiagent systems. *Dissertation*, 2006.

[23] L. Kuyer. Multiagent reinforcement learning and coordination for urban traffic control using coordination graphs and max-plus. *MS Thesis*, 2007.

[24] B. Kveton, M. Hauskrecht, and C. Guestrin. Solving factored MDPs with hybrid state and action variables. *JAIR*, 2006.

[25] Lamber and Sinno. Bioinspired resource management for multi-sensor target tracking systems. In *MIT Lincoln Laboratory Project Report MD-26*, 2011.

[26] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. *AAMAS*, 2003.

[27] L. Panait and S. Luke. Cooperative multi-agent learning - the state of the art. *JAAMAS*, 2005.

[28] A. Raghavan, S. Joshi, A. Fern, P. Tadepalli, and R. Khardon. Planning in factored action spaces with symbolic dynamic programming. *AAAI*, 2012.

[29] E. Reed and O. Mengshoel. Scaling bayesian network parameter learning with expectation maximization using mapreduce. In *NIPS Big Learning*, 2012.

[30] A. Rogers, A. Farinelli, and N. Jennings. Self-organising sensors for wide area surveillance using the max-sum algorithm. *Lecture Notes in Computer Science. Self-Organizing Architectures*, 2010.

[31] B. Sallans and G. Hinton. Reinforcement learning with factored states and actions. *JMLR*, 2004.

[32] A. Sherstov and P. Stone. Function approximation via tile coding - automating parameter choice. In *In Proceedings of teh Symposium on Abstraction, Reformulation, and Approximation (SARA)*, 2005.

[33] A. Strehl, C. Diuk, and M. Littman. Efficient structure learning in factored-state mdps. *AAAI*, 2007.

[34] R. Sutton and A. Barto. *Reinforcement Learning An Introduction*. MIT Press, Cambridge, MA, 1998.

[35] M. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey. Reinforcement learning agents providing advice in complex video games. *Connection Science*, 2013.

[36] M. A. Thathachar and P. S. Sastry. *Networks of learning automata: Techniques for online stochastic optimization*. Springer, 2004.

[37] L. Tyler and C. Boutilier. Eliciting additive reward functions for markov decision processes. *IJCAI*, 2011.

[38] S. Whiteson, M. Taylor, and P. Stone. Adaptive tile coding for value function approximation. In *AI Technical Report AI-TR-07-339, UT Austin*, 2007.

[39] S. Williamson, E. Gerding, and N. Jennings. Reward shaping for valuing communications during multi-agent coordination. In *AAMAS*, 2009.

[40] L. Yliniemi and K. Tumer. Extending the difference reward to multi-objective congestion problems. In *AAMAS (OPTMAS)*, 2012.