Using Reward/Utility Based Impact Scores in Partitioning (Extended Abstract)

William Curran
Oregon State University
Corvallis, Oregon
curranw@onid.orst.edu

Adrian Agogino
NASA AMES Research Center
Moffet Field, California
adrian.k.agogino@nasa.gov

Kagan Tumer
Oregon State University
Corvallis, Oregon
kagan.tumer@oregonstate.edu

ABSTRACT

Reinforcement learning with reward shaping is a well-established but often computationally expensive approach to multiagent problems. Agent partitioning can assist in this computational complexity by treating each partition of agents as an independent problem. We introduce a novel agent partitioning approach called Reward/Utility-Based Impact (RUBI). RUBI finds an effective partitioning of agents while requiring no prior domain knowledge, provides better performance by discovering a non-trivial agent partitioning, and leads to faster simulations. We test RUBI in the Air Traffic Flow Management Problem, where there are simultaneously tens of thousands of aircraft affecting the system and no intuitive similarity metric between agents. When partitioning with RUBI in the ATFMP, there is a 37% increase in performance, with a 510x speed up per simulation step over non-partitioning approaches.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents

Keywords

Multiagent Partitioning, Multiagent Learning

1. INTRODUCTION

Two key elements in a multiagent reinforcement learning system is minimizing computation time and maximizing coordination. Reward shaping is a field in multiagent reinforcement learning that focuses on the design of rewards, and has been shown to assist in multiagent coordination. This reward shaping is typically at a large cost to computation time, and in large, highly coupled domains reward shaping quickly becomes computationally intractable.

Partitioning agents into hierarchies [3] or teams [2] speeds up computation time for extremely large domains (approx. 10000-40000 agents) while still using the reward shaping technique without approximation error. In this paper we introduce Reward/Utility-Based Impact (RUBI) scores. RUBI partitions agents by determining the effect of one agent's action on another agent's reward. Using this metric it develops similarity metrics between all agents in order to usefully partition and therefore reduce the complexity of the learning problem. In contrast to many other partitioning approaches, this has the advantage of requiring *no domain knowledge*.

The contributions of this work are: **Generality**: RUBI requires no prior knowledge of the domain, essentially treating the domain as a black box to obtain rewards from. **Usability**: RUBI removes the need to derive similarity metrics, removing the need for domain experts in situations where a domain expert isn't available. **Performance**: RUBI discovers non-trivial agent partitioning by using a reward function to partition agents. **Speed**: RUBI leads to a larger number of partitions without losing performance, leading to more independence and therefore faster simulations.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. RUBI

In this work we introduce an autonomous partitioning algorithm requiring no domain knowledge, the Reward/Utility Based Impact algorithm. We develop an initial agent similarity matrix that uses no knowledge about the domain, and partitions agents together based on the impact of one agent to another. This matrix can then be used as an input to a hierarchical agglomerative clustering algorithm.

If one agent's action heavily impacts another agent's reward (positively or negatively), those agents are coupled enough to be partitioned together. The RUBI algorithm computes a localized reward for each agent with agent i in the system, and then compares that reward to the localized reward for each agent if agent i is not in the system. This partitioning algorithm is based around the central idea: $|L_i(z) - L_i(z - z_j)| > |L_k(z) - L_k(z - z_j)| \Rightarrow SIM(i,j) > SIM(k,j)$, where $L_i(z - z_j)$ is the localized reward of agent i if j is not in the system, $L_k(z - z_j)$ is the localized reward of agent k if j is not in the system, and L_i and L_k are the localized rewards of i and k when all agents are in the system. This means that if the localized reward of agent i changes more than the localized reward of agent k when agent j is taken out of the system, agent j has more effect on agent i than agent k.

Implementation: The RUBI algorithm first initializes an N x N matrix C, where N is the number of agents within the system. It then calculates actions based on the $ACT()$ function, which is typically random action selection. A simulation is then ran with all of the agents in the system and the localized reward is calculated for every agent. We then remove an agent from the system, recalculate the reward for each agent (since this is a localized reward, this is typically

Algorithm 1 Reward/Utility Based Impact Algorithm

```
1: function RUBI(sim)
2:     C ← NxN
3:     for i ← 1 to iterations do
4:         actions ← ACT()
5:         sim.run(actions)
6:         L(z) ← sim.getRewards()
7:         for r ← 1 to N do
8:             sim.removeAgent(r)
9:             L(z − z_r) ← sim.getRewards()
10:            for a ← 1 to N do
11:                C_{r,a} ← C_{r,a} + |L_a(z) − L_a(z − z_r)|
12:            end for
13:            sim.addAgent(r)
14:        end for
15:    end for
16: end function
```

a fast operation), and update the impact table C.

Reward/Utility based impact: The impact data used to compute the similarity matrix is obtained from a localized reward or utility with respect to an agent. Learning in congestion problems with local rewards typically leads to a poor solution, as agents following local rewards do not optimize the system-level reward. In RUBI, we do not want to learn, but instead analyze the local impact one agent has on another, therefore local rewards are an ideal choice.

RUBI is simply an accumulation of impact scores (line 11 of Algorithm 1). Given enough iterations, this accumulation is informative enough to perform accurate partitioning. In this research we are interested more in the relative impact score from one agent to another, rather than what the explicit impact score is.

Benefits of RUBI: One of the key strengths of RUBI is its sheer simplicity and generality combined with computing highly informative similarity scores, leading to well-performing partitions. It needs no prior knowledge about the domain to perform partitioning, and simply needs a localized reward from each agent to build the similarity matrix. This makes RUBI highly generic and can be applied to any multiagent domain.

Additionally, partitions built using RUBI are likely to be greater in number without loss of performance. Domain-based partitioning based on agent similarity encodes how often two agents impact each other. RUBI on the other hand looks more into how the actions of one agent impact another agents reward. If over a few thousand trials the reward impact of two agents is always 0, those agents actions never impact each others reward. The same is true if the reward impact is always the same non-zero value, the actions do not affect the reward, therefore they are not partitioned. This is a key feature of RUBI, and leads to finding more partitions without loss of performance.

3. RUBI PERFORMANCE IN THE ATFMP

We test RUBI in the Air Traffic Flow Management Problem (ATFMP). The approach used here is the same as in Curran et al. [2] and Agogino and Rios [1, 4], except utilizing RUBI.

The system-level reward in the ATFMP focuses on the cumulative delay (δ) and congestion (C) throughout the system: $G(z) = -(C(z) + \delta(z))$. Agogino and Rios originally had the idea of adding a greedy scheduler to algorithmically remove congestion from the system, while simultaneously using learning to minimize delay. We follow this approach, and therefore our system-level reward is simply: $G(z) = -\delta(z)$

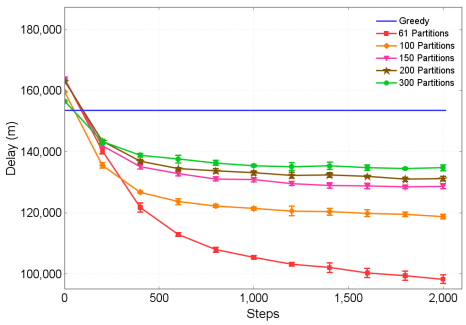

Figure 1: As the number of partitions decreases, performance improves while time complexity increases.

With so many agents, tens of thousands of actions simultaneously impact the system, causing the reward for a specific agent to become noisy with the actions of other agents. A difference reward shaping function reduces much of this noise, and is easily derived from the system-level reward: $D_i(z) = \delta(z - z_i + c_i) - \delta(z)$, where $\delta(z - z_i + c_i)$ is the cumulative delay of all agents with agent i replaced with counterfactual c_i. Without RUBI, this reward shaping technique makes the ATFMP computationally intractable.

Partitions developed using RUBI uses similarity metrics that encapsulated the agent coupling. Partitioning with RUBI and the difference reward outperformed the greedy scheduler. Figure 1 shows a variety of partitions out performing the greedy scheduler. The key benefit of RUBI-based partitioning was that a reward independent partition involved 61 partitions, but in domain-based partitioning the smallest was 3. This leads to a 10% faster processing time, at no cost to performance and using no domain knowledge.

Acknowledgments: This work was partially supported by the National Science Foundation under Grant No. CNS-0931591.

4. REFERENCES

[1] Adrian Agogino. Evaluating evolution and monte carlo for controlling air traffic flow. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, 2009.

[2] William J. Curran, Adrian Agogino, and Kagan Tumer. Addressing hard constraints in the air traffic problem through partitioning and difference rewards. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013.

[3] C. Holmes Parker and K. Tumer. Combining difference rewards and hierarchies for scaling to large multiagent system. In *AAMAS-2012 Workshop on Adaptive and Learning Agents*. Valencia, Spain, June 2012.

[4] J. Rios and J. Lohn. A comparison of optimization approaches for nationwide traffic flow management. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois*, August 2009.