Time-Extended Policies in Multi-Agent Reinforcement Learning

Kagan Tumer
NASA Ames Research Center
ktumer@mail.arc.nasa.gov

Adrian K. Agogino
UC Santa Cruz
adrian@email.arc.nasa.gov

Abstract

Many algorithms such as Q-learning successfully address reinforcement learning in single-agent multi-time-step problems. In addition there are methods that address reinforcement learning in multi-agent single-time-step problems. However, unmodified single-agent multi-time-step methods and multi-agent single-time-step methods cannot necessarily be combined to solve multi-agent multi-time-step problems due to strong coupling between multi-agent interactions between time steps. Rewards that result in multi-agent collaboration for a single time-step may result in poor collaboration in future time-steps. This paper shows how to avoid this problem.

1. Background

There are many problems which can only be properly addressed by having a set of autonomous agents act independently and have their *joint* sequence of actions (z) maximize a pre-set global utility function ($G(z)$). Examples of such problems include control of a constellation of satellites, construction of distributed algorithms, routing over a data network, and control of a collection of planetary exploration vehicles (e.g., rovers on Mars, or submersibles under Europa's ice caps). Instead of maximizing $G(z)$ directly, though, in such problems, each agent i tries to maximize its **private utility function** $g_i(z)$.

To have agents acting to optimize their own private utilities result in also optimizing the provided global utility, the agents' private utilities must be "factored" with the global utility. Formally, a system is **factored** if for each agent i:

$$g_i(z) > g_i(z') \Leftrightarrow G(z) > G(z') \quad \forall z, z' \ s.t. \ z - z_i = z' - z'_i \quad .$$

where z_i is agent i's contribution to the system state z. Intuitively, for all pairs of states z and z' that differ only for agent i, a change in i's state that increases its private utility cannot decrease the global utility. As a trivial example, any system in which all the private utility functions equal G is fully factored [1]. Unfortunately in large systems, such a private utility yields poor results, because it is difficult for an agent to see the effect of its own actions on its own utility (this concept, called **learnability** is discussed in detail in [2]. To allow agents to learn efficiently, we introduced **Wonderful Life (WL)** utility functions [2]:

$$WLU_i^c = G(z) - G(z - z_i + c) \quad (1)$$

where $z - z_i + c$ is the state where agent i's state is replaced by an arbitrary vector c. For any choice of c, WL utilities are factored, and have higher learnability than team games [2].

2. Time-Extended Rewards

Consider a system, where the global utility, G, is a function of a sequence of actions, z, of all the agents. Also assume that the global utility is an undiscounted sum of global rewards (GRs): $G(z) = \sum_t GR_t(z)$. Using a WLU-based reward offers a solution, but care must be taken in how the WL *utility* is broken down into single step WL *rewards*. The straight forward, naive WLR is given by:

$$NWLR_{i,t}^{c_{i,t}}(z) = GR_t(z) - GR_t(z - z_{i,t} + c_{i,t}) \quad (2)$$

where $c_{i,t}$ is i's a fixed action for time step t. This reward is factored at each time step t, since i's actions at time step t cannot affect the value of the second term. However this reward is *not* factored *through* time (i.e., the sum of these rewards factored with GR does not produce a utility factored with the global utility). To illustrate this problem, consider a simple two-time-step, multi-agent problem where each agent can take one of two actions at each time step. For an agent i, we can draw a reward graph showing the outcome of its actions given the actions of the other agents. Potentially agent i could have a different reward graph for each possible combination of actions of all of the other agents. Consider one of these reward graphs as illustrated in Figure 1 (top), showing how the global reward values depend on i's actions.

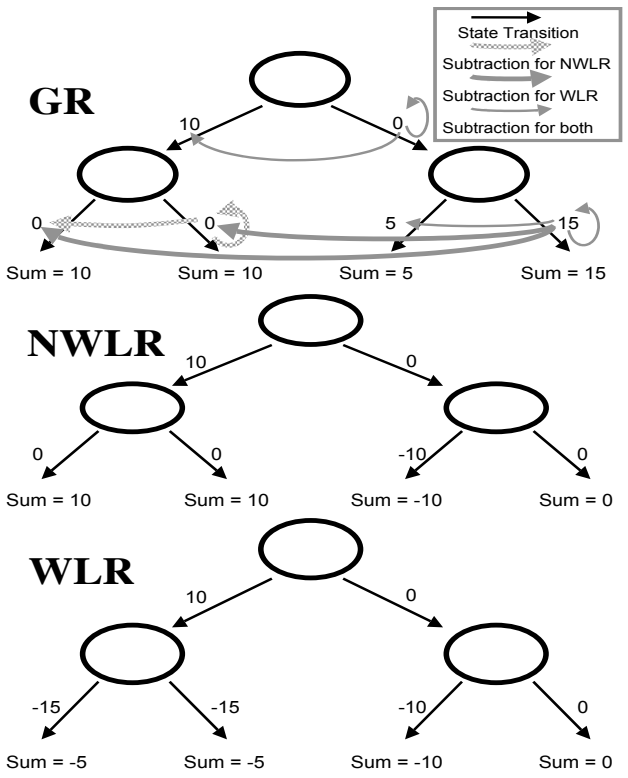

Figure 1. WLR sum ranks action sequences in same order as GR sum

Figure 1 shows that if agent i moves left on the first time step, the global reward will be ten on the first time step and zero on the second time step. If the agent moves right on the first time step the global reward will be zero for the first time step, but it will be either five or fifteen on the second time step depending on whether its second action is left or right respectively. When the actions of all of the other agents are held constant, if agent i is using a Sarsa learner with the global reward, it should form the policy of moving right for both time steps, maximizing the global utility (sum of global rewards). However consider the values produced by the NWLR, shown in Figure 1 (middle). If $c_{i,t}$ represents the go-right action, then the NWLR will equal GR, for the first time step, but will be different if it takes the go-right action on the second time step. Instead of five or fifteen, the NWLR will evaluate to negative ten or zero on the second time step. If the agent is using a Sarsa learner, we would expect the agent to take the left action at the first time step, which is sub-optimal with respect to the global utility.

The solution to this problem is to subtract out the full **sequence** of actions, z_i, instead of the single action, $z_{i,t}$, in the second term of the WLR and replace it with a constant full sequence of actions:

$$WLR^{c_i}_{i,t}(z) = GR_t(z) - GR_t(z - z_i + c_i) \qquad (3)$$

where c_i is a full sequence of actions. This version of WLR therefore differs from the previous version in that:

1. The subtracted single action $z_{i,t}$ is replaced with the *sequence* of actions z_i
2. The constant action $c_{i,t}$ is replaced with a constant *sequence* of actions c_i

For the example problem the values produced by this utility are shown in Figure 1 (bottom). In this case, c is set to the "right-right" sequence of actions (i.e., 15 is subtracted from GR). Note that now the sum of rewards for both GR and WLR have the same ordering, that is the two utilities are factored. With this utility an agent using a Sarsa learner forms the correct policy.

Figure 2 shows the effectiveness of using WLR on a multi-agent grid-world problem with eighty-five agents. In this problem, agents using the WL reward were compared to agents using the global reward, and to agents using a natural and selfish (non-factored) reward. Agents trying to maximize the WL reward significantly outperformed the other two sets of agents. The results confirm that the WL reward is highly learnability while being factored both through time and between agents.

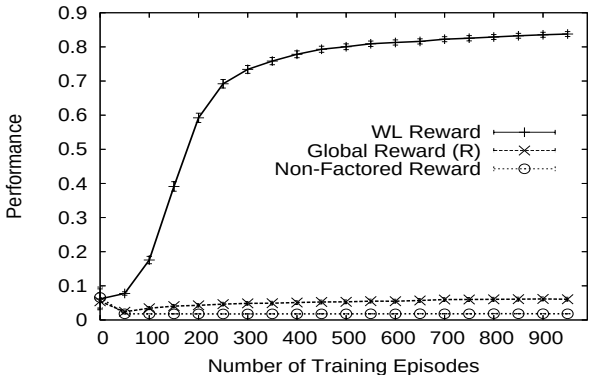

Figure 2. Agents perform better using WLR.

References

[1] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In Touretzky, Mozer, and Hasselmo, eds., *Advances in Neural Information Processing Systems - 8*, pages 1017–1023. MIT Press, 1996.

[2] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.