Multi-Objective Multiagent Credit Assignment Through Difference Rewards in Reinforcement Learning

Logan Yliniemi and Kagan Tumer

Oregon State University
Corvallis, Oregon, USA
logan.yliniemi@engr.orst.edu
kagan.tumer@oregonstate.edu

Abstract. Multiagent systems have had a powerful impact on the real world. Many of the systems it studies (air traffic, satellite coordination, rover exploration) are inherently multi-objective, but they are often treated as single-objective problems within the research. A very important concept within multiagent systems is that of credit assignment: clearly quantifying an individual agent's impact on the overall system performance. In this work we extend the concept of credit assignment into multi-objective problems, broadening the traditional multiagent learning framework to account for multiple objectives. We show in two domains that by leveraging established credit assignment principles in a multi-objective setting, we can improve performance by (i) increasing learning speed by up to 10x (ii) reducing sensitivity to unmodeled disturbances by up to 98.4% and (iii) producing solutions that dominate all solutions discovered by a traditional team-based credit assignment schema. Our results suggest that in a multiagent multi-objective problem, proper credit assignment is as important to performance as the choice of multi-objective algorithm.

1 Introduction

Cooperative multiagent systems focuses on producing a set of autonomous agents to achieve a system-level goal [12]. Multiagent frameworks have been used to study complex, real-world systems like air traffic [10], teams of satellites [3], and extra-planetary rover exploration [1]. In each case, the goal is to optimize a single, well-defined objective function.

But, in many of these cases, the problems lend themselves more naturally to multiple objectives: for example, air travel should be as safe and as expedient as possible. Satellites may need to make observations for multiple separate institutions. Extra-planetary rovers should acquire multiple different types of scientific data. However, most research in multiagent systems does not take a multi-objective viewpoint: they typically seek to find a single usable solution, without considering the tradeoffs between potential alternatives that would increase one objective's value at the cost of another. These tradeoff solutions, which form the *Pareto front*, are a key solution concept in multi-objective problems.

Developing successful agent policies in multiagent systems can be challenging. One successful approach is to use adaptive agents with tools like reinforcement learning.

Each agent seeks to maximize its own reward; with a properly designed reward signal, the whole system will attain desirable behaviors. This is the science of credit assignment: determining the contribution each agent had to the system as a whole. Clearly quantifying this contribution on a per-agent level is essential to multiagent learning. This is an issue that has not been studied within the context of multiple objectives. In this work we address the challenges that arise when multiagent systems are combined with multi-objective problems.

The primary contribution of this work is to develop the concept of credit assignment for multi-objective problems. This broadens the traditional multiagent learning framework to account for the multiple objectives present in many real world problems. This improves system-level performance by (i) increasing learning speed by up to 10x (ii) reducing sensitivity to unmodeled disturbances by up to 98.4% and (iii) producing solutions that dominate all solutions discovered by a traditional team-based credit assignment schema.

The remainder of this work is organized as follows: Section 2 describes the necessary background. Section 3 describes a stateless coordination domain, the multi-objective bar problem (MOBP), and presents the results in this domain. Section 4 describes a stateful, time-extended coordination domain, the collective transport domain (CTD), and presents the results in this domain. Finally, Sec. 5 draws the conclusion to this work and identifies future directions for this line of research.

2 Background

We limit the scope of this work to consider reinforcement learners using difference rewards as a feedback signal with "a priori" scalarization of objectives. This allows us to examine the performance of multi-objective difference rewards in two scenarios in which they have been shown to out-perform a global "team" reward in a single-objective case. "A posteriori" methods, such as multi-objective evolutionary algorithms, though more generally successful, are explicitly out of the scope of this work.

2.1 Multi-objective problems

In a multi-objective problem, there is typically not one "best" solution, but instead an array of optimal tradeoffs that are *incomparable*. For example, a man with 1 kilogram of bread and 1 kilogram of wine might be just as happy as a man with 0.9 kilograms of bread and 1.2 kilograms of wine. The two are incomparable [6]. However, both of these solutions are strictly better than a man with no bread and no wine, which is *dominated* by both of the others.

Non-dominated set (NDS) The NDS is the set of discovered feasible solutions that are not dominated by any other solution. The process for calculating the NDS is illustrated in Figure 1. Any optimizer or search will develop a set of non-dominated solutions; the globally best-possible NDS is known as the Pareto front, and the goal of any multi-objective approach is to develop an NDS that is a close approximation to the Pareto front.

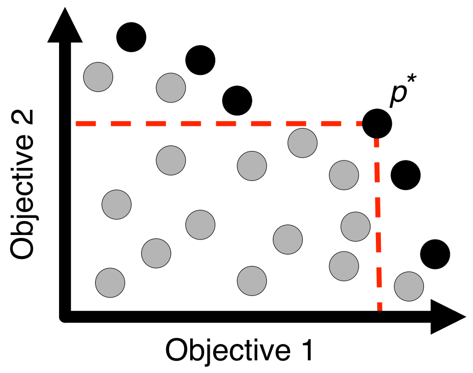

Fig. 1. Domination. The point p^* is a point in the NDS, and all points which score worse on all objectives than (below and to the left of) p^* are dominated by p^*. The three grey points not dominated by p^* are dominated by other (black) points in the NDS.

Scalarization of objectives Within the class of *a priori* methods for multi-objective problems, there are many different ways to scalarize the objectives into a single reward signal. In this work we examine two: a linear combination and a hypervolume calculation. In each case we normalize the objectives to the range [0:1] before combining them in one of two ways:

$$R_+ = \sum_{c \in C} w_c f_c^{\text{norm}} \quad \Big| \quad R_\lambda = \prod_{c \in C} f_c^{\text{norm}} \tag{1}$$

where R_+ is the linear combination reward delivered to the reinforcement learner, R_λ is the hypervolume reward delivered to the reinforcement learner, C is the set of all criteria or objectives, and f_c^{norm} is the normalized score on objective c. In each case we give all agents either R_+ or R_λ, but never any combination of the two. The form which f_c^{norm} takes varies depending on the credit assignment schema used, which is discussed in the following section. Other types of scalarizations do exist, like an exponentially weighted set of objectives or distance from a target point, but we limit the scope of this work to consider only these two.

2.2 Reinforcement learning

In this work we use a team of independent reinforcement learners (Action-value learners for the MOBP and Q-learners for the CTD [5, 9]), with the standard notation of a learning rate α, a discount factor γ, and a reward R.

2.3 Multiagent Credit Assignment Structures

In a multiagent system, it is important to reward an agent based on its contribution to the system. This is difficult due to the other agents acting in the environment, obscuring the agent's individual contribution to system performance. We consider three popular *credit assignment structures* for addressing these concerns.

A **local reward** (L_i) is the reward based on the part of the system that an agent i can directly observe. Using this reward signal often encourages "selfish" behavior, in which the agent may act at cross-purposes with other agents while blindly increasing its own reward, causing poor system performance.

The **global reward** (G) is the system performance used as a learning signal. This encourages the agent to act in the system's interest, but includes a large amount of noise from other agents acting simultaneously. An agent's own contribution to the global reward may be dwarfed by the contribution of hundreds of other agents, resulting in a low "signal to noise ratio" [11].

The **Difference reward** (D_i) is a shaped reward signal that helps an agent learn the consequences of its actions on the system objective by removing a large amount of the noise created by the actions of other agents active in the system [11]. It is defined as

$$D_i(z) = G(z) - G(z_{-i}) \tag{2}$$

where $G(z)$ is the global system performance for the system considering the joint state-action z, and $G(z_{-i})$ is $G(z)$ for a theoretical system without the contribution of agent i. Any action taken to increase D_i simultaneously increases G, while agent i's impact on its own reward is much higher than its relative impact on G [11].

3 Multiobjective bar problem (MOBP)

The first domain we consider in this work is an extension of the El Farol Bar Problem originally introduced by Arthur [2]. In this extension, a group of agents $\mathbf{A}$ are each assigned a static type m or f and must independently choose to attend one of several bars. There are multiple objectives: first, the agents wish to attend a bar that is not too crowded, and not too empty. Second, the agents wish to attend a bar with an even mixing of agents of type m and f.

The first "capacity" objective for each bar is modeled as a smooth curve that takes on a value of 0 with no agents attending, near 0 with many agents attending, and a maximum at the ideal capacity ψ. This models the enjoyment of the agents (of quantity x_b) attending bar b. The second "mixture" objective for each bar is maximized when $M_b = F_b$, where these are the number of agents of type m and f attending bar b, *regardless of the number of agents at the bar*. Formally:

$$L_{\text{cap}_b} = x_b \cdot e^{\frac{-x_b}{\psi}} \qquad \Bigg| \qquad L_{\text{mix}_b} = \frac{\min(M_b, F_b)}{(M_b + F_b)W} \tag{3}$$

where W is the number of bars available for the agents to choose from. L_{mix_b} evaluates to 0 if the agents are all of the same type, and $0.5/W$ if there is an equal mixture of types. The number of bars, W, is a constant (and therefore does not change the reinforcement learning process), and serves to limit G_{mix} to values in the range [0:0.5] for easier interpretation of results.

The global rewards for each of these objectives are simply the sum of the local rewards across all bars:

$$G_{\text{cap}} = \sum_{b \in B} L_{\text{cap}_b} \quad \Bigg| \quad G_{\text{mix}} = \sum_{b \in B} L_{\text{mix}_b} \tag{4}$$

And the Difference rewards for each are calculated by Equation 2 as the global reward minus the global reward in a fictional world had agent i never attended any of the bars:

$$D_{\text{cap}_i} = x_{\text{a}} \cdot e^{\frac{-x_{\text{a}}}{\psi}} - (x_{\text{a}} - 1) \cdot e^{\frac{-(x_{\text{a}}-1)}{\psi}} \tag{5}$$

$$D_{\text{mix}_i} = \begin{cases} \frac{\min(M_{\text{a}},F_{\text{a}})}{(M_{\text{a}}+F_{\text{a}})W} - \frac{\min((M_{\text{a}}-1),F_{\text{a}})}{(M_{\text{a}}+F_{\text{a}}-1)W} & : i \in m \\ \\ \frac{\min(M_{\text{a}},F_{\text{a}})}{(M_{\text{a}}+F_{\text{a}})W} - \frac{\min(M_{\text{a}},(F_{\text{a}}-1))}{(M_{\text{a}}+F_{\text{a}}-1)W} & : i \in f \end{cases} \tag{6}$$

where x_{a} is the attendance in the bar attended by agent i, and M_{a} and F_{a} are the number of agents of types m or f respectively that attended the same bar as agent i. D_{mix_i} depends on the type of the agent; the second term represents the system with agent i removed from bar b.

Procedure The procedure for running the MOBP is simple. Each agent simultaneously selects a bar to attend based on no sensory information. The local rewards L_{cap_b} and L_{mix_b} are calculated for each bar b. Then the global rewards G_{cap} and G_{mix} are calculated. Finally, D_{cap_i} and D_{mix_i} are calculated for each agent i. Once these are calculated, the selected reward type (local, global, or difference) is normalized and put through Equation 1 depending on the desired scalarization. The result is then provided to the agent as the reward R, calculated with a value of $\gamma = 0$, because the problem is only a single step.

Tradeoffs and independence of objectives We take measures to prevent a trivial solution for either objective, or a single dominating solution:

– G_{cap}: There are many more agents (100) than capacity across all bars (a capacity of 5 for 7 bars).
– G_{mix}: Agent types are 70% type m, 30% type f.
– *Tradeoff*: L_{cap_i} is maximized at 5 agents; L_{mix_i} is maximized only when an even number of agents attend a bar.
– *Tradeoff*: A maximum G_{mix} case involves many bars with one agent of each type and the rest attending a single bar, which conflicts with G_{cap}.

We calculate the coefficient of determination (R^2) value for the correlation between the two objectives across 10^6 random Monte Carlo trials using a linear, exponential, and polynomial fit. The maximum value was the linear fit at 0.0034, which reinforces that the objectives are distinct, though they are coupled through the actions of the agents.

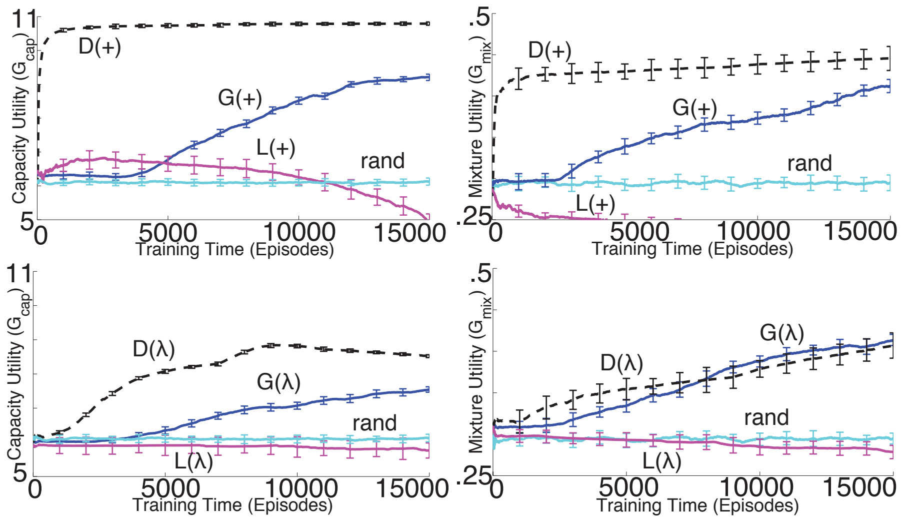

Fig. 2. Performance on G_{cap} (left) and G_{mix} (right), for agents trained on the linear scalarization (+,top) and hypervolume calculation (λ,bottom) of the three reward structures (D,G,L) and the random baseline (rand). Each of these objectives is to be maximized.

3.1 MOBP results

To exhibit the benefits of Difference rewards in multi-objective problems, we examine 4 types of results:

- Average system performance on both system objectives (Figure 2)
- Dominance and NDS (Figure 3)
- Impact of training time (Figure 3)
- Robustness to disturbances (Figure 4)

Simulation information We execute 30 statistical runs of the MOBP for seven independent experiments: training all agents on each structure-scalarization combination in turn (D+, Dλ, G+, Gλ, L+, Lλ), and on a random policy (rand).

Each agent selects an action using an ϵ-greedy mechanism, with an initial $\epsilon = 0.05$ for local and difference rewards, and $\epsilon = 0.1$ for global rewards[1] (both multiplied by a factor of 0.999 every episode to reduce exploration), with a learning rate $\alpha = 0.10$.

We performed a full sweep through w_c values, but due to the large effect each agent has on the overall system performance near the Pareto front, we found that an even weight, combined with the natural exploration, resulted in a spread of solutions discovered along the Pareto front.

In Fig. 2, a 100-episode moving average (across 30 statistical runs) of system performance was used. Error bars report the error in the mean, calculated as $\frac{\sigma}{\sqrt{N}}$, where N

[1] These values were chosen through a parameter sweep to create the best performance for each reward, though the results are not very sensitive to ϵ or α values.

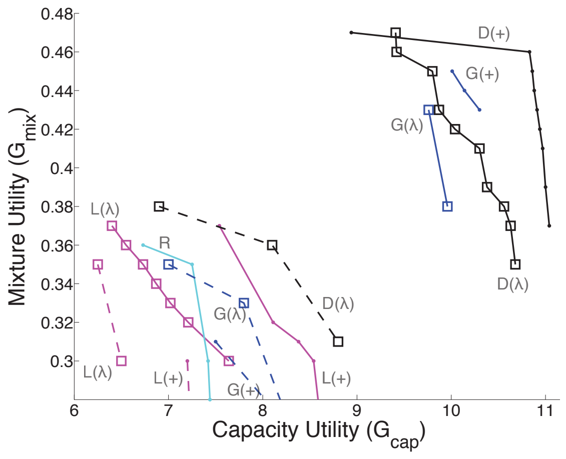

Fig. 3. The set of non-dominated episodes created over the entire training process through using hypervolume (λ) or a linear combination ($+$); dotted lines show the NDS after 1500 learning episodes; solid lines, the NDS after 15,000 episodes. Agents trained on D(+) peak in performance before 1500 episodes, so both D(+) NDSs are identical.

is the number of statistical trials. We identify the NDS for each structure-scalarization combination (e.g. "Hypervolume of Global Reward", Gλ), across all 30 statistical runs, and aggregate these into a single non-dominated set, for clarity [4].

3.2 Average performance on system objectives

It is informative to look at the performance of the system on each objective individually (Figure 2), as this performance drives the behavior of the non-dominated set. For both the linear combination and hypervolume scalarization, the local reward (L+,Lλ) performs poorly; the agents work at cross-purposes, undermining each other's efforts by all trying to attend low-attendance days. This leads to low performance, and will never lead to good system behavior, even with an extreme amount of training time. The global reward (G+,Gλ) does learn, slowly. For the hypervolume scalarization, Dλ increases system performance at a slightly higher rate than Gλ. The linear combination of difference rewards, D+, performs at a very high level very quickly, and reaches near its final performance after only 1500 episodes.

3.3 Non-dominated sets (NDS) and training time

In addition to performing well on the individual objectives, solutions produced by D+ or Dλ produce superior NDS compared to the global and local rewards with the same scalarization. The NDS are shown in Figure 3. In fact, every solution produced by the local or global rewards is dominated by a solution produced by the difference reward.

The dotted lines in Figure 3 represent the NDS produced in the first 1500 episodes (10% of the training). In all cases the NDS improve between 1500 and 15000 episodes,

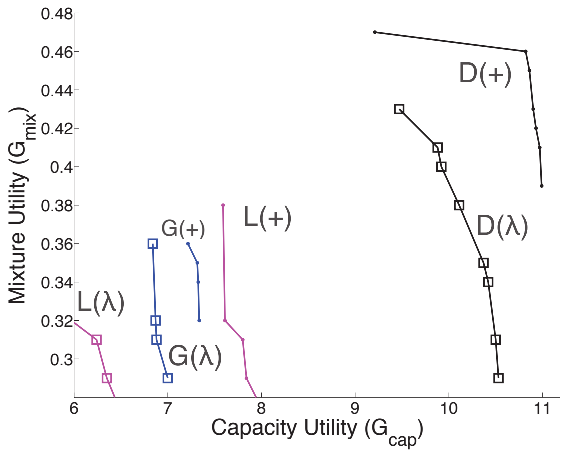

Fig. 4. The NDS produced in the 5000 training episodes after the conversion of 20% of agents to "selfish" behavior. Compare with solid lines in Figure 3: D+ and Dλ recover well; G$+$ and Gλ suffer a disastrous drop in performance.

except D+, which has already produced its best episodes (dominating all other credit assignment/scalarization combinations). Solutions produced by Dλ dominate the solutions produced by other methods in the same time, except D+.

3.4 Robustness to disturbances

To model outside disturbances, after 15000 episodes 14 agents of type m and 6 of type f "fail". They have their Q-table reset to zero values and continue learning using the local reward policy regardless of the learning signal they were using previously (acting selfishly). The remaining 80 agents continue learning using the same signal they were using previously. Additional exploration was found to be necessary in this case, so we reset ϵ to initial values. All agents continue the learning process as before.

Figure 4 shows that D+ maintains its dominant NDS. G+ and Gλ are affected catastrophically by the selfish agents, while D+ loses 98.4% less dominated hypervolume. Dλ only loses performance on G_{mix}.

4 Collective transport domain

We additionally performed experiments in a collective transport domain, modeled after [7], in which a team of small robots must cooperate to transport an item (which we also refer to as a body or load) across a surface in much the same way that ants transport objects.

We formulate this as a time-extended, stateful reinforcement learning problem in which the robot agents try to (i) collectively transport the object as quickly as possible to the goal, while (ii) expending minimum effort.

Each robot is given discretized state information about the load's position and velocity, and is allowed to take one of nine actions, applying a force to the load in a cardinal direction (N,S,E,W), an intermediate direction (NE,SE,SW,NW), or no force. The robots are assumed to be attached to the object, and receive discretized state information based on the object's current location and speed.

The body's acceleration (acc), velocity (vel), position (pos) at time t are found with particle kinetics:

$$acc(t) = \sum_{i \in \mathbf{A}} [F_{\text{x}}(i)\hat{\imath}] + \sum_{i \in \mathbf{A}} [F_{\text{y}}(i)\hat{\jmath}] - F_f \hat{f} \tag{7}$$

$$vel(t) = vel(t-1) + acc(t) \cdot t_{\text{step}} \tag{8}$$

$$pos(t) = pos(t-1) + vel(t) \cdot t_{\text{step}} \tag{9}$$

where $\mathbf{A}$ is the set of all agents, $F_{\text{x}}(i)$ is the force applied by agent i in the $\hat{\imath}$ direction, $F_{\text{y}}(i)$ is the force applied by agent i in the $\hat{\jmath}$ direction, and F_f is the force of friction, which acts in the $\hat{f}$ direction, which points opposite the direction of motion of the body. We omit mass from this calculation of Newton's second law because we assume the mass of the body and transporting robots to be 1 unit total. In this context a local reward loses some meaning as all agents are collectively acting to move the same object, so we only look at global and difference reward in this case.

The first objective (proximity) is to move the load close to the goal as quickly as possible. This takes the form:

$$G_{\text{prox}}(t) = -Tdist(t) \tag{10}$$

$$D_{\text{prox}_i}(t) = -Tdist(t) + Tdist_{-i}(t) \tag{11}$$

where $Tdist()$ is a function that returns the body's Euclidian distance from the target at time t, and $Tdist_{-i}()$ returns the distance from the target if agent i took no action during timestep t.

The second objective is to minimize the effort exerted by the team to move the load to the desired target location:

$$G_{\text{effort}}(t) = \sum_{i \in \mathbf{A}} [1 - E_{i,t}] \tag{12}$$

$$D_{\text{effort}_i}(t) = 1 - E_{i,t} \tag{13}$$

where $E_{i,t}$ is 1 if the agent applied a force to the object at time t, and 0 if the agent did not apply a force.

We perform a Q-update at every time step. To visualize the performance, we aggregate these into one point for each time the load reaches the goal state. For the purpose of learning, however, we use the distance to the goal state after each time step, as this provides a smoother gradient for learning [5]. The process for conducting this experiment is described in Algorithm 1. For each credit assignment schema and scalarization combination, step 19 would use the proper evaluation (one of L_i, G, or D_i), and use the desired scalarization from Equation 1.

In this domain the two objectives are in conflict with one another: minimizing the time to deliver the load will maximize the effort required, and minimizing effort will lead to a longer time.

4.1 CTD results

In the collective transport domain, we examine two types of results:

- Dominance and NDS (Figure 5, Left)
- Impact of training time (Figure 5, Right)

Simulation Information We perform 4 different trials following Algorithm 1; one each for G+, Gλ, D+, and Dλ. For each, we conduct 30 statistical runs of 5000 time steps for teams of 50 agents attempting to transport a load across a surface with maximum static force of friction $F_f = 8$ units and kinetic force of friction of $F_f = 2$ units. The body's starting state is initialized as $(x, y) = (1, 1)$, with the goal as a square at $\{x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\} = \{900, 1000, 900, 1000\}$. The boundaries are a larger square at $\{x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\} = \{0, 1000, 0, 1000\}$. Though the calculations of the body's velocity and position are continuous, we us an approximation via tile coding [9] and discretize into 10 states each for $(x_{\text{vel}}, y_{\text{vel}}, x_{\text{pos}}, y_{\text{pos}})$ creating 10,000 states. In this

Algorithm 1 Collective Transport Domain using Difference Reward of Dominated Hypervolume (Dλ)

```
 1: initialize Q-values to zero: $Q(s, a) = 0 \; \forall \; s, a$
 2: initialize body position to starting location
 3: initialize velocity and acceleration to **0**.
 4: **for** $timestep = 1 \rightarrow max\_timesteps$ **do**
 5:     **for** $i = 1 \rightarrow total\_agents$ **do**
 6:         choose an action to take with $\epsilon$-greedy action selection: {none,N,NE,E,SE,S,SW,W,NW}
 7:         add force contribution to body ($F_x(i)$,$F_y(i)$)
 8:     **end for**
 9:     evaluate body acceleration (Equation 7)
10:     evaluate body velocity (Equation 8)
11:     evaluate body position (Equation 9)
12:     **if** body position is out of bounds **then**
13:         set body position to nearest in-bounds position
14:         set body velocity to **0**
15:     **end if**
16:     evaluate global reward (Equations 10, 12)
17:     **for** $i = 1 \rightarrow total\_agents$ **do**
18:         evaluate difference rewards (Equations 11, 13)
19:         evaluate $R \leftarrow R_\lambda$ (Equation 1)
20:         update $Q(s, a)$ values
21:     **end for**
22:     **if** body is in goal state **then**
23:         set body to starting location
24:         set velocity and acceleration to **0**.
25:     **end if**
26:     reduce $\epsilon$
27: **end for**
```

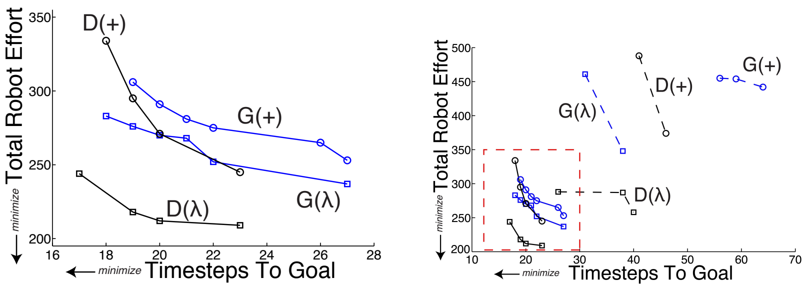

Fig. 5. (Left) Collective Transport Domain results. D(λ) creates solutions that dominate all other methods (solutions below and to the left are superior in this domain). D+ outperforms G+, and creates an overlapping Pareto front with G(λ). (Right) Dotted lines denote early system performance after 500 time steps. The denoted highlighted area is the range of the figure on the left.

domain, we find the best performance when we vary the weights for the objectives as a function of learning step, starting by with a value of $\{w_{\text{prox}}, w_{\text{effort}}\} = \{1, 0\}$ changing linearly to $\{0, 1\}$ at the final learning step. This produces policies which do find the goal state, and learn to reduce effort over time. This produces better initial performance and a spread of solutions along the NDS. Initial weights favoring the effort objective led to policies of inaction, never reaching the goal.

4.2 Dominance and NDS

Figure 5 shows the final NDS for each method. The teams of agents trained on the scalarizations of the difference reward (D+, Dλ) outperform their global counterparts in the final produced NDS. In this domain, however, the hypervolume calculations (λ) perform better than the linear combinations ($+$). We find nearly equivalent performance between Gλ and D+, suggesting that using the proper multi-objective scalarization is as important as proper multiagent credit assignment. The Dλ result shows that these benefits can be symbiotic.

4.3 Impact of training time

We also identify the NDS produced by each solution after 10% of the training time in Figure 5. Again, the difference reward using the preferable scalarization attains performance close to its final performance very quickly, while the global methods are not as near their final performance values. Dλ dominates all solutions formed by other scalarizations.

Additionally, in this domain we noticed that the performance of the global reward signals was sensitive to the learning parameters, while the difference reward signals were robust to these changes. In additional trials we found the agents trained with the difference reward to be robust to noisy actuators, noisy sensors, failing agents, and unmodeled disturbances (externally applied forces) as well.

5 Conclusion

Multiagent systems are a powerful concept for dealing with complex systems. Many multiagent systems are intrinsically multi-objective, but this has received scant attention. In this work we explicitly addressed one of the key concerns in multiagent systems — credit assignment — under the conditions of a multi-objective problem. We found that credit assignment is important under multi-objective conditions: our results show (i) a 10x increase in learning speed, (ii) a 98.4% increase in robustness to unmodeled disturbances, and (iii) the production of solutions which dominate all solutions found by a traditional global reward. These results show that proper credit assignment is of paramount importance in a multiagent multi-objective system. However, the choice of multi-objective algorithm is still extremely important. Difference rewards boosted performance in both domains, for both scalarizations. The gains from credit assignment through difference rewards were independent of the scalarization used and the domain.

Difference rewards are not limited to reinforcement learning or a priori methods, however. Future work on this topic includes an examination of the effects that credit assignment can have on multiagent implementations of well-established *a posteriori* multi-objective evolutionary algorithms.

6 Acknowledgements

This work was partially supported by the National Energy Technology Laboratory under grants DE-FE0011403 and DE-FE0012302.

References

1. A. K. Agogino and K. Tumer. Analyzing and visualizing multi-agent rewards in dynamic and stochastic domains. *JAAMAS*, 17(2):320–338, 2008.
2. W. B. Arthur. Inductive reasoning and bounded rationality (the El Farol Problem). *American Economic Review*, 84(406), 1994.
3. S. Damiani, G. Verfaillie, and M.-C. Charmeau. An earth watching satellite constellation: How to manage a team of watching agents with limited communications. *AAMAS*, 2005.
4. C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. *Lecture Notes in Computer Science*, 1141:584–593, 1996.
5. L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.
6. Vilfredo Pareto. *Manual of Political Economy*. MacMillan Press Ltd., 1927.
7. M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal. Collective transport of complex objects by simple robots: Theory and experiments. *AAMAS*, 2013.
8. A. A. Sherstov and P. Stone. Function approximation via tile coding: Automating parameter choice function approximation via tile coding: Automating parameter choice. *SARA*, 2005.
9. R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
10. C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509 – 521, APRIL 1998.
11. D. H. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6), 2000.
12. Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, 2002.