Improving Multi-Agent Learners Using Less-biased Value Estimators

Sherief Abdallah
Faculty of Eng. and IT, British University in Dubai, U.A.E
School of Informatics, University of Edinburgh, U.K.
Email: shario@ieee.org

Michael Kaisers
Centrum Wiskunde & Informatica
Science Park 123, 1098 XG Amsterdam, The Netherlands
Email: kaisers@cwi.nl

***Abstract*—Many different value-based or policy-search reinforcement learning algorithms have been applied to multi-agent settings. Value-based learners estimate the expected return (value) for each state-action combination and then derive a policy from these expectations. Policy-search learners optimize the agent's policy directly by using a parameterized representation of the policy and then optimizing the parameter values to maximize the expected return. While the two classes of algorithms have been considered as contrasting one another, we note that several policy-search algorithms (e.g., Weighted Policy Learner and Infinitesimal Gradient Ascent) need a method for estimating the expected returns. In practice, these policy-search algorithms internally use an update equation for incrementally improving value estimates. In this paper we present the first detailed study of the effect of using different value-based learning algorithms as components of policy-search learners. Our results show that the particular choice can significantly affect performance.**

I. Introduction

Using reinforcement learning algorithms to optimize decisions in multi-agent systems has been an active research direction for quite some time now [5], [4], [15], [16], [2], [21], [1]. Typically, algorithms can be cast into one of two classes: value-based learners and policy-search learners. Value-based learners estimate the expected return (value) for each state-action combination and then derive a policy from these expectations. Q-Learning (QL) is perhaps the most widely-used and widely-studied value-based learning algorithm due to its ease of implementation, intuitiveness, and effectiveness over a wide range of problems [19], [18]. Although Q-learning was originally designed for single-agent domains, Q-learning and its variants have also been reasonably successful in multi-agent settings [5], [11], [1]. Policy-search learners optimize the agent's policy directly by using a parameterized representation of the policy and then optimizing the parameter values to maximize the expected return. Some of these policy search learning algorithms have been invented specifically for multi-agent applications [2], [3], [6], [21].

While the two classes of algorithms have been considered as contrasting to one another [12], it is interesting to note that most Policy-search learning algorithms need a method for estimating the expected returns for the different actions at different states [4], [6], [2], [21]. In practice, most of the policy-search algorithms used (or recommend the use of) the basic Q-learning algorithm to estimate expected returns.

An important question that we raise and answer in this paper is: does the value-based learning algorithm, which is used internally as a component of a policy-search algorithm, significantly affect the overall performance? Despite its popularity, the common Q-learning is subject to several limitations that have been addressed in more recent variants, including Frequency Adjusted Q-learning (FAQL) [11] and Repeated Update Q-learning (RUQL) [1]. In this paper we present the first detailed study of the effect of integrating different value-based learning algorithms within policy-search learners, and we show that the particular choice can significantly affect performance. In particular, we investigate Weighted Policy Learner (WPL) [2] and Policy Gradient Ascent with Approximate Policy Prediction (PGA-APP) [21] as case studies. Our experiments show that integrating the advanced value-based learning algorithms results in significant improvement in performance in multi-agent settings (in terms of minimizing the regret).

In summary, the main contributions of this paper are:

- Proposing the integration of advanced value-based learners within multi-agent policy-gradient learners.
- The first experimental analysis of the effect of the underlying value-based learners when used as internal components of policy-search learners, in a multi-agent setting.

The paper is organized as follows. The next section provides the necessary background, including different policy search and value-based learning algorithms. This is then followed by our proposal to integrate both classes of learners. We then present the experimental results that compare the performance of 6 combinations of policy search and value based learners. This is followed by a discussion of related work.

II. Background

Reinforcement learning, in its most basic form, attempts to find the optimal solution (policy) for a given Markov Decision Process (MDP). An MDP is defined by the tuple $\langle S, A, P, R\rangle$, where S is the set of states representing the system and $A(s)$ is the set of actions available to the agent at a given state s. The function $P(s, a, s')$ is the transition probability function and quantifies the probability of reaching state s' after executing action a at state s. Finally the reward function, $R(s, a, s')$,[1] gives the average reward an agent acquires if the agent executes action a at state s and reaches state s'.

[1]Other variations of the reward function definition exist.

The Q-learning algorithm [19] is a model-free reinforcement learning algorithm that is guaranteed to find the optimal policy of an MDP. The Q-learning algorithm relies primarily on the following update equation:

$$Q^{t+1}(s,a) = Q^t(s,a) + \alpha \left(r + \gamma \max_{a'} Q^t(s',a') - Q^t(s,a) \right) \tag{1}$$

The function $Q^t(s,a)$ represents the current best estimate at time t of the expected discounted sum of future rewards, which the agent believes it would get if it executes action a at state s and following its policy subsequently (or intuitively, what the agent believes, at time t, to be the worth of action a at state s). The parameters α and γ are tunable learning parameters (α is called the learning rate and γ is called the discount factor). The variables r and s' refer to the (sample) immediate reward and the next state, both of which are observed after executing action a at state s. Q-learning does not require knowing the underlying transition probability function of the MDP model.

Ideally, the agent should gain information about the value of each action at every state in order to become more certain over time that the action with the highest estimate truly is the optimal action. However, an agent can only execute one action at a time and the agent only receives feedback (reward) for the action that was actually executed. As a result, in Q-learning, the rate of updating an action relies on the probability of choosing that action. In other words, Q-learning (and to a lesser extent Sarsa) has *policy-bias*. The theoretical comparison of updating one vs. all actions at each time step reveals that Q-learning counter-intuitively decreases the probability of optimal actions under some circumstances, which leads to drawbacks in non-stationary environments [11], [20], [1].

Scaling the learning rate inversely proportional to the policy has been proposed to overcome this limitation, thereby approximating the simultaneous updating of all actions every time a state is visited. This concept has been initially studied to modify fictitious play [8], and inspired several modifications of Q-learning named Individual Q-learning [13], Frequency Adjusted Q-learning (FAQL) [11], and Repeated Update Q-learning (RUQL) [1]. The FAQL algorithm uses the following update equation:

$$\begin{aligned} Q^{t+1}(s,a) &= Q^t(s,a) \\ &+ \min(1, \frac{\beta}{\pi(s,a)})\alpha\Big(r + \gamma \max_{a'} Q^t(s',a') - Q^t(s,a)\Big) \end{aligned} \tag{2}$$

where β is a tuning parameter that safeguards against the cases where $\pi(s,a)$ is close to zero. The function $\pi(s,a)$ returns the probability of choosing action a at state s. The RUQL algorithm, on the other hand, uses the following update equation:

$$\begin{aligned} Q^{t+1}(s,a) &= [1-\alpha]^{\frac{1}{\pi(s,a)}} Q^t(s,a) \\ &+ \left[1 - (1-\alpha)^{\frac{1}{\pi(s,a)}}\right] [r + \gamma \max_{a'} Q^t(s',a')] \end{aligned} \tag{3}$$

Q-learning and its variants (including RUQL and FAQL) attempt to estimate the expected return (value) for each state-action combination and then derive a policy from the expected return. There is another class of algorithms that optimize the agent's policy directly. These algorithms use a parameterized representation of agent policy and optimize parameters values to maximize the expected return. Among the most successful of this class of algorithms are gradient-ascent-based (GAB) learners. Such algorithms have shown great success in challenging multi-agent settings where Q-learning and similar algorithms fail [2], [3], [21]. Two recent algorithms that belong to this class are Weighted Policy Learner (WPL) [2] and Policy Gradient Ascent with approximate policy prediction (PGA-APP) [21]. The WPL algorithm uses the following update rule:

$$\Delta\pi_i(a) \leftarrow \frac{\partial V_i(\pi)}{\partial \pi_i(a)} \cdot \eta \cdot \begin{cases} \pi_i(a) & \text{if } \frac{\partial V_i(\pi)}{\partial \pi_i(a)} < 0 \\ 1 - \pi_i(a) & \text{otherwise} \end{cases}$$

$$\pi_i \leftarrow projection(\pi_i + \Delta\pi_i)$$

The term $\frac{\partial V_i(\pi)}{\partial \pi_i(a)}$ refers to the partial derivative of the value function w.r.t to the current policy. In other words, how the total (discounted) expected reward will change w.r.t to a change in the policy (the probability of choosing actions). The term η is a learning parameter that controls the rate of learning. The function $projection(x)$ projects a point x to the closest valid policy [22]. The PGA-APP has more complex policy update rule that we will not present here, but interested readers can refer to the original paper [21].

III. Using Less-Biased Value-based Learners Within Policy-search Learners

While the two classes of algorithms, the algorithms that search for values and the ones that search for policies, have been considered as contrasting to one another [12], it is interesting to note that a particular class of policy-search learners rely internally on value-based learners. The class of gradient-ascent based (GAB) learning algorithms, which have been used extensively in multi-agent learning setting [4], [22], [2], [21], need a method for estimating the expected returns. Most of these algorithms, in practice, use Q-learning or a similar algorithm to estimate the expected returns for each action. For example, both of WPL and PGA-APP algorithms rely on the Q-learning update rule in their evaluation [2], [21] to estimate the expected returns of different state-action pairs, which is then used to estimate the gradient and update the policy. We formalize the relationship between value-based learners and GAB learners in Algorithm 1.

To our knowledge, the effect of the algorithm that is used to estimate action values *inside* GAB learners has not been studied yet. Researchers usually overlooked this detail. We

Algorithm 1: Integrated Gradient-Ascent Based Learners and Value-Based Learners

```
1  begin
2      Initialize functions $Q$ and policy $\pi$ arbitrarily.
3      Observe the current state $s$.
4      repeat
5          Choose an action $a$ according to the policy $\pi$.
6          Execute action $a$ (and observe the resulting
           reward $r$ and the next state $s'$).
7          Update $Q(s,a)$ according to Value-Based
           learner.
8          Update $\pi(s)$ according to Policy-Search learners
9      until done
10 end
```

hypothesize that using value-based learners with less policy-bias is more suitable for multi-agent domains. The reason for the improvement is that the value-based learners with less-bias toward the execution policy (such as RUQL and FAQL) are more effective (than QL) in estimating the values regardless of current underlying policy (which is controlled by the policy-search algorithm). Our experimental results verify our hypothesis and show significant improvement in performance.

IV. Experimental Analysis

In this section we evaluate the benefit of incorporating RUQL and FAQL into gradient-ascent-based (GAB) MARL algorithms. Since the strength of GAB-MARL algorithms appears in games with mixed Nash Equilibria (NE), we focus on games with this property. Table I lists three games that have been used in the literature as benchmarks [2], [3], [4], [21]. The matching-pennies game is the most well-known game and has mixed NE=[$(\frac{1}{2},\frac{1}{2})$ for the row player & $(\frac{1}{2},\frac{1}{2})$ for the column player], or NE=[$(\frac{1}{2},\frac{1}{2})_r$ & $(\frac{1}{2},\frac{1}{2})_c$] for short. The tricky game also has one mixed NE=[$(\frac{1}{2},\frac{1}{2})_r$ & $(\frac{1}{2},\frac{1}{2})_c$] but was shown to be more challenging for some multi-agent learning algorithms [2]. The biased game exposes the bias toward or against uniform policy by having mixed NE=[$(0.15, 0.85)_r$ & $(0.85, 0.15)_c$] (while other games have mixed NE that is uniform).

TABLE I. Benchmark games used in evaluating multi-agent learning algorithms. Both *matching-pennies* and *tricky* games have one mixed Nash Equilibrium (NE), where all actions are played with equal probability, i.e., NE=[$(\frac{1}{2},\frac{1}{2})_r$ & $(\frac{1}{2},\frac{1}{2})_c$], while the biased game yields NE=[$(0.15, 0.85)_r$ & $(0.85, 0.15)_c$].

(a) biased

	a1	a2
a1	1.0,1.85	1.85,1.0
a2	1.15,1.0	1.00,1.15

(b) matching pennies

	H	T
H	1,-1	-1,1
T	-1,1	1,-1

(c) tricky

	a1	a2
a1	0,3	3,2
a2	1,0	2,1

A. Performance Metric

Using average payoff to gauge performance can be ineffective, e.g., in the matching-pennies game the average payoff for strategies that cycle around the Nash equilibrium is zero. A better performance metric in competitive games is *regret* [3]. The term regret refers to the difference between the maximum payoff an agent could get, and the payoff an agent actually achieved using its current (learned) policy. The maximum payoff an agent could get at time t can be defined as the payoff of the deterministic best reply, *knowing the opponent policy at time t*. In other words, we define the (expected) regret for a particular algorithm ALG at time t to be:

$$regret^t(ALG) = \sum_{i \in agents} \Bigg[\max_{x \in \Pi_i} \text{payoff}_i(x, \pi^t_{-i}) - \text{payoff}_i(\pi^t_i(ALG), \pi^t_{-i}) \Bigg],$$

where $\text{payoff}_i(\pi_i, \pi_{-i})$ is the payoff agent i would get given its policy π_i and the opponent policy π_{-i}, the term $\pi^t_i(ALG)$ refers to the policy agent i has learned at time t using algorithm ALG, and Π_i is the set of possible deterministic policies for agent i. Since regret can vary over time, we are actually interested in the *accumulated regret*:

$$R^t(ALG) = \textstyle\sum_{i=0}^{t} regret^i(ALG)$$

And since we are mainly interested in measuring the improvement in performance if we switch from the traditional Q-learning (QL) to RUQ-learning and FAQ-learning, we define the Regret Reduction (RR) as the reduction in regret due to the switch:

$$RR^t(ALG) = 100 \times \frac{R^t(QL) - R^t(ALG)}{R^t(QL)}$$

Note that RR provides the percentage of reduction in regret (normalized w.r.t. the original Q-learning regret). We define the final RR, $RR(ALG) = RR^T(ALG)$, where T is the total simulation time. In other words, our goal of *minimizing* expected regret R is equivalent to *maximizing* regret reduction RR.

B. Self-Play Results

In self-play, the games are played with both agents using the same value estimator and policy-search algorithm. Table II shows the average RR over 10 simulation runs and 4 different initial joint policies corresponding to the four combinations near the policy boundary[2] (total of 40 samples) for the benchmark games in Table I and the two MARL algorithms WPL [2] and PGA-APP [21]. The learning rate α of both QL and RUQL was set to 0.01 while it was set to $\sqrt{0.01} = 0.1$ for FAQL. The policy learning rate η was set to 0.00005. The simulation was run for 2,000,000 time steps for each trial.

Using RUQL and FAQL results in significant improvement over the use of traditional QL, as shown in Table II. All entries in the table are positive (higher is better) and statistically significant w.r.t. Q-learning, at p-value of 0.05. Furthermore, RUQL consistently outperforms FAQL, indicated by higher RR in all the entries, and statistically significant results in two thirds of the entries (shown in bold).[3]

[2]The Initial Q values were set to either 0 (corresponding to which action is initially selected with probability 0.0001) or 5 (corresponding to which action is initially selected with probability 0.9999).

[3]We have used paired-two-sample t-tests (the same random seed and initial policy were used for each pair of samples).

TABLE II. The average and standard deviation of RR for both FAQL and RUQL when used instead of Q-learning within GAB algorithms. All the RR entries are statistically significant w.r.t. Q-learning. When comparing RUQL against FAQL, the results were statistically significant only in the cases shown in bold text; for the remaining cases no statistically significant results could be obtained in either direction. RUQL consistently achieved positive RR (better than QL), and also outperformed FAQL.

Algorithm	WPL	PGA-APP
Game		
Tricky		
FAQL	24.53 ± 7.57	18.74 ± 3.04
RUQL	**27.22** ± 8.57	18.83 ± 2.95
Matching Pennies		
FAQL	13.11 ± 2.66	24.76 ± 2.71
RUQL	**14.65** ± 2.77	24.82 ± 2.69
Biased		
FAQL	22.08 ± 3.84	3.64 ± 1.37
RUQL	**26.89** ± 7.25	**3.85** ± 1.21

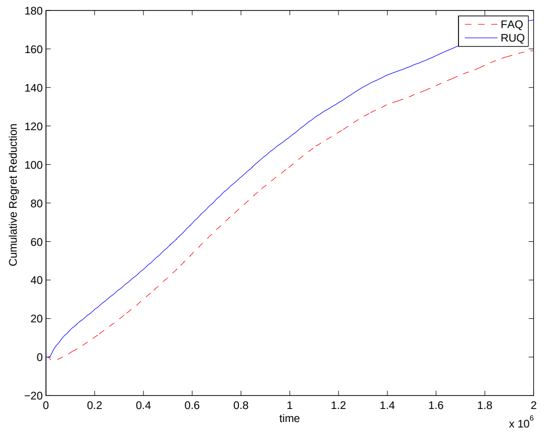

Fig. 1. The cumulative reduction in regret for the WPL algorithm in the matching-pennies game. RUQL and FAQL achieve significant improvements, compared to the original WPL using Q-learning, with RUQL slightly surpassing FAQL.

To gain more insight into the results presented in Table II, Figure 1 and Figure 2 show the evolution of regret reduction for some of the table entries, along with the corresponding policies that the agents learn. Figure 1 shows the accumulated difference in regret between QL and RUQL (FAQL), for the WPL algorithm [2] in the matching pennies game (higher is better). Note teh figure plots the numerator of the RR equation (without the normalization, unlike the values in Table II which gives RR). The cumulative regret reduction is monotonously increasing for both RUQL and FAQL. Furthermore, there is a gap between RUQL and FAQL. The gap widens in the beginning and then remains almost constant. This means that RUQL's edge over FAQL is during the early transitional state of learning. This is consistent with previous theoretical analysis [1]: because both algorithms start from an almost deterministic initial policy (0.0001 and 0.9999), it is harder for FAQL to adapt quickly due to the β-limitation ([1]) in the initial stage. While the relative performance of WPL using RUQL vs WPL using FAQL may be affected by the initial setting, it is clear that using either RUQL or FAQL clearly outperforms the use of traditional QL.

Figure 2 plots the policy learned using WPL in combination with each of the three algorithms: QL, FAQL, and RUQL. QL consistently overshoots, because of its policy-bias limitation. This causes the continuous increase in RR(RUQL) that is observed in Figure 1. On the other hand, the asymptotic performance is very similar for both of RUQL and FAQL in this case, which explains the fixed gap in RR between RUQL and FAQL.

C. Mixed-Play Results

The results presented so far assumed self-play. That is, both agents playing the game are using exactly the same algorithm. The next experiment investigates the performance when players are using two different learning algorithms: one agent using gradient ascent based WPL with the original QL while the other agent uses WPL with RUQL. Table III shows the results when one agent uses QL for estimating the values while the other uses RUQL. It answers whether an improvement in RR (compared to the pure QL case) is observed, and if such an

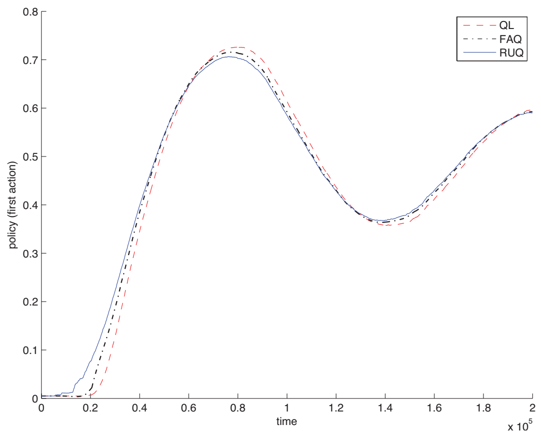

Fig. 2. The policy (probability of choosing the first action) for the matching pennies game using WPL in combination with each of QL, FAQL, and RUQL. The NE for this game is to play all actions for both players with probability 0.5. We can see that RUQL is always a step ahead of QL, which results in a widening gap in regret. FAQL on the other hand catches up with RUQL after some transitional phase (RUQL remains ahead but only slightly).

improvement is consistent over different games and gradient-ascent-based learning algorithms. Using RUQL in a mixed population resulted in consistent performance improvement.

V. Related Work

Q-learning was originally designed for static single-agent environments, but yet performed adequately in multi-agent settings. However, there is a growing collection of algorithms that were designed specifically for multi-agent learning. The Win-or-Learn-Fast heuristic [4], [3] resulted in the first gradient-ascend-based (GAB) multi-agent learning algorithms that successfully converged to mixed Nash Equilibrium in small general-sum games with minimum knowledge of the underlying game (only the player's own payoff). This was followed by the more recent GAB algorithms [2], [21]. Other multi-agent algorithms that assumed knowledge of the underlying game were also proposed and were able to converge in larger games [9], [6]. All of these algorithms could benefit from improved value approximation techniques, as we showed in our experiments for two of these algorithms [2], [21].

Some learning algorithms specifically targeted non-stationary environments [7], [14] and can potentially improve GAB algorithms similar to RUQL and FAQL. However, their integration with GAB algorithms are not as straightforward. Specifically, one approach is specific to single-state domains and yet more complex than the simple update equations of

TABLE III. The average and standard deviation of RR for one agent using RUQL against another one using Q-learning for value estimation. RUQL against QL consistently reduces regret compared to QL against QL.

Algorithm	WPL	PGA-APP
Game		
Tricky	14.17 ± 7.27	10.55 ± 2.90
Matching Pennies	7.81 ± 2.03	14.80 ± 2.23
Biased	13.60 ± 10.96	1.67 ± 2.21

RUQL and FAQL [14]. The RL-Conext-Detection (RL-CD) algorithm is model-based and assumes finite sets of stationary contexts that the environment switches among [7]. For each stationary context a model is learned. Such an approach is challenging to use in multi-agent setting when the non-stationarity is a result of an adaptive opponent (with potentially infinite contexts).

Aside from Q-learning and its variations and extensions, several reinforcement learning algorithms were proposed to optimize the exploration and achieve more efficient learning [17], [10]. Aside from being more complex than Q-learning, the algorithms that optimize exploration are difficult to integrate with GAB-MARL algorithms, because GAB-MARL algorithms control the exploration. RUQL and FAQL, which we propose to use instead of QL, have simple update equations that can be easily integrated with GAB-MARL algorithms, and exhibit improved performance in dynamic settings.

VI. Conclusions

In this paper we proposed the use of recent value-based learning algorithms as internal components in multi-agent policy search learning algorithms. We evaluated six combinations of two policy-search algorithms (WPL and PGA-APP) and three value-based learning algorithms (QL, RUQL, and FAQL). We experimentally showed that using value-based-learners with less policy-bias in combination with the state-of-the-art MARL algorithms results in significant improvements in performance, both in self-play and in mixed-play with other variants. While related work has adopted the basic Q-learning update equation by default, the findings of this paper alert researchers to the importance of learning action values with algorithms that fit domain assumptions, especially for gradient-ascent-based algorithms and in multi-agent settings.

References

[1] S. Abdallah and M. Kaisers. Addressing the policy-bias of q-learning by repeating updates. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, pages 1045–1052, 2013.

[2] S. Abdallah and V. Lesser. A multiagent reinforcement learning algorithm with non-linear dynamics. *Journal of Artificial Intelligence Research*, 33:521–549, 2008.

[3] M. Bowling. Convergence and no-regret in multiagent learning. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, pages 209–216, 2005.

[4] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[5] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence*, pages 746–752, 1998.

[6] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.

[7] B. C. Da Silva, E. W. Basso, A. L. Bazzan, and P. M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd international conference on Machine learning*, pages 217–224. ACM, 2006.

[8] D. Fudenberg and D. K. Levine. *The theory of learning in games*. MIT press, 1998.

[9] J. Hu and M. P. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[10] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *The Journal of Machine Learning Research*, 99:1563–1600, 2010.

[11] M. Kaisers and K. Tuyls. Frequency adjusted multi-agent q-learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 309–316, 2010.

[12] S. Kalyanakrishnan and P. Stone. An empirical analysis of value function-based and policy search reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 749–756, 2009.

[13] D. S. Leslie and E. J. Collins. Individual q-learning in normal form games. *SIAM J. Control and Optimization*, 44(2):495–514, 2005.

[14] I. Noda. Adaptation of stepsize parameter to minimize exponential moving average of square error by newtons method. In *9th International Conference on Autonomous Agents and Multiagent Systems, pp. M-2–1*. Citeseer, 2010.

[15] Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007.

[16] P. Stone. Multiagent learning is not the answer. it is the question. *Artificial Intelligence*, 171:402–05, May 2007.

[17] A. L. Strehl, L. Li, and M. L. Littman. Reinforcement learning in finite mdps: Pac analysis. *The Journal of Machine Learning Research*, 10:2413–2444, 2009.

[18] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1999.

[19] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[20] M. Wunder, M. L. Littman, and M. Babes. Classes of multiagent q-learning dynamics with ϵ-greedy exploration. In *Proceedings of the International Conference on Machine Learning*, pages 1167–1174, 2010.

[21] C. Zhang and V. Lesser. Multi-Agent Learning with Policy Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 927–934, 2010.

[22] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the International Conference on Machine Learning*, pages 928–936, 2003.