COOPERATIVE REINFORCEMENT LEARNING USING AN EXPERT-MEASURING WEIGHTED STRATEGY WITH WOLF

Kevin Cousin and Gilbert L. Peterson
Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH, 45431, USA
kevin.cousin,gilbert.peterson@afit.edu

ABSTRACT

Gradient descent learning algorithms have proven effective in solving mixed strategy games. The policy hill climbing (PHC) variants of WoLF (Win or Learn Fast) and PDWoLF (Policy Dynamics based WoLF) have both shown rapid convergence to equilibrium solutions by increasing the accuracy of their gradient parameters over standard Q-learning. Likewise, cooperative learning techniques using weighted strategy sharing (WSS) and expertness measurements improve agent performance when multiple agents are solving a common goal. By combining these cooperative techniques with fast gradient descent learning, an agent's performance converges to a solution at an even faster rate. This statement is verified in a stochastic grid world environment using a limited visibility hunter-prey model with random and intelligent prey. Among five different expertness measurements, cooperative learning using each PHC algorithm converges faster than independent learning when agents strictly learn from better performing agents.

KEY WORDS

Multiagent, cooperative reinforcement learning, weighted strategies.

1. Introduction

Multiagent research has seen an increase in activity over the past decade with applications in a variety of fields. Some recent research has focused on the convergence of policy gradient ascent techniques for use in rational algorithms beneficial to multiple agent coordination. Likewise, techniques for exploiting cooperative learning [1] using weighted strategy sharing [2, 3] have been developed and shown to increase the learning rate of agents under some conditions by measuring the effectiveness of an agent while performing a task of meeting an objective. It seems natural to combine these ideas to increase a multiagent group's learning rate in domains that require generalization and function approximation. At a minimum, learning should improve through sharing information on explored spaces.

By combining the ideas of variable rate learning algorithms using policy hill climbing with expertness weighted strategy sharing, performance of groups of agents is generally increased. The WoLF and PDWoLF learning algorithms are implemented with policy sharing measured by five expertness metrics: normal, absolute, positive, negative and gradient. The convergence properties of the learning algorithms are preserved, however, experiments in a stochastic environment (grid world modeled with hunter-prey) demonstrates that applying these techniques by incorporating the different experiences of better performing agents into "losing" agents under proper measurement results in an increase of the overall performance of a group through individual enhancement.

In the following section, three variations on policy hill climbing algorithms are discussed, specifically PHC (Q-Learning), WoLF-PHC, and PDWoLF-PHC. This is followed by a discussion of expert measures and their applicability to WSS. Section 3 outlines the experiment design that implements the combined implementation of PHC and WSS in the hunter-prey domain. The following section highlights the results of the experiments. The last section concludes with some final thoughts and future issues to examine.

2. Background

2.1. Policy Hill Climbing (PHC) Algorithms

Three PHC variants are examined: standard PHC, WoLF-PHC and PDWoLF-PHC. These algorithms have been shown to convergence in increasing rates [4] as shown in fig. 1. The particular implementation of the algorithms relies on two tables. One table, Q, holds the expected reward over time using a typical temporal-difference formula to iteratively update the table's reward function approximation [5]. The second table, π, holds the policy, the probabilities used to select an action from some state. In general, the table initializations use the following values:

$$
\begin{gathered}
Q(s,a) \leftarrow 0 \\
\pi(s,a) \leftarrow \frac{1}{|A|} \\
\alpha \in (0,1] \\
\delta, \delta_l < \delta_w \ni \delta_{any} \in (0,1]
\end{gathered}
$$

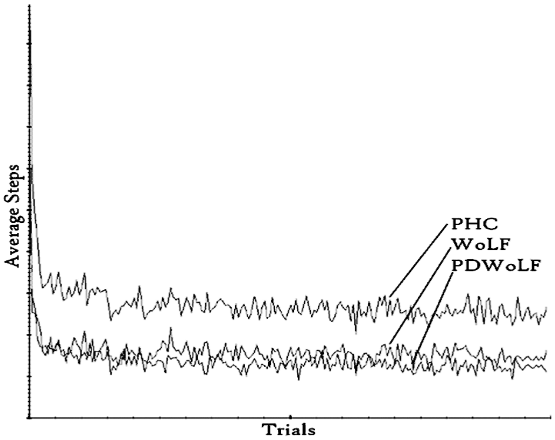

Figure 1. Comparison of PHC, WoLF-PHC and PDWoLF-PHC convergence vs an intelligent prey.

where the tables Q and π are functions over a state, s, and action, a, pair for the set of actions A. The learning rate α is a step-size parameter used to partially control the rate of gradient descent. The values of δ, δ_l and δ_w are each learning rates applied to each of the policy hill climbing approaches. The tables, Q and π, are updated according to the following rules for each algorithm.

$$
\begin{gathered}
Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a' \in A} Q(s',a')\right) \\
\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa} \\
\Delta_{sa} = \begin{cases} -\delta_{sa} \ \ if \ \ a \neq \max\limits_{a' \in A} Q(s,a') \\ \sum\limits_{a' \neq a \in A} \delta_{sa'} \ \ otherwise \end{cases} \\
where \ \delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A|-1}\right).
\end{gathered}
$$

The expected rewards table, Q, requires a reward r and a discount rate γ for updating. The policy, π, only requires an update from a Δ value based on the selection of δ_{sa} from a comparison of the existing policy value and a δ learning rate parameter chosen through additional criteria.

Generally, PHC only requires a constant δ learning rate to update the policy tables. No additional initialization or methodical testing is required. However, WoLF-PHC and PDWoLF-PHC both make use of a dynamic learning rate resulting in a faster convergence over general PHC. Because of this, both WoLF-PHC and PDWoLF-PHC use stronger selection criteria for choosing δ.

2.2. Win or Learn Fast (WoLF)

WoLF [6, 7] uses additional tables to estimate the average policy value in its calculation of the learning rate. The initialization of that table is

$$
\begin{gathered}
C(s) \leftarrow 0 \\
\bar{\pi}(s,a) \leftarrow \frac{1}{|A|}
\end{gathered}
$$

representing a counting function and an estimated average policy value. These functions are updated by

$$
\begin{gathered}
C(s) \leftarrow C(s) + 1 \\
\forall a' \in A, \bar{\pi}(s,a') \leftarrow \bar{\pi}(s,a') + \frac{\pi(s,a') - \bar{\pi}(s,a')}{C(s)}.
\end{gathered}
$$

The delta selection criteria used by WoLF for determining the learning rate uses this rule:

$$
\delta = \begin{cases} \delta_w \ if \ \sum\limits_{a' \in A} \pi(s,a')Q(s,a') > \sum\limits_{a' \in A} \bar{\pi}(s,a')Q(s,a') \\ \delta_l \ otherwise \end{cases},
$$

which is used in the calculation of Δ_{sa} of the PHC update rules.

2.3. Policy Dynamics Win or Learn Fast (PDWoLF)

Likewise, PDWoLF [4] uses additional estimation tables initialized by

$$
\begin{gathered}
\Delta(s,a) \leftarrow 0 \\
\Delta^2(s,a) \leftarrow 0
\end{gathered}
$$

where Δ and Δ^2 represent changing rates within the policy and are estimates of the slopes of the decision space. Each of these are updated with

$$
\begin{gathered}
\Delta^2(s,a) \leftarrow \Delta_{sa} - \Delta(s,a) \\
\Delta(s,a) \leftarrow \Delta_{sa}
\end{gathered}.
$$

The delta selection is then

$$
\delta = \begin{cases} \delta_w \ if \ \Delta(s,a)\Delta^2(s,a) < 0 \\ \delta_l \ otherwise \end{cases}.
$$

WoLF-PHC and PDWoLF-PHC each make use of known information in the decision space to approximate a good δ for improving the policy. The fundamental difference lies in the criteria used to select δ. WoLF relies on an average policy estimation while the PDWoLF uses the rate of change in the space to better approximate the change. Each algorithm, though, is based on the same core set of policy table update functions. Likewise, Q-Learning, WoLF and PDWoLF maintain expected value tables.

2.4. Expertness Measures and Weighted Strategy Sharing (WSS)

A variety of expertness measures can be applied to cooperative agents using WSS [2, 3, 8]. Of these, we consider the following five:

1) Normal: the sum of all reinforcement values.

$$e_i^{Nrm} = \sum_{t=1}^{T} r_i(t).$$

2) Absolute: the sum of absolute values.

$$e_i^{Abs} = \sum_{t=1}^{T} \left| r_i(t) \right|.$$

3) Positive: the sum of all positive-only reinforcements.

$$e_i^{P} = \sum_{t=1}^{T} r_i^{+}(t), r_i^{+}(t) = \begin{cases} 0, r_i(t) \le 0 \\ r_i(t), otherwise \end{cases}.$$

4) Negative: the sum of all negative-only reinforcements.

$$e_i^{N} = \sum_{t=1}^{T} r_i^{-}(t), r_i^{-}(t) = \begin{cases} 0, r_i(t) > 0 \\ r_i(t), otherwise \end{cases}.$$

5) Gradient: the sum of reinforcements from a certain point in time.

$$e_i^{G} = \sum_{t=c}^{T} r_i(t).$$

Each expertness measurement *e* is the summation of the rewards an agent receives over time. All, except the gradient, sum over the entire time period of a trial. The gradient measure sums over smaller intervals bound by a user defined constant c.

During WSS, each agent calculates its expertness measure, and then chooses to incorporate policy information from agents that are performing better based upon the weighting function (W_{ij}) and updates using the policy update rule:

$$W_{ij} = \begin{cases} 1-\rho_i, \; if \;\; i = j \\ \rho_i \dfrac{e_j - e_i}{\sum_{k=1}^{n} e_k - e_i}, \; if \;\; e_j > e_i \\ 0, \; otherwise \end{cases}$$

$$\pi_i \leftarrow (1-\rho_i)\pi_i + \rho_i \sum_{j \in Expert(i)} W_{ij}\pi_j$$

$$where\ Expert(i) = \left\{ j \middle| e_j > e_i \right\}.$$

for agent *i* and agent *j* with an expertness measure *e* for each of them. An impressibility factor *ρ* is used to further control how much an expert agent's policy influences any other agent.

Previous results using Q-learning agents [8] showed that when learning from better performing agents, the absolute, positive and negative measurements improved the performance of the group for random and intelligent moving prey. The goal is to apply the same methodologies to PHC algorithms in a stochastic environment and measure any performance changes.

3. Implementation

The combination of PHC algorithms and expertness measuring WSS [8] has each cooperating agent complete a trial by collecting its rewards and updating its expertness measure. At certain iterations, the agents collectively communicate their weights among the group and each selects the agent(s) from which they will use the expected reward value from to improve their behavior. The combination of WSS with the WoLF versions of PHC follows similarly with exception that agents improve behavior based on their policy tables and only cooperate with agents who have a better performance.

Verification testing combines each version of PHC with each expert measure for WSS by creating groups of three or more agents which compete in the classic hunter-prey grid world model [9, 10]. Although convergence to an optimum solution is not guaranteed, faster convergence is expected using multiagent cooperation rather than using a single agent solution. To increase the complexity of the search space, a series of tests are separately run against 1) a randomly moving prey and 2) an intelligently moving prey using potential fields with limited visibility. The agent will also have less visibility than the prey.

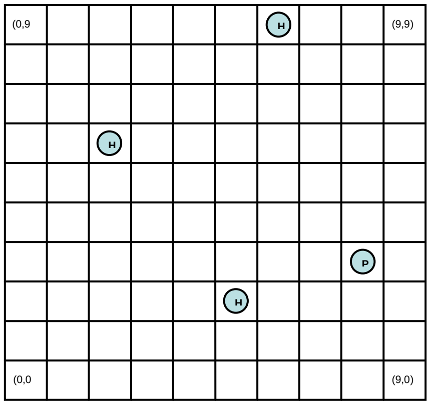

Figure 2. Sample hunter-prey grid world.

Table 1.
Algorithm for PHC Gradient Descent WSS-Cooperatively Learning Agents.

```
Select expertness measurement $E$.  Initialize $Q$, $\pi$ and let $q_0 = 0.1$.

For each agent A, repeat {
    Choose action $a$ from state $s$ by $a = \begin{cases} \max \pi(s), q \le q_0 \\ random(s), q > q_0 \end{cases}$.

    Observe distance, $d$, to prey.

    Observe reward $r_i$ =
```

From \ To	$d > v$	$1 < d \le v$	$0 < d \le 1$	$d = 0$
$d > v$	$-1/k^2$	$1/k$	$1/k$	k^2
$1 < d \le v$	$-1/k$	$-1/k^2$	$1/k$	$1/k$
$0 < d \le 1$	$-1/k$	$-1/k$	$-1/k^2$	$1/k$
$d = 0$	$-1/k$	$-1/k$	$-1/k$	$-1/k^2$

```
    Update    $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r_i + \gamma \max_{a' \in A} Q(s',a')\right)$

    Update    $\Delta_{sa} = \begin{cases} -\delta_{sa} \ if \ a \ne \max_{a' \in A} Q(s',a') \\ \sum_{a' \ne a \in A} \delta_{sa'} \ otherwise \end{cases}, where \ \delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A|-1}\right)$
              $\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa}$

    Update expertness $e_i \leftarrow e_i + e_i^E, E = \{Nrm, Abs, P, N, G\}$.

    If (i ≡ 0 mod $n_c$) then cooperate by
        $W_{ij} = \begin{cases} 1-\rho_i, if \ i = j \\ \rho_i \frac{e_j - e_i}{\sum_{k=1}^{n} e_k - e_i}, if \ e_j > e_i \\ 0, otherwise \end{cases}$.
        $\pi_i \leftarrow (1-\rho_i)\pi_i + \rho_i \sum_{j \in Expert(i)} W_{ij}\pi_j, where \ Expert(i) = \{j | e_j > e_i\}$
} until $d = 0$.
```

The hunter-prey model used for testing is a 10 x 10 discrete grid with three cooperating agents, one non-cooperating agent (for control), and one prey as shown in fig 2. The three cooperative agents use the same expertness metric, and all four agents use the same PHC algorithm to select actions leading to a capture of the prey according to its policy table. For each trial, all four agents start at the same randomly chosen location. The prey starts in a different randomly chosen location. The starting location of each trial changes after all agents have captured the prey. Once an agent captures the prey, it no longer continues to take action that can influence the prey nor does it cooperate with any other agents. One trial is complete when all agents have captured the prey.

The state representation is a Cartesian coordinate and tiled location of the prey. The location was decomposed to a direction, one of nine values (eight 45 degree angles and collocated), coupled with one of four distance values (on, one away, visible, not visible). Only 18 rational pairings of direction and distance are considered as a part of the state. Logically, the agent can choose one of nine actions, directly related to the directions of a prey. Only legal actions can be chosen, e.g., an agent is unable to go left if it is on the left side of the area. Hence, a completely filled policy consists of 16,200 state-action values. For this specific implementation, a sparse matrix is used to store only visited states to reduce memory overhead and involve more agents simultaneously.

An agent's actions are chosen based upon its policy table (π), however, if the agent is unable to "see" the prey, it uses a default state, denoting non-visibility, to make a selection. The intelligent prey makes use of a potential field to steer it away from the hunters. For these trials, the hunter agents have visibility of 2 units and the prey has 3.

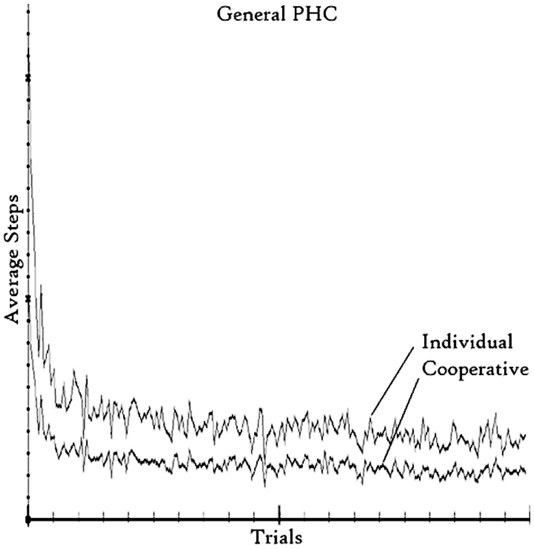

Figure 3a. Average PHC results.

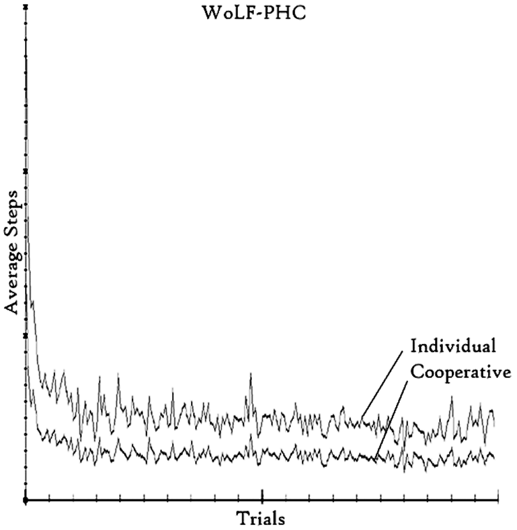

Figure 3b. Average PHC-WoLF results.

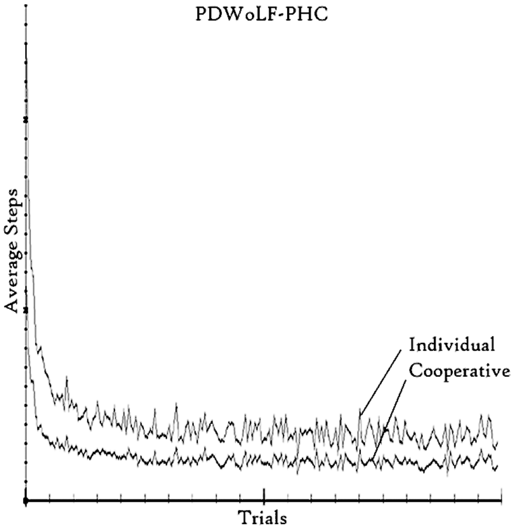

Figure 3c. Average PDWoLF-PHC results.

Reinforcement reward values are based on how an action moves an agent to the prey. If the action moved the agent from not visible to visible or visible to one away, a small, positive value based on the width of the grid world was given. In particular, such a move was rewarded with *1/k* for a *k* x *k* grid. Moving to a more distant level was rewarded with *-1/k*. Moves that did not change the distance level were awarded $-1/k^2$. Capturing the prey has a larger reward of k^2.

During the simulation, only the weighting of the policy tables (π) will be subject to WSS. The expected rewards (Q) and other support tables will remain unaltered during trial runs. Likewise, only the policy will be normalized to a standard distribution.

A complete experiment consisted of averaging 10 runs of *n* trials on each of the five expertness measures for each of the PHC algorithms. The results are for trials of *n* = 2,000. For each PHC algorithm, $\alpha = 0.3$, $\delta = 0.5$, $\delta_w = 0.25$, $\delta_l = 0.75$, and $\gamma = 0.8$. ρ_i was set to 0.75. Cooperation occurred every 15 steps. An algorithm for the experiment is listed in table 1.

4. Results

The test results show that under most measurements, there is a speedup in learning. As shown in fig. 3, each of the three PHC algorithms retained its convergence property for each expertness measurement. The data represents the average number of steps for an individual learner and an average of three cooperative learners averaged over 10 runs of 2,000 trials typical for any of the expert metrics. Each tick on the vertical axis measures 10 steps for each figure.

As shown in fig. 4 and fig. 5, the percentage of improvement in the number of steps required to capture the prey as an average of the three cooperative agents over the individual learner. These experiments were also averaged over 10 runs of 2,000 iterations. Select experiments with up to 5,000 iterations produce similar results.

These results indicate some level of improvement over the results of the Q-learning experiment in that none of the five chosen expertness measurements completely fails to provide an improvement in group learning. The data suggests that in either case of an intelligent of randomly moving prey, some amount of improvement is achieved.

However, as expected, no single expertness metric proved best in all situations. Clearly, the data suggests that using a positive metric, the only function to produce a negative value, generally fares the worst for group improvement. The negative result of using positive measurement (e^P) with PHC is fairly insignificant being less than 1%. In the case of the randomly moving prey, the difference in performance of the measurements are less distinct and statistically similar for absolute, positive and negative measures across each of the PHC algorithms.

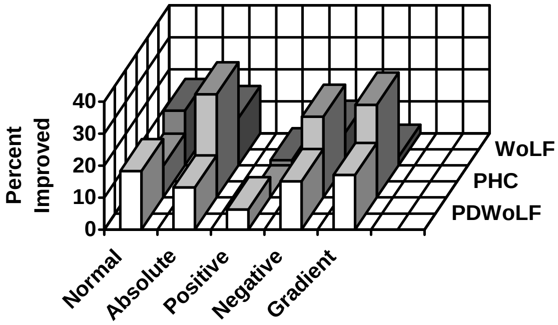

Figure 4. Improvement of cooperative group learning matched against an intelligent prey.

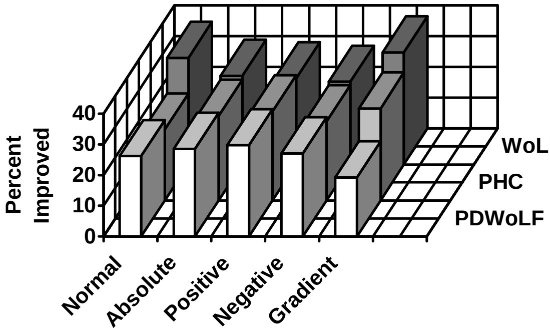

Figure 5. Improvement of cooperative group learning matched against a random moving prey.

5. Conclusions

Clearly, using weighted strategy sharing in a multiagent setting accelerates the learning of fast PHC algorithms, such as WoLF and PDWoLF. These experimental results suggest as much as a 36% increase in performance when using cooperative expertness to affect policy learning. This is largely due to allowing multiple agents to share their experiences at intervals while retaining their own identity. This allows the cooperative group to overall outperform a collection of independent agents.

However, multi-objective or competitive agents have not been addressed using this technique. Further examination should reveal more of whether applying these expertness metrics provide improved learning. Likewise, this test was simple, using some readily identifiable expertness metrics. Could there be other, less identifiable, metrics that radically improve the performance of this model? A slight change in any of these parameters can easily reorder the convergence ranking of the PHC algorithms. Expertness measuring adds a second dimension to optimizing performance. Are expertness measuring and PHC tuning related? By studying these questions within a stronger mathematical framework, promising new techniques could be revealed in multiagent cooperation.

6. Acknowledgements

The work on this paper was supported by the Digital Data Embedding Technologies group of the Air Force Office of Scientific Research. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Air Force Institute of Technology, or the U.S. Government.

References

[1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98),* pages 746--752, July 1998.

[2] N. M. Ahmadabadi, M. Asadpour, S. H. Khodaabakhsh and E. Nakano. Expertness Measuring in Cooperative Learning. *Proceedings of IROS 2000.*

[3] S. M. Eshgh and M. N. Ahmadabadi. An Extension of Weighted Strategy Sharing in Cooperative Q-Learning for Specialized Agents, Neural Information Processing, 2002. ICONIP '02. *Proceedings of the 9th International Conference,* 18-22 Nov. 2002, 106- 110 vol.1.

[4] B. Banerjee and J. Peng: Adaptive policy gradient in multiagent learning. *AAMAS 2003*: 686-692.

[5] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[6] M. H. Bowling and M. M. Veloso: Multiagent learning using a variable learning rate. *Artif. Intell. 136(2)*: 215-250 (2002).

[7] M. H. Bowling and M. M. Veloso. Scalable learning in stochastic games. *AAAI Workshop Proceedings on Game Theoretic and Decision Theoretic Agents,* Edmonton, Canada, 2002.

[8] M. N. Ahmadabadi and M. Asadpour: Expertness based cooperative Q-learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B 32*(1): 66-76, 2002.

[9] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. *11th International Conference on Machine Learning,* New Brunswick. NJ, Morgan Kaufmann, San Mateo, CA, 1994, pp. 157-163.

[10] M. Tan, Multi-agent reinforcement learning: Independent vs. cooperative agents. *Proc. 10th International Conference on Machine Learning,* Amherst, MA, 1993, pp. 330-337.