**Delft University of Technology**

**Delft Center for Systems and Control**

Technical report 05-017

# Multiagent reinforcement learning with adaptive state focus*

L. Buşoniu, B. De Schutter, and R. Babuška

*If you want to cite this report, please use the following reference instead:*
L. Buşoniu, B. De Schutter, and R. Babuška, "Multiagent reinforcement learning with adaptive state focus," *Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2005)* (K. Verbeeck, K. Tuyls, A. Nowé, B. Manderick, and B. Kuijpers, eds.), Brussels, Belgium, pp. 35–42, Oct. 2005.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: `https://www.dcsc.tudelft.nl`

*This report can also be downloaded via `https://pub.deschutter.info/abs/05_017.html`

# MULTIAGENT REINFORCEMENT LEARNING WITH ADAPTIVE STATE FOCUS

Lucian Buşoniu Bart De Schutter Robert Babuška

*Delft Center for Systems and Control, Delft University of Technology*
*Mekelweg 2, 2628 CD Delft, The Netherlands*
{*i.l.busoniu,b.deschutter,r.babuska*}*@dcsc.tudelft.nl*

**Abstract**

In realistic multiagent systems, learning on the basis of complete state information is not feasible. We introduce *adaptive state focus Q-learning*, a class of methods derived from Q-learning that start learning with only the state information that is strictly necessary for a single agent to perform the task, and that monitor the convergence of learning. If lack of convergence is detected, the learner dynamically expands its state space to incorporate more state information (e.g., states of other agents). Learning is faster and takes less resources than if the complete state were considered from the start, while being able to handle situations where agents interfere in pursuing their goals. We illustrate our approach by instantiating a simple version of such a method, and by showing that it outperforms learning with full state information without being hindered by the deficiencies of learning on the basis of a single agent's state.

**Keywords:** *multiagent learning, adaptive learning, Q-learning, coordination.*

## 1 Introduction

So far, reinforcement learning (RL) has been the leading paradigm for learning from interaction with the environment [8]. Q-learning belongs to the most popular model-free reinforcement learning methods [9]. The comprehensive understanding and the availability of convergence results for Q-learning in the single-agent case make its application to the multiagent (MA) learning problem very appealing. However, in the MA case, the results of one agent's actions depend on the actions of the other agents, so the agents need to *coordinate* their actions in order for learning to be effective and to converge.

There are various methods to coordinate learning in MA systems. Friend-or-foe Q-learning assumes that opponents can be classified as either "friends" that always act towards the learner's goal, or "foes" that always act against it [6]. Nash-Q removes this restrictive assumption by using the game-theoretic notion of Nash equilibrium [4]. Nash-Q is representative for the game-theoretic approach to MA-RL. The coordination issue appears here as the equilibrium selection problem – in a given state, there might be several equilibria, and agents need to choose one in a consistent fashion. Win-or-learn-fast Q-learning is a sound heuristic that combines Q-learning with gradient ascent [1].

Almost all the work in MA-RL implicitly considers the complete state information to be known and used by the agents. The tasks the algorithms were tested on were either stateless [7] or had very small, simple state spaces [1, 4, 6], in which potential problems with using the complete state were not considered. However, in realistic problems, considering the whole state vector throughout learning is not feasible due to several reasons: learning takes large amounts of resources and is slow; some parts of the state may not be observable, so they need to be constantly communicated; and last but not least, not all of the state information is relevant to the learner.

Therefore, we introduce a class of MA-RL algorithms that start learning focused only on a small part of the state, essential to the agent. When an agent detects the lack of convergence due to interference with other agents, it adds new dimensions to its state space (or only to a part of it), so that the learning problem can be resolved. As a proof of concept, we present a simple instantiation of such a method that starts learning with the state of a single agent, and upon lack of convergence adjusts its focus in one leap to the complete world state information. We show that the method performs better than when the complete state information is considered

from the beginning of learning, while being able to solve situations that single-agent learning cannot deal with.

The work that relates most closely to ours [5] analyzes statistics on expected returns in order to detect coordination relationships among agents. The coordination relationships are expressed in terms of agent actions, maintaining the already encountered monolithic view on the world state. A related approach expands the learning structures of the agent with the actions of other agents that seem to have a significant effect on its performance [3]. This approach also disregards the world state issue. Our method attempts to complement this work by dealing with situations where learning on the basis of the complete state is impractical.

The remainder of this paper is organized as follows. Some necessary background in RL and Q-learning is introduced in Section 2. We describe our class of adaptive state focus methods, give an example, and discuss some parameters and exploration issues in Section 3. We then present and discuss our experimental results in Section 4. Section 5 concludes the paper and gives a brief overview of planned future work.

# 2 Background: Q-Learning

In the independent learner case, the RL task is formalized as a *Markov decision process*:

**Definition 1.** *A Markov decision process (MDP) is a tuple* $\langle S, A, r, p \rangle$*, where* $S$ *is the discrete state space,* $A$ *is the discrete action space,* $r : S \times A \to \mathbb{R}$ *is the reward function of the agent, and* $p : S \times A \times S \to [0, 1]$ *is the transition function, mapping states and actions into a probability distribution over subsequent states.*

The behavior of the agent is described by its *policy* $\pi$, a mapping from states into actions (for deterministic policies, $\pi : S \to A$), or from states into a probability distribution over actions (for stochastic policies, $\pi : S \times A \to [0, 1]$). The goal of the agent is to maximize the discounted return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where $t$ denotes the discrete time, $r_t$ is the reward received at time step $t$ and $\gamma \in (0, 1]$ is the discount factor [8].

The *Q-learning* algorithm [9] employs so-called *Q-values* (action values). The Q-value $Q^\pi(s, a)$ is the expected return value of taking action $a$ in state $s$ and following the policy $\pi$ thereafter:

$$Q^\pi(s, a) = E_\pi \{R_t | s_t = s, a_t = a\}, \quad \forall s \in S, a \in A. \tag{2.1}$$

The optimal Q-values, denoted by $Q^*$, are those obtained by an optimal policy – i.e., a policy under which the expected returns are maximal. There may be several such optimal policies, but the optimal Q-values are unique. Q-learning iteratively approximates the optimal Q-values by applying the temporal difference formula:

$$Q_{t+1}(s_t, a_t) = (1 - \alpha_t) Q_t(s_t, a_t) + \alpha_t \Big[ r_{t+1} + \gamma \max_{a \in A} Q_t(s_{t+1}, a) \Big], \tag{2.2}$$

each time the agent observes a transition from state $s_t$ to state $s_{t+1}$, as a consequence of its action $a_t$ and associated with a reward $r_{t+1}$. The parameter $\alpha_t \in [0, 1)$ is the learning rate. Assuming that all state-action pairs are visited infinitely often, and that the learning rate decays properly (the series $\alpha_t$ sums to infinity, but $\alpha_t^2$ sums to a finite value), the algorithm is guaranteed to converge in the limit to the optimal Q-values [9]. Given these values, a (deterministic) optimal policy follows immediately:

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s, a), \quad \forall s \in S. \tag{2.3}$$

The agent typically uses a greedy policy similar to (2.3), but this is not enough to fulfill the convergence conditions: the agent must sometime *explore* worse-looking actions in order to improve its knowledge. This is usually done by following a random action instead of the greedy one with a certain probability $\varepsilon \in (0, 1)$, decaying over time as the agent becomes more certain of its knowledge.

The formalization of the *multiagent* RL task is typically a Markov (or "stochastic") game, which is a straightforward extension of the MDP to the MA case:

**Definition 2.** *A stochastic game (SG) is a tuple* $\langle n, S, A^1, \ldots, A^n, r^1, \ldots, r^n, p \rangle$*, where* $n$ *is the number of agents,* $S$ *is the discrete state space,* $A^i$ *is the discrete action space and* $r^i : S \times A^1 \times \cdots \times A^n \to \mathbb{R}$ *the reward function of agent* $i$*,* $i = 1, \ldots, n$*, and* $p : S \times A^1 \times \cdots \times A^n \times S \to [0, 1]$ *is the transition probability.*

Note that this time, the transitions and rewards depend on the *joint* action of the agents. In our setting, a given agent (say the $i^{\text{th}}$) will be characterized by a (local) state vector $s^i \in S^i$, where $S^i \subseteq S$ is the (local) state space of agent $i$. The rest of the state space $S$ is given then by state elements referring to other agents, or to the environment in general without reference to a specific agent.

# 3 Adaptive state focus in Q-learning

## 3.1 Motivation

As already noted, the literature devoted to MA-RL typically assumes the complete state information to be known and used. However, as the size of an agent's state space and the number of agents increase, taking into account the complete state starts posing two problems:

- the memory required to store the Q-values rapidly increases;
- the learning speed decreases, due to the need of exploring a large state space.

Observing the whole state in large problems is not cheap, either. Consider the following hypothetical example: a rescue team of ground robots and UAVs assisting in rescue operations after a highway crash. These robotic agents may not always be able to directly measure each other's state (e.g., positions), and some parts of their state might never be directly accessible (e.g., fuel level). Thus, in real-life problems, observing the complete state typically requires that agents continuously exchange their states via communication. We aim at alleviating this situation by resorting to complex state information only when necessary.

It has been observed that basic Q-learning (2.2) works well in certain MA problems (see e.g., [2]). This happens if no conflicts or interferences between agents occur. If this is not the case, single-agent Q-learning will not converge. On the contrary, Q-values will oscillate significantly late into learning, without settling on a valid policy, due to the Q-learner's inability to distinguish between world states in which the states of other agents differ. In these situations, giving more detailed state information to the Q-learner will help it get over the difficulty and solve the problem.

## 3.2 Approach

We consider a cooperative setting with non-conflicting agent goals, but where agents may interfere while working towards their goals. We do not address the equilibrium selection problem, focusing instead on a more relaxed type of coordination, appearing in tasks with relatively large state spaces and solvable by a correct decision taken prior to reaching the state where equilibrium selection becomes necessary. In our experiments, we exemplify this type of coordination problem with the *critical shared resource* problem, where several agents need some unique resource to achieve their goal, but that resource cannot be used by more than one agent at a given time. This problem is solvable if the agents manage to properly sequence their arrivals at the resource.

The method we propose is motivated by the considerations in Section 3.1 and builds upon the basic Q-learning algorithm (2.2). Learning begins with single-agent Q-learning:

$$Q^i_{t+1}(s^i_t, a^i_t) = (1 - \alpha^i_t) Q^i_t(s^i_t, a^i_t) + \alpha^i_t \Big[ r^i_{t+1} + \gamma \max_{a^i \in A^i} Q^i_t(s^i_{t+1}, a^i) \Big], \tag{3.1}$$

where $s^i_t \in S^i$ is the state of agent $i$ at time step $t$ and all other agent-specific terms have been similarly superscripted by $i$. The agent monitors the evolution of its Q-values while learning, looking for the above mentioned oscillatory patterns caused by a failure of convergence. Upon detecting a failure of convergence, the agent adds new dimensions to its state space in order to overcome the learning difficulty.

For the sake of space, we present an algorithm that adjusts the agent's state focus in one leap from the single agent state to the complete world state. Various finer adjustments are possible; we discuss the possibilities in the end of this section.

Considering that $S = S^1 \times \cdots \times S^n$ (i.e., the complete state is the concatenation of all the agent's state vectors), the structure and sizes of the $i^{\text{th}}$ agents' Q-tables before and after the expansion are:

$$\begin{aligned} &Q^i_{\text{before}}(s^i, a^i), && \dim(Q^i_{\text{before}}) = \big|S^i\big| \cdot \big|A^i\big| \\ &Q^i_{\text{after}}(s^1, \ldots, s^n, a^i), && \dim(Q^i_{\text{after}}) = \big|S^1\big| \cdots |S^n| \cdot \big|A^i\big| \end{aligned} \tag{3.2}$$

where $|\cdot|$ denotes the cardinality of a set. The exponential increase in memory requirements and processing time is clear.

The expanded Q-table will be initialized so that the learned Q-values are maintained. One Q-value in the smaller table will now span an entire slice of the enlarged state space:

$$Q^i_{\text{after}}(\cdots, s^i, \cdots, a^i) = Q^i_{\text{before}}(s^i, a^i), \quad \forall s^i \in S^i, a^i \in A^i. \tag{3.3}$$

If $Q^i(s^i, a^i)$ is a good approximation of $Q^i(s^1, \ldots, s^n, a^i)$, i.e., if the agents do not severely interfere with each other, we expect (3.1) to converge quickly and efficiently without any state expansion taking place. However, the algorithm should still be able to handle situations where agents interfere, by adjusting its focus to the complete world state.

This type of focus adjustment, moving in one leap from the single agent state to the complete state, is just an example. By considering various expansion magnitudes, an entire class of methods is obtained: first, almost certainly the coordination problem will only arise in a small region of the state space (in our example, two ground robots need to coordinate when navigating between two crashed cars, but not necessarily in other places). So, an agent needs to expand only parts of its own state space with external state information. Equally important, in a realistic multiagent system many agents may coexist, only a small subset of which intersect in some region of the world leading to a coordination problem (e.g., to pass between the two cars the ground robots need not coordinate their navigation with the UAV). Thus, the agent needs to expand its state space only with information from relevant agents. Third, not all the state information of these agents is relevant (e.g., the fuel level of another robot is not relevant when sharing a passageway, unless it is very low in which case that robot will not move).

Several focus adjustments could be applied in an incremental manner, so that only the amount of information necessary to solve the problem is considered. In some situations, it might even be useful to discard dimensions of the state space. One such situation is when the problem has been solved and will not reappear, thus rendering the state space dimensions added for solving it irrelevant.

We hereafter call Q-learning algorithms applying such state focus adjustments "adaptive state focus Q-learning" — ASF-Q. Note that the approach is applicable to other RL algorithms as well.

## 3.3 Analysis of the convergence behavior

The analysis of the convergence behavior of Q-learning is clearly an essential part of any ASF-Q algorithm. The mechanism must be reliable: if it does not detect lack of convergence, the learning task will not be solved; if it erroneously detects lack of convergence, it will use significantly more resources without benefit, probably even slowing down learning. It must also be fast, as it runs on-line, in parallel with learning. We focus here on a heuristic method, based on analyzing the average *absolute temporal difference* at consecutive time steps:

$$\Delta Q^i_t = \frac{\sum_{a^i \in A^i, s^i \in S^i} \left| Q^i_t(s^i, a^i) - Q^i_{t-1}(s^i, a^i) \right|}{c^i_t}, \quad t \geq 1, \tag{3.4}$$

where $c^i_t$ is the number of pairs $(s^i, a^i)$ whose Q-values have changed at step $t$, i.e., $Q^i_t(s^i, a^i) \neq Q^i_{t-1}(s^i, a^i)$.

The absolute difference $\Delta Q^i_t$ should decrease as Q-learning converges, whereas it should attain positive high values if Q-learning does not converge. For robustness and speed we will analyze a sliding window of such differences. The window moves continuously over the entire learning process, without considering individual trials separately. The window width is given in number of time steps and is denoted by $w$. We will look at the evolutions of the mean of this sliding window, and the average first-order difference of this mean. For speed of analysis, the mean will be subsampled $\sigma$ times per window. The convergence will be examined at the same time steps when the mean is subsampled.

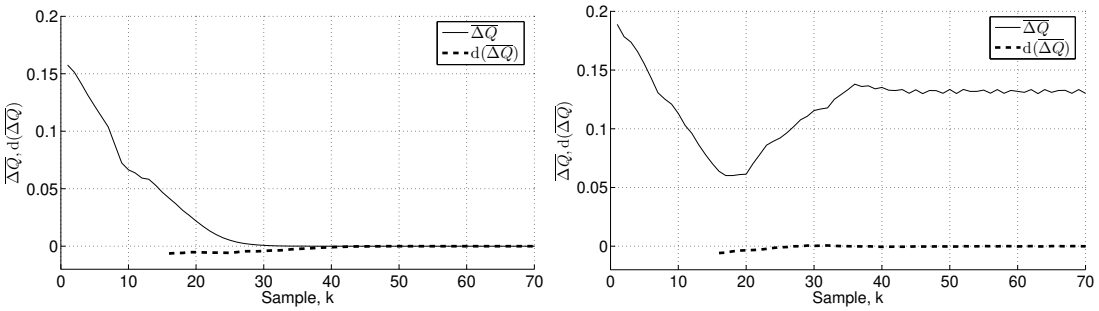
The subsampled mean is therefore computed by averaging the $\Delta Q^i_t$ window once each $m = w/\sigma$ steps, waiting $w$ steps for the window to be fully populated prior to starting the computation:

$$\overline{\Delta Q^i_k} = \frac{\sum_{j=km-w+1}^{km} \Delta Q^i_{w+j}}{w}, \quad k \geq 1, \tag{3.5}$$

where $k$ is the sample index. The mean first-order difference of $\overline{\Delta Q^i_k}$ is computed by averaging the difference of the current $\overline{\Delta Q^i_k}$ window, waiting for it to be fully populated. Thus, as (3.5) is

in its turn applied only after a full window has elapsed, the first-order difference computation can start only after $2w$ steps:

$$\mathrm{d}(\overline{\Delta Q^i_k}) = \frac{\sum_{j=k-\sigma+1}^{k} \left( \overline{\Delta Q^i_j} - \overline{\Delta Q^i_{j-1}} \right)}{\sigma}, \quad k > \sigma. \tag{3.6}$$




(a) Convergence.

(b) No convergence.

Figure 1: Evolution of convergence measures $\overline{\Delta Q}$ and $\mathrm{d}(\overline{\Delta Q})$.

Figures 1(a) and 1(b), respectively, present typical evolutions of the sliding mean and mean difference of $\Delta Q$ when Q-learning converges and when it does not[1]. These figures outline a clear behavior of $\Delta Q$: if converging, its mean will most of the time decrease approaching zero (thus its mean difference will approach zero from negative values); if not converging, the mean will eventually start increasing and/or reach a positive plateau, driving the mean difference close to zero. When converging, constant or slightly increasing transients may still appear in the evolution of the mean. They are due to the discovery of unexplored areas of the state space where Q-values are in need of adjustment, or to non-critical interactions with other agents. The overall tendency will however be decreasing and the mean will eventually reach zero.

For robustness, we will consider that convergence has failed if the mean difference falls within a zero-centered tolerance band with width $\tau \in (0, 1)$ of its absolute maximum value, while the mean is still situated above a similar zero-centered tolerance band with the same relative width $\tau$.

## 3.4 Parameters

ASF-Q needs more parameters than the basic Q-learning algorithm. The variant presented above requires three: the convergence analysis window width $w$, the number of samples per window $\sigma$, and the width of the zero-band $\tau$. Based on our current experience, only one of them is crucial: $w$, while for the others reasonable defaults should work across a large spectrum of problems. This is because the $\sigma$ parameter does not influence the nature of the convergence analysis, but only dictates how often it should be performed. Any $\sigma$ between 10 and 20 should work well. Due to the fact that the mean and difference of $\Delta Q$ evolve together, different values of the relative tolerance $\tau$ should perform similarly. They must, however, not be too great or too small, as the robustness of the algorithm would be lost. A value around 0.1 seems to be a reasonable choice.

The window width $w$, however, is essential: if it is too short, transient fluctuations in convergence will be interpreted as convergence failure, while if too long, large initial values may bias the sliding averages and the focus adjustment would be delayed.

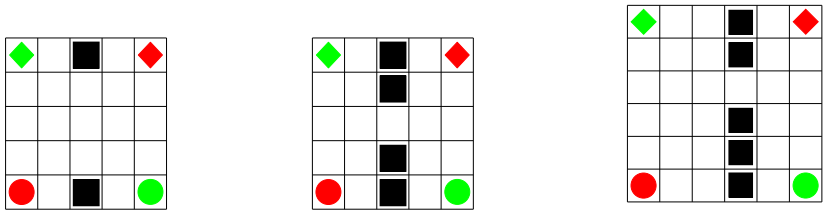
*Remark.* An issue not touched in the discussion above is the effect of the focus changes on the exploration behavior. Intuitively, since the state space has just become larger, it seems the learner should perform a new epoch of exploration. However, experiments on this issue (not shown here due to limited space) indicate that, at least in the considered situations, this is not the case. This is probably due to the fact that the Q-values $Q^i(s^i, a^i)$ learned by the agent prior to the expansion are already a rough approximation of the expanded Q-values $Q^i(s^1, \ldots, s^n, a^i)$. After the expansion, a few more adjustments in the region of the state space where the influence of the other agents is significant, suffice to solve the problem. There is however no guarantee that this behavior will extend to settings other than those presented here.

[1]The evolutions were recorded for the left hand agents using single-agent Q-learning on the gridworlds of Figures 2(a) and 2(b). The parameters were set as described in Section 4.

# 4 Experiments

## 4.1 Setting

As learning environments for our experiments, we use *gridworlds*, which are popular test beds for MA-RL algorithms in the literature (see e.g., [1, 4]). They are abstractions of relevant real domains such as robotic navigation, and provide interesting learning problems such as the critical shared resource issue. The gridworld is a rectangular two-dimensional surface divided into cells of three types: empty, obstacle, and goal cells. Agents can only move in the four compass directions or stand still. They have distinct goal cells, can only travel through empty cells, with at most one agent in a given cell at a given time, and cannot swap places. A learning trial starts with the agents in their initial positions and ends when all agents have reached their corresponding goal cells (they may reach them at different moments in time).



(a) $5 \times 5$ gridworld with no coordination problem.

(b) $5 \times 5$ gridworld with a shared passage.

(c) $6 \times 6$ gridworld with a shared passage.

Figure 2: Experimental setting.

Figure 2 presents the three gridworlds used here as agent environments. Agents are colored discs, their corresponding goals are identically colored diamonds, and black squares are obstacle cells. Two agents suffice to illustrate the relevant issues; they are represented in their starting positions. In the gridworld of Figure 2(a), no problems arise. In the other two a critical shared resource exists: the single-cell passage in the row of obstacles. The two agents need to learn how to sequence their passing through it. Figure 2(b) presents a symmetrical problem, whereas Figure 2(c) is a larger, asymmetrical setup.

While a variety of focus changes are possible, we use the algorithm described in Section 3.2: the state space is expanded from one agent's position to the position of all the agents. According to (3.2) the sizes of the Q-tables before and after the expansion will be $WH \cdot 5$ and $(WH)^n \cdot 5$, respectively, where $W$ and $H$ are the width and height of the gridworld, $n$ the number of agents, and 5 the number of agent actions: move east, south, west, north, and stand still. For the gridworld of Figure 2(c), the Q-table sizes before and after the expansion would therefore be 180 and 6480, respectively. For the gridworlds of Figures 2(a) and 2(b), these sizes would be 125 and 3125, respectively.

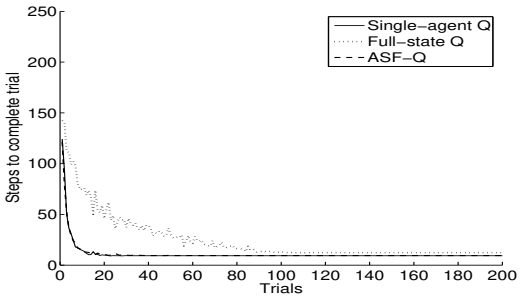
Our experiments compare the performance of three algorithms: single-agent Q-learning, full-state Q-learning (where the agents use the complete state information from the start of learning), and ASF-Q. The learning parameter settings are: constant learning rate $\alpha = 0.3$, discount factor $\gamma = 0.95$, and random exploration with $\varepsilon = 0.3$ decaying with the trials count ($\varepsilon \leftarrow \varepsilon / N_{\text{trials}}$ at the beginning of each trial). Though a constant learning rate violates the theoretical convergence requirements of Q-learning, in this setting it actually poses no problem for convergence. Instead, learning is more robust as the Q-learners are allowed to continuously adapt to an environment made dynamic by the presence of other agents.

The convergence analysis parameter values are $w = 256$ steps, $\sigma = 16$, giving $m = 16$ steps, and $\tau = 0.1$. Since we are interested in the overall convergence of Q-learning, and not along a trial, the window width $w$ is not related to the length of individual trials.

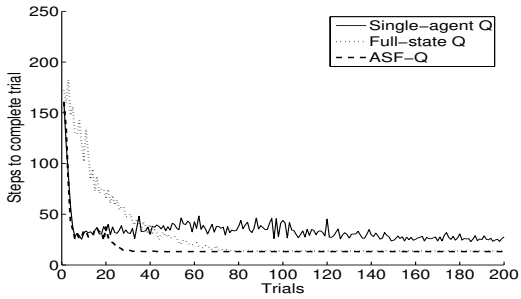
We also use an eligibility trace with the decay factor $\lambda = 0.5$ [8]. The eligibility trace mechanism maintains an exponentially decaying trace of the path followed through the state-action space up to the current step. This trace is used at each step to update not only the Q-value corresponding to the last taken action, but the entire sequence Q-values along the path. This increases the convergence speed of Q-learning, without influencing its qualitative properties. Thus, without an eligibility trace, larger values of $w$ might be required.

### 4.2 Results and discussion

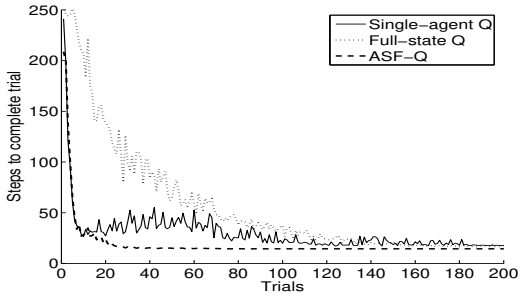
Figure 3 presents the results, averaged over 25 independent runs for each configuration. The convergence is measured by plotting the number of steps the agents need to reach the goal (i.e., complete a trial) against the number of elapsed trials (and thus the learning time).




(a) Gridworld of Figure 2(a).




(b) Gridworld of Figure 2(b).




(c) Gridworld of Figure 2(c).

Figure 3: Convergence of single-agent, full-state and adaptive state focus Q-learning.

It is clear that allowing the agents to learn on the basis of the full state information is very inefficient: besides taking more resources, learning is considerably slower — these reasons were part of our motivation for introducing ASF-Q. Full state learning even learns (on average) slightly worse policies than the single-agent version on the simple problem of Figure 2(a), and than ASF-Q on the problems of Figures 2(b) and 2(c).

On the simple problem of Figure 2(a), ASF-Q performs as well as single-agent Q-learning, being in fact equivalent to it as no focus adjustment takes place. When the need for coordination arises, single-agent Q-learning is not anymore able to solve the learning task (see Figures 3(b) and 3(c)): the policies of the agents keep oscillating until the end of learning (clearly visible in Figure 3(b), less visible but still present in Figure 3(c)), and if at any time learning would be stopped the agents would not be able to reach their goals. They are able to do that during learning because the learning rate $\alpha$ is maintained constant. ASF-Q, however, soon detects the convergence problem and switches to full state information (consistently around trial 20 in Figure 3(b) and around trial 10 in Figure 3(c)). After a short while, it converges – much faster and (on average) to a slightly better policy than full-state Q-learning.

## 5 Conclusions and future work

We presented a class of multiagent reinforcement learning methods that adapt the dimensions of an agent's state space on the basis of the convergence analysis of the learning process. Our results indicate that learning on the basis of the full state information is not effective in problems of realistic complexity, due to the computational requirements, slow learning speed, and even lower quality of the learned behavior. On the other hand, agents using only their own state information might not be able to learn at all when the need for coordination arises. It appears that the adaptive state focus techniques we introduced are able to combine the simplicity and efficiency of single-agent learning with the power of using more complex, relevant parts of the state.

Regarding the ideas on the focus adjustment magnitude outlined in Section 3.2, two questions arise: how to determine (i) the regions of the state space in need of expansion, and (ii)

the relevant dimensions along which to expand? An answer to the first question could be to monitor second-order information in Q-values, discriminating by state. Projection methods such as Principal Component Analysis could provide answers to the second question.

Further study of the algorithm is required on settings other than gridworlds. We are also planning to investigate strategies for setting the analysis window width $w$, as well as other convergence analysis methods.

**Acknowledgments**

This work was undertaken within the Interactive Collaborative Information Systems (BSIK-ICIS) project, funded by the Dutch Ministry of Economic Affairs (grant no. BSIK03024).

# References

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

[2] Robert H. Crites and Andrew G. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 8, pages 1017–1023, 1996.

[3] Nancy Fulda and Dan Ventura. Dynamic joint action perception for Q-learning agents. In *Proceedings of the 2003 International Conference on Machine Learning and Applications - ICMLA 2003*, pages 73–79, Los Angeles, California, USA, June 2003.

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

[5] Jelle R. Kok, Pieter Jan 't Hoen, Bram Bakker, and Nikos Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG'05)*, pages 29–36, Colchester, United Kingdom, April 2005.

[6] Michael L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328, San Francisco, California, US, June 2001. Morgan Kaufmann.

[7] Rob Powers and Yoav Shoham. New criteria and a new algorithm for learning in multi-agent systems. In *Advances in Neural Information Processing Systems*, number 17, pages 1089–1096. Cambridge, MA, 2005.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[9] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.