Tentative Exploration on Reinforcement Learning Algorithms for Stochastic Rewards

Luis Peña[1], Antonio LaTorre[2], José-María Peña[2], and Sascha Ossowski[1]

[1] Artificial Intelligence Department, Universidad Rey Juan Carlos
{luis.pena,sascha.ossowski}@urjc.es
[2] Computer Architecture Department, Universidad Politécnica de Madrid
{atorre,jmpena}@fi.upm.es

Abstract. This paper addresses a way to generate mixed strategies using reinforcement learning algorithms in domains with stochastic rewards. A new algorithm, based on Q-learning model, called TERSQ is introduced. As a difference from other approaches for stochastic scenarios, TERSQ uses a global exploration rate for all the state/actions in the same run. This exploration rate is selected at the beginning of each round, using a probabilistic distribution, which is updated once the run is finished. In this paper we compare TERSQ with similar approaches that use probability distributions depending on state-action pairs. Two experimental scenarios have been considered. First one deals with the problem of learning the optimal way to combine several evolutionary algorithms used simultaneously by a hybrid approach. In the second one, the objective is to learn the best strategy for a set of competing agents in combat-based videogame.[1]

1 Introduction

Stochastic games (SGs) and Markov decision processes (MDPs) have been studied in the literature as typical agent-based scenarios in which reinforcement learning (RL) has been successfully applied. Although SGs differ from MDPs in the existence of multiple agents performing simultaneous actions (and all their actions determine the next state), there are other multiagent problems, like matrix games (sometimes referred as strategic games), that also consider the existence of multiple actors in the problem. However, SGs and MDPs have one common characteristic: "*state transitions are non-deterministic*". Transitions depend on the action performed by the agent in MDPs, or the combination of actions of all agents in SGs, but in a stochastic way. This characteristic motivates the use of stochastic-specific learners [1,2,3].

[1] The present work has been partially funded by the Spanish Ministry of Science and Innovation through the projects THOMAS-COIN (grant TIN2006-14630-C03-02), PEO-HCDP (grant TIN2007-67148), and also with the Madrid Regional Education Ministry IV PRICT. This project has been carried out in collaboration of CeSViMa supercomputing center.

E. Corchado et al. (Eds.): HAIS 2009, LNAI 5572, pp. 336–343, 2009.

RL methods also use a reward function which evaluates the effect of carrying out one action in a given situation. This expected value is obtained by a reward function on the new state. In many RL problems, reward functions, as well as state transitions, are non-deterministic.

In this paper, a new RL algorithm is presented. This algorithm, named Tentative Exploration by Restricted Stochastic Quota (TERSQ), is compared against two state-of-the-art RL algorithms for stochastic problems. TERSQ optimizes a stochastic quota through different learning executions. This quota parametrizes a binomial decision process that selects whether the algorithm should perform a deterministic or a biased exploration. This quota acts as a variable learning rate in the RL algorithm. In order to evaluate TERSQ performance, two problems (one MDP and a SG) have been considered: (i) adaptative features for hybrid evolutionary algorithms, and (ii) behavior of competing characters in videogames.

The rest of the paper is organized as follows: section 2 reviews the related work on RL algorithms for stochastic games. Section 3 defines TERSQ algorithm. In section 4, the experimental scenarios for both problems are described. Finally, section 5 details the conclusions derived from this study.

2 Related Work

The objective of the agents is to select the best actions (best response) to maximize a reward function (attenuated by a factor of γ). In the context of non-deterministic learning scenarios, the agent should be able to identify mixed strategies which are functions that assign a probability distribution to agent's next action $\rho_i : \mathcal{S} \rightarrow PD(\mathcal{A}_i)$.

Despite deterministic RL algorithms (such as Q-learners) are not appropriate to deal with MDPs and SGs, some variants of these algorithms have been successfully applied on these scenarios. PHC and WoLF [1] are extensions to the Q-learner algorithm particularly designed to deal with stochastic scenarios.

In [4], the authors extends WoLF algorithm to incorporate the concept of Infinitesimal Gradient Ascent (IGA) presented by [5] to define the "wining" situations required to update the learning rate in WoLF. GIGA-WoLF [6] is an extension of the latter considering the concept of Generalized IGA [7]. BL-WoLF [2] is an enhanced version of WoLF that provides a bounded-loss where the cost of learning is measured by the losses the learning agent accrues (rather than the number of rounds). Another variant is Hyper-Q [3], in which values of mixed strategies rather than base actions are learned, and in which other agents strategies are estimated from observed actions via Bayesian inference. WPL (Weighted Policy Learner) [8] is a new RL algorithm which does not assumed an agent knew the underlying game structure.

3 TERSQ Algorithm

In this work we introduce the TERSQ algorithm for reinforcement learning environments. The main idea of this algorithm is to use a global stochastic quota,

σ, in order to select the action to be executed. A binomial decision process is performed in such a way that action with best Q-values are selected with a probability of σ while the rest of the actions are stochastically selected with a probability of $1 - \sigma$ according to their Q-value ranking.

Let
$\mathcal{A}$ be the set of possible actions for the state s, and $A_i \in \mathcal{A}$ one action for this state,
α, γ the learning parameters,
$\sigma \in \Gamma = \{0.0, 0.1, \ldots, 1.0\}$ the global quota used to select A_{max},
$\tau(\sigma)$ the average performance of σ

1. Let σ be selected from Γ following the specific criteria of the current phase.
2. Initialize $Q(A_i, s) = 0$.
3. Repeat until the round has finished:
   (a) For each action A_i on each state s, a basic probability $\pi(s, A_i)$ is obtained by a ranking process where actions are sorted according to their Q-values in an increasing order:

$$\{A'_i\} = sort(\{A_i\}) \tag{1}$$

$$\pi(s, \{A'_i\}) = i \times \pi_0 \quad / \quad \sum \pi(s, \{A'_i\}) = 1 \tag{2}$$

   (b) These probabilities are adjusted by the σ quota as follows,

$$\widehat{\pi(s, A_i)} = \pi(s, A_i) \times (1 - \sigma), \quad A_i \neq A_{max} \tag{3}$$

   and for the A_{max} (action with the best actual Q-value)

$$\widehat{\pi(s, A_{max})} = (\pi(s, A_{max}) \times (1 - \sigma)) + \sigma \tag{4}$$

   (c) Select action A_i with probability $\widehat{\pi(s, A_i)}$.
   (d) Q-values are updated observing reward r and next state s',

$$Q(A_i, s) = (1 - \alpha)Q(A_i, s) + \alpha \left(r + \gamma \max_{A_j} Q(A_j, s') \right)$$

4. Update the $\tau(\sigma)$ according to the evaluation of the round.

The σ value is selected for each round based on three different criteria. From these criteria, three phases can be established: (1) *Tentative Phase* in which the algorithm tries all the possible σ values (from a finite set of values, named Γ) to get an initial estimation of the performance of every possible σ value, (2) σ *Adjustment Phase* where σ values are proportionally chosen according to their average performance $\tau(\sigma)$ (which is updated at the end of each round), and (3) *Optimal* σ *Phase* where the σ value with highest average performance is selected

for the rest of the learning process. The usual Q-learning technique is applied during all the process.

4 Experimental Scenarios

4.1 Hybrid Evolutionary Algorithm

Preliminaries. According to Sinha and Goldberg [9], three are the main reasons for hybridization in Evolutionary Algorithms:

1. An improvement in the performance of the Evolutionary Algorithm (for example, the speed of convergence).
2. An improvement in the quality of the solutions obtained by the Evolutionary Algorithm.
3. To incorporate the Evolutionary Algorithm as a part of a larger system.

One alternative to deal with Hybrid Evolutionary Algorithms is a dynamically adjusted framework, named Multiple Offspring Sampling (MOS). This framework is able to simultaneously handle several evolutionary approaches to produce the new offspring and dynamically adjust the participation of each of these approaches according to their current performance. [10] provides a completed presentation of MOS.

Learning the Hybrid Strategy. When working with hybrid evolutionary algorithms it is hard to guess which is the best way to handle each of the different offspring mechanisms present in the hybrid approach. Static participation ratios could lead to suboptimal results and to a waste of resources (creation of solutions by means of algorithms with a poor performance). A dynamic adjustment of the participation of each of the reproductive techniques that compose the hybrid algorithm can solve most of these problems. However, several executions of the same dynamically adjusted hybrid algorithm can result in different curves of participation. At this point, the question is if it is possible to learn a nearly optimal way to adapt the participation of the different available techniques on the overall search process through different stages of the algorithm execution.

For this purpose, a hybrid evolutionary algorithm with reinforced-learning capabilities has been proposed. It is able to learn the optimal way for adapting participation by using one of the following RL algorithms: TERSQ, described in this contribution, Policy Hill Climbing (PHC) and WoLF [1]. In this algorithm, an action is the creation of an offspring individual by a particular reproductive mechanism. The set of possible states is defined by a discretized participation value for each of the evolutionary techniques. A state transition is performed when the ratio of individuals produced by a technique introduces a change in the discretized participation value.

Experimentation. For this experimentation two continuous optimization functions proposed for the CEC'08 Special Session and Competition on Large Scale Global Optimization [11] have been selected. Shifted Rastrigin's function is a multi-modal, shifted and separable function with a huge number of local optima. Schwefel's Problem is an unimodal, shifted and non-separable function. Both are good examples of hard optimization functions were a hybrid evolutionary approach can be successfully exploited to obtain better results than with single algorithms.

	UCUM	BCUM	UCGM	BCGM
Evolutionary Model	GA			
Initializer	Uniform			
Crossover	Uniform	BLX-α	Uniform	BLX-α
Mutator	Uniform		Gaussian	

Previous table presents the set of techniques used by the hybrid evolutionary algorithm. These four reproductive mechanisms are simultaneously used by the hybrid algorithm. Each time an individual is created, this action is recorded and the Q-values updated. The three aforementioned RL algorithms are tested on the two proposed functions. For the TERSQ algorithm, 11 rounds are performed in the Tentative Phase, 50 rounds in the σ Adjustment Phase and 50 rounds in the σ Optimal phase. For the other two algorithms, PHC and WoLF, 111 rounds are executed.

Phase	PHC	WoLF	TERSQ	
Tentative	–	–	1,97E-01	Rastrigin
	–	–	5,87E+00	Schwefel
σ Adjustment	–	–	1,96E-01	Rastrigin
	–	–	5,69E+00	Schwefel
σ Optimal	3,81E-01	2,22E-01	1,76E-01	Rastrigin
	7,87E+00	7,16E+00	5,64E+00	Schwefel

This table presents the results obtained by the three RL algorithms. From these results, we can observe that the average error reported by the TERSQ algorithm is smaller in both functions even in the Tentative Phase. Moreover, the TERSQ algorithm is able to improve its results in both the σ Adjustment and the σ Optimal phases, obtaining average errors 54% and 21% smaller compared to PHC and WoLF, respectively, on the Rastrigin function, and 28% and 21% smaller on the Schwefel Problem.

4.2 Videogame Characters

Environment Description. The second experiment is the combat between two characters to bring enemy's Hit Points counter to 0. Every character has two state counters, Hit Points (`HPs`) that represents the remaining life for this character and the Exahustion Points (`EPs`) counter that shows the fatigue level of the character. If HPs reach 0 the character is dead and thus it is defeated. On

the other hand, if EPs are below 0 the character cannot do anything but rest until it is recovered.

A character can perform two types of actions: Offensive and Defensive Actions. Offensive Actions take a fixed amount of time to be executed, named Action Points (`APs`), which represent the time that the action takes to be triggered after it is called. In addition, Offensive Actions consume some EPs when they are triggered. Once an Offensive Action is fired it has a probability of hitting the target and inflicting some damage. The damage of an action can be of three types: (1) HPs damage, (2) EPs damage and (3) Stun damage. The first two damage types represent a direct amount to be substracted to the respective counter of the enemy. The stun damage works in a different way: this type of damage makes the target to cancel his present declared action and makes that the target cannot declare any other action until he gets recovered from the stun.

Table 1. Results for videogames characters scenario

Wins							
		A-TERSQ	A-PHC	A-WoLF	B-TERSQ	B-PHC	B-WoLF
Vs	A-TERSQ		50,56%	50,44%	44,61%	39,32%	38,38%
	A-PHC	49,44%		50,18%	36,49%	34,91%	33,32%
	A-WoLF	49,56%	49,82%		35,87%	34,57%	33,13%
	B-TERSQ	55,39%	63,51%	64,13%		53,94%	53,02%
	B-PHC	60,68%	65,09%	65,43%	46,06%		49,42%
	B-WoLF	61,62%	66,68%	66,87%	46,98%	50,58%	
	N Wins	55,33%	59,19%	59,45%	41,96%	42,55%	41,34%

(a) Ratio of wins versus other characters

		A-TERSQ	A-PHC	A-WoLF	B-TERSQ	B-PHC	B-WoLF
Vs	A-TERSQ		37,38%	36,52%	52,23%	33,96%	33,05%
	A-PHC	62,62%		49,47%	49,07%	41,62%	40,52%
	A-WoLF	63,48%	50,53%		49,91%	42,93%	41,58%
	B-TERSQ	47,77%	50,93%	50,09%		51,71%	48,71%
	B-PHC	66,04%	58,38%	57,07%	48,29%		48,98%
	B-WoLF	66,95%	59,48%	58,42%	51,29%	51,02%	
	N Wins	61,56%	51,53%	50,47%	50,13%	44,27%	42,61%

(b) Results on fixed σ stage

APs of Defensive Actions represent the time the defense is active when it is declared, consuming the EPs when it finishes. If a character is hit and he has a Defensive Action declared, he has a probability of blocking the attack. If the Defensive Action blocks the attack, the damage taken by the character is reduced by a factor applied to the HPs and EPs damages and, if the Defensive Action specifies it, to the stun damage.

Once the combat begins, every character choices which action he wants to declare and the APs are added to the current instant counter. When an action is triggered, it is resolved depending on its type, and then the character choices another action if he has any remaining EPs. If he has none, the character must rest for a fixed amount of time to recover some EPs.

Experimentation. To evaluate the TERSQ algorithm on this environment, two different character profiles, A and B, have been created. Each profile defines specific HPs, EPs and action characteristics. For each of these profiles, three RL algorithms are used: TERSQ, PHC and WoLF resulting six different characters (each of the two profiles and each of the three RL algorithms). The experiment consists of 10 executions of 200000 combats (rounds). For each combat, two characters are randomly selected from the six available characters. The Q-values and the learning rates are reseted when each execution begins.

Table 1.a) presents the winning percentage for each pair of characters averaged for the 10 executions. These results show a better performance of PHC or WoLF for each of the character profiles.

Table 1.b) shows the winning ratios restricted to the last combats of every experiment, once σ has been selected. These last results emphasize a significative better performance of the two characters controlled by TERSQ algorithm. The two character profiles are different: **B** is worse than **A**, but despite of this B-TERSQ character profile beats nearly half of times against **A**-profiles, outperforming others **B**-profiles.

The figure 1 shows the evolution of the winnig ratio along the combats with the inflexion points that marks the differents phases at 2000 (end of Tentative) and 120000 (end of σ Adjustment).

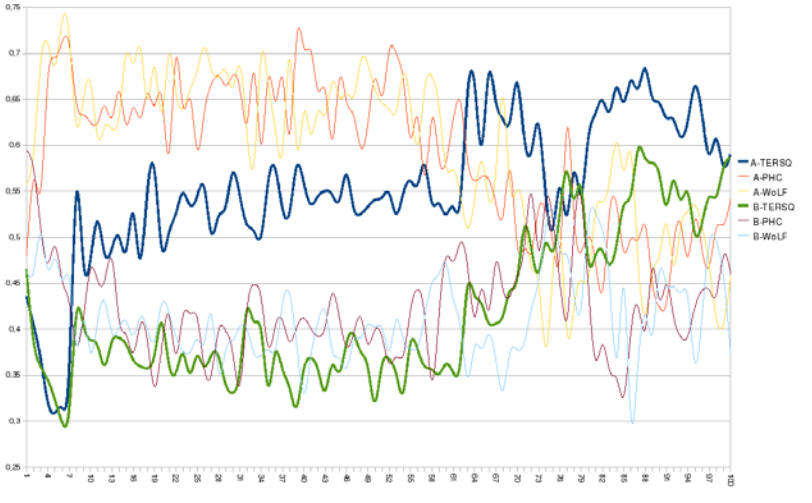

Fig. 1. Ratio evolution across the combats, sampled by 2000

5 Conclusions

In this paper, a new RL algorithm named TERSQ has been presented. This algorithm differs from PHC and WoLF in how the selection probability for each state-action pair is computed. PHC and WoLF maintain a separate matrix for

these probabilities. This forces these two algorithms to learn both the attenuated reward value (Q-value) and the selection probability. TERSQ implements a mechanism for directly computing this probability from the Q-values using a global quota, σ. Two different experimental scenarios have been proposed. In the first scenario, the proposed RL algorithm shows better average performance on the two proposed functions. Furthermore, TERSQ is able to improve its own average performance in each of the subsequent phases. In the second scenario, once the optimal σ value has been selected, its results clearly outperform those of the competing algorithm. This behavior can be explained as a more explorative training phase, in which results are not very good, provides better information that can be exploited in the last phase.

References

1. Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. Artificial Intelligence 136, 215–250 (2002)
2. Conitzer, V., Sandholm, T.: Bl-wolf: A framework for loss-bounded learnability in zero-sum games. In: International Conference on Machine Learning (ICML), pp. 91–98 (2003)
3. Tesauro, G.: Extending q-learning to general adaptive multi-agent systems. In: Advances in Neural Information Processing Systems, vol. 16, p. 2004. MIT Press, Cambridge (2004)
4. Bowling, M., Veloso, M.: Convergence of gradient dynamics with a variable learning rate. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 27–34. Morgan Kaufmann, San Francisco (2001)
5. Singh, S., Kearns, M., Mansour, Y.: Nash convergence of gradient dynamics in general-sum games. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 541–548. Morgan Kaufmann, San Francisco (2000)
6. Bowling, M.: Convergence and no-regret in multiagent learning. In: Advances in Neural Information Processing Systems, vol. 17, pp. 209–216. MIT Press, Cambridge (2005)
7. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 928–936 (2003)
8. Abdallah, S., Lesser, V.: A Multiagent Reinforcement Learning Algorithm with Non-linear Dynamics. Journal of Artificial Intelligence Research 33, 521–549 (2008)
9. Sinha, A., Goldberg, D.: A survey of hybrid genetic and evolutionary algorithms. Technical Report 2003004, Illinois Genetic Algorithms Laboratory, IlliGAL (2003)
10. LaTorre, A., Peña, J., González, S., Robles, V., Famili, F.: Breast cancer biomarker selection using multiple offspring sampling. In: Proceedings of the ECML/PKDD 2007 Workshop on Data Mining in Functional Genomics and Proteomics: Current Trends and Future Directions, Warsaw, Poland. Springer, Heidelberg (2007)
11. Tang, K., Yao, X., Suganthan, P., MacNish, C., Chen, Y., Chen, C., Yang, Z.: Benchmark functions for the cec 2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China (2007)