

Coordinated Exploration in Multi-Agent Reinforcement Learning: An Application to Load-balancing

Katja Verbeeck
Computational Modeling Lab
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussel,
Belgium
kaverbee@vub.ac.be

Ann Nowé
Computational Modeling Lab
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussel,
Belgium
ann.nowe@vub.ac.be

Karl Tuyls
Theoretical Computer Science
Group
University of Limburg
Diepenbeek, Belgium
Karl.Tuyls@luc.ac.be

ABSTRACT

This paper is concerned with how multi-agent reinforcement learning algorithms can practically be applied to real-life problems. Recently, a new coordinated multi-agent exploration mechanism, called Exploring Selfish Reinforcement Learning (ESRL) was proposed. With this mechanism, a group of independent agents can find optimal fair solutions in multi-agent problems, without the need for modeling other agents, without the need of knowing the type of the multi-agent problem confronted with and by using only a limited form of communication. In particular, the mechanism allows for using natural reinforcement signals coming from the application itself. We report on how ESRL agents can solve the problem of load-balancing in a natural way, both as a common interest and as a conflicting interest problem.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems*

General Terms

Algorithms

Keywords

Reinforcement Learning, Exploration, Load-balancing

1. INTRODUCTION

A central issue in applying Reinforcement Learning (RL) to real-life problems is how to set-up an appropriate reinforcement signal so that the RL agent is able to learn to behave optimal. In a multi-agent environment, this issue is complicated by the non-stationarity created by the collective of agents. The feedback an agent experiences is usually influenced by the other agents. In contrast to single-agent

environments, uncontrolled exploration in a multi-agent setting will not lead to a better understanding of the environment and may even lead to policy oscillations. Recently, a MARL technique that focuses on a coordinated form of exploration was described, i.e. Exploring Selfish Reinforcement Learning (ESRL) [3]. ESRL imposes little assumptions upon the agents and the external environment. Unique is also that the agents do not need to know beforehand which type of problem they are trying to solve, i.e. agents may have common interests or not. In the following, we describe how ESRL agents organize their exploration together in an efficient way. Next we illustrate how ESRL can naturally be used in practice on the problem of load-balancing.

2. COORDINATED EXPLORATION

The main idea of ESRL is that the group of agents explores the joint action space of a stochastic game as efficiently as possible by shrinking it temporarily during different periods of play. During what is called the exploration phase, we let the team of agents converge to an attractor point of the system. ESRL agents are based on learning automata [1] and LA convergence results guarantee that ESRL agents will converge to an interesting attractor point of the stochastic game. As a consequence of the LA update scheme we use this attractor point is a pure joint action that in case pure Nash equilibria exists is Nash. After this convergence, a synchronization phase takes place, i.e. the agents collect information on the attractor they have reached and all agents temporarily exclude the private action they have converged to from their private action space. As such the attractor that was reached is removed from the joint action space and the team of agents gets the opportunity to look for possibly better solutions in a reduced space. After playing enough exploration/synchronization phases¹, the team can jointly find the best attractor point(s) that was(were) visited before. Of course, which attractor is best depends on the type of game the agents play. We assume that the agents do not know the type of game they play and thus initially the agents take into account that others may have conflicting interests. Figure 1 depicts the general exploration scheme for an individual ESRL agent. For a detailed overview of the ESRL mechanism we refer the reader to [2].

¹When the agents' private action space gets empty, all temporarily excluded actions are restored, action probabilities initialized and learning can continue.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

Copyright 2005 ACM 1-59593-094-9/05/0007 ...\$5.00.

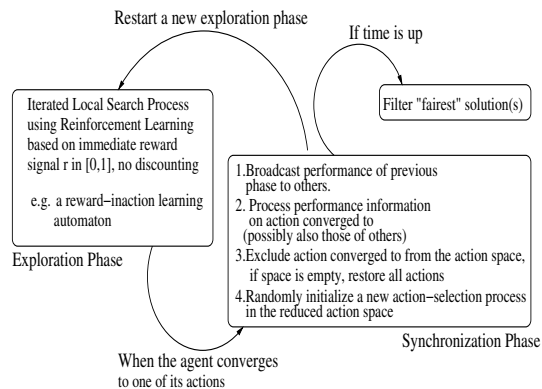


Figure 1: The general exploration scheme for a single ESRL-agent.

Table 1: Average gain in master idle time and total blocking time.

#slaves	#actions	% idle gain	% blocking gain
2	3	55.6	50.7
2	4	54.3	51.3
2	5	47.7	38.7
3	3	30.3	20.7
3	5	26.8	20.7
5	3	4.0	-3.4
5	5	0.98	3.20
7	3	9.70	3.00
7	5	11.6	3.00

3. A COMMON INTEREST PROBLEM

Load balancing is crucial for parallel applications since it ensures a good use of the capacity of the parallel processing units. A traditional architecture to use, is the master-slave software architecture. In this architecture the slaves (one per processing unit) repeatedly request a chunk of a certain size from the master. As soon as the data has been processed the result is transferred to the master and the slave sends a new request. This classical approach is called job farming. Instead of having all slaves asking the same chunk size of data to the master, in our experiment ESRL slaves will learn to request a suitable chunk size, so as to minimize the total computation time of the parallel application. This in turn ensures a reduced total idle time of the master and reduced blocking times for the slaves. The action set of the agents consists of a discrete number of chunk sizes the slave can request. All possible combinations constitute the joint action space. The reward collected during an exploration phase is the overall computation time needed for the data to be processed by the parallel system during this phase. This reward is the same for all agents, and thus the problem is translated as a common interest game. Table 1 gives an overview of the gain in blocking overhead when using ESRL slaves. The idle time given, concerns the idle time of the master, the blocking time represents the cumulated blocking time of all the slaves.

4. A CONFLICTING INTEREST PROBLEM

In our second load-balancing experiment, a group of autonomous agents, that generate jobs, have to learn to select

passive processors who will execute these jobs. The objective is to learn a policy so that the load is nicely distributed and the overall job around time of all the jobs generated in the system is minimized. In our setting an agent has only two action choices. Either it uses a private processor, or it chooses a public processor, which is faster but which can be overloaded by the group. Therefore, the agents have to develop a resource selection behavior complementary to the behavior of others and to influencing factors such as load and resource capacities. The immediate reinforcement a processor sends back to an agent is the time a packet, coming from that agent, needed to be processed. Note that reinforcements are delayed, jobs are created probabilistically and also processor rates follow an exponential distribution. Figure 2 shows that using ESRL, payoffs are equalized and the obtained average is better than what the agents would get if they always choose their private processor.

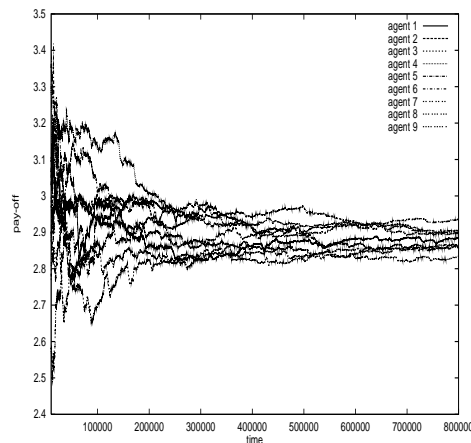


Figure 2: Average job around time for 9 ESRL agents.

5. CONCLUDING REMARKS

ESRL can handle asynchronous action selection and delayed rewards in a very natural way. During the exploration phases, agents are selfish reinforcement learners and able to cope with asynchronism in both the action selection as the reward signal. What is important for ESRL agents to know is when the other agents have converged. In practice, the agents just wait until they have received all broadcast messages from the others, so that they can start their synchronization phase together.

6. REFERENCES

- [1] K. Narendra and M. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall International, Inc, 1989.
- [2] K. Verbeeck. *Coordinated Exploration in Multi-Agent Reinforcement Learning*. PhD thesis, Computational Modeling Lab, Vrije Universiteit Brussel, Belgium, 2004.
- [3] K. Verbeeck, A. Nowe, M. Peeters, and K. Tuyls. Multi-agent reinforcement learning in stochastic single and multi-stage games. In *Kudenko et al (Eds): Adaptive Agents and Multi-Agent Systems II*, pages 275–294. Springer LNAI 3394, 2005.