Learning to Weigh Basic Behaviors in Scalable Agents

Olivier Buffet Alain Dutech François Charpillet

{buffet,dutech,charp}@loria.fr
http://www.loria.fr/~{buffet,dutech,charp}

LORIA, BP 239
F-54506 Vandœuvre-lès-Nancy

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent systems*; I.2.6 [**Computing Methodologies**]: Learning

General Terms

Algorithms, Design

1. INTRODUCTION

We are working on the use of Reinforcement Learning (RL)[3] algorithms to design automatically *reactive situated* agents limited to only *local perceptions*. Unfortunately, as good RL algorithms suffer from combinatorial explosion, their use is generally limited to simple problems.

As shown on the tile-world example of figure 1, we propose to overcome these difficulties by making the hypothesis, as in Brook's subsumption architecture [1], that a complex problem can be efficiently dealt with if considered as a combination of simple problems.

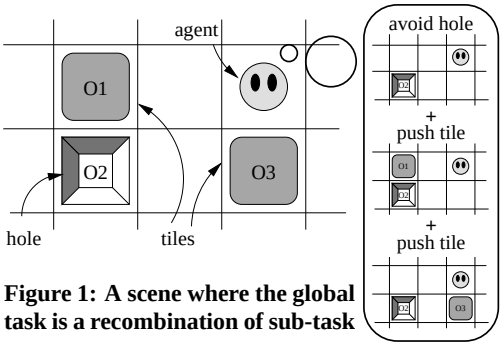

Figure 1: A scene where the global task is a recombination of sub-task

Thus, we propose here to *recombine behaviors* that have already been learned by reinforcement for each simple problem, trying to be both *adaptive* and *scalable*.

AAMAS'02, July 15-19, 2002, Bologna, Italy.

This short presentation gives basic ideas on RL algorithms (section 2). Then the three steps of our method are presented: how basic behaviors are learned for each basic motivation (sec. 3), how the scene is decomposed in key figures to find the basic behaviors currently involved (sec. 4), and how to combine them into a complex global behavior using learned weights (sec. 5). A few words are given on the experiments conducted on the tile-world problem (sec. 6) and precede a conclusion.

2. REINFORCEMENT LEARNING AND LIMITATIONS

Reinforcement Learning (RL) methods are very appealing ways to have agents learn optimal reactive behaviors, as only a scalar feedback from the system to the agents is required.

But the convergence of RL algorithms (like *Q-Learning* or $TD(\lambda)$) has only been proven for Markov Decision Processes (MDP). A MDP is defined as a $\langle \mathcal{S}, \mathcal{A}, T, r \rangle$ tuple, $\mathcal{S}$ being a finite set of states and $\mathcal{A}$ a finite set of actions. When the system is in given state s, an action a being chosen, the probability for the system to end in state s' is given by $T(s, a, s')$. After each transition, the environment generates a reward $r(s, a)$. The problem is then to find the optimal mapping $\pi(s, a)$ between states and actions so as to maximize the reward received over time, usually expressed as a utility function $Q(s, a) = \sum_{t=0}^{\infty} \gamma^t (r_t | s_0 = s, a_0 = a)$. Such a mapping is called a policy and, for a MDP, it is well known that an optimal *deterministic* policy exists.

As our agent only has a partial view of its environment, we abusively make the assumption that the agent faces a Markovian problem. This weak approximation is partially corrected since the learning algorithms used look for *stochastic* policies, which are prone to give better solutions. Nevertheless, the problem of combinatorial explosion remains, even though the locality of the agent's perceptions helps reducing the size of the state-space.

3. BASIC BEHAVIORS

Each basic behavior b is linked to a basic motivation, and thus to a set of types of objects concerned by this motivation. This set is called a **type of configuration** $\mathcal{C}^T(b)$. In a given scene, many configurations (sets of objects) may have the same type, leading to many uses of the same basic behavior (in figure 1 are two occurences of the `[push tile]` behavior).

Moreover, a basic behavior is defined by a stochastic decision policy learned by reinforcement and the utility of this policy. These last elements are stored in two tables [1]:

[1] Both tables have the same definition set $\mathcal{C}^T(b) \times \mathcal{A}$

- $P_b(c, a)$: the probability to choose action a while seeing configuration c, and
- $Q_b(c, a)$: the expected discounted reward when choosing action a for the configuration c.

4. SCENE DECOMPOSITION

When confronted to a complex situation, an agent has to decompose its observation of the scene o into "useful" configurations related to known basic behaviors.

In fact, for each behavior $b \in \mathcal{B}$, the agent will look for all configurations c related to behavior b in the observation o (this set is called $\mathcal{C}(b, o)$). Scalability derives mainly from the fact that one basic behavior (resp. configuration) can be associated to more than one configuration (resp. basic behavior). This is all the more interesting than, due to the locality of perceptions, the number of useful configurations changes.

In the tile-world scene presented on figure 1, the agent's perceptions concern objects O_1, O_2 and O_3. With two possible behaviors: *a*voiding the holes (b_a) associated to `(hole)` and *p*ushing blocs in those holes (b_p) associated to `(hole,tile)`, the agent has to take into account three different $(behavior, configuration)$ pairs: $(b_a, \{O_2\})$, $(b_p, \{O_1, O_2\})$ and $(b_p, \{O_3, O_2\})$.

5. BASIC BEHAVIORS COMBINATION

To combine our basic behaviors in a complex one, we propose to simply compute a new policy $\mathcal{P}(o, a)$ as a linear combination of the $P_.$ policies (linked to configurations present in current perceptions) weighted by a function of the Q-values. In this paper, we just present one of the many possible formulas:

$$\mathcal{P}(o, a) = \frac{1}{K} \sum_{b \in \mathcal{B}} e^{\zeta_b} [\sum_{c \in \mathcal{C}(b,o)} W_b(c).P_b(c, a)]$$

Each occurence of a basic behavior b has a weight proportionnal to $W_b(c) = max_a|Q_b(c, a)|$ (gives importance to policies linked to negative as to positive reward) and e^{ζ_b}, where ζ_b is learned by a gradient ascent to adjust the W_b functions [2].

6. EXPERIMENTS

The tile-world problem also illustrates our experiments. Thus, to evaluate the method presented, we had to compare global behaviors learned tabula rasa by an agent to global behaviors obtained by recombination of previously learned basic behaviors (*hunt* and *avoid*).

Due to convergence difficulties, we could learn a policy through a gradient ascent in the `[1 tile/1 hole]` case only. In the other situations, only a modified Boltzmann Q-learning gave us satisfying results, although possibly far from optimum solutions.

These behaviors have to be compared with the results of our method, for which the two basic behaviors have been learned previously. These basic behaviors' recombinations are compared to the policies learned tabula rasa in table 1, where the weights ζ have been learned. The use of noisy behaviors in column **c(2)** is a simple way to overcome some blocking situations (see figure 2) and improve the agent's efficiency.

The quality of the solutions is very satisfying if you consider that learning the weights takes less than 10000 simulation steps (that prevents us from having significant graphics) and that the average reward reached is not far, and sometimes better, than the one obtained with the classical approach.

[2]Only one weight for each behavior, regardless of its number of occurences.

objects		reward (for 10000 steps)		
#tiles	#holes	**c(1)**	**c(2)**	tabula rasa
1	1	938	745	**1200**
2	1	110	**444**	**300**
1	2	~ 0	**90**	~ 0
2	2	180	**277**	**200**

Table 1: Comparative table between policies obtained by recombination and tabula rasa

- **c(1)**: simple recombination
- **c(2)**: recombination of more noisy basic behaviors

The same efficiencies of recombined policies were also obtained when just reusing weights learned in the `[1 tile/1 hole]` case for environments with more objects, showing that the scalability of the behaviors' weights is really satisfying in this problem. Even if the average reward largely decreases when adding more tiles, the learned weights can be reused directly.

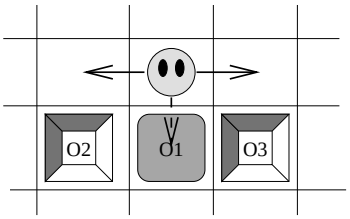

Figure 2: `[1 tile/2 holes]`: the difficulty

7. CONCLUSION AND FURTHER WORKS

Further studies and experiments on this subject have been conducted. A part of them are detailed in [2], along with a more complete description of our methodology.

8. REFERENCES

[1] R. Brooks. A robust layered control system for a mobile robot. Technical report, September 1985. A.I. Memo No.864.

[2] O. Buffet, A. Dutech, and F. Charpillet. Adaptive combination of behaviors in an agent. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'02)*, 2002. [to appear].

[3] R. Sutton and G. Barto. *Reinforcement Learning*. Bradford Book, MIT Press, 1998.