On the Characteristics of Sequential Decision Problems and their Impact on Evolutionary Computation

[Extended Abstract]

André M. S. Barreto
Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil
amsb@lncc.br

Douglas A. Augusto
COPPE—Universidade Federal do Rio de Janeiro
Rio de Janeiro, RJ, Brazil
douglas@coc.ufrj.br

Helio J. C. Barbosa
Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil
hcbm@lncc.br

ABSTRACT

In this paper we argue that the performance of evolutionary computation on sequential decision problems strongly depends on the characteristics of the task at hand. On "error-avoidance" tasks, in which the decision process is interrupted every time a bad decision is made, evolutionary methods usually perform well. However, the same is not true for "goal-seeking" tasks, in which the objective is to find one or more target locations. In this case, it is not clear how to evaluate the unsuccessful candidate solutions, and the performance of evolutionary computation may depend on prior knowledge about the problem. Even though the hypothesis of this paper is essentially a conceptual one, we support our ideas with a computational experiment.

Categories and Subject Descriptors: I.2.8 [Computing Methodologies]: Solving, Control Methods, and Search

General Terms: Performance

Keywords: Sequential Decision Problems, Evolutionary Computation, Reinforcement Learning

1. INTRODUCTION

One of the main issues regarding the solution of sequential decision tasks is the so-called *temporal credit-assignment problem*: how to apportion credit to individual decisions by looking at the outcome of a sequence of them [2]. Evolutionary methods perform the temporal credit-assignment implicitly, by evaluating each decision policy as a whole and then combining the most successful ones in the hope that better policies will come forth [1]. In this work we argue that this strategy is not equally effective on all kinds of sequential decision tasks. In particular, we identify two categories of decision problems in which the performance of evolutionary computation is quite different.

We call *error-avoidance tasks* problems in which the decision process lasts until the agent makes a mistake. The task of balancing a pole hinged to a cart, a classic control problem used for years as a benchmark, is an example of this type of decision task [4]. Evolutionary methods perform well on error-avoidance tasks because it is usually straightforward to evaluate poor decision policies in these problems. For example, in the pole-balancing task one can easily rank the unsuccessful candidate policies based on the number of steps they could balance the pole for. However, there also exist problems where bad policies cannot be ranked in a meaningful way. This is the case of *goal-seeking tasks*, in which the objective is to find a goal region whose location is not known beforehand (like in a maze, for instance). Since in these problems it is not possible to compute the distance from a given state to the goal, it is not clear how to differentiate between decision policies that never reach it.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
ACM 978-1-60558-325-9/09/07.

2. MAIN HYPOTHESIS

Error-avoidance tasks present two characteristics that make them particularly favorable for the type of search carried out by evolutionary computation. Specifically, it is fairly easy to evaluate candidate solutions in these problems because:

1. One can rank unsuccessful decision policies based on the amount of time they could avoid making a mistake.
2. Usually, the worse the performance of a candidate decision policy, the faster its evaluation.

It is not hard to see that item 1 is absolutely fundamental for the evolutionary process: if candidate decision policies that have failed in the task cannot be ranked, there is no selective pressure towards better solutions. Although not essential, item 2 is also very desirable, since it induces a smart allocation of computational resources. When item 2 is true, the evolutionary process will quickly evaluate a large number of poor solutions and eventually focus on individuals that are worth the computational effort.

Unfortunately, not all sequential decision problems have the properties listed above. As described, in goal-seeking tasks one is interested in finding a decision policy able to reach a specific set of states. This makes it hard for optimization techniques in general, and evolutionary methods in particular, to differentiate between candidate solutions that are unable to accomplish the task. Notice that this question is not restricted to any specific task, neither is it related to design choices such as the type of representation or genetic operators used by the evolutionary method. In fact, this issue is intrinsic to any goal-seeking task and any optimization method which bases its search on the relative merit of candidate solutions. Since the information that really matters is not available during the search, one must find an alternative source of information that provides an estimate of a decision policy's quality—that is, of how far it is from completing the task. This information may or may not be available.

3. ILLUSTRATIVE EXPERIMENT

The central argument of this work is a conceptual issue and as such it can be intuitively understood. For the sake of completeness, though, we present in this section a computational experiment providing evidence for our hypothesis.[1] In particular, we present a sequential decision task that can be easily configured as either an error-avoidance or a goal-seeking task. We show how the performance of a genetic algorithm degenerates when one switches from one version of the problem to the other. In order to put the results of the genetic algorithm into perspective, we contrast them with the results obtained by Q-learning, a popular reinforcement-learning algorithm commonly used to solve sequential decision problems [3].

In the decision task we propose an agent must learn how to perform by directly interacting with a two-dimensional maze. In order to get more reliable results, instead of using one specific maze we tested the algorithms on a set of mazes with similar characteristics. All mazes were based on $n \times n$ grids and had one entrance and one exit. The entrance was constrained to be on the upper-left corner of the maze, while the exit was always on the bottom-right corner. The mazes were "perfect" or "simply connected," meaning that there was one and only one path from each state to any other state of the maze.

We used two different formulations of the task. In the first one the agent must find *the longest path* it can travel from the entrance of the maze without visiting the same state twice (thus, in this version of the task the exit of the maze is ignored). Also, every time the agent hits a wall it is repositioned at the entrance of the maze. This configuration of the problem results in an error-avoidance decision-task. For this version of the problem the fitness of a candidate solution was defined as the number of steps performed until it hits a wall or returns to a previously visited state.

In the second version of the problem the agent must find the path from the entrance to the exit of the maze, and it remains at the same state when it runs into a wall. Obviously, this is a goal-seeking task. Since in this problem it is not possible to compute the distance from a particular state to the goal, the evaluation of a candidate policy was based on the number of steps it performed inside the maze. This strategy is a good example of how prior knowledge about the problem can be incorporated into an evolutionary method: since one knows the entrance and the exit of the mazes are always located at opposite corners, one can favor those decision policies that travel farther into the maze. Even though this fitness function does not reflect the true objective of the problem, it is not clear how to do better with the information available on the task.

We fixed an upper bound of $n^2 \times 25,000$ transitions for the genetic algorithm and Q-learning to find a solution for each version of the problem. Table 1 and Figure 1 show the results obtained by both methods on the experiments. The algorithms were executed using a set of parameters specifically optimized for each configuration of the task. The results correspond to an average obtained over 150 mazes of each dimension. As shown in Table 1, in the first version of the problem both methods perform similarly. In contrast, when the problem is formulated as a goal-seeking task, the reinforcement-learning algorithm significantly outperforms the evolutionary method (see Figure 1). These results clearly support our hypothesis.

[1]Due to space limitation we are unable to give all the details of the experiment. A longer version of the paper with a full description of the experiment will be made available soon.

Table 1: Length of the Solution Found Relative to the Longest Path (%). S.D. = "standard deviation"

Size	Mean	Max.	Min.	S.D.
	Genetic algorithm			
5×5	99.25	100.00	80.00	2.96
10×10	81.38	100.00	18.52	20.81
15×15	57.38	92.98	12.32	18.14
20×20	37.61	87.80	6.44	16.86
	Q-learning			
5×5	96.11	100.00	61.11	7.96
10×10	80.35	100.00	35.59	16.40
15×15	52.54	100.00	16.67	16.79
20×20	34.99	82.93	12.83	11.55

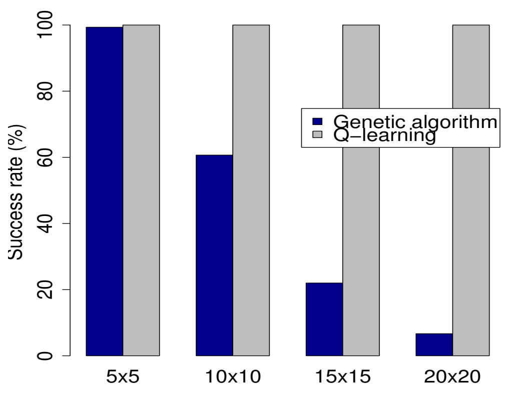

Figure 1: Proportion of Runs in Which the Agent Learned How to Escape from the Maze

4. REFERENCES

[1] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

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

[3] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, England, 1989.

[4] D. Whitley, S. Dominic, R. Das, and C. W. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13(2-3):259–284, 1993.

Acknowledgments

The authors would like to thank the support provided by "Conselho Nacional de Desenvolvimento Científico e Tecnológico" (CNPq) and "Agência Nacional do Petróleo" (ANP).