Designing Artificial Tetris Players With Evolution Strategies and Racing

Amine Boumaza
Univ. Lille Nord de France, F-59000 Lille
ULCO, LISIC, F-62100 Calais, France
boumaza@lisic.univ-littoral.fr

ABSTRACT

This article describes how racing procedures in evolution strategies can help reduce the number of evaluations. This idea is illustrated on learning Tetris players which can be addressed as a stochastic optimization problem. Different experiments show the benefits of the racing procedures in evolution strategies which can significantly reduce the number of evaluations.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms

Keywords

Evolution strategies, racing algorithms, Tetris players

1. INTRODUCTION

Designing artificial game players has been, and still is, studied extensively in the artificial intelligence community. Among these games, Tetris [3] is a single player game where the goal is to place randomly falling pieces onto a 10×20 game board. Each completed horizontal line is cleared from the board and scores points to the player and the goal is to clear as much lines as possible before the board is filled.

In the literature, artificial Tetris players use evaluation functions that evaluate future actions by assigning numerical values to game states. This evaluation function is generally a weighed linear combination of some feature functions. In the present work we choose the set of eight features used in [6] and [2]. To our knowledge these are the best performing players to this date.

In both of these studies, authors report long evaluation times. Since the score is a random variable, evaluations need to be performed on a series of games, which may lead to long evaluation times. One way to overcome this problem is to take into account confidence bounds around the estimated fitness value using racing methods [1]. These methods adapts the number of evaluations dynamically until the algorithm reaches a reliable ranking in the population. [5] proposed to use such methods with CMA-ES[4] and reported some success on different machine learning problems. We propose to do the same for learning Tetris.

The racing procedure we used is inspired from [5]. At each iteration of CMA-ES, the λ offspring undergo multiple evaluations until either : 1) μ outstanding individuals get out of the race, in which case we are sure (with probability $(1-\delta)$) that they are distinct. Or 2) the number of evaluations reaches some upper limit r_{limit}. Furthermore, if the actual number of evaluations necessary to distinguish μ individuals is less than the limit r_{limit}, this one is reduced: $r_{limit} := \max(1/\alpha\, r_{limit}, 3)$ with $\alpha > 1$, and 3 is the minimum number of evaluations. On the other hand if r_{limit} evaluations where not enough to distinguish μ individuals, then r_{limit} is increased: $r_{limit} := \min(\alpha\, r_{limit}, r_{max})$ where r_{max} is a maximum number of evaluations. Initially $r_{limit} = 3$ and it is adapted at each iteration of CMA-ES using the above two rules.

After r re-evaluations, the empirical bounds $c_{i,r}$ around the mean fitness of each individual $\hat{X}_{i,r} = \frac{1}{r}\sum_{j=1}^{r} f^j(x_i)$ where $i = 1\ldots\lambda$, are computed with:

$$c^h_{i,r} = R\sqrt{\frac{\log(2n_b) - \log(\delta)}{2r}} \tag{1}$$

using the Hoeffding bound and

$$c^b_{i,r} = \hat{\sigma}_{i,r}\sqrt{2\frac{\log(3n_b) - \log(\delta)}{r}} + 3R\frac{\log(3n_b) - \log(\delta)}{r} \tag{2}$$

using Bernstein. Where $n_b \leq \lambda r_{limit}$ is the number of evaluations in the current race, $\hat{\sigma}_{i,r} = \sqrt{\frac{1}{r}\sum_{j=1}^{r}\left(f^j(x_i) - \hat{X}_{i,r}\right)^2}$ is the standard deviation of the fitness for individual x_i, and $(1-\delta)$ is the level of confidence we fix. $R = |a-b|$ such that the fitness values of the offspring are almost surely between a and b.

After each evaluation in the race the lower bounds $lb_{i,r}$ and the upper bounds $ub_{i,r}$ around the mean of each offspring are updated to the tightest values:

$$lb_{i,r} = max\left(lb_{i,r-1}, \hat{X}_{i,r} - c^{h/b}_{i,r}\right)$$

and

$$ub_{i,r} = min\left(lb_{i,r-1}, \hat{X}_{i,r} + c^{h/b}_{i,r}\right)$$

Beside the above empirical bounds, there exists for the game of Tetris estimated bounds on the mean score of the player [6]. We have: $\frac{|X-\hat{X}|}{\hat{X}} \leq \frac{2}{\sqrt{n}}$ with probability 0.95, where $\hat{X}$ is the average score of the player on n games and X is the expected score. We will refer to this bound as the Tetris bound.

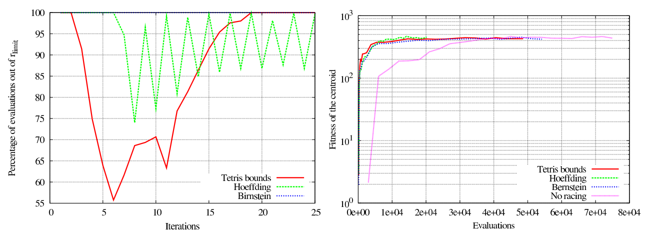

Figure 1: Left : the number of evaluations with racing using different bounds versus iterations. Right : fitness of the mean vector versus the number of evaluations.

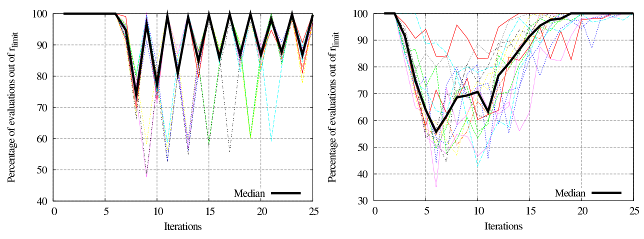

Figure 2: The number of evaluations for 16 runs. The thick line represents the median value. Left:with Hoeffding races. Right:with Tetris bounds.

2. EXPERIMENTS

In all experiments the initial search point x_0 was drawn randomly with $\|x_0\| = 1$ and the initial step-size $\sigma_0 = 0.5$. We let the algorithm run for 25 iterations, we set $r_{max} = 100$ and $\delta = 0.05$. There are eight feature functions in the evaluation function therefore the dimension of the search space is $n = 8$.

In order to reduce the learning time, we adopt the same scheme as in [2] and evaluate the offspring on harder games, in which "S" and "Z" appear four times more frequently than in the standard game. Furthermore, as in [2] the offspring are normalized at each iteration, and all play the same series of games. When racing is applied, the number of evaluation can go from 3 to r_{max}. When it is not applied it is r_{max}. The initial bounds $a = 150$ and $b = 2000$ were fixed experimentally. The games were simulated on the MDPTeris platform[1].

2.1 Results and discussions

The number of evaluations is reduced without a loss of performance. Fig. 1 on the right shows the fitness of the mean vector with and without racing. Hoeffding bounds perform the best reaching the performance for the smallest evaluation time, followed by Tetris bounds and Bernstein. Fig. 1 on the left shows the median number of evaluations out of r_{limit} used. With Tetris bounds this number of evaluation is reduced at the beginning of the learning process and increases towards the end reaching r_{limit}. For Hoeffding bounds the number of evaluations oscillates between r_{limit} and lower values. Finally, Bernstein bounds were not efficient compared to Hoeffding bounds, the number of evaluations is not reduced and at each race each individual is evaluated r_{limit} times. This is due to the fact that the standard deviation of the fitness is of the same order of its average, in other words $\hat{\sigma_{i,t}}$ is large compared to R in eq. 2. We noticed also that there is a large variability in the racing procedure. Fig. 2 shows the rate of evaluation out of r_{limit} of all 16 runs for Hoeffding and Tetris bounds. In order to assess the results of the racing procedure, we test the learned players on a standard game and compare the scores with previously published scores. We follow the same scheme as in [2] to construct the player. The score of a player is the average score on 100 games with a confidence bound of ±20% [6]. Table 1 present the scores of the players learned on CMA-ES with racing compared to the score reported by [2]. We notice that for both games instances the scores are statistically equivalent. Therefore we can conclude that the racing

[1]MDPTetris is available at http://mdptetris.gforge.inria.fr

Table 1: Comparison of the scores of the learned strategies on two game sizes

Strategy	10×16	10×20	Evaluations
No racing [2]	8.7×10^5	36.3×10^6	100%
Tetris bounds	8.1×10^5	31.5×10^6	67%
Hoeffding	7.8×10^5	33.2×10^6	27%

procedures do not affect the performance of the player. Furthermore, the same performance was obtained using, in the case of Hoeffding bounds two thirds less evaluations and in the case of Tetris bounds one third less evaluations.

3. CONCLUSIONS

In the present work, we have described the use of racing methods to reduce the evaluation time in learning artificial Tetris players. Using Hoeffding and Tetris bounds allowed to reduce the evaluation time, on the other hand Bernstein bounds were inefficient in all our problem instances due the properties of the Tetris fitness function. This also was the case when the confidence level was lowered. Testing on the standard game instances allowed to show that players designed using CMA-ES with racing have the same performance as the best existing players.

4. REFERENCES

[1] J.-Y. Audibert, R. Munos, and C. Szepesvári. Tuning bandit algorithms in stochastic environments. In M. H. et. al., editor, *Algorithmic Learning Theory*, volume 4754 of *LNCS*, pages 150–165. Springer, 2007.

[2] A. Boumaza. On the evolution of artificial tetris players. In *Proceeding of the IEEE Symposium on Computational Intelligence and Games 2009 (CIG'09)*, pages 387–393. IEEE, June 2009.

[3] C. P. Fahey. Tetris AI, Computer plays Tetris, 2003. On the web `http://colinfahey.com/tetris/tetris_en.html`.

[4] N. Hansen, S. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[5] V. Heidrich-Meisner and C. Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *Proc. of the 26th ICML*, pages 401–408, New York, 2009. ACM.

[6] C. Thiery and B. Scherrer. Improvements on Learning Tetris with Cross Entropy. *International Computer Games Association Journal*, 32, 2009.