Apply Ant Colony Optimization to Tetris

Xingguo Chen, Hao Wang, Weiwei Wang, Yinghuan Shi, and Yang Gao*
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
{chenxgspring, leafwanghao, elegate, oarsman23}@gmail.com,gaoy@nju.edu.cn

ABSTRACT

Tetris is a falling block game where the player's objective is to arrange a sequence of different shaped tetrominoes smoothly in order to survive. In the intelligence games, agent imitates the real player and chooses the best move based on a linear value function. In this paper, we apply Ant Colony Optimization (ACO) method to learn the weights of the function, trying to search an optimal weight-path in the weight graph. We use dynamic heuristic to prevent premature convergence to local optima. Our experimental result is better than most of traditional reinforcement learning methods.

Categories and Subject Descriptors: G.3 [PROBABILITY AND STATISTICS]: Markov processes

General Terms: Experimentation

Keywords: Ant Colony Optimization, Tetris, Weight Graph, Reinforcement Learning

1. INTRODUCTION

Tetris is a falling-blocks puzzle video game originally designed and programmed by Alexey Pajitnov in June 1985. It is an interesting sequential decision-making problem. There are some results in mathematics about tetris:

1. There is no way to win at tetris. It has been mathematically proven that it is possible to generate a sequence of tetrominoes that will guarantee the eventual termination of any game of Tetris played in a well of width 2(2n + 1), with n being any integer [3].
2. Ron Breukelaar proved that it is NP-complete to maximize the number of rows removed while playing the given piece sequence [4].
3. Ron Breukelaar also proved that given an initial gameboard and a sequence of p pieces, for any constant $\epsilon > 0$, it is NP-hard to approximate the maximum number of rows that can be removed without a loss within a factor of $p^{1-\epsilon}$ [4].

These properties make Tetris an appealing problem for testing machine learning algorithms and an interesting benchmark in RL-competition 2008 and 2009. In this paper, we apply ACO to learn the weights of the value function, corresponded to the policy of the intelligent Tetris agent.

*Corresponding Author

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

2. VALUE FUNCTION

In each state of the Tetris, agent rates all possible subsequent game boards by a state-value function and selects the best move. This state-value function is a linear combination of 16 feature functions as denoted $V_w(s) := \Sigma_{i=1}^{16} w_i \phi_i(s)$, where s is a Tetris state, $\phi_i(s)$ is the feature function of s, each of which is scaled to $[0, 1]$. We use Feature 1-12 from [2], and added Feature 13-16 as following:

13. Highest Hole: The height of the highest hole on the board.
14. Blocks Above Highest Hole: The number of blocks above the Highest Hole above.
15. Potential Lines: The number of lines which are above Highest Hole and occupied by more than 8 cells.
16. Smoothness: The sum of all absolute differences of adjacent column height, including difference of the first and last column.

Thus, our objective is to find a weight vector w as better as possible: $w^* = \arg\max_w E(w)$, where $E(w)$ is an evaluated function.

3. ANT COLONY OPTIMIZATION

3.1 Weight Graph

Ant colony optimization was firstly proposed by Marco Dorigo in 1992 to solve Travelling Salesman Problem (TSP) [6]. Following such an approach, we code the weight vector w into a 16-bit vector, thus in all $16 * 16 = 256$ bits. Every bit is joined by two vertices of the graph, and corresponded to two edges, each of which indicates '0' or '1'. The task of ant is to go from the first vertex to the last, deciding which edge to go through ('0' or '1'), resulting a path. Such a path is a solution to the task, corresponding to a weight vector in the weight vector space. And it's exactly a solution to Tetris.

3.2 Pheromone and Dynamic Heuristic

When all the ants finish their task, they release pheromone all the way. The quantity of pheromone is depended on the performance of the path (corresponding to a weight vector), which is defined as average removed lines (arl) of several games. We use $arlBest$ to record the most average removed lines up to current iteration. Then, let $pheromone \leftarrow$

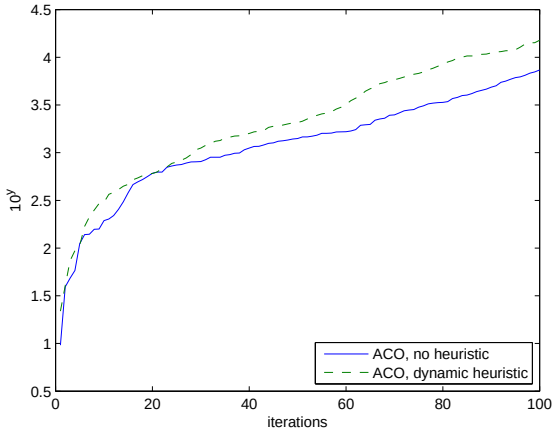

Figure 1: Average removed lines of 30 evaluation runs versus iteration number. Note the log scale.

Table 1: Performance of various algorithms.

	Method	Lines	Games
Non-RL	Hand-coded [7]	631.167	n. a.
	GA [2]	586,103	3000
RL	RRL-KBR [5]	≈50	120
	Policy iteration [1]	3,183	1500
	LSPI [10]	<3,000	≈17
	LP+Bootstrap [8]	4,274	n. a.
	Natural Policy gradient [9]	≈6,800	≈10000
	Noisy Cross Entropy + RL [11]	348,895	5000
	ACO + RL, no heuristic	7,368	4000
	ACO + RL, dynamic heuristic	17,586	4000

$pheromone + \frac{arl}{arlBest}$. When all ants finish releasing pheromone, the environment will execute evaporating pheromone. For each edge of the graph, let $pheromone \leftarrow pheromone * (1 - \rho)$, where ρ is evaporating rate.

Here, we try to utilize dynamic heuristic to prevent premature convergence t o local optima. We set heuristic as none at the beginning, and keep the record of the pheromone, which guide the ant to a best weight path up to current iteration. Then, when we finish updating the pheromone, we copy the recorded pheromone to heuristic, and update parameters (α and β).

3.3 Ant Action Selection

At each vertex of the graph, there are only two edges needed to choose, except the last vertex. Ants make the decision based on the pheromone and heuristic. At time t, at the vertex i, the probability of selecting edge j ($j \in \{0, 1\}$) $p_i^j(t)$ is $\frac{[pheromone_i j(t)]^\alpha \times [heuristic_i j(t)]^\beta}{\Sigma_{j=0}^{1}[pheromone_i j(t)]^\alpha \times [heuristic_i j(t)]^\beta}$.

4. EXPERIMENTAL RESULTS

We set the number of ants $n = 40$, the pheromone factor $\alpha = 1.5$. In our first experiment, we use no heuristic. Known from Figure 1, it converges to a premature optima and the performance settles at about 7,000 removed lines on average. In our second experiment, we use dynamic heuristic, and set $\beta = 0.5$, such heuristic information guides the ants to find better weight paths. With this setting, the performance (average removed lines) exceeds 17,000, and the best weight exceeds 23,700 [1](w=(-17636, -18574, -23919, 16797, 21146, -11704, -30208, -7444, 30106, -21136, -31399, -29620, -32550, -9138, 14900, -30487)).

5. FUTURE WORK

In Table 1, our experimental result is better than most of documented RL-tetris playing programs. However, compared with the methods of the Noisy Cross Entropy and Genetic Algorithms, there could be something to do to improve the performance, such as the design of the weight graph and utilization of function approximation.

The linear value function could not depict the real properties of Tetris, exponential value function is promising to get better performance. Another improvement could be done to the design of heuristics and the automatic settings of all parameters.

Acknowledgement: The work is supported by the Natural Science Foundation of China (No.60775046 and No.60721002) and National Grand Fundamental Research 973 Program of China (No. 2009CB320700).

6. REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena, 1996.

[2] N. Böhm, G. Kókai, and S. Mandl. An evolutionary approach to tetris. In *6th Metaheuristics International Conference*, pages CD–ROM, 2005.

[3] J. Brzustowski. Can you win at tetris? Master's thesis, University of British Columbia, 1992.

[4] E. Demaine, S. Hohenberger, and D. Liben-Nowell. Tetris is hard, even to approximate. In *Proceedings of the 9th International Computing and Combinatorics Conference (COCOON 2003)*, 2003.

[5] K. Driessens. *Relational Reinforcement Learning*. PhD thesis, Catholic University of Leuven, 2004.

[6] H.-B. DUAN. *Ant Colony Algorithms: Theory and Applications*. Science Press, 2005.

[7] C. P. Fahey. *Tetris AI*. http://www.colinfahey.com.

[8] V. F. Farias and B. van Roy. Tetris: A study of randomized constraint sampling. In *Probabilistic and Randomized Methods for Design Under Uncertainty*. Springer-Verlag UK, 2006.

[9] S. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, 2002.

[10] M. G. Lagoudakis, R. Parr, and M. L. Littman. Least-squares methods in reinforcement learning for control. In *In SETN '02: Proceedings of the Second Hellenic Conference on AI*, pages 249–260. Springer-Verlag, 2002.

[11] A. Szita, I. Lorincz. Learning tetris using the noisy cross-entropy method. *Neural Computation*, 18:2936–2941, 2006.

[1]The performance is based on the evaluation at the end of each iteration, but the best weight appears in computing $arlBest$