Tetris Artificial Intelligence

Wei-Tze, Tsai
NTUEE, Taiwan
Junior Student
b98901111@ntu.edu.tw

Chi-Hsien, Yen
NTUEE, Taiwan
Junior Student
b98901112@ntu.edu.tw

Wei-Chiu, Ma
NTUEE, Taiwan
Junior Student
b98901159@ntu.edu.tw

Tian-Li, Yu
NTUEE, Taiwan
Assistant Professor
tianliyu@cc.ee.ntu.edu.tw

***Abstract*—Recently, more and more researchers are interested in the Tetris video game. Many have studied about how many rows can be eliminated at most. In real world Tetris game, however, players will get extra bonus while eliminating rows consecutively (i.e., Combo) or using advanced skills (e.g., T-Spin). As a result, strategies without taking extra bonus into consideration may end up with lower points than those have already considered it. In this paper, we proposed an artificial intelligence applying human players strategies. Instead of using reinforcement learning, we use Breadth First Search (i.e., BFS) to find the best action in each Board state with features evaluation function. The experiment result shows the implemented strategies steadily outperform the original one.**

I. Introduction

Tetris is a video game which was originally invented by Russian programmer Alex Pajitonv at 1985. Then It has been popular since the day [1] [2].

The game area of Tetris, usually called board, is a matrix of cells with a height of 20 and width of 10. The blocks, usually called Tetrominos, fall from the top of the board to the bottom, each time a step. Once a block reaches the bottom of the board, it is placed. Traditionally, there are seven kinds of Tetrominos, which are shown below in Fig. 1. From left to right are: I, J, L, O, S, T and Z. In addition to moving the Tetrominos left and right, one can rotate them by 90 degrees, either clockwise or counterclockwise. When all cells in one row are occupied after a certain action, the row will be eliminated, and the player will get some points. For more detail, visit Faheys website [2].

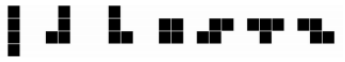

Fig. 1. From left to right : I, J, L, O, S, T, Z

As we have discussed above, Tetriss rules are simple. It needs, however, a lot of practice for us to gain high points. As for artificial intelligence, it may face the problem that there are too many board states (about 10^60 states [3]). It is not a wise decision to go through each of them. In order to reduce the number of states, we use features, which we will explain its detail in the following paper.

Generally, players know the current and the next Tetromino. In this paper, we assume that the artificial player can see at most three Tetrominos a time. It can then try all possible rotations and translations and decide where to place the Tetromino based on what it sees. One should remember that consecutive elimination is encouraged.

For the remaining paper, we will give a brief review on the existing approaches (section II). Define problem settings (section III). Analysize performance (section IV). Discuss the results (section IV).And finally, we conclude our discussion and present future ideas we intend to follow to improve our settings (section V).

II. Previous Review

Before reviewing what previous research have achieved, we first step back and define what an artificial intelligent agent is, then find out what it can do. In general, a Tetris agent can select, among all possible actions, a best way at any given state. The more rewards the agent receive, the better the action is. It is, however, sometimes difficult to give each action a value. As a consequence, we evaluate states instead.

Since almost all the best algorithms are using features and evaluation function, including those from Dellacherie, Fahey and Bohm, we cannot lag behind. We use evaluation function to give every state a numerical value, and the agent can use it to determine the best action which will lead to the state with highest score. For example, given a Tetromino, an agent can make it rotate and translate several times in order to simulate and evaluate all possible board states, after considering all possibilities, the agent chooses the board state whose value is the highest among the possible next states. The extension to our game setting(i.e., next three Tetrominos are known advanced) is also straightforward. The agent only needs to simulates two more level in depth before selecting an action for the current Tetromino.

Now, we have known how an action is chosen under certain principles. We turn our attention to how an evaluation function is defined. As we have mentioned above, Tetris game has almost 10^{60} different states. It is not very wise to use such a great amount of states as the nucleus of classification, instead, features may be much better. Through features, we can classify different states into same clusters, and significantly decrease the number of states. Some popular features, such as peak_height, landing_height, number_of_holes, row_transition, column_transition and eroded_piece_cells are widely used by several researchers. We, however, only take some features above into consideration (see table. 1), due to the reason that other features are not that important. The resulting features then combined with each other through a weighted parameter, and thus we get an evaluation function.

Let fi denotes a feature, fi is a function that maps a state

into a real value. The value of a state s is then defined as :

$$V(s) = \sum_{i=1}^{N} Wi * fi(s) \quad (1)$$

where N is the number of features and wi are the weights of each of them.

For many researchers, not only the weights but also the features are learned by reinforcement learning, but here, we use heuristic instead. Both of these two parameters are chosen by hand.

III. Problem Settings

Since Tetris game has been evolved for almost three decades, a variety of rules has been announced. In this paper, we use the game rules designed by Alexey Pajitnov, which is applied in Tetris Battle [4], a popular Facebook Application. The specific rules are following.

A. Matrix Setting

- The game area is a rectangular with 10 columns wide and 20 rows high.
- From beginning, every 7 pieces contains each type of Tetrimino with random order. That is, a given type of Tetrimino must arrive in next 7 pieces.
- Before dropping down a Tetrimino , player could ”hold” the current Tetrimino once into the Hold Queue, and take out the Tetrimino in the Hold Queue if exists.
- Player could see the Tetrimino type of the next 3 pieces.

B. SendLine System

- When clearing lines, one player sends ”Garbage lines” to force its opponent to the brink.
- If a player has nowhere to move its pieces, its opponent wins one ”K.O.”.
- Anyone who wins five ”K.O.” first will win.
- The number of ”Garbage lines”, i.e., Line-Sent, depends on the moves according to the Fig. 2.

IV. Analysis

Nowadays, the most common Artificial Intelligence (A.I.) for Tetris is simple and straightforward. We call this strategy ”Greedy A.I.”, which is named after its policy for eliminating rows. Greedy A.I. finds the possible states and eliminate lines once achievable, keeping the height of existing block as small as possible. In addition, if the heuristic function for reducing unnecessary states is well developed, Greedy A.I. could performs at a rapid speed. However, this strategy becomes inefficient under the game rules described in Section III. Since Greedy A.I. eliminate rows once reachable, the row elimination is generally single-line and discontinuous. Under the game rules, single-line row elimination is not rewarded and the discontinuous elimination ends up the Combo bonus, which both significantly reduce the Line Sent of Greedy A.I..

Because of the special bonus of Tetris Battle game rules, several strategies have been developed to make full use of the bonus under the game rules. In this paper, we focus on two strategies, Tetris A.I. and 2-Wide Combo. There are still more strategies, such as 3-Wide Combo, 4-Wide Combo or T-spin, which are out of the scope of this paper.

Move	Lines Sent (Lines per combo step)
Single	0
Double	1
Triple	2
Tetris	4
B2B Tetris	6
Perfect Clear	10
T-Spin	0
T-Spin Mini Single	1
T-Spin Single	2
T-Spin Double	4
T-Spin Triple	6
B2B T-Spin Mini Single	2
B2B T-Spin Single	3
B2B T-Spin Double	6
B2B T-Spin Triple	9
0 Combo	0 (0)
1 Combo	1 (1)
2 Combo	2 (1)
3 Combo	4 (2)
4 Combo	6 (2)
5 Combo	9 (3)
6 Combo	12 (3)
7 Combo	16 (4)
... Combo	... (4)

Fig. 2. Line-Sent table

A. Tetris A.I.

The strategy of Tetris A.I. maximize the use of Tetris Move and B2B Tetris Move, which are rewarded by 4 Line Sent and 6 Line Sent respectively. Tetris Move is the action that drop an I-Tetrimino into a gap and eliminate four rows at the same time. B2B (Back-to-back) Tetris Move is the Tetris Move performed after another Tetris Move and no other type of row elimination occurs between the two actions. That is, a series of Tetris Moves will be rewarded by 4 Line-Sent for the first one, and 6 Line-Sent for every other moves. The goal of the Tetris A.I. is therefore to make consequent Tetris Moves as many as possible.

To perform Tetris Move, one should stack on 9 columns and left the other one column blank for the I-Tetriminos. We let Tetris A.I. keeps stacking on the left 9 columns. The reason is that it is easier to stack on a wider board than on a narrower board in Tetris. Choosing a continuous 9 columns could simplify the problem. Besides, stacking on the right 9 columns is in the same situation and thus do not affect the results. The algorithm of stacking is explained later.

Tetris A.I. performs a Tetris Move once the precondition is satisfied, including the height of left 9 columns is larger than 4 and an I-Tetrimino is available. Consequently, Tetris A.I. will keep on stacking the left 9 columns and make a Tetris Move

features	*description*
coloumn_transition	*when the flow of the height changes, transition increases by one*
transition_height	*the total height difference between the neighbor columns*
peak_height	*the highest height of all columns*
$right - left$	*whether the left side is higher than the right side*

TABLE I
2-WIDE COMBO A.I. FEATURES

if possible.

The stacking algorithm of Tetris A.I. is Breadth-first Search (BFS). From the current state, it generates the possible next states, including those states after "hold". To exclude obvious adverse states, we remove the states that have "holes" in the stacking. A hole is a blank Mino underneath an occupied Mino, thus makes the existing blocks slower to eliminating. Tetris A.I. do not consider the states containing one or more holes while stacking. Tetris A.I. then expands the states to an adjustable depth, evaluates each final states, finds the optimal solution, and follows the corresponding actions.

Tetris A.I. evaluates states by several features and weights, which are listed in Table I.

B. Two Wide Combo A.I.

"Greedy A.I." always performs row elimination if possible. "Tetris A.I." performs row elimination whenever the current Tetromino is "I" and the height of block is larger than four. Compared to these both, "Two Wide Combo A.I." is much more complicated.

Roughly speaking, "Two Wide Combo A.I." can be separated into two states, one representing "Stacking" and the other representing "Combo". During the Stacking state, the agent uses BFS to find the best way to stack the Tetrominos on the left eight columns. While in the Combo state, the agent tries its best to consecutively eliminate rows, in order to get the Combo bonus. The details of these two states and when "Two Wide Combo A.I." transit from one to the other is shown as following.

Unlike "Tetris A.I." and "Greedy A.I.", which eliminate rows easily, "Two Wide Combo A.I." has much more to concern. "Two Wide Combo A.I." cannot directly use "holes" to determine whether such action is feasible or not, rather, it has to trace deep enough to see the last board configuration to make decisions, due to the reason that "holes" may not be "holes" after taking some specific actions. Therefore, "Two Wide Combo A.I." doesnt care about whether there is a hole at the right two columns or not, the only thing it cares is the shape and if it is good to continue Combo.

V. EXPERIMENTS

In order to compare these three kind of A.I., we conducted experiments to test how fast A.I. can play. The experiment setting is the same as our game setting(i.e., next three Tetrominos are known advanced), and the tested A.I. play with a time limit of 30 seconds for 10 times. The result of Line-Cleared and Line-Sent is showed below:

Fig. 3. Two Wide Combo

	Greedy A.I.		Tetris A.I.		Two Wide Combo A.I.	
	Line-Cleare	Line-Sent	Line-Cleare	Line-Sent	Line-Cleare	Line-Sent
	61	69	64	94	60	102
	62	76	64	94	56	95
	63	71	62	88	57	94
	66	81	64	94	58	88
	62	73	64	94	54	92
	64	79	66	94	56	92
	63	74	59	82	62	104
	62	77	63	88	60	96
	64	80	65	94	57	98
	62	69	64	94	57	94
Average	62.9	74.9	63.5	91.6	57.7	95.5
Stdev	1.449138	4.408325	1.900292	4.195235	2.359378	4.790036
C/P value	1.190779		1.44252		1.655113	

Fig. 4. Experiment Results

- Greedy A.I. and Tetris A.I. have roughly the same number of Line-Cleared, while Two Wide Combo A.I. performed less Line-Cleared than the above two A.I.. This result indicates that Greedy A.I. and Tetris A.I. holds the same time complexity, and Two Wide Combo A.I. holds higher time complexity.
- Tetris A.I. dominates Greedy A.I. in the number of Line-Sent(i.e., Tetris A.I. has higher C/P value). This make sense because Greedy A.I. performs row elimination whenever possible, therefore it sends zero line when only one single line is cleared.
- When we takes a clear look at the performance of Tetris A.I. and Two Wide Combo A.I., we can observe that Two Wide Combo A.I. sent more lines than Tetris A.I. even though it has less number of Line-Cleared. Moreover, Two Wide Combo A.I. has the highest C/P value among these three A.I.s. This result is prominent, since it infers that Two Wide Combo is more efficient than B2B Tetris Move, let alone Combos with higher wide.

VI. CONCLUSION

In this paper, We have implemented "Tetris A.I." and "2-Wide Combo A.I.". Experiment results show that the two strategies both outperform the original "Greedy A.I.". in all aspects. Furthermore, according to the c/p values comparison,

"2-Wide Combo A.I." is more efficient than "Tetris A.I."; that is, have more Line-Sent at the same row elimination situation. Therefore, "2-Wide Combo A.I." may be preferred among all existing A.I. strategies.

VII. Future work

A. Implement more strategies

We already have three types of A.I., namely, Greedy A.I., Tetris A.I., and Two Wide Combo A.I.. Next step we are going to implement more types of A.I., like Three Wide Combo A.I. and Four Wide Combo A.I.. Though it seems easy to extend Two Wide to Three Wide or more, it is not the case since we have to handle many problems such as rules for combo, environment settings for spin, and back off policies.

B. Fusion of strategies

With enough strategies in hand, we can construct a powerful A.I. which has the ability to choose different strategies for different conditions. First we can employ reflective agent to handle this, then we turned up the agent, and we can use a learning agent to find the appropriate timing to change strategies.

VIII. Reference

References

[1] Niko B hm, Gabriella K kai and Stefan Mandl, An evolutionary approach to Tetris, 6th Metaheuristics International Conference, 2005

[2] Colin Fahey, Tetris website, http://www.colinfahey.com/tetris/tetris en.html, 2011

[3] Amine Boumaza, On the evolution of articial Tetris players, Computational Intelligence and Games 2009, 2009

[4] http://www.tetrisfriends.com/help/tips_Battle.php

[5] Xu Pu, Feng Jiali, A Tetris AI with an Adventurous Strategy Based on Attribute Theory Method, International Conference on Computer Science and Network Technology, 2011