CS3243 Introduction to Artificial Intelligence
Learn to play Tetris with Big Data

National Univeristy of Singapore
Wang Zhe, Wei Yihao, Zhang Kai, Zhang Wei
A0111785E,A0105593E, A0119378U, A0101349N

Introduction

The goal of this project is to develop a competent utility-based Tetris agent to play the Tetris game. The agent is approaching Tetris as an example of exploratory problem, occuring on a 20 rows by 10 columns board. A piece is randomly chosen from the set of seven shapes - O, I, T, L, J, S and Z.The performance measure given as the number of rows removed. Weight matrix is concurrently generated via modified Genetic Algorithm(GA) and the agent is tasked with deciding the optimum move for each piece given.

Methodology

Problem Formalisation

The environment under consideration is partially observable, single agent, stochastic, episodic, static, discrete and known. We propose the definition of 'state' as a vector of features that describe the board, stacking and landing piece. The action employed by the agent is a instance of the legal moves of a given piece, composed of the rotation and alignment. Given as such, the step cost is negligible. Instead of employing a goal-test function, the agent will terminate in compliance with Tetris rule of topping out.
The board position and shape of current falling object is represented by i and respectively. Prior to starting the game, the agent heuristically generates the weight factor, w, in order to construct the utility function. The agent then proceeds to evaluate the board, stacked pieces and falling piece to generate the state f, a vector of six features.
Hence the agents utility function is defined as:

$$\mu(i, y) \;= argmaxV(\overline{f}\,) = \overline{w} \cdot \overline{f} \qquad (1)$$

The Tetris agent is an utility-based agent, which is making decision based on the structured representation of the state, which will be explained in our list of features.

Heuristics

List of features:

- Landing height: (maximum height of the last piece + height of the column) /2

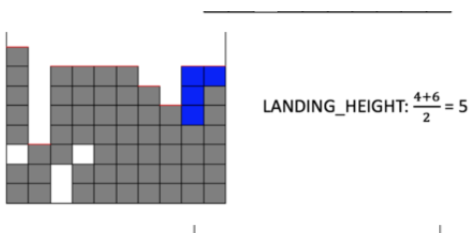


Figure 1: Landing Height

- Rows eliminated: number of row cleared by a move
- Row Transitions: when an empty cell is adjacent to a filled cell on the same row

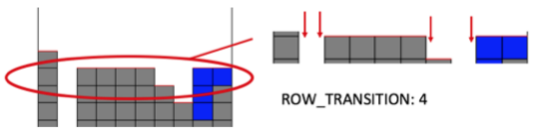


Figure 2: Row Transition

- Column Transitions: when an empty cell is adjacent to a filled cell on the same column

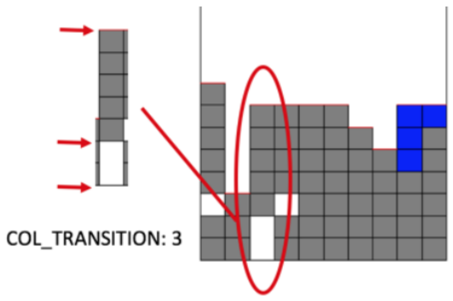


Figure 3: Column Transition

- Number of Holes: an empty cell with at least one filled cell above it

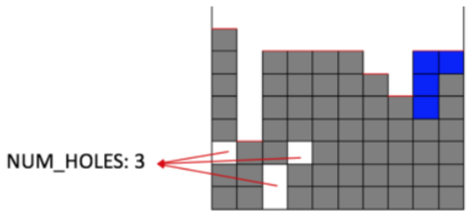


Figure 4: Number of Holes

- Well sums:an empty cells with their left cells and right cells filled

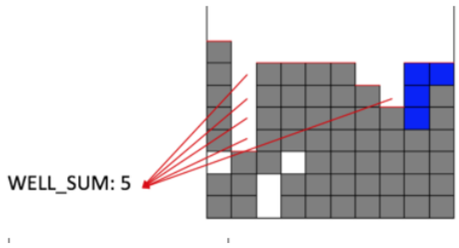


Figure 5: Well Sums

Program Specification

Class: Genetic Algorithm, GA

In order search for a weight that will allow the agent to pick the best move in playing Tetris, we adopt the genetic algorithm which is inspired by natural evolution.

If no weight matrix has been provided, which is the default state, GA is tasked as the entry function.The optimum weight matrix is generated and saved to specified file path, waiting to be called by AIPlayer.

- *Initialisation*: We choose to have a population of 1000, each of which is randomized from the range of -10 to 10 at each weight value(forming a set of 6), in the hope that it is able to cover the search space and converge to global maximum. As we have noted that the variance of scores from each round of Tetris is quite large, each set of weights are tested by playing 10 rounds of game and being assigned the average score.
- *Selection and Inheritance*: In the selection process, we sort our population according to the scores obtained and adopt a selection rate of 0.05 to preserve the best gene. The selected population will be treated as parents to produce the next generation.
- *Crossover and Mutation*: Crossover is a process of reselection and combination of parents gene in the next generation. To fully mimic natural evolution, we also set a mutation rate as 0.1 to maintain genetic diversity. Should mutation does occur, we are mutating only 1 feature weight.

The special features of GA:

1. When searching in the weight solution space, we fix all our weighs to positive or negative. For example, we set score weight to be positive in a sense that we reward it if it increases. As a consequence, the solution space is reduced to (2^6 =)64 times smaller, which definitely leads to a higher convergence rate.
2. By assuming that higher scores implies better gene, those parents with higher scores will be chosen on higher probabilities. The selection method is formuled as follows:
$$P(chosen) = \frac{score_i}{\sum_{k=1}^{n} score_i}$$
where n is the number of parents, $score_i$ is the score of parent i.

Class: AIPlayer

The class AIPlayer dictates the agent's behaviour through the life time of the play round. It assumes that all pieces are generated randomly.

- Simulator: The simulator would simulate all the possibility of next move given current state.
- Analyzer: The analyzer would retrieve the values of all the features from the simulated state and pass it to the AIPlayer.

AIPlayer would compute the heuristic function by assigning weight values to the collected feature vector. Afterward, AIPlayer would proceed to pick move with the best score as its next move.

The special features of AIPlayer:

1. Predictor: The predictor is tasked with frequency predication. It would calculate frequency distribution of all the generated pieces number given the record of generated pieces. It would modify its estimation of the probability of random piece generation.
2. Markov Chain: It is a stochastic model describing a sequence of possible events- in which the probability of each event depends solely on the state attained in the previous event. It specifies the probability of transitioning from one state to another state. It would pass the probability of the transition between previous two generated pieces and each of following pieces to the AIPlayer.

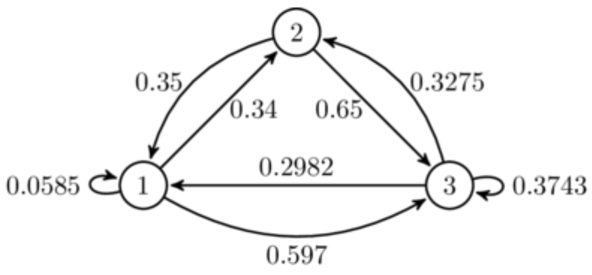
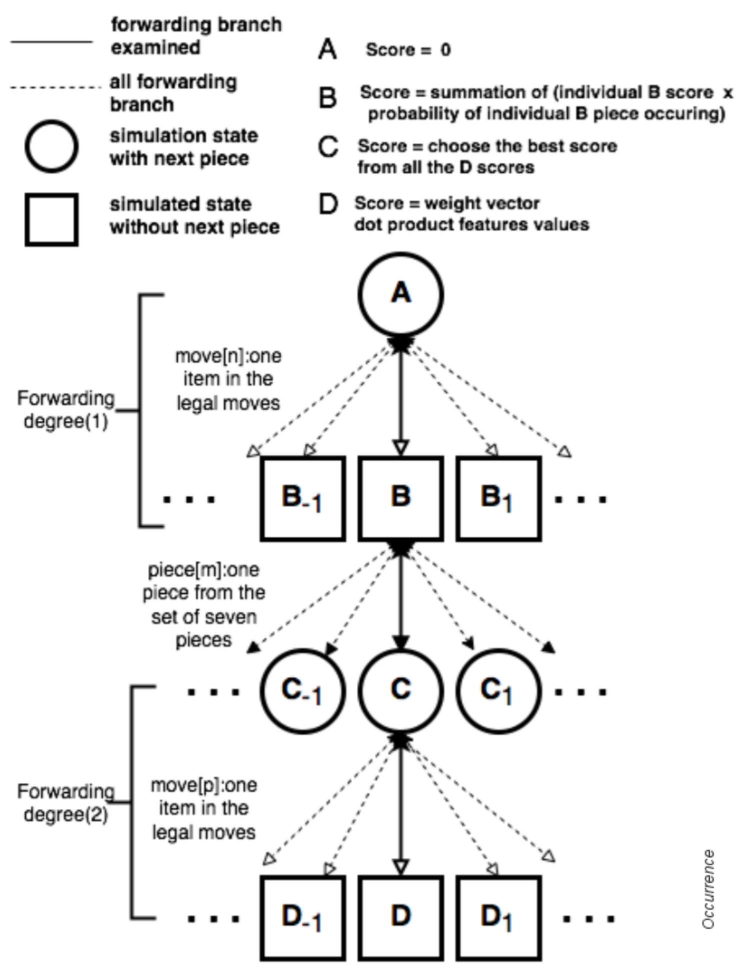


Figure 6: Markov Chain

3. Forwarding:



forwarding branch examined
all forwarding branch
simulation state with next piece
simulated state without next piece
A Score = 0
B Score = summation of (individual B score x probability of individual B piece occuring)
C Score = choose the best score from all the D scores
D Score = weight vector dot product features values
A
Forwarding degree(1)
move[n]:one item in the legal moves
B-1
B
B1
piece[m]:one piece from the set of seven pieces
C-1
C
C1
Forwarding degree(2)
move[p]:one item in the legal moves
D-1
D
D1

Figure 7: Tree model of the Forwarding process

The forwarding employed in AIPlayer involves two stages of the forwarding process. Firstly, the current state (A) generates one simulation state(B) of the current move employing one move from the list of legal moves generated. This is the first level of forwarding.
The simulated state then proceed to generate a second-generation simulation state (C). The state C, then proceed to generate the last simulation state D via employing one of the list of legal moves available at C.This is the second stage of forwarding. The first forward process is repeated for all moves from the legal list at A. The generation of second-generation simulation state is repeated for all 7 possible pieces, and the second forwarding is repeated for all moves in the list of legal moves at C.
The score at D is generated via the utility function, specified weight vector dot product the state at D. The score at C is evaluated via selecting the best (in our case, the max) from the scores of all the simulation states at D level. The score at B is obtained from the summation of the multiple of each of score at B level with the probability of the piece moved occurred. At state A, the agent decides the best move from the array of scores at B level.

Optimisation: Speed-up

Parallel Computing

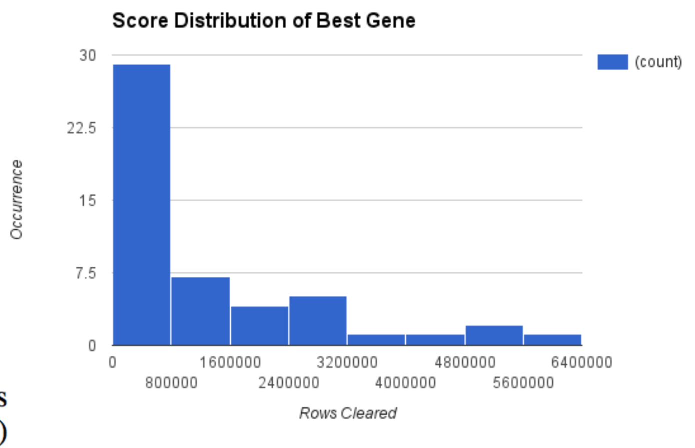
The GA utilises multithreading to run multiple individual class simultaneously to play lots of games at the same time. It will fasten the generation of prime genes, which are used to crossover and mutate for finding the best weight assignment of the features.

Enhanced Difficulty

- Reduce the Board Size: The board size can be set to smaller, which will increase difficulty of the game, and therefore shorten the playing time.
- Modify shape generation probability function: By increasing the probability of Z and S shape, it will directly increase the difficulty of the game, and also will shorten the game time. This will also speed up the running time of the algorithm.

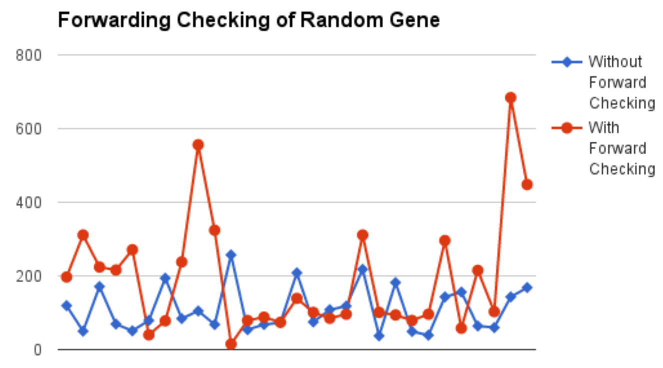
Results

Data



Statistical Analysis of Best Gene Score

Average	1263750.84
Standard Deviation	1427352.20
Lowest	1015
Highest	5706962



Discussion

In 20 generations, the best population converges to our final weight result as shown in above table. As you can see, the frequency of the score in the range (0, 800000) is quite high. However, we still have an average of 1263750.84. This is due to a large standard deviation of the scores range from one thousand to 5 millions. To further confirm our result, we calculated the confidence level within 50 rounds of games.

We chose p = 95% for our confidence intervals and we have

$$|u - \widetilde{u}| \leq \frac{K\widetilde{u}}{\sqrt{N}}$$

Where u is the average score of N games and u' is the expected score. Since we choose 95% confidence interval, we have k=2. As a result of 50 simulated games, the expected score is within

$$(1263750.84 \pm 28.3\%)$$

We would like to substantiate that the enhanced difficulty part of the optimisation section is proposed as a potential speed-up solution. However, we have not included it in our code due to the following reasons:
In accordance with the enhanced features mentioned above, we would like to elaborate that reducing the board size not only directly contradicts the problem definition, but is also proven ineffective in practice(despite its attractiveness in theory). In actuality, this is not implemented.

Furthermore, the second part of the enhanced feature, regarding modified probability of the shapes generated, it is not also not implemented for the question has specified that the assumption of uniform probability of each shape being generated.