James Kasakyan

CSC 44800 Artificial Intelligence

Stephen Lucci

**Project 2 Report: Genetic Algorithm for Sprouts**

Table 1- Player 1 Minimax vs Player 2 Random

|  |  |
| --- | --- |
| Player 1 (minimax) | Player 2 (random play) |
| Total wins after 100 games | |
| 75 | 25 |

Table 2- Player 1 Minimax vs Player 2 GA (5 generations)

|  |  |
| --- | --- |
| Player 1 (minimax) | Player 2 (genetic algorithm) |
| Total wins after 50 games, number of generations = 5 | |
| 3 | 47 |

Table 3- Player 1 Minimax vs Player 2 GA (10 generations)

|  |  |
| --- | --- |
| Player 1 (minimax) | Player 2 (genetic algorithm) |
| Total wins after 50 games, number of generations = 10 | |
| 4 | 46 |

Table 4- Player 1 Minimax vs Player 2 GA (50 generations)

|  |  |
| --- | --- |
| Player 1 (minimax) | Player 2 (genetic algorithm) |
| Total wins after 50 games, number of generations = 50 | |
| 0 | 50 |

**String Encoding:**

Regardless of whether the game finishes in 3n-1 moves (n = 2, 3n-1 = 5) or 2n moves (n = 2, 2n = 4), the total number of game decisions for player 2 is 2. At most, a game state has 4 valid plays. N unique states can be represent by Log2(N) bits; N = 4, so each play needs to be encoded with Log2(4) = 2 bits. Thus a complete solution requires a string of length: 2 (bits to encode one move) \* 2 (number of game decisions) = 4.

**Encoding:**

00 = node.adjacencyArray[0]

01 = node.adjacencyArray[1]

10 = node.adjacencyArray[2]

11 = node.adjacencyArray[3]

Since the number of decisions possible at a certain game state for player 2 can range from 2-4, provisions must be made to prevent the production of invalid solutions that would result in attempting to make an invalid play. For example, a random initial solution might be of the form 1111. However, on the second turn of player 2 the maximum number of decisions to choose from is three, and a solution string of 1111 corresponds to choosing the fourth possible option twice. To reconcile this and similar conditions from occurring, whenever a string corresponds to an invalid move, the next highest possible choice will be made. Thus a solution string of 1111 will be treated like a string of 1110 if there are 3 possible decisions from the second game state, 1101 if there are 2 possible decisions, and 1100 if there is only one possible decision from the second game state.

Strategy:

As the second player in a two node game of sprouts, a winning strategy is available that guarantees victory. It involves, whenever possible, restricting the number of playable nodes for the opponent. Usually this entails connecting two vertices of degree two (thus making them both degree 3 and "dead") such that the new vertex placed on this new edge is unable to connect to any remaining edge of degree 2 (usually because it inside a "loop" whose edges act as a sort of barricade). In the game tree used in this program, the third choice at any game state (if no third choice exists, the next highest choice) corresponds to a move employing this strategy. This strategy guarantees a win against an opponent.

Fitness Function f(*s*)

Since sprouts is a game of perfect information, the fitness function is relatively straightforward. A candidate solution string *s* simulates a match against a minimax opponent. If the solution results in a win for the player, its fitness is increased by 1. However since the minimax opponent does not always make the same play (if all choices are equally bad, he will make a random choice), a win for the player does not guarantee that this candidate string *s* corresponds to a string representing the strategy above, which is guaranteed to win regardless of the decisions of the opponent. Thus, to reduce the possibility of a random chance solution happening to be chosen as the solution string, the fitness function simulates 25 matches, adding a fitness of 1 for each victory. Probabilistically, when a candidate solution string has a fitness of 25, it corresponds to this winning strategy, and further generation can be stopped.