Evolving a Hex-Playing Agent

Michael McCarver and Rob LeGrand, PhD

Intro with related work:

Hex is an adversarial board game in which there is always exactly one winner. The game was invented in 1942 by Piet Hein, and independently reinvented in 1948 by John Nash. The board is in the shape of an nxn parallelogram and is made up of nxn hexagon shaped tiles. Each player has a store of tiles, colored differently than their opponents'. Players alternate placing tiles on the hexagons of the board, capturing that hexagon. The goal of the game is for a player to connect opposing sides of the board. The sides of the board each player aims to connect is determined before the game starts. See Figure 1 below for an example of a completed game. John Nash offered an existence proof in 1949 of a first-player advantage, but no general winning strategy exists for the game (Gardner).

(figure of hex board)

Research into Hex is often concerned with solving the game. Here, solving generally means finding perfect moves that give the player a guaranteed win. Levels of 'solved' vary from knowing only the best first move (weakly solved), to knowing the best move at all points in the game (strongly solved). As of 2011, humans have only been able to solve some center moves weakly for boards sized 8x8 and 9x9 (Beyond Humans pg 2-3). In 2014, Pawlewicz and Hayward developed a Scalable Parallel DFPN Search and used it to solve all previously intractable 9x9 openings and one 10x10 opening. The hardest 9x9 opening took 111 days to solve (Scalable Parallel DFPN Search pg 1).

Other avenues of research have been explored in the context of board game play. Dr. David Fogel trained a neural network with an evolutionary algorithm to teach it to play checkers. The neural network was accompanied with a minimax search of variable depth, where the depth was determined by the amount of time available to make a move. The final agent was able to perform better than 99% of players on a checkers-playing forum (Blondie 24). Young, Vasan, and Hayward used Deep Q-Learning to train a CNN (convolutional neural network) to play Hex on a 13x13 board. They began their experiment with supervised training over a database of generated games to accelerate the speed with which the CNN learned early and mid-game strategy. Then, they used self-play to update their heuristic values with a Deep Q-learning algorithm. They did not utilize minimax during gameplay, and they were able to win 20.4% of games as first player and 2.1% of games as second player against a version of the ICGA Olympiad Hex champion (neurohex).

Experimental Setup:

Our research is inspired by the work of Dr. David Fogel's Blondie 24, and similarly, we set out to evolve the weights of an ANN through genetic algorithm. We create a population of 100 Hex players, arranged on a torus so that players at the top and bottom of the map are neighbors, as well as players on the left and right. Each position that a player occupies is a hexagon, so each player is neighbored by six other players. One iteration of evolution consists of 600 matches, so that each player plays a match as first player and a match as second player against their neighbors. Our algorithm keeps track of the number of games won by each player; we use this statistic as a fitness function during the breeding phase. In breeding, we move through each weight of each player and determine whether to keep the original weight or to replace it, and if we replace it, we use a weighted probability, determined by our fitness function, to choose between our neighbors and our currently looked-at player. After a weight is chosen, we shake the weight by applying a normal distribution with a mean set as the original weight. Finally, with some probability, we randomly swap weights within the ANN.

One key difference between our research and Fogel's research is that we do not use a minimax search to guide our evolutions. When a minimax search is used, the player is aware of future board evaluations following a line of hypothetical moves, and aware of winning moves in future states of game play. Minimax allows the player to evaluate the consequences of their moves, both from their perspective and their opponent’s perspective. Also, once a winning path of moves is discovered, the player will stop using its heuristic function and rely entirely on the minimax search to determine its next moves. By eliminating the minimax search, we hope to test the bounds of what our ANN is capable of learning concerning the strength of a board positions at any point of gameplay. The only mechanics of gameplay that our players are aware of is the validity of potential moves. A sufficiently complicated ANN is capable of learning move validity also, but that is outside the scope of our current research.

We created a set of ‘vanilla’ players so that we could focus on small variable changes and the effects of those changes on the experiment. Our vanilla players are two-layered ANN with five nodes leading into a final output node. The board state is read into a one-dimensional vector, where every index refers to a hex tile on the board. Tiles owned by our player are given a value of one, tiles owned by opponents have a value of negative one, and unoccupied tiles have a value of zero. The players are initialized with random values determined by a normal distribution with standard deviation of one and a mean of zero. During the breeding phase, the probability that players keep their original weight, what we call inertia, is set to zero. The fitness function is a simple comparison of the number of games won by each player, which means that players with zero wins have a zero probability of passing on their weights. The normal distribution used for shaking each weight has a standard deviation of one. After the breeding phase of evolution, they will always swap weights within their ANN exactly once.

Experimental Results:

The variables we changed during experimentation were: the size and shape of the ANN, the probability/number of swaps per player, the level of inertia (probability of keeping a player’s original weights), and the function used to apply number of wins as a fitness function. To compare the experiments with each other, we took populations of different strategies generated after 10,000 iterations and played all players of each population against each other. We generated eight experiments with the vanilla strategy to ensure a diverse baseline, and we generated two experiments of variable strategies (time constraints were a factor). Of our eight vanilla experiments, the seventh experiment contained the strongest players, so we focused on this population for comparisons.

One variable we tested was the probability of swapping weights during the breeding phase. Instead of always swapping once, we applied a negative binomial distribution to determine whether to swap. We chose two probabilities for experimentation: 50% chance of failure and 25% chance of failure. Chance of failure represents the odds of performing another swap. So, with a 50% chance of failure, there is a 50% chance that no swaps will occur and a 50% chance that at least one swap will occur. After each swap, the probability is re-evaluated to decide whether to swap again. With the 50% chance of failure, both of our experiments performed worse than the vanilla players, although by a small margin. One experiment produced more winning players than vanilla, and one experiment produced less. Both experiments produced more losing players than vanilla. Producing losing players isn’t necessarily undesirable since we are concerned only with the wining players an experiment produces, so these players may have skewed the statistics. The experiments with a 25% chance of failure performed better than vanilla, although again by a small margin. This experiment consistently produced better winning players, however, and produced less losing players.

We also experimented with a three-layer ANN. One experiment performed better than vanilla, and one experiment performed worse. Both experiments produced many winning players, and even the losing population produced fewer losing players than vanilla.

<Better results forthcoming>

Conclusions:

Based on the results of our experiments, a few strategies stand out as promising. Always swapping weights exactly once during the breeding phase seems to inject too much chaos into our evolutions. The relative success of 50% failure and 25% failure negative binomial distributions suggest that swapping less frequently allows populations to conserve momentum and to better find stable configurations during evolutions. The three-layer ANN didn’t perform noticeably better than the vanilla experiments; however, this isn’t indicative of a poor strategy. It is possible, and probable, that a more complicated network topology requires more iterations to successfully learn playing strategy. None of our strategies produced players that would beat human players familiar with the game Hex. This is unsurprising, as the state-space for Hex is vast. ANN are capable of learning any function, even Hex board states, but it is likely that more complicated networks are required, as well as longer experiment iterations, before competitive players are produced.

Future Work:

There are many variations to our experiment that would yield interesting research. Growing an ANN without any prior knowledge to game mechanics, including move validity, captures the spirit of our research as well as that of Dr. David Fogel’s. Other possibilities include: changing the shape of the population space (100 players arranged on a torus with hexagonal neighborhoods); re-arranging the positions of players based on their performance during the previous iterations; copying weights instead of swapping weights during the breeding phase; changing the method of reading in board states (perhaps two different vectors for opponent-owned tiles and player-owned tiles, or considering whether a tile is connected to a side); the standard deviation by which to shake genes during breeding; adding more information to be considered into the fitness function (possibly consider number of moves played to incentivize players that win quickly); different probabilities used during the swap function; and devising a hardness scale (similar to the mohs hardness scale) with which to accurately compare experiments to each other.

Citations

|  |
| --- |
|  |
|  |

Young, Kenny, Gautham Vasan, and Ryan Hayward. "NeuroHex: A Deep Q-learning Hex Agent." Computer Games. Springer, Cham, 2016. 3-18.

Pawlewicz, Jakub, and Ryan B. Hayward. "Scalable parallel DFPN search." International Conference on Computers and Games. Springer, Cham, 2013.

Fogel, David B. Blondie24: Playing at the Edge of AI. Morgan Kaufmann, 2001.

Arneson, Broderick, Ryan B. Hayward, and Philip Henderson. "Solving Hex: Beyond Humans." Computers and Games 6515 (2010): 1-10.

Gardner, Martin. "The game of Hex." Hexaflexagons and Other Mathematical Diversions: The First Scientific American Book of Puzzles and Games (1959): 73-83.

Questions/Notes for Dr. LeGrand

* I haven’t mentioned our original experiments, which include the no-swap agents. I thought explaining them might complicate the paper. What do you think?
* Which citation format should I use. The citations here are MLA, but I haven’t formatted the inline citations yet.
* The experimental results aren’t really ready yet. I haven’t generated a set of experiments with the specs we decided on for vanilla players, and I haven’t written code yet for players to play against random agents. The experimental results I talk about refer to the old vanilla players, so that section will probably change considerably. Still, I’d like feedback on my approach for writing that section.