Evolving Game-Playing Agents Through Coevolution

Lucas Ellgren
Advisor: Nic McPhee

Division of Science and Mathematics
University of Minnesota, Morris

April 28, 2012

Motivation

- Push the boundaries of artificial intelligence
- Coevolution is a relatively unexplored way to create intelligent agents
- Can be adapted to other kinds of problems
- Scientific curiosity

Outline

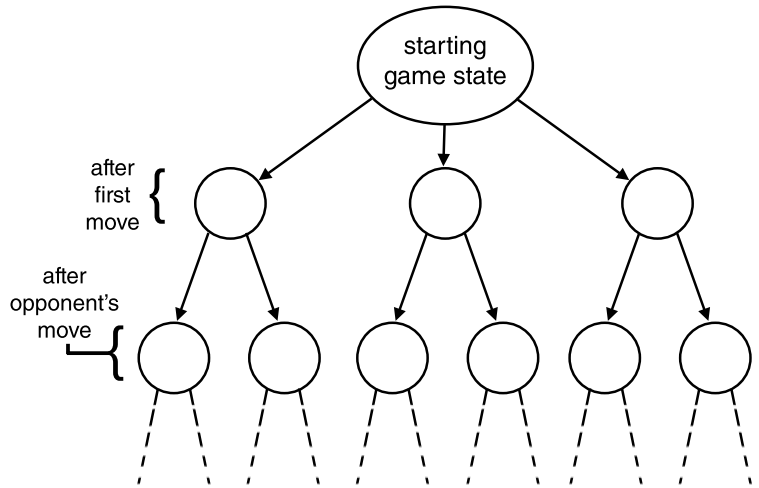
Traditional Game-Playing Agents

- Playing games used as a benchmark in artificial intelligence
- Early work - solving tic-tac-toe and checkers
- IBM's Deep Blue - beats world champion chess player [1]
- Current work - Othello, Go, many others
- Two kinds of agents:
  - Search-based agents
  - Artificial neural networks

Image by James the photographer
http://bit.ly/I0L6PM

Search-Based Agents

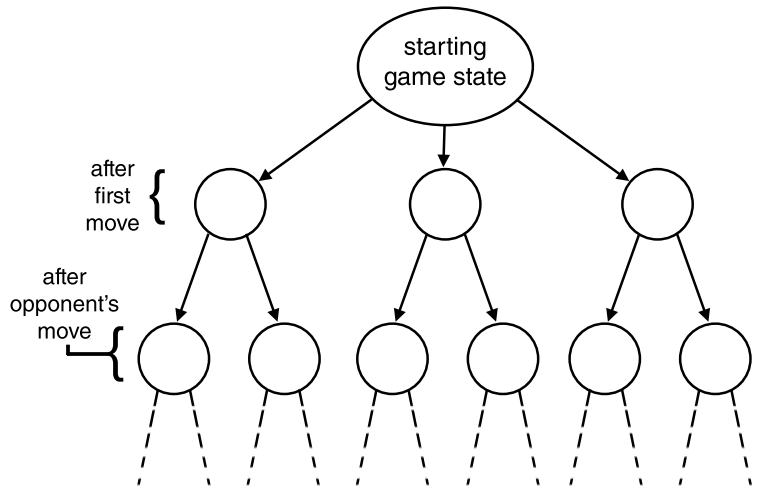
- *Game-tree:* tree structure representing all possible moves
- Search algorithms look for win/loss states in tree
- Choose the move with the highest chance of success
- Depth-first search and breadth-first search used for simple games
- Depth-first iterative deepening is a more recent example



An example of a game-tree for a two-player game

Depth-first Iterative Deepening

- Does depth-first search iteratively into tree [5]
- Continues until reaching estimated goal depth
- Always starts from root node
- Searches lots of nodes, but uses less memory
- Successfully used to solve checkers [2]



An example of a game-tree for a two-player game

Artificial Neural Networks

- Based on the way real-life neurons work
- Composed of many connected neurons
- As information flows through network it gets processed
  - Input neurons receive data, pass it along to connected neurons
  - *Weight values* are applied when data is passed
  - Hidden neurons process data further
  - Final data collected in output neurons
  - Final results are outputted
- Computation can be changed by modifying weight values

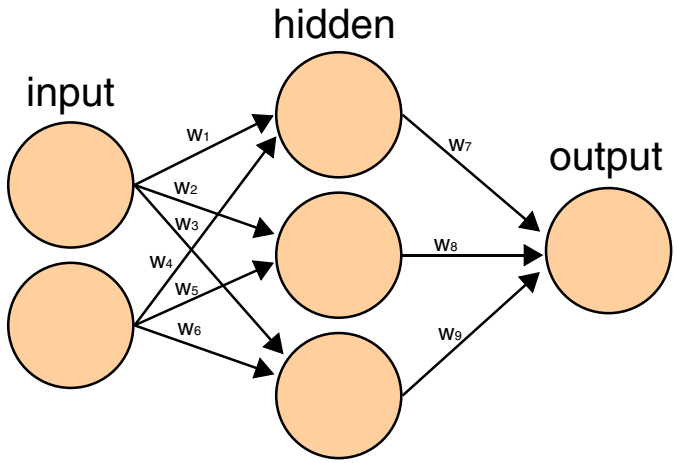
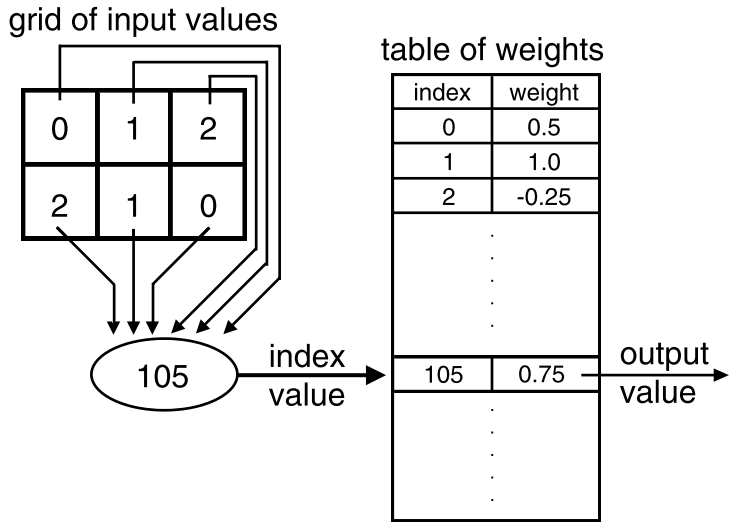


Image by Cburnett
http://bit.ly/IbWk8y

N-tuple Neural Networks

- Originally designed for optical character recognition
- Each n-tuple neuron:
  - Takes input from a grid of values
  - Contains a *table of weights*
  - Uses input to compute index value into table of weights
  - Item returned through table is used as output
- Output of all neurons in network are used to compute final result



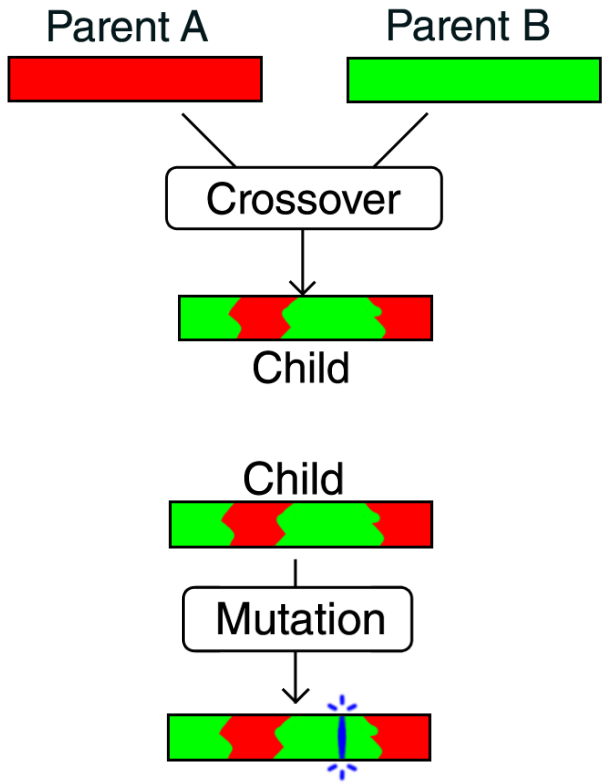
How a 6-tuple neuron processes data from a grid.

Evolutionary Algorithms (EAs)

- Use basic principles of biological evolution to *evolve* solutions to problems
- Represent potential solutions to a problem as *individuals*
- Individuals are given a *fitness* value depending on how well they solve the given problem
- A higher fitness value means a better chance to reproduce and pass on solution
- Good solutions survive while bad solutions do not

Evolutionary Processes

- EAs create new individuals through:
  1. *Selection* - two parents chosen based on fitness values
  2. *Crossover* - components from both parents are combined into a new individual
  3. *Mutation* - a tiny part of the individual is randomly changed
- Crossover and mutation allow new solutions to be found



The process of crossover and mutation illustrated

Why Use Coevolution?

- Traditional EAs only work with problems that can be evaluated *objectively*
- They are not as effective for *subjectively* evaluated problems
- Most game-playing agents must be evaluated subjectively
- *Coevolutionary algorithms* (CEAs) - EAs that use coevolution
- Fitness is evaluated through interactions between individuals
- CEAs have been used successfully to evolve game-playing agents [3]

Coevolutionary Algorithms

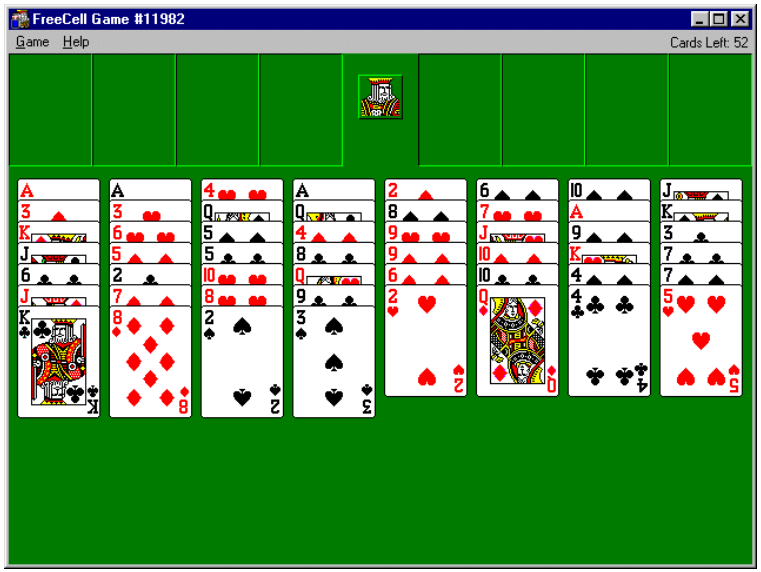
- Reproduction of individuals works the same
- *Single population* - interaction occurs within one big population
- *Multiple population* - individuals only interact with those from another population
- *All vs. Best Evaluation* - individuals are evaluated against best from previous generation
- *Tournament Evaluation* - individuals evaluated through tournament brackets

Solving FreeCell

- Study by Achiya Elyasaf and collegues to solve FreeCell [4]
- *GA-FreeCell* - coevolutionary algorithm used to evolve agents that play FreeCell
- Expands on past work done with search-based algorithms

The Game of FreeCell

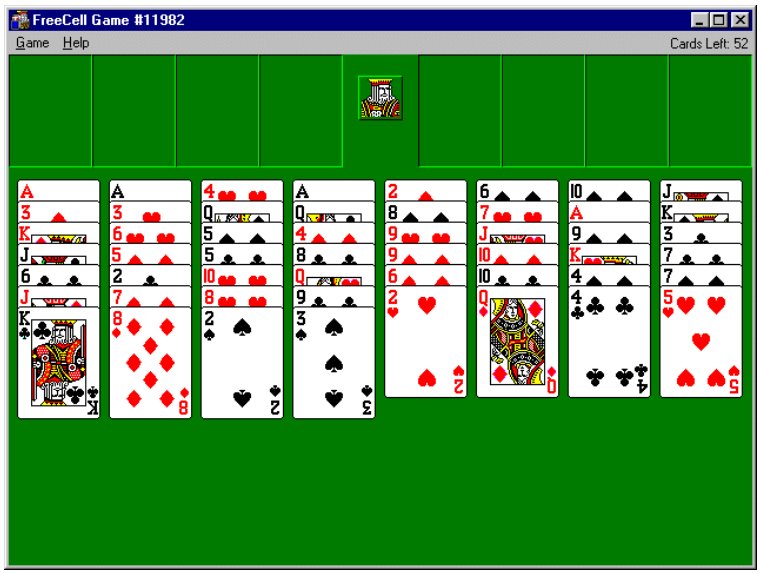
- A single-player Solitaire-like game
- All 52 cards arranged into 8 piles called *cascades*
- Object: arrange cards into four *foundation* piles
- Four FreeCells where any card may be placed temporarily



The FreeCell game included with Windows 95

The Game of FreeCell

- Cascade cards must be placed in descending order with alternating colors
- Basic strategy involves organizing cascades as much as possible
- Proven to be NP-Hard
- Made popular through Windows 95
- *Microsoft 32k* - set of 32,000 solvable deals included



The FreeCell game included with Windows 95

Before GA-FreeCell

- Initially used depth-first iterative deepening
  - Could not solve any problems in Microsoft 32k
- Used heuristic developed by George Heineman to estimate goal distance
  - Peformed better, but not good enough
- Implemented Heineman's Staged Deepening Algorithm (HSD) completely (for comparison)
- Created GA-FreeCell to evolve new search heuristics

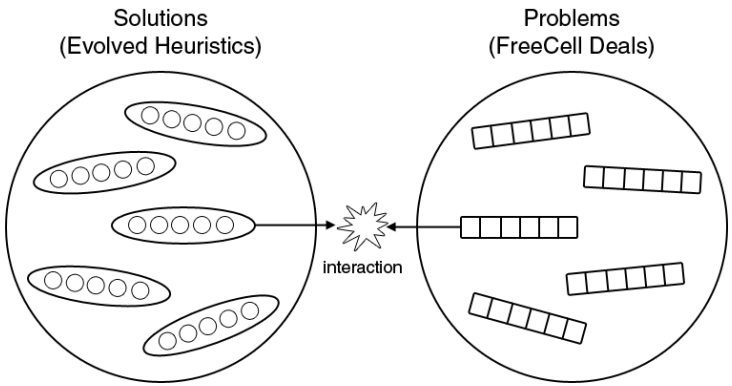
GA-FreeCell Individuals

- Individuals are sets of heuristics that guide a staged deepening search
- Heuristics use input from game state
- Outputs are normalized to between 0 - 1
- Then multiplied by corresponding weight value
- Values are summed to provide total heuristic value

Name	Description
HSDH	Heineman's staged deepening heuristic (Explained in-depth in his paper)
NumberWellPlaced	Number of well-placed cards in cascade piles
NumCardsNotAtFoundations	Number of cards not at foundation piles
FreeCells	Number of free FreeCells and cascades
DifferenceFromTop	Average value of top cards in cascades minus average value of top cards in foundation piles
LowestHomeCard	Highest possible card value minus lowest card value in foundation piles
HighestHomeCard	Highest card value in foundation piles
DifferenceHome	Highest card value in foundation piles minus lowest one
SumOfBottomCards	Highest possible card value multiplied by number of suits, minus sum of cascades' bottom card

Coevolutionary Algorithm

- Used *Hillis-style coevolution*
- Population of solutions evolves alongside population of problems
- Problems represented by sets of six FreeCell deals
- Fitness of problems inversely proportional to fitness of solutions
- As the CEA continues progressing:
  - Problems find more difficult deals
  - Heuristics get better at solving them



A representation of two populations in Hillis-style coevolution

Experimental Setup

- Two populations containing 40-60 individuals each
- Runs lasted 300-400 generations
- 20% chance of reproduction
- 70% chance of crossover
- 10% chance of mutation
- Mutation operator replaced weights with random value between 0 - 1

GA-FreeCell Results

- Reduced amount of search by 87%
- Time to find solution reduced by 93%
- Number of moves needed reduced by 41%
- GA-FreeCell solved 98% of Microsoft 32k

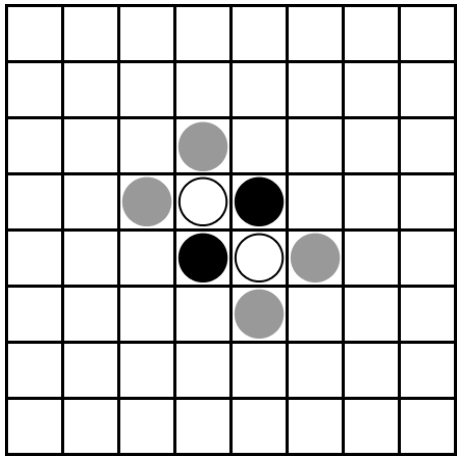
Algorithm	**Average Time to Solve**	**Deals Solved**
HSD	709 seconds	30,859
GA-FreeCell	150 seconds	31,475

Solving Othello

- Study by Edward Manning to solve Othello (aka Reversi) [6]
- Uses coevolutionary algorithm to evolve agents that play Othello
- Makes use of artificial neural networks

The Game of Othello

- Two-player, played on an 8×8 grid
- Starts with two pieces of each color in center of grid
- Pieces must be placed across from another of the same color, with one or more opponent pieces "sandwiched" in-between
- These pieces are then captured and turn to the opposite color
- Object: have most pieces in your color in the end



Starting positions for the game of Othello. Potential next moves for the black player are shown in gray.

Othello Individuals

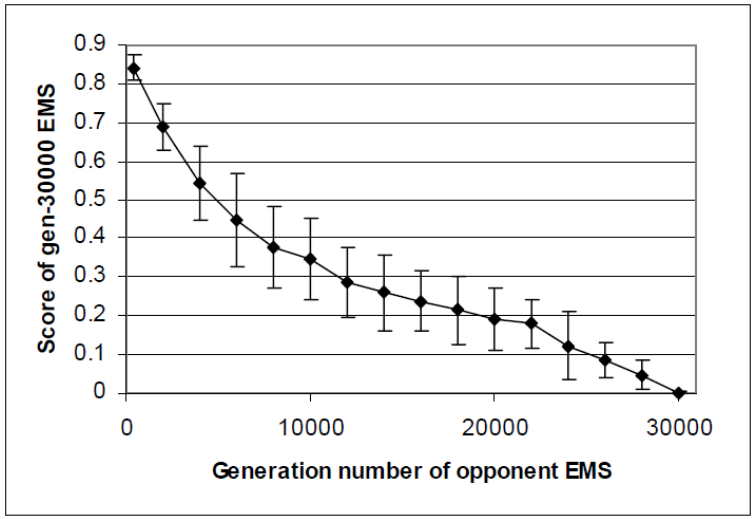
- Individuals are n-tuple neural networks
- A collection of 12 6-tuple neurons
- Neurons take input from different locations on game board
- Values in weight tables correspond to the perceived effectiveness of different moves
- Output from neurons are used by network to decide best move

Coevolutionary Algorithm

- Single population, uses a modified form of all vs. best evaluation
- *Support set* - maintained set of the best individuals seen so far
- Individuals play games against 40 random peers and members of support set
- Individuals that do poorly against peers get replaced
- Crossover and mutation happen similar to most CEAs

Experimental Setup and Results

- Performed 5 runs with 30,000 generations each
- Each generation had an *equilibrium mixed strategy* (EMS) - combination of best individuals into one
- There was a steady increase in competency of the EMS in later generations
- 30,000 generation EMS would win against 400 generation EMS 92% of the time.



Comparison of fitness score between 30,000 generation EMS against previous generations. Error bars show standard deviation of the sample.

Conclusions

- CEAs evolve highly competent game-playing agents
- Don't need to rely on human experience
- Could potentially be applied to other problems
  - Spatial navigation
  - Computer vision
  - Controlling robots

References

[1] M. Campbell, A. H. Jr., and F. hsiung Hsu.
Deep Blue.
Artificial Intelligence, 134:57 – 83, 2002.

[2] D. B. Fogel.
Evolving a checkers player without relying on human experience.
Intelligence, 11:20–27, June 2000.

[3] N. Franken and A. P. Engelbrecht.
Evolving intelligent game-playing agents.
In *Proceedings of the 2003 annual research conference of the South African Institute for Computer Scientists and Information Technologists*, SAICSIT '03, pages 102–110, , Republic of South Africa, 2003. South African Institute for Computer Scientists and Information Technologists.

[4] A. Hauptman, A. Elyasaf, M. Sipper, and A. Karmon.
Gp-rush: using genetic programming to evolve solvers for the rush hour puzzle.
In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 955–962, New York, NY, USA, 2009. ACM.

[5] R. E. Korf.
Depth-first Iterative-Deepening: An optimal admissible tree search.
Artificial Intelligence, 27:97–109, 1985.

[6] E. P. Manning.
Coevolution in a large search space using resource-limited Nash memory.
In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 999–1006, New York, NY, USA, 2010. ACM.