Evolving a Checkers Player

Dirk Thierens

Universiteit Utrecht
The Netherlands

Figure 1. The opening board position. The picture is from an Internet gaming site, www.zone.com, where people can log in to play a variety of games. In this case, the game is checkers. Players can chat by using the "chat box" below the board.

Evolving a Checkers Player

Can we build intelligent systems to **learn** to play checkers ?

- No expert knowledge provided to the learning system.
- Programs simply have to play against themselves, and figure out how to play.

Evolutionary computation feasible approach ?

Game playing

- Board representation
- Move search: minimax algorithm
- Traditional game playing programs: board evaluation functions are extensively knowledge based
  1. weighted feature function
  2. opening games
  3. end games table look-up

$\Rightarrow$ they do not learn by themselves !

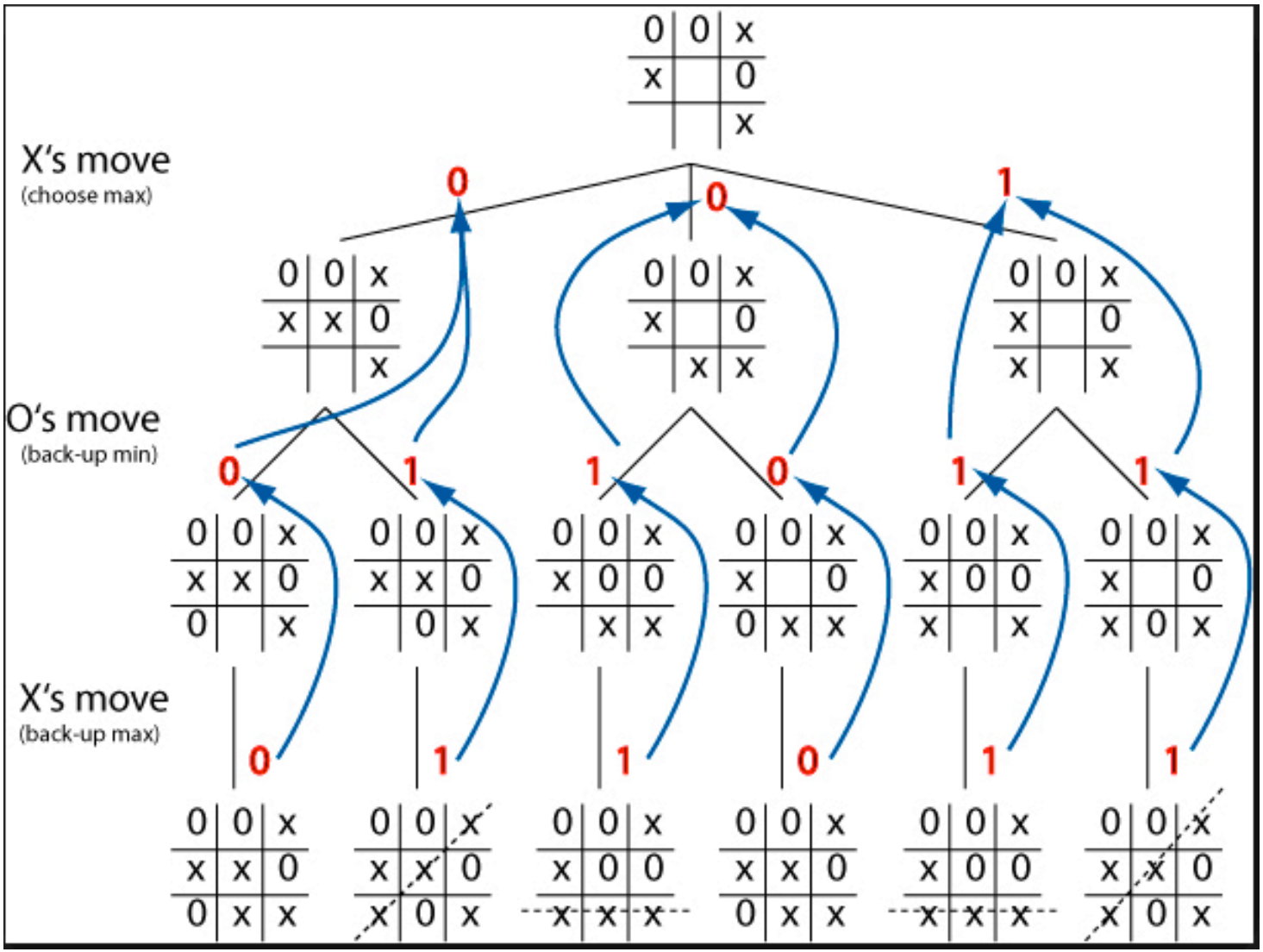
Here: Evolving Neural Networks to Play Checkers

Board representation

- Output $\in [-1 \ldots +1]$
  -1: loss positions
  +1: win positions
  $\rightarrow$ closer to $+1 \Rightarrow$ better evaluations
- Input: vector of 32 possible positions, 5 possible values
  1. - K : king opponent
  2. - 1 : checker opponent
  3. 0 : empty
  4. + 1 : checker self
  5. + K : king self

  K $\in [1 \ldots 3]$: exact value evolved

Mini-Max algorithm



Game lookahead

- The further we can lookahead the better.
- Computational restrictions: number of possible board positions grows very fast with increasing number of lookahead levels.
- Deep Blue when defeating chess champion Garry Kasparov made 200 million chess board evaluations per second !
- Here only lookahead search of 2 moves each side when evolving.
- When testing against players on Internet: lookahead search of 3 moves each side.

Board Evaluation

- Evaluation function represented by an artificial neural network

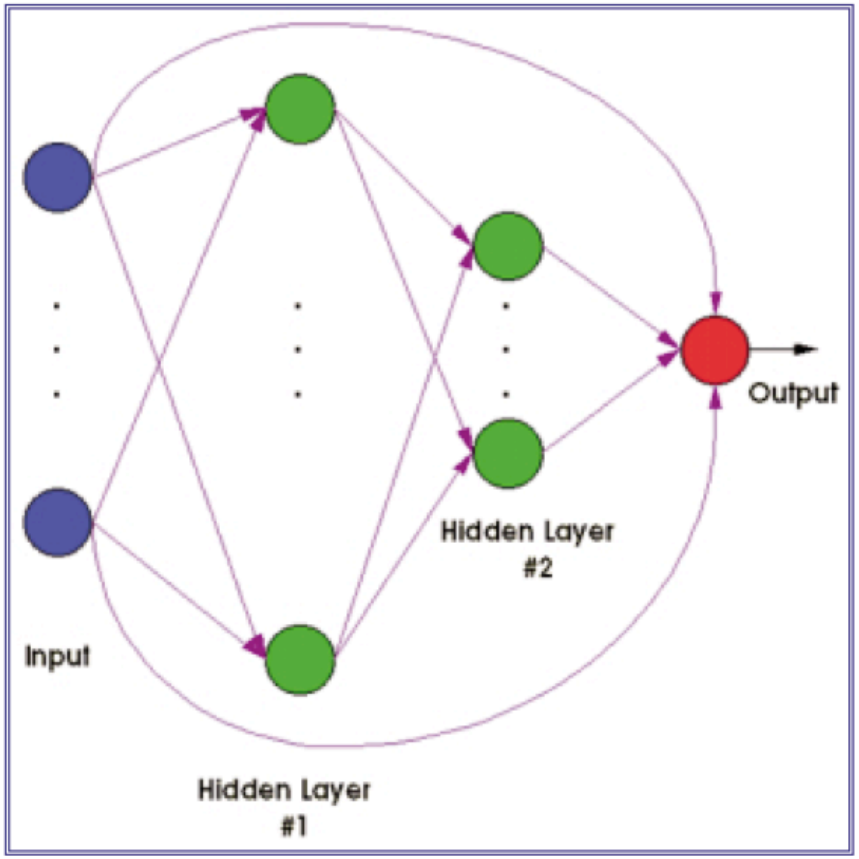


Figure 2. The neural network topology chosen for the evolutionary checkers experiments. The networks have 32 input nodes (blue) that correspond to the 32 possible positions on the board. The two hidden layers (green) comprise 40 and 10 hidden nodes, respectively. All input nodes are connected directly to the output node (red) with a weight of 1.0. Bias terms affect each hidden and output node as a threshold term (not pictured).

Neural Network Architecture

- Input Layer: 32 inputs
- First hidden layer: 40 neurons
- Second hidden layer: 10 neurons
- Direct input-output connections with weight 1.0
- Total number of neural network weights (incl. bias term) = (32 + 1) x 40 + (40 + 1) x 10 + (10 + 1) x 1 = 1741

Need to determine values for the 1741 weights

$\Downarrow$

Use evolutionary algorithm to co-evolve the weights

Evolutionary search for network weights

- Mutate neural network weights by adding a small random, Gaussian distributed number to each weight.
- Each weight has its own Gaussian distribution (different widths or standard deviations).
- The width or variance of each Gaussian distribution also evolves by mutation.
- For each neural network NN_i all the $N_w(= 1741)$ neural network weights $w_i(j)$ $(j = 1 \ldots N_w)$ gets associated with the corresponding standard deviation $\sigma_i(j)$ $(j = 1 \ldots N_w)$ of the Gaussian mutation distribution (mean value is always zero).

Self-adaptive Mutation

First mutate the N_w widths of the Gaussian distributions, then mutate the N_w weights ($j = 1 \ldots N_w$):

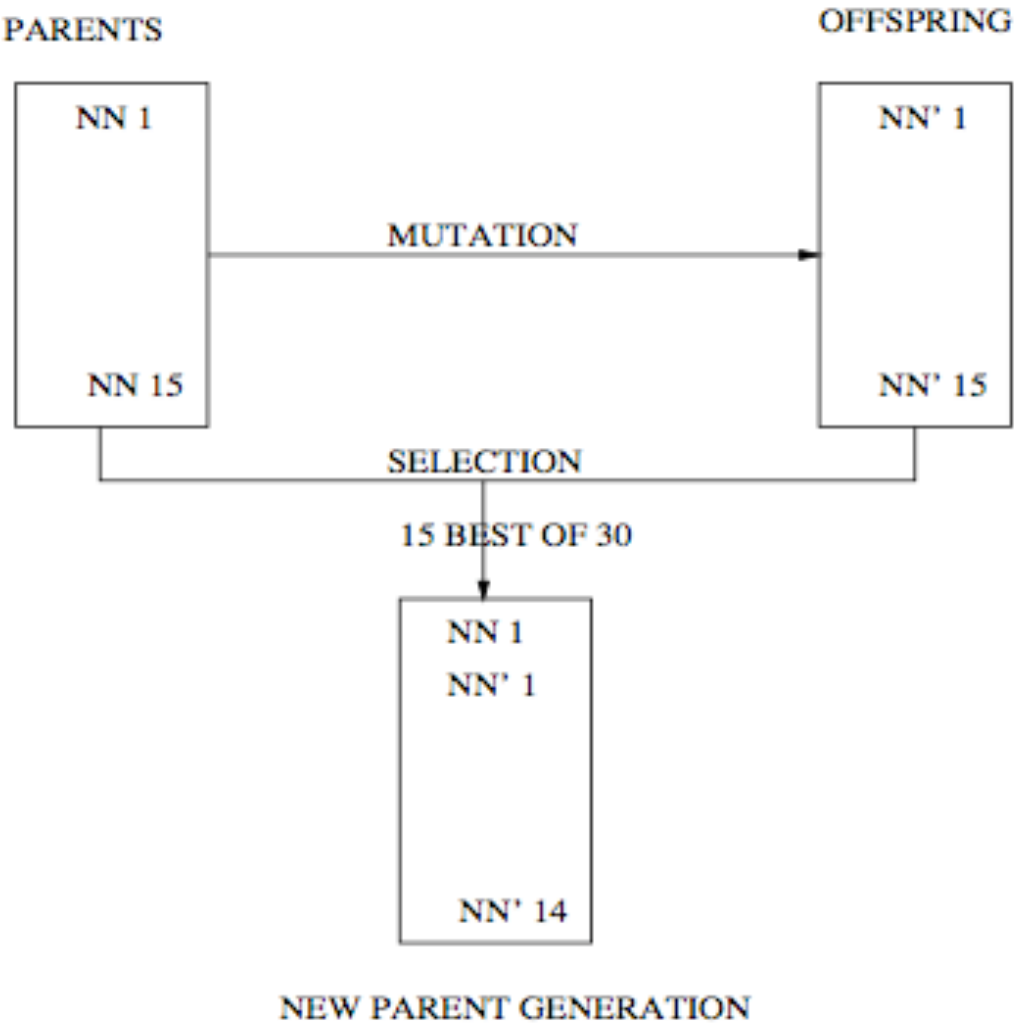
- Self-adaptive mutation of the mutation step-size:

$$\sigma_i'(j) = \sigma_i(j)\ exp(\frac{RandNorm_j(0,1)}{\sqrt{2\sqrt{N_w}}})$$

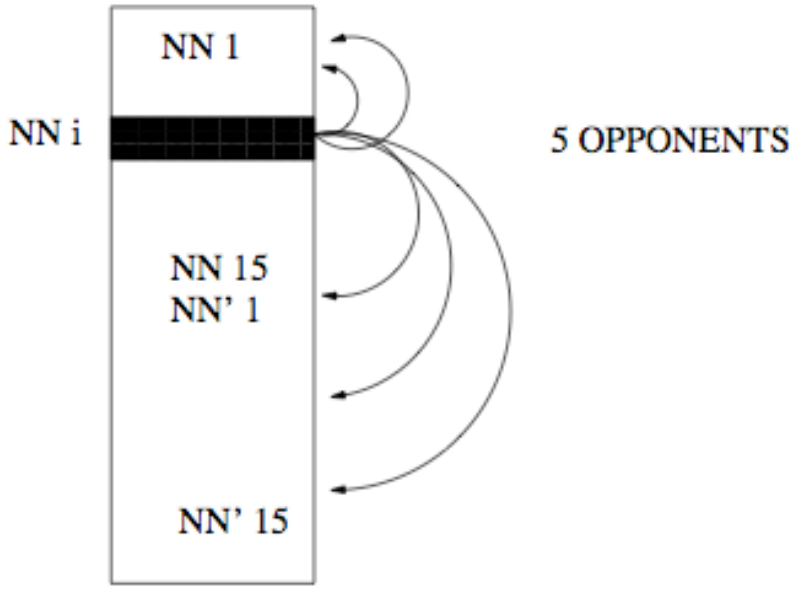
- Mutation of the neural network weights:

$$w_i'(j) = w_i(j) + \sigma_i'(j) RandNorm_j(0,1)$$

Evolutionary Cycle



Fitness evaluation



- 15 parents + 15 offspring neural networks
- Each NN competes against 5 randomly chosen NN
- Score: win : + 1; draw : 0; loss : -2
- Fitness: sum of scores

Darwinian system ?

1. Structures ?
   → **neural networks**
2. Structures are copied ?
   → **15 parents ⇒ 15 offspring**
3. Copies partially vary from the original
   → **weights Gaussian mutated**
4. Structures are competing for a limited resource
   → **fixed population size each generation**
5. Reproductive success depends on environment
   → **winning strategies survive**

Experiment

- total of 250 generations evolved
- 15 neural networks each generation
  $\Rightarrow$ 15 x 250 = 3750 neural networks created
- fitness evaluation: 15 parents + 15 offspring: each competing against 5 others
  $\Rightarrow$ 30 x 5 x 250 = 37500 games played

Checkers Rating

Class	Rating
Senior Master	2400+
Master	2200-2399
Expert	2000-2199
Class A	1800-1999
Class B	1600-1799
Class C	1400-1599
Class D	1200-1399
Class E	1000-1199
Class F	800-999
Class G	600-799
Class H	400-599
Class I	200-399
Class J	below 200

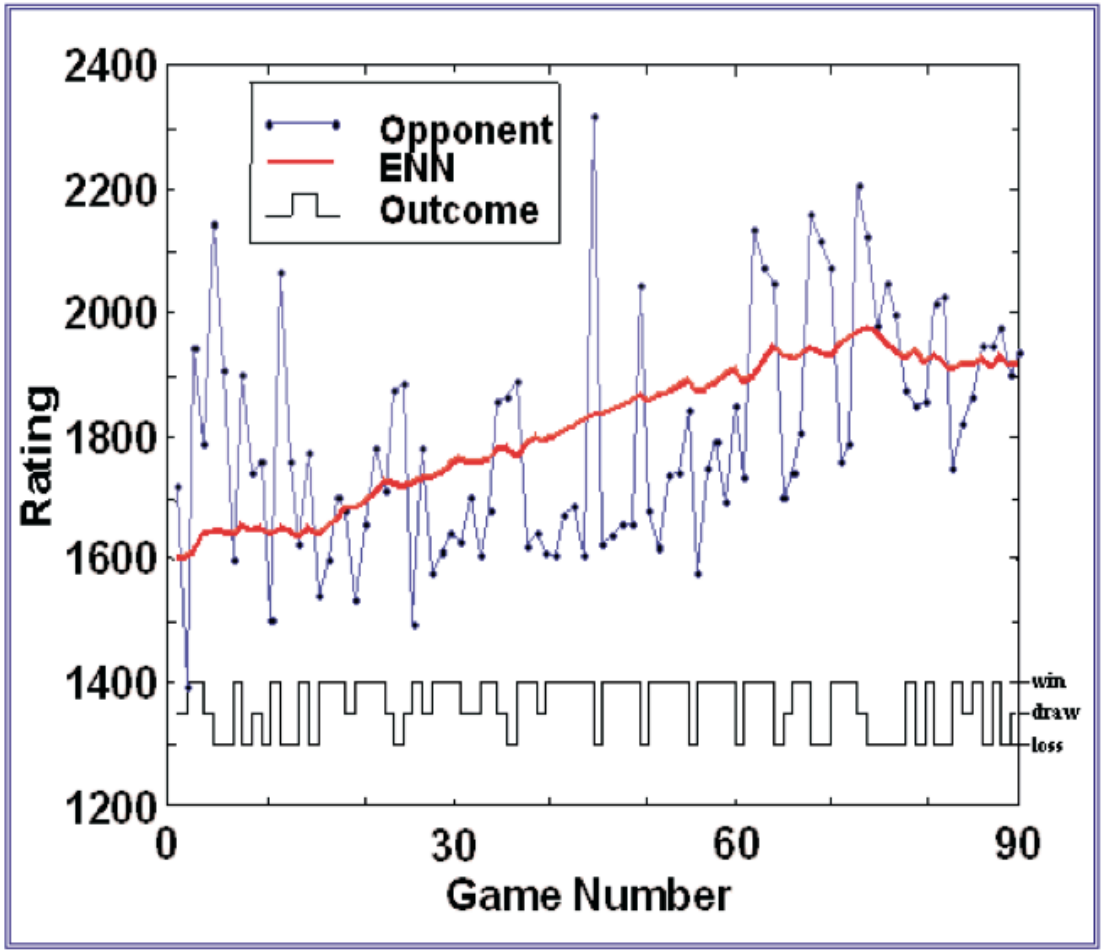


Figure 4. The sequential rating of the evolved neural network (ENN) throughout the 90 games played against human opponents. The graph indicates both the network's rating and the corresponding rating of the opponent on each game, along with the result (win, draw, or loss). The highest rating for the ENN was 1975.8 on game 74. The evolved network completed the 90 games by earning a Class A rating.

- Evolved neural network rating: 1902

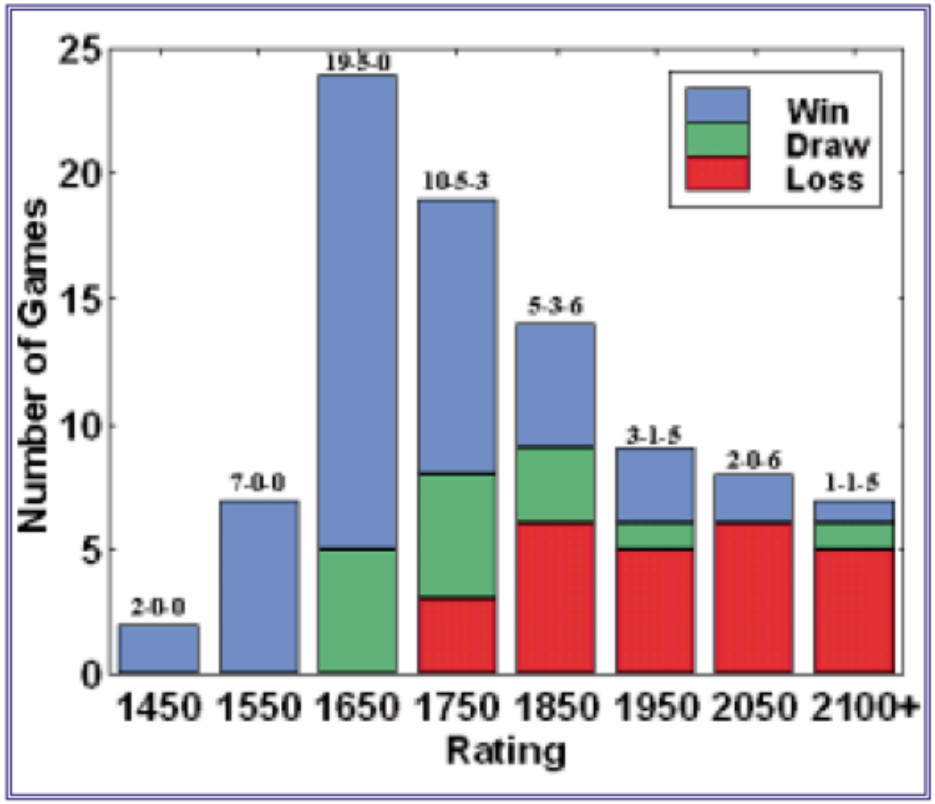


Figure 3. The performance of the evolved neural network after 250 generations, played throughout 90 games against human opponents on www.zone.com. The histogram indicates the rating of the opponent and the associated performance against opponents with that rating. Ratings are binned into intervals of 100 units (that is, 1650 corresponds to opponents who were rated between 1600 and 1699). The numbers above each bar indicate the number of wins, draws, and losses, respectively. Note that the evolved network generally defeated opponents who were rated below 1800 and played to about an equal number of wins and losses with those who were rated between 1800 and 1899.

- 1 draw against player rated 2207 , ie. master level, ranked 18 out of 40000 listed players

Discussion

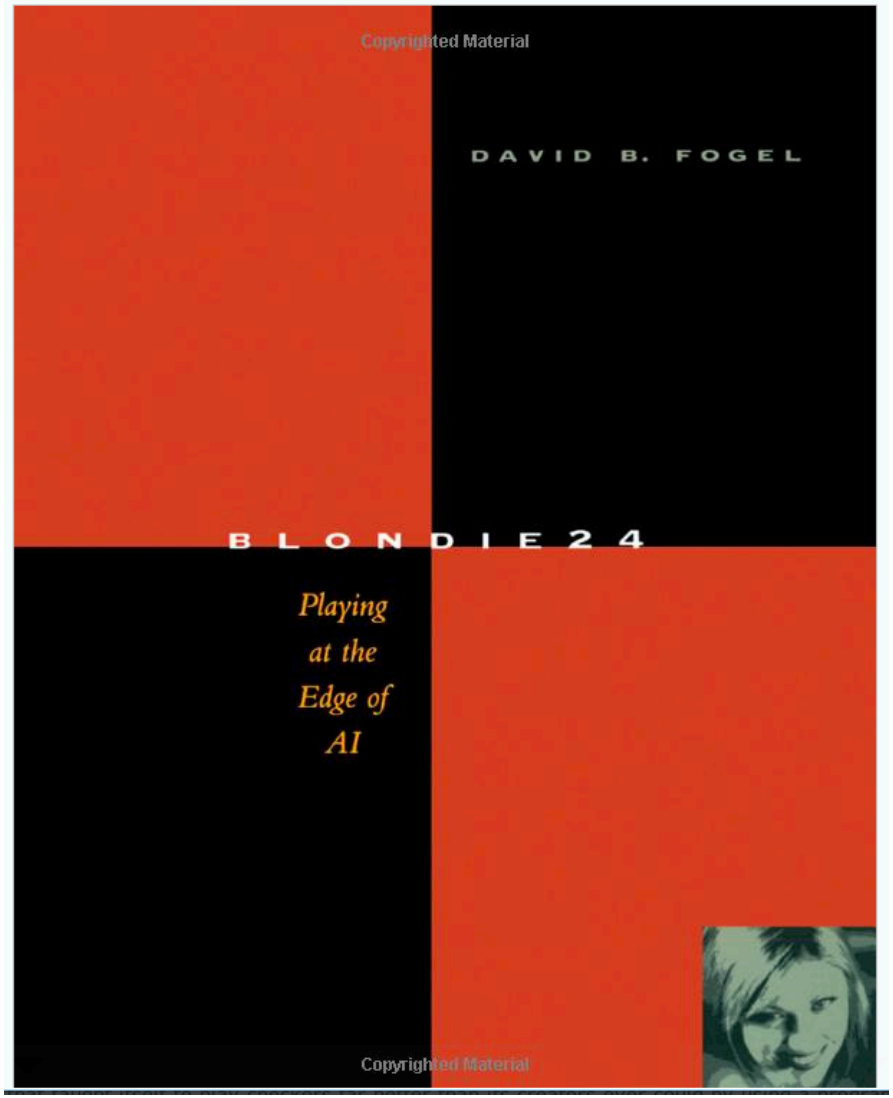
- Chinook: opening games, end games table look-up, hand-crafted evaluation function
- Deep Blue: 200 million board evaluations per second vs. 3500 here
- Payoff: summed score over 5 games $\rightarrow$ no immediate feedback about winning or losing a single game
- Input to neural networks do not give spatial information: only 1 x 32 vector of {-K,-1,0,1,K}

Allen Newell:

"It is extremely doubtful whether there is enough information in 'win, lose, or draw' when referred to the whole play of the game to permit any learning at all over available time scales"

Conclusion

- Building intelligent systems by evolutionary computing.
- Learn to play checkers at high level by competing against themselves.
- No tedious domain knowledge extraction.
- Learning by evolution: feasible approach.


Copyrighted Material
DAVID B. FOGEL
BLONDIE24
Playing at the Edge of AI
Copyrighted Material