Phoenix: A Self Learning Chess Program

M. Tech Thesis

Rahul A R

Outline

- Background
- Chess programming basics
- Initial ideas
- Genetic Algorithms
- Implementation
- Result

Background

- Chess is a test bed for AI research
- Structured with well defined rules
- Readily available expertise
- Study human reasoning modes
- Problem solving and uncertainty management

Chess Programming

An incomplete summary

Chess Notation

- SAN, PGN, FEN, etc
- Rows are Ranks (1-8)
- Columns are Files (a-h)
- Pieces are denoted using first letters
- E.g.: 1.d4 d5 2.c4 Nf6 3.Nc3 dxc4 4.e4 Nc6 ....
- We mainly use PGN

Bit Boards

- 1 bit for each square
- 64 squares in 1 word
- 1 bit board to represent 1 piece
- Other type of information can be stored (legal moves, king ring, etc)
- Easy to manipulate (bitwise operations)
- E.g.: all possible moves & ~(all other pieces of the same color) to get the legal moves of a piece

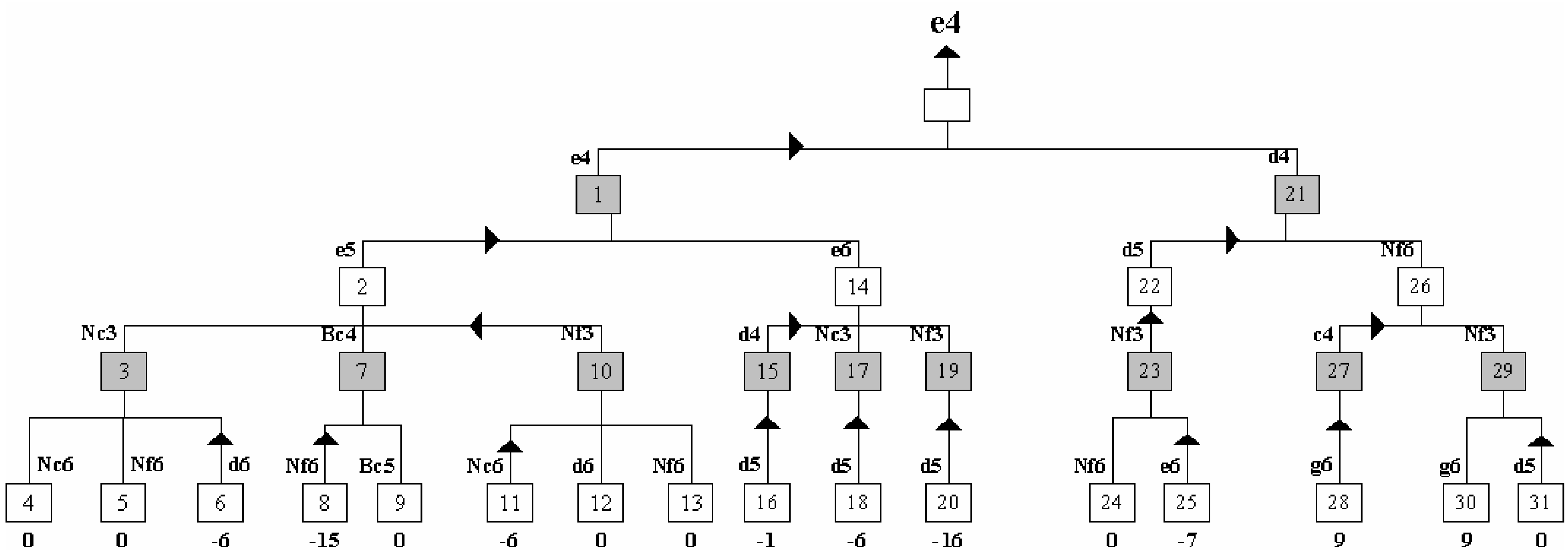
Move Search

- Can be represented as a tree
- Can we traverse till the leaves?
- $30^{120} = 1.8 \times 10^{177}$ (30-35 replies)
- $5^{120} = 7.5 \times 10^{83}$ (4-5 replies)
- But we don’t play this way!
- There is more than what meets the eye

α-β pruning

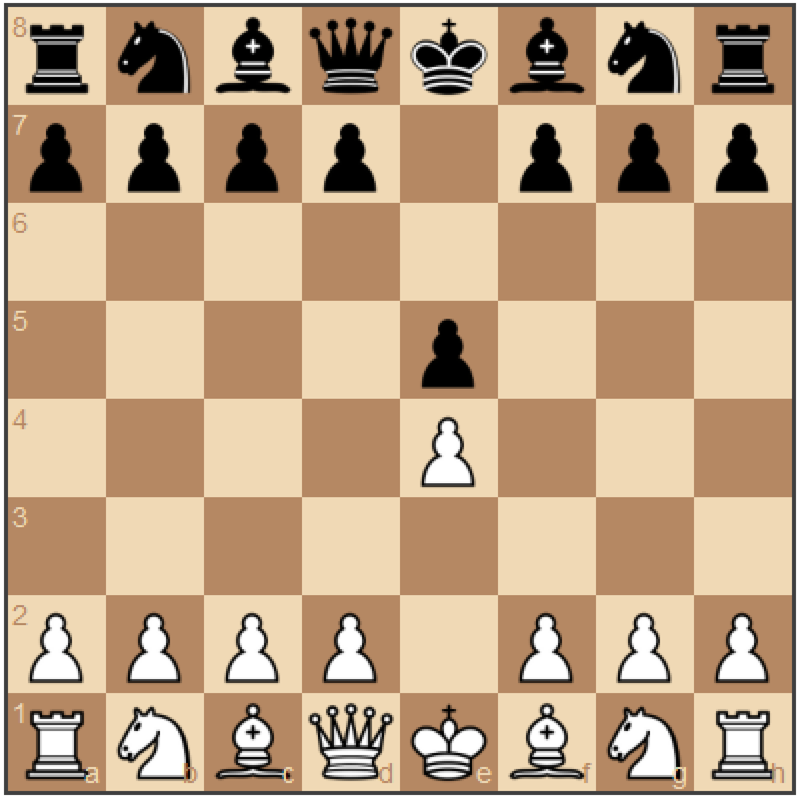
- Machines understand only numbers
- Uses Minimax strategy to search
- But there is a better way
- Cut off sub-trees without evaluation
- Saves a lot of time
- Increases depth per unit time

α-β pruning



e4
e4
d4
1
21
e5
e6
d5
Nf6
2
14
22
26
Nc3
Bc4
Nf3
d4
Nc3
Nf3
Nf3
c4
Nf3
3
7
10
15
17
19
23
27
29
Nc6
Nf6
d6
Nf6
Bc5
Nc6
d6
Nf6
d5
d5
d5
Nf6
e6
g6
g6
d5
4
5
6
8
9
11
12
13
16
18
20
24
25
28
30
31
0
0
-6
-15
0
-6
0
0
-1
-6
-16
0
-7
9
9
0

Transposition Tables



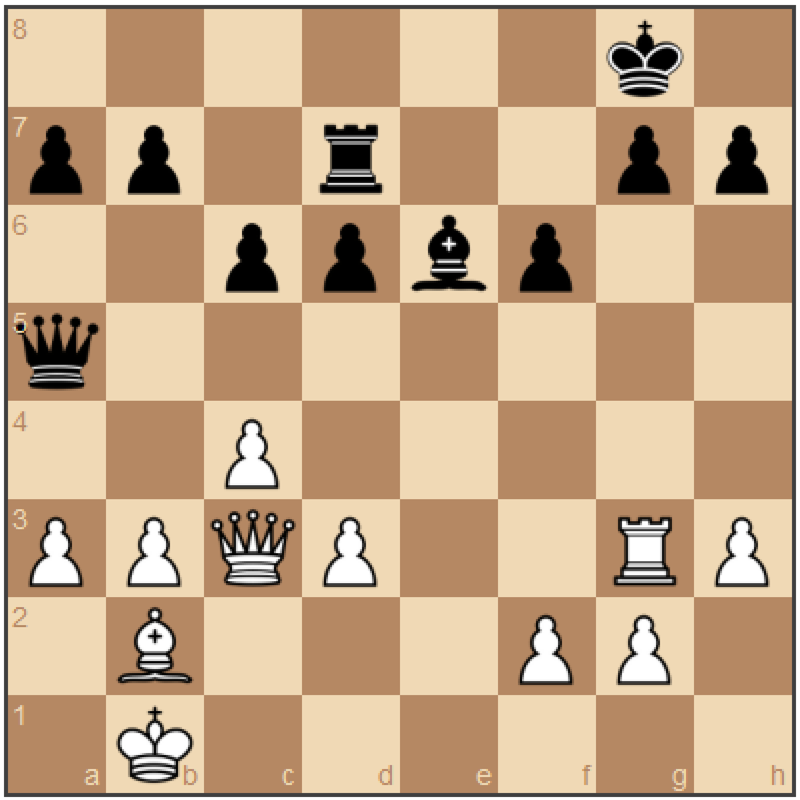
Transposition Tables

- To avoid redundant evaluations
- Use unique ID for each position
- Zobrist Hashing
  - Each piece on every square has a random ID
  - $12 \times 64 = 768$ random numbers
  - Empty squares are 0
  - All the IDs are XORed
  - Maintain another set just in case

Move Ordering

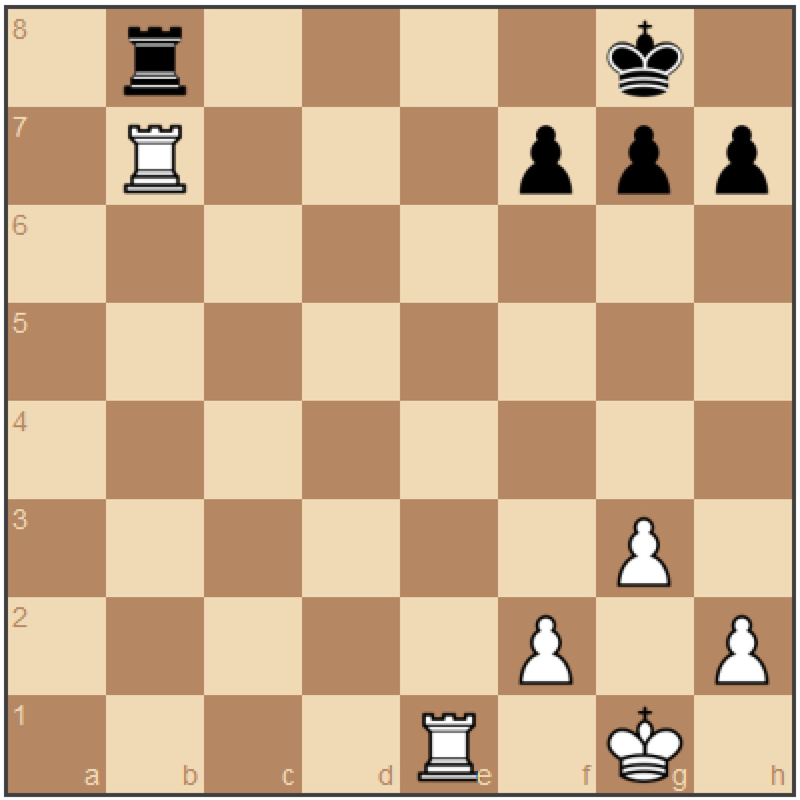
- Best moves always enhance pruning efficiency
- But they are hard to find!
- Order moves using some heuristic
  - First consider moves causing material imbalance
  - Next consider check moves
  - Consider everything else later

Killer Moves



- It doesn't matter if Black moves to h6 or h5
- White's reply is Qxa5
- Therefore the sub-tree rooted at queen capture can be pruned
- Such moves which allow major pruning are called Killer Moves

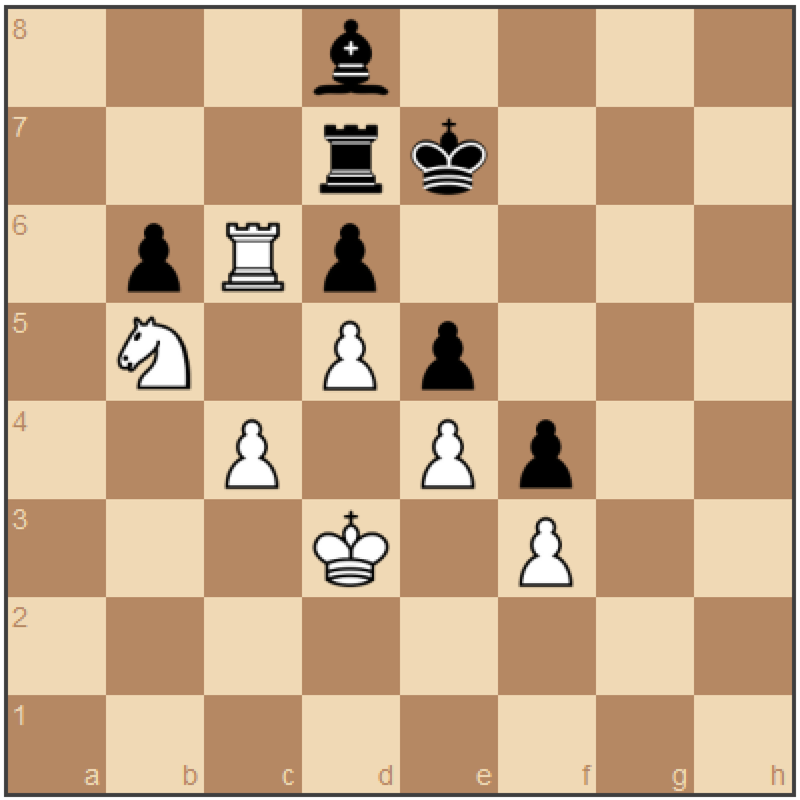
Quiescence Search



Black to move

- Assume this is at depth 4 in a 5 ply engine
- Horizon Effect
- Evaluate only quiescence (read dead) positions
- Put extra effort into captures, promotions till all changes appear

Null Moves



Zugzwang

- Allow opponent to play twice
- If nothing bad happens, cut off the sub-tree
- Saves a lot of time
- Takes only 3% of total time. So no loss on failure
- Zugzwang positions are an exception and are almost always losing
- They occur during end-games and so null move application is stopped after the number of pieces on the board is less

Opening Books

- Opening theory is heavily analyzed
- No need to re-invent the wheel
- Usually stored for efficient retrieval
- End game tablebases are also available
- Common endings are stored assuming perfect play with scores ($+\infty$, $-\infty$, 0)
- When these positions occur, they are considered as leaves and no further evaluation takes place

Evaluation

Evaluation Function

- Traversal of game tree till leaves is impossible
- We need some measure to quantify moves
- Features are extracted from positions and weighted according to importance
- Their sum gives an *Evaluation Score*

$$F = \sum_{i=1}^{N} x_i \cdot \ v_i \,, \qquad \begin{cases} x_i \in \{0,1\} \\ v_i \in R \end{cases} , i = \overline{1,N}$$

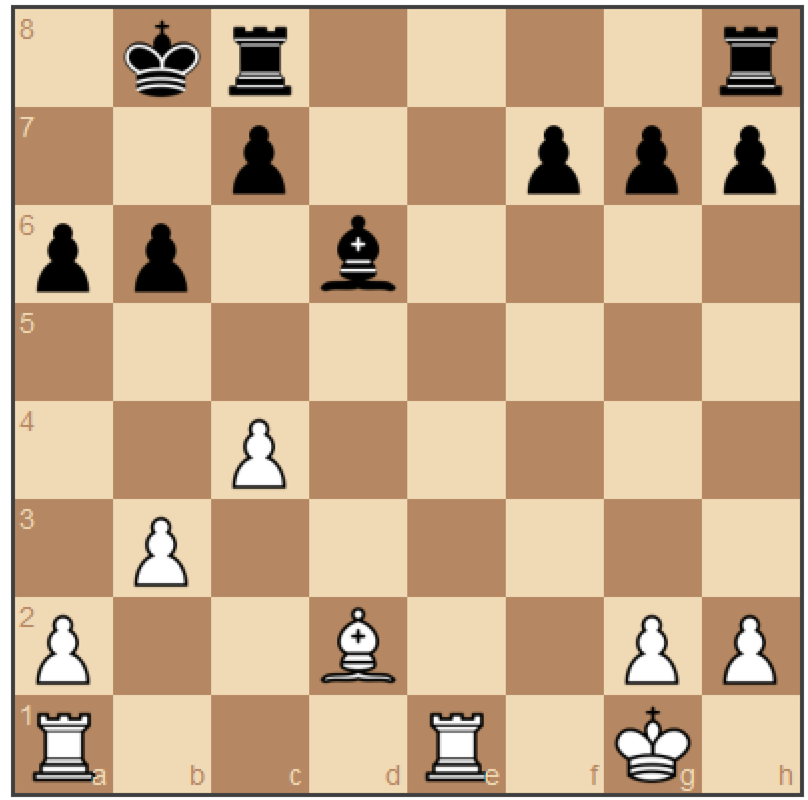
Evaluation Function

- Simplest evaluation: Material balance
- But useful only in trivial cases
- Players often sacrifice pieces as a gambit to gain space (called non-material advantage)
- Humans can see more into a position
- They are called Positional Parameters
- Explicitly extracted in all engines

Positional Parameters

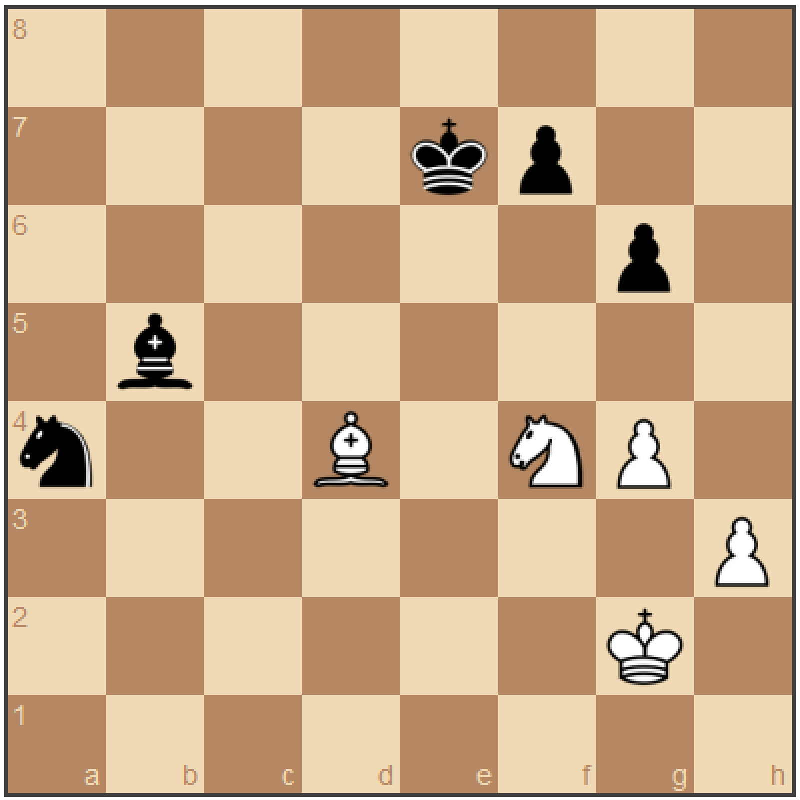
Castling

Rook on a (semi) open file

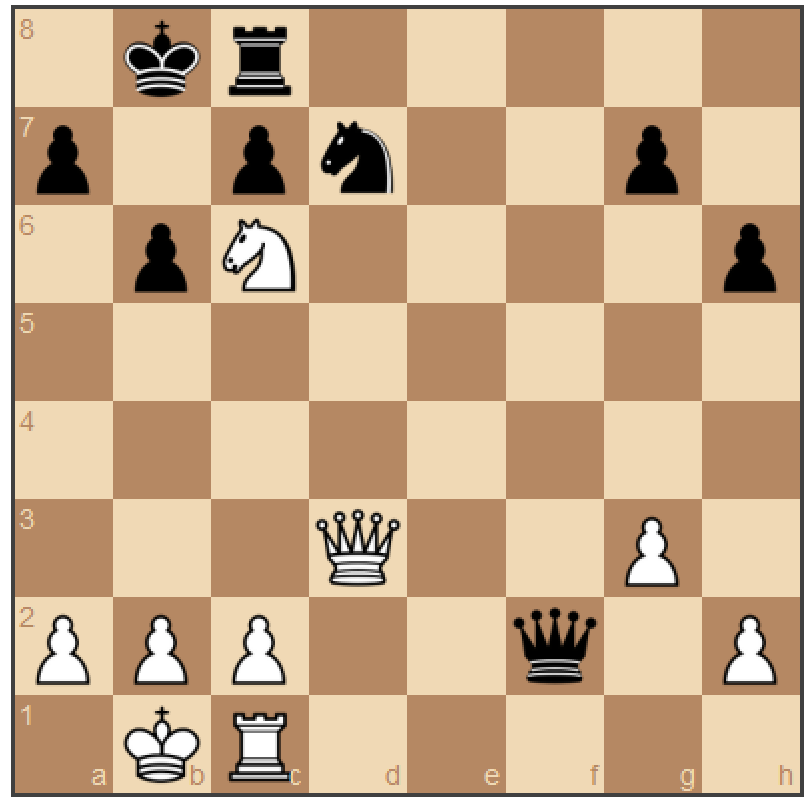


Positional Parameters

Knight's mobility

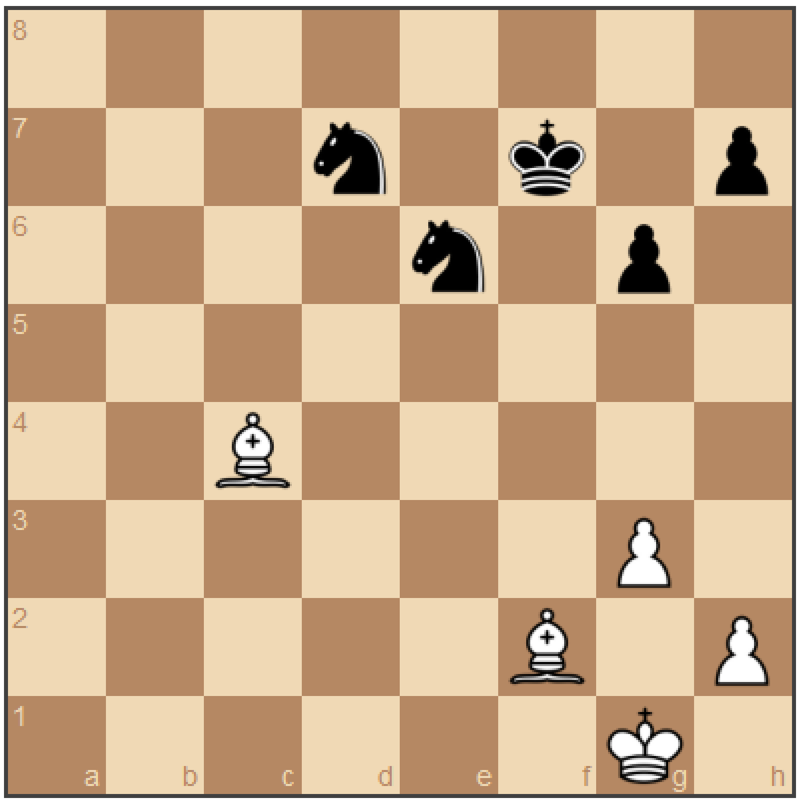


Outposts



Positional Parameters

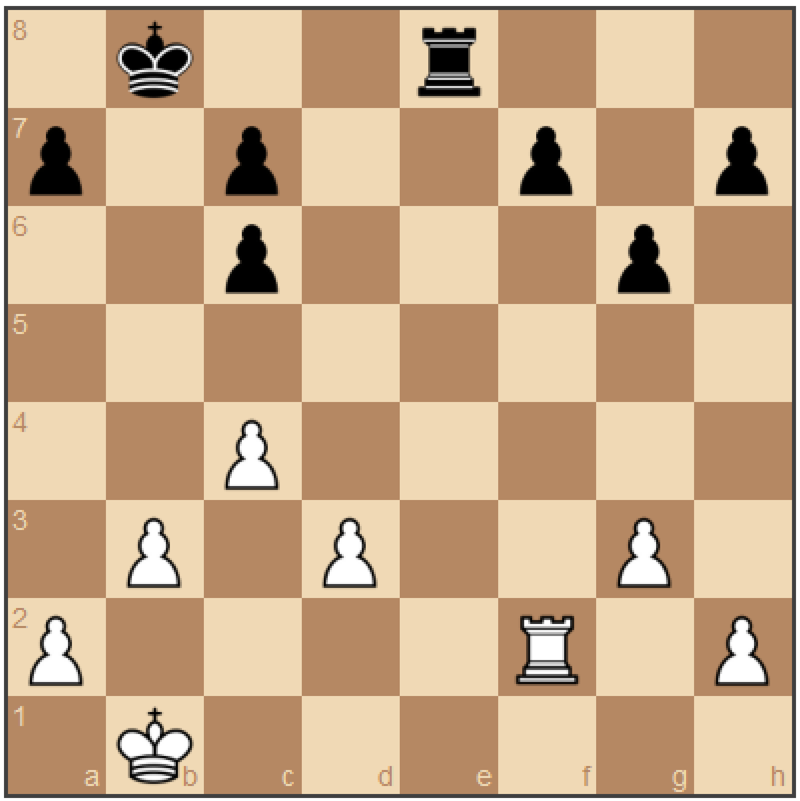
Bishop pair/long diagonals



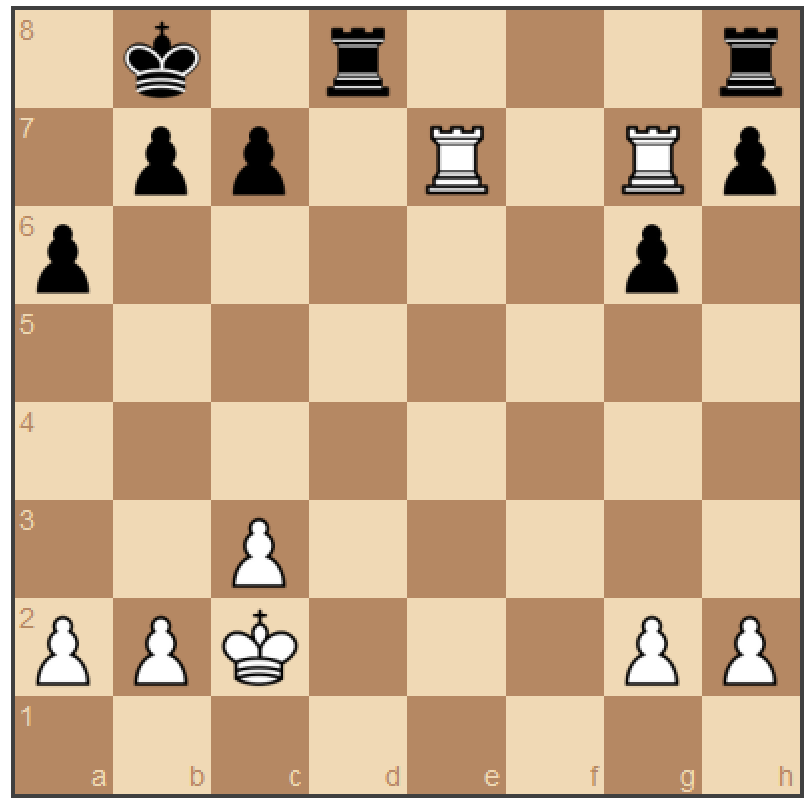
Central pawns

Positional Parameters

Doubled/Backward Pawns

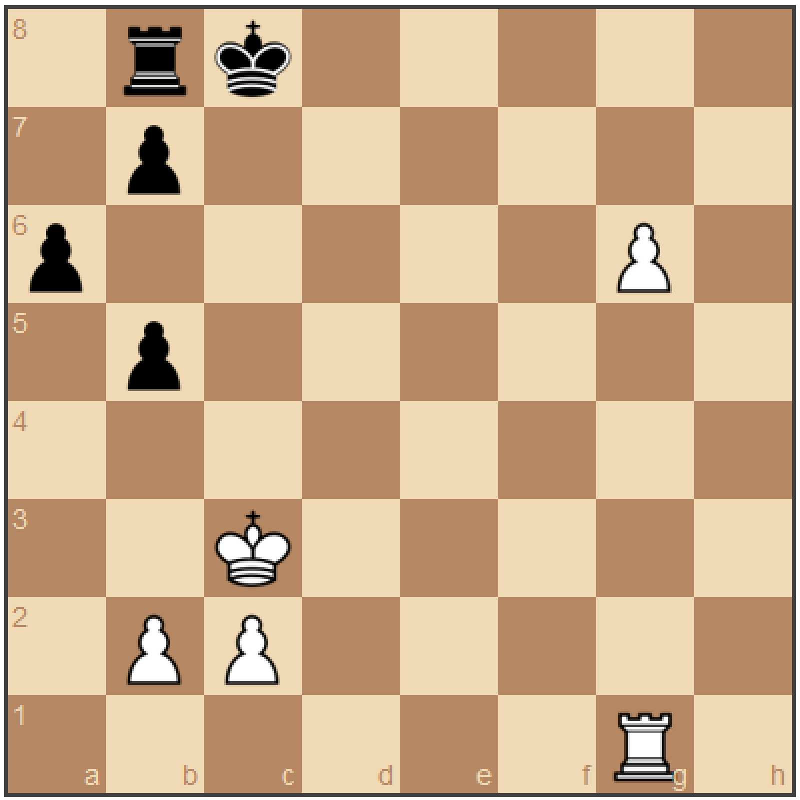


(Connected) rooks on the 7th rank



Positional Parameters

(Rook supported) passed pawn



- There are many other parameters which are considered in engines
- But these are the most important
- We need to find a way to make our program learn these parameters on it’s own

Initial Ideas

Which failed

Neural Networks

Training Data Generation

[1 0 0 1 1 1 0 0 1 0 1 0 1 .... : +0.1]

=

Converter

[rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1]
to
[1 0 0 1 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 0 1 0 1 0 0 1 1 1 0 0]

+

Evaluation module

[e4: +0.1, d4: +0.1, Nc3: +0.07, ...]

+

Move Generator

[e4, d4, Nc3, Nf3, ...]

Neural Networks

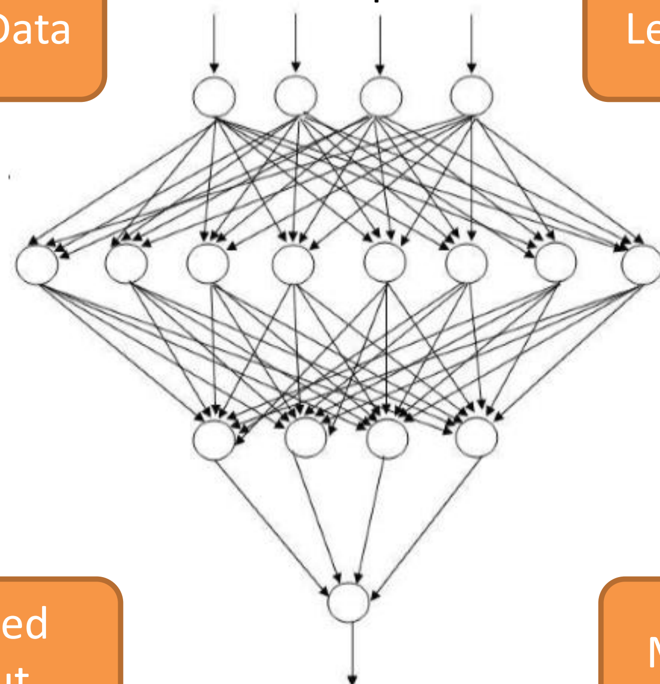
Learning

Training Data

262 inputs

Legal Moves

Learning



Playing

Expected Output

1 output

Make Move

Deep Learning

- Convolutional NN
  - Used to detect spatial correlations
  - 3 CNNs were used for 3 stages of the game
  - Idea was to detect local patterns on the board
  - Attacks are localized if we consider the king ring
  - Defense is localized in terms of good placement of pieces in the king's quadrant
- Autoencoders
  - Tries to find features (similar to PCA)
  - Can be stacked to form a non-linear feature extractor

Deep Learning

- The features are not spatially correlated which means that it is not easy to recognize patterns
- A universal mathematical function for chess evaluation does not exist.
- If it does, its dimensionality will be so high that it would require extremely large networks, huge training data and very long training phases for any learning to happen
- Learning happens only when the input given to networks have a direct relation with what is to be learnt.
- In chess however, no such relation exists among the squares, pieces and positions
- We were asking the network to recapitulate knowledge developed over centuries in a very limited time and with limited computational power/training data

Implementation

Genetic Algorithms

- Borrowed from natural evolution of life forms
- Adaptability, Natural Selection
- Survival of the fittest
- Used in Optimization problems
- Fitter individuals allowed to reproduce
- Solutions get better over Generations
- Hope is to find the optimal solution

Genetic Algorithms

Natural Genetics	Genetic Algorithms
Phenotype	A set of parameters to be optimized
Genotype	A population of individuals
Chromosome	Individual
Gene	A parameter from the set
Locus	Position of the parameter (index)
Allele	Value of the parameter

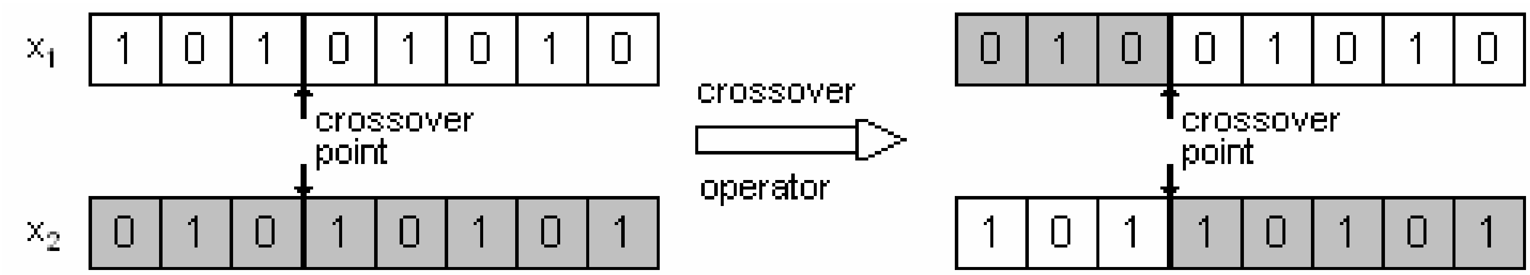
Genetic Algorithms

- 2 important stages:
  - Formulation phase (avoid wrong parameters)
  - Optimization phase
- Fitness function is unique to each problem
- Solution optimality depends on diversity in population, selection pressure, replacement strategy and number of generations
- **Crossover and Mutation**

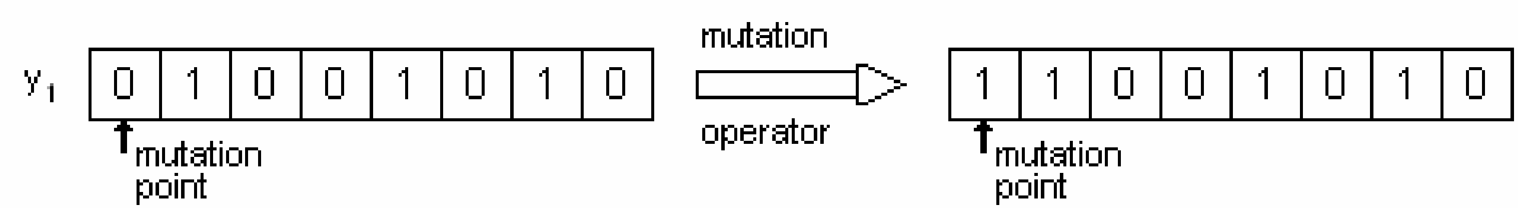
Crossover and Mutation

of Binary Chromosomes

- Crossover

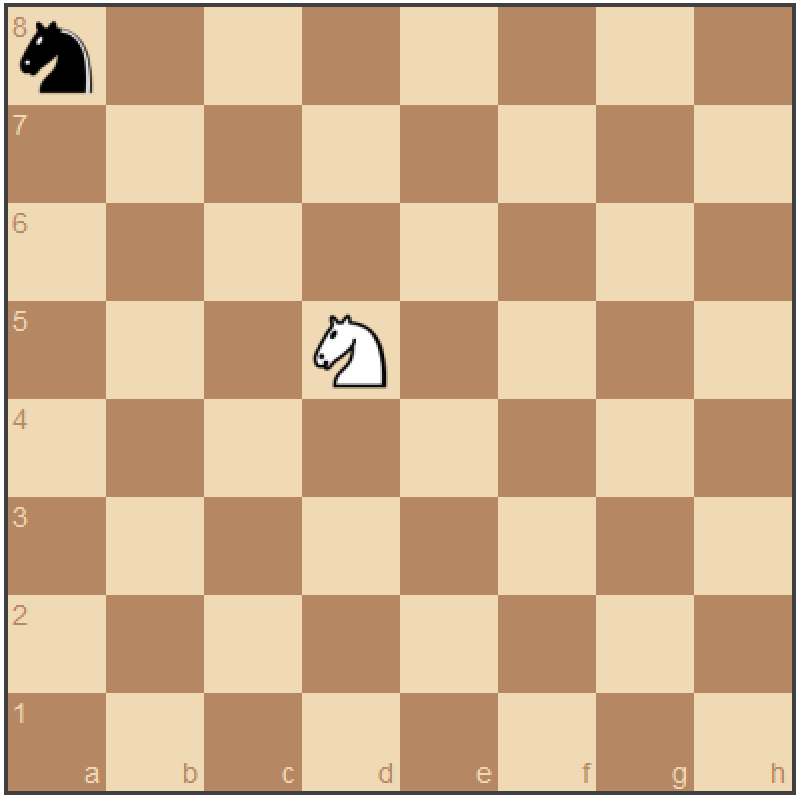


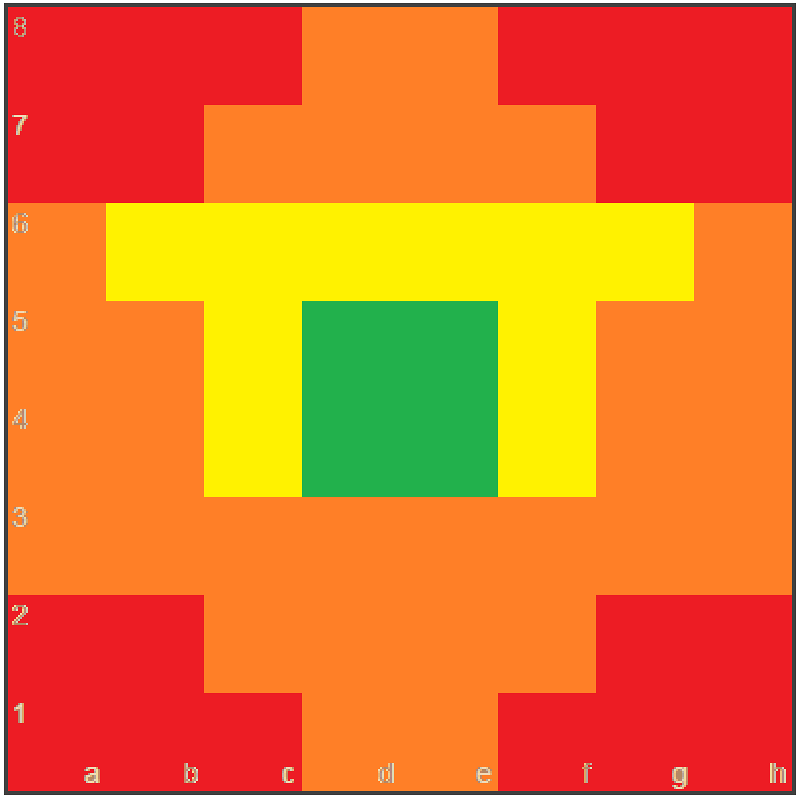
- Mutation



Problem Formulation

PVT for a Knight





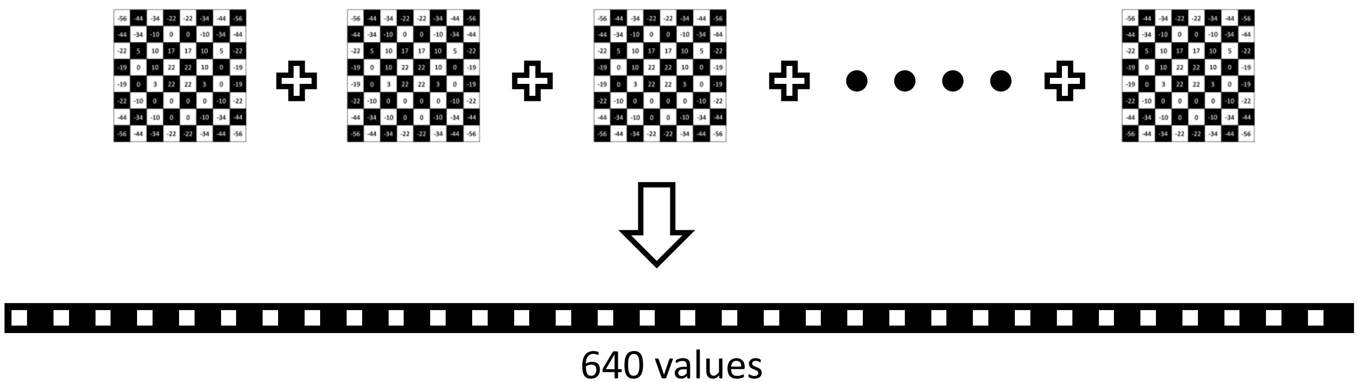
Positional Value Tables

-56	-44	-34	-22	-22	-34	-44	-56
-44	-34	-10	0	0	-10	-34	-44
-22	5	10	17	17	10	5	-22
-19	0	10	22	22	10	0	-19
-19	0	3	22	22	3	0	-19
-22	-10	0	0	0	0	-10	-22
-44	-34	-10	0	0	-10	-34	-44
-56	-44	-34	-22	-22	-34	-44	-56

- This is an ideal PVT for a black knight during middle game
- Pieces have different objectives at different times in a game
- There are 2 PVTs for each piece*; one for middle game and one for end game
- The tables are mirrored for white and black
- 10×64=640 values
- They are the optimization parameters

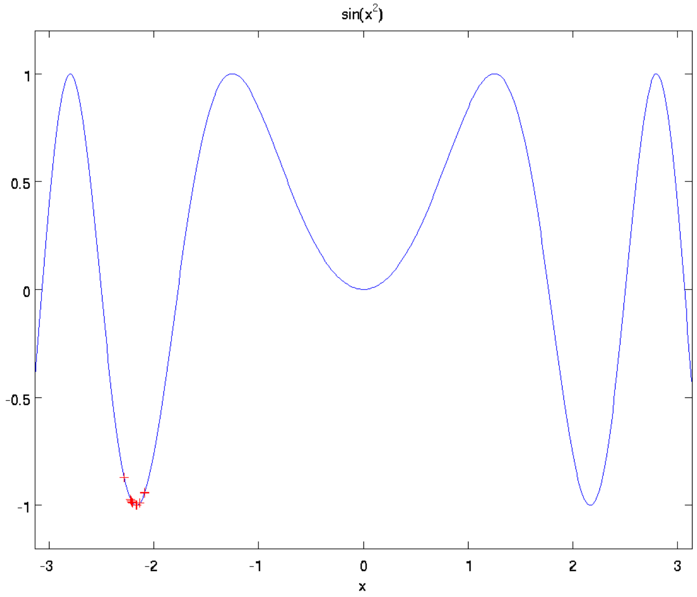
*Except for the Rook and Queen

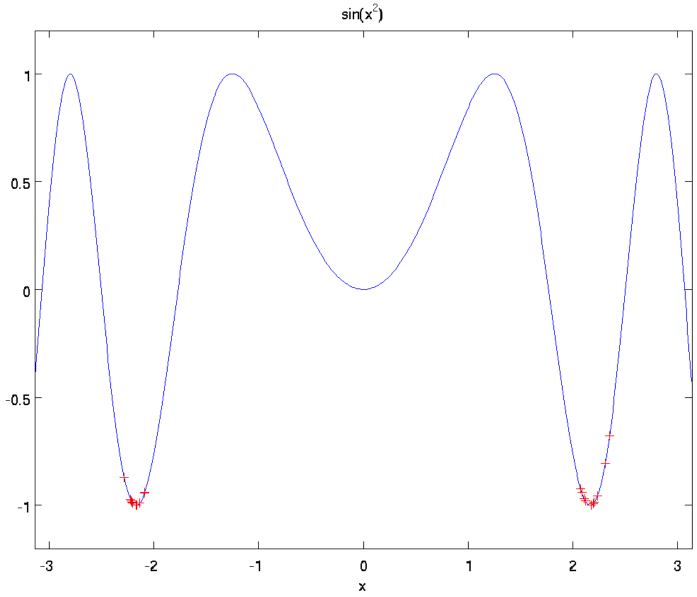
Optimization



- 20 players, 1000 generations, 2 Elites, 10 replacements
- Tournament selection with each player playing at least 3 games (+1 for win, 0 for loss and 0.5 for draw)
- One point crossover with $p(x) = 0.8$, Simple mutation with $p(x) = 0.02$ ($-100 < x < +100$)
- Gets stuck in some local optimum, no real improvement in gameplay

Niching





- A general class of techniques which promotes the formation and maintenance of stable sub-populations
- 2 main objectives:
  - Multi-modal Optimization
  - Prevention of premature convergence

Multi-Niche Crowding

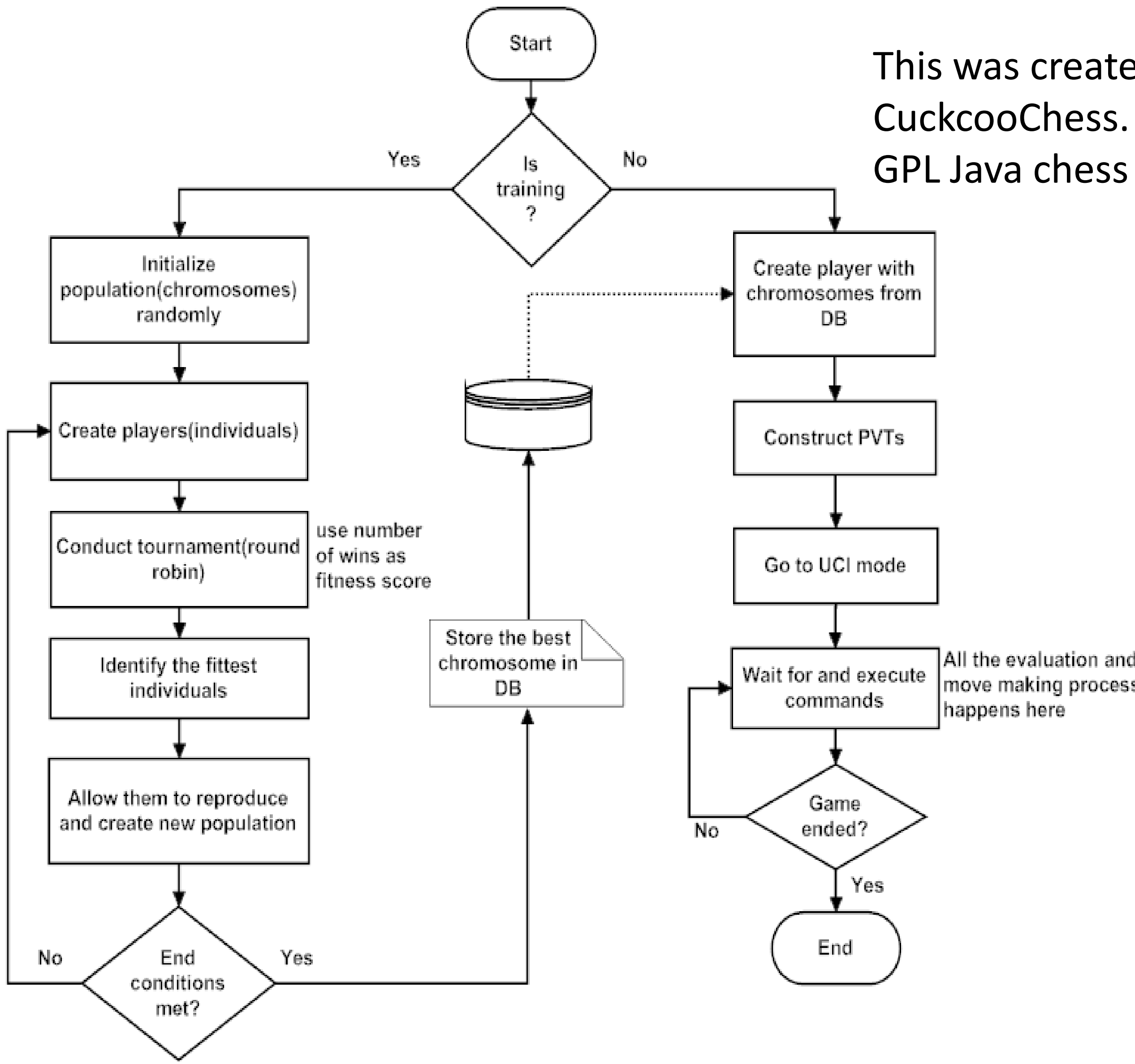
- Idea is to eliminate the selection pressure caused by FPR (Fitness Proportionate Reproduction)
- Encourage mating and replacement within members of the same niche while allowing for some competition for slots among the niches
- Maintains diversity among the Niches

Multi-Niche Crowding

How it works

- Select an individual 'i' randomly
- Among c_s randomly picked individuals, select a mate 'm' which is most similar to 'i'
- Pick c_f groups of 's' individuals from all the niches randomly
- From each of these c_f groups, pick the individuals which are most similar to 'i'
- From these c_f individuals, pick the one with the lowest fitness value for replacement

Overview



This was created from CuckcooChess. A GNU-GPL Java chess engine

Change Log

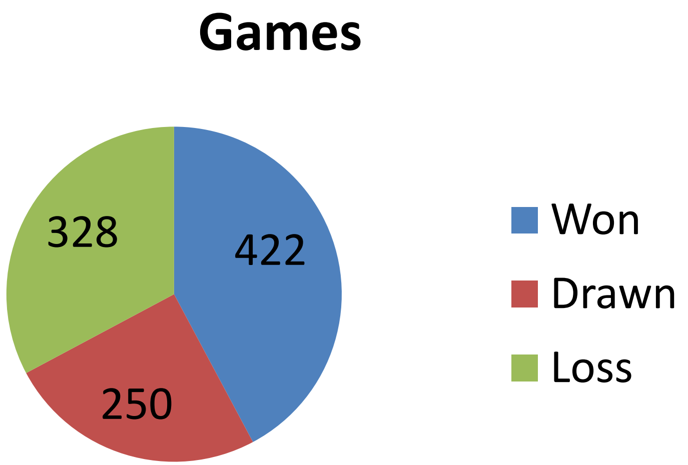
- Replacing the entire evaluation module with our new module
- Adding a genetic training system with a tournament selector
- Adding PVTs which is the core of the new evaluation function
- Modifying search behavior to take advantage of the new PVTs
- A tournament simulator which is used for both fitness evaluation and testing

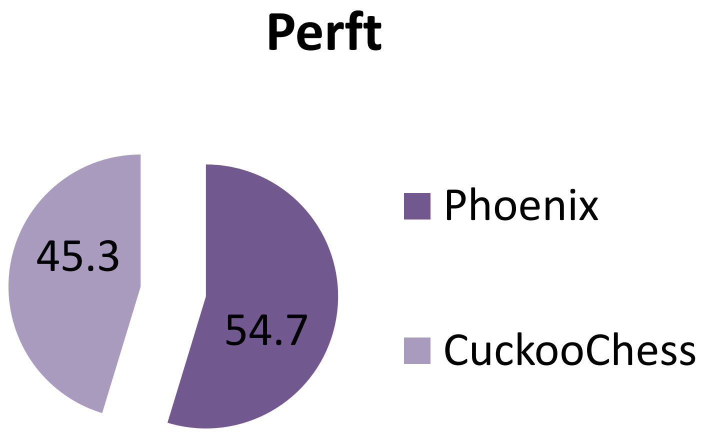
Result & Conclusion

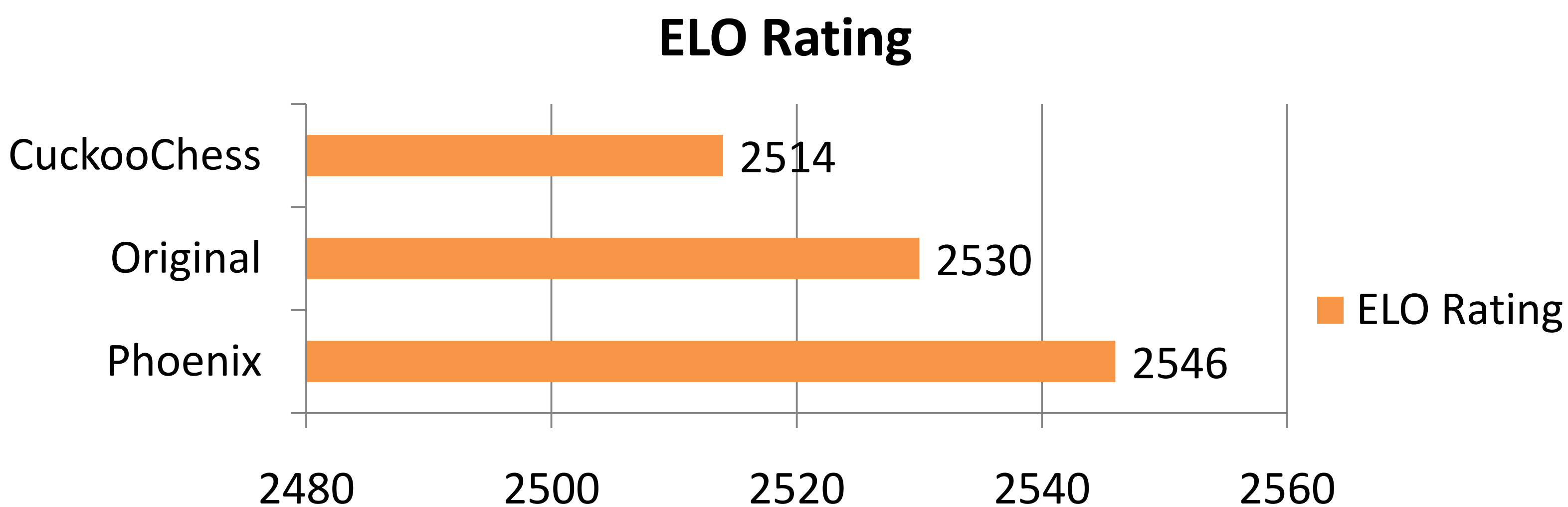
Testing

- Not easy to test chess engines
- Best way to test is by playing tournaments
- UCI is a communication protocol facilitating gameplay between engines
- A tournament of 1000 games were played against CuckooChess
- Time control: 3 seconds/move
- Ratings computed using EloStat

Performance







Conclusion

- A new perspective for designing complex self-learning systems
- Intention was to find a way to reduce abstract concepts into a group numbers which can be understood by the computer
- Not a world class engine. Attains IM level
- Neural/Deep networks not suited for chess
- Research in Multi-modal optimization is needed
- Can be improved in the following ways:
  - No control on how Mutation takes place
  - Still depends on search algorithms
  - Use past experience to prune the search tree quicker

Useful Links

- Source code: http://goo.gl/n0yVej
- Original engine: http://goo.gl/LUaibx
- Misc tools developed: http://goo.gl/MlJPyg
- Tournament PGN: http://goo.gl/3u1z87
- Thesis report: http://goo.gl/hiQHSr