Alpha Go
and
Reinforcement Learning

Vincent Lepetit and Pascal Fua

ParisTech   EPFL
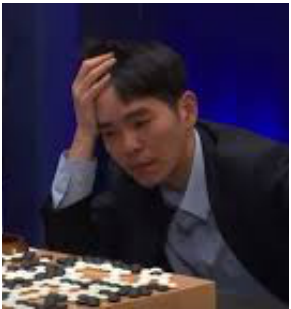
Historical Perspective

1997: Chess program Deep Blue defeats Garry Kasparov.

Until 2015: Go programs still play at amateur level.

October 2015: AlphaGo defeats Fan Hui, European Go champion, 5-0. First time a program has beaten a professional player.

March 2016: AlphaGo Lee defeats Lee Sedol, one of the best world players, 4–1.

2017: Publication of AlphaGo Zero. 100-0 victory against AlphaGo after 3 days of training.

2018: Publication of AlphaZero. Same algorithm learns to play Go, Chess, and Shogi. Defeats all previous programs at Chess and Shogi.

2018: AlphaStar, first Artificial Intelligence to defeat a top professional player at StarCraft II.

Artificial Intelligence

Expert Systems   A*

                                        min-max

Machine Learning

        Support Vector Machines

    Boosting

                    Random Forests

Deep Learning

    Lenet   VGG

            ResNet

Artificial Intelligence:
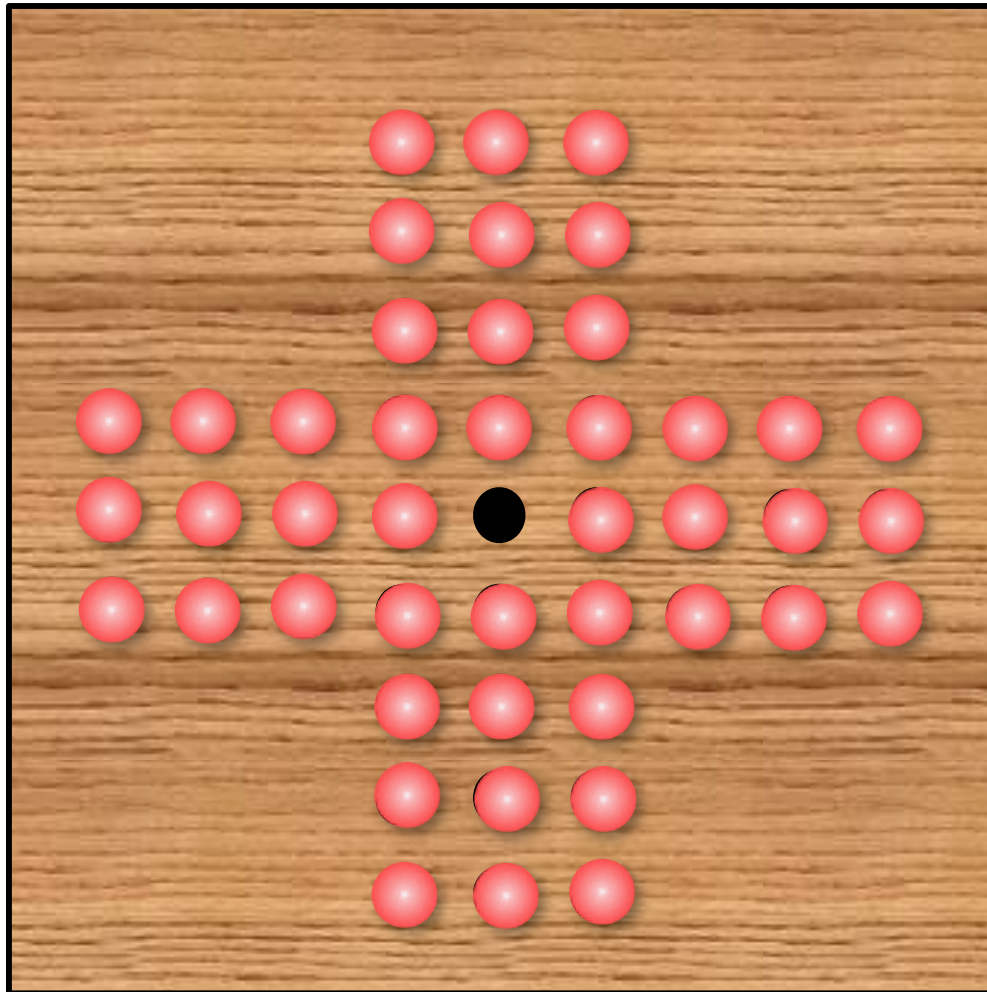   Algorithms that search the space of possible solutions.

Machine Learning:
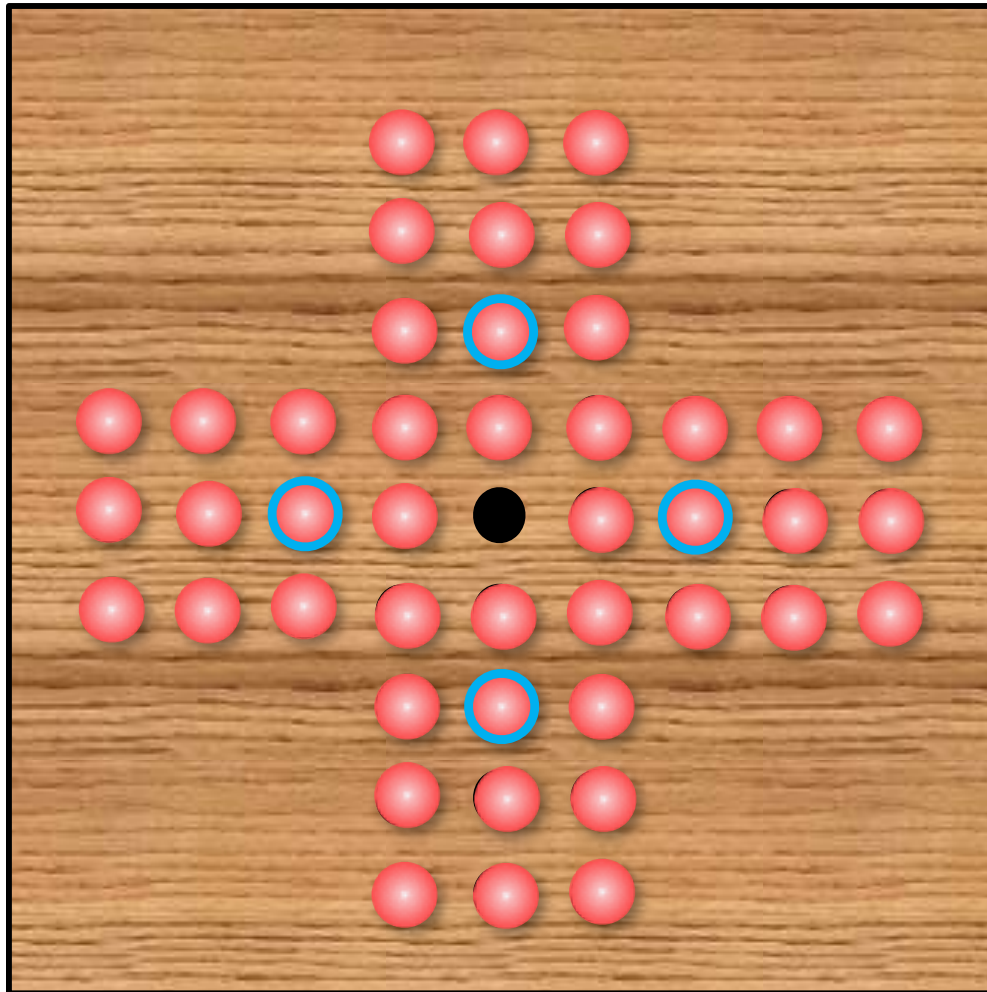   Algorithms whose performance can be improved using training data.
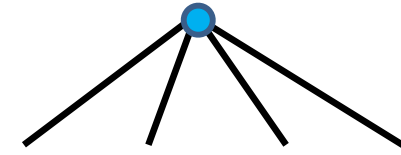
Deep Learning:
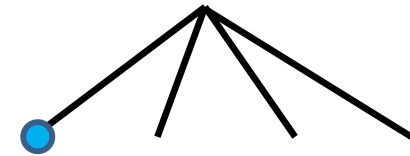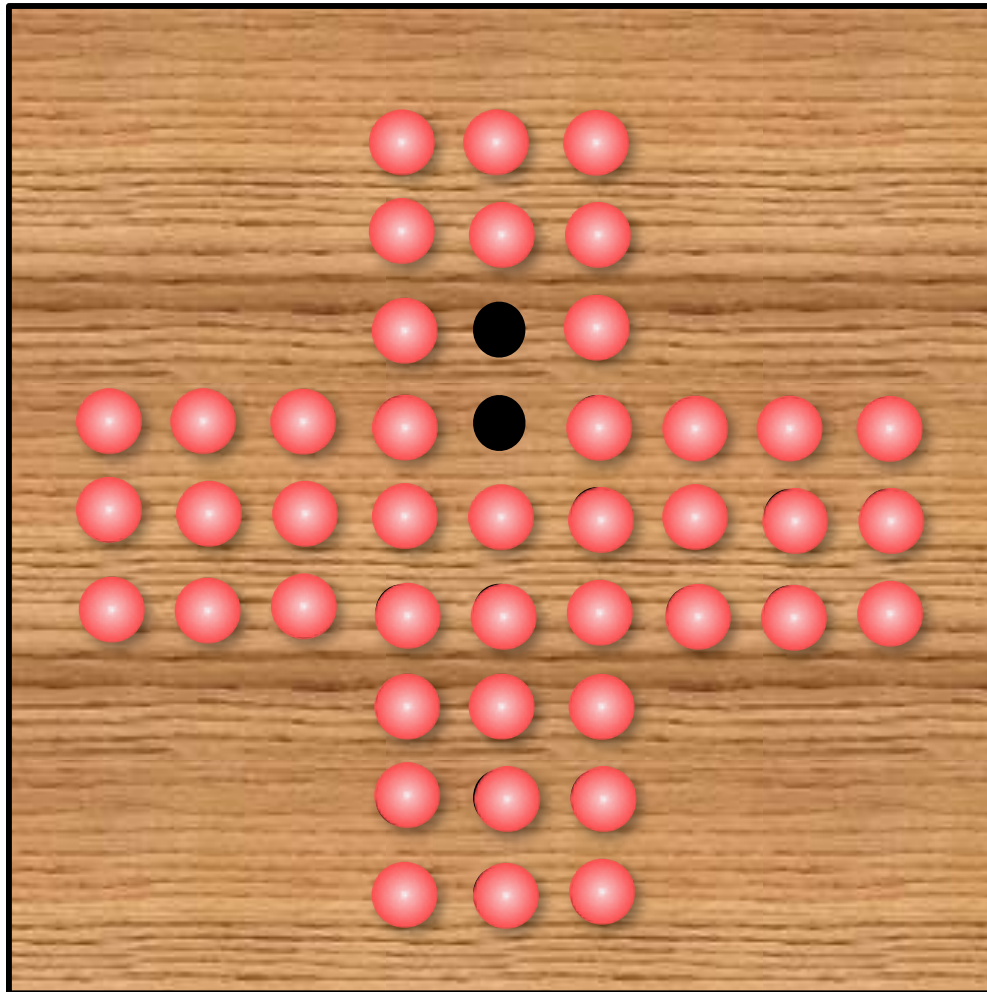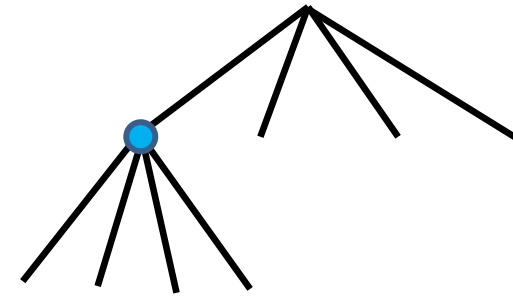   Algorithms whose performance can be improved using a lot of training data.
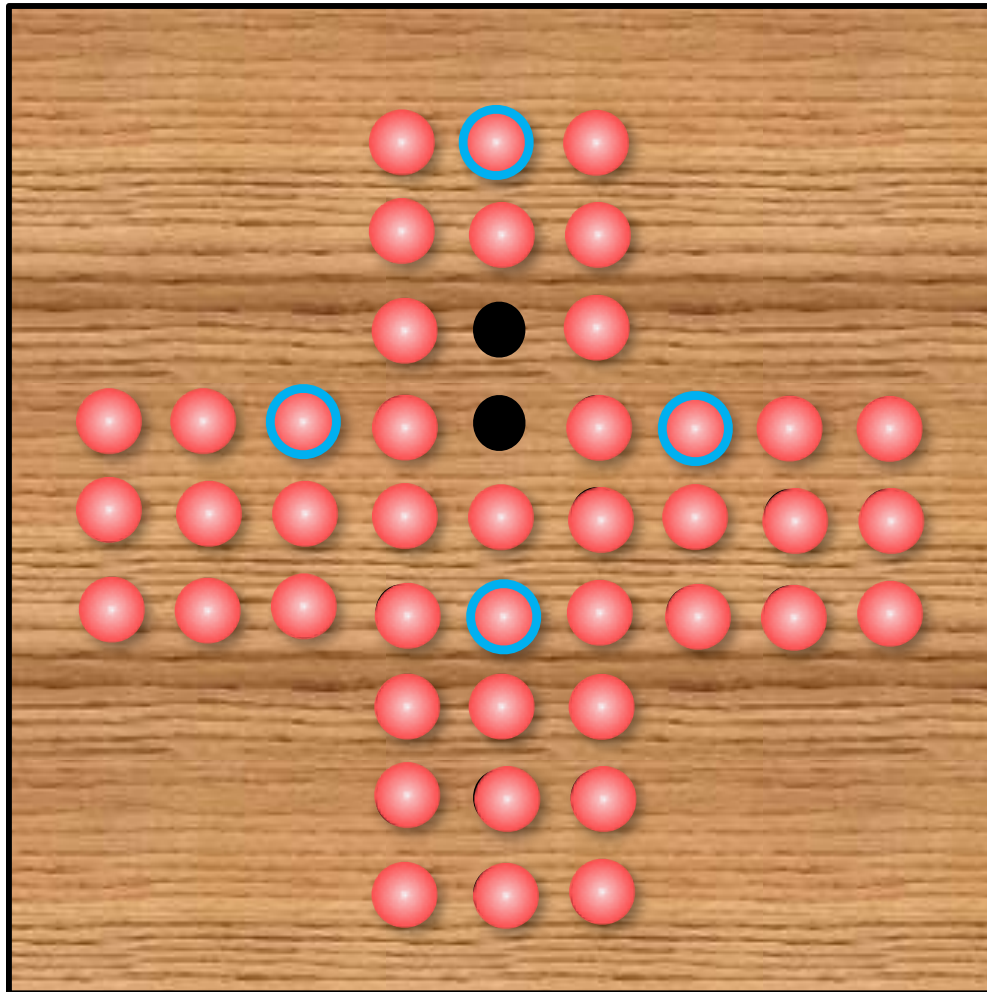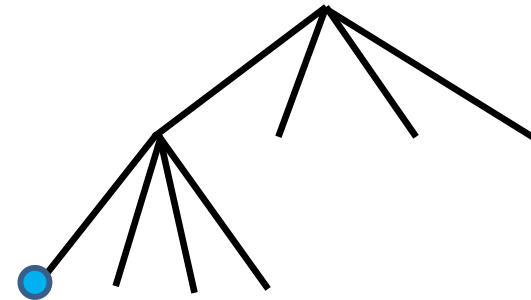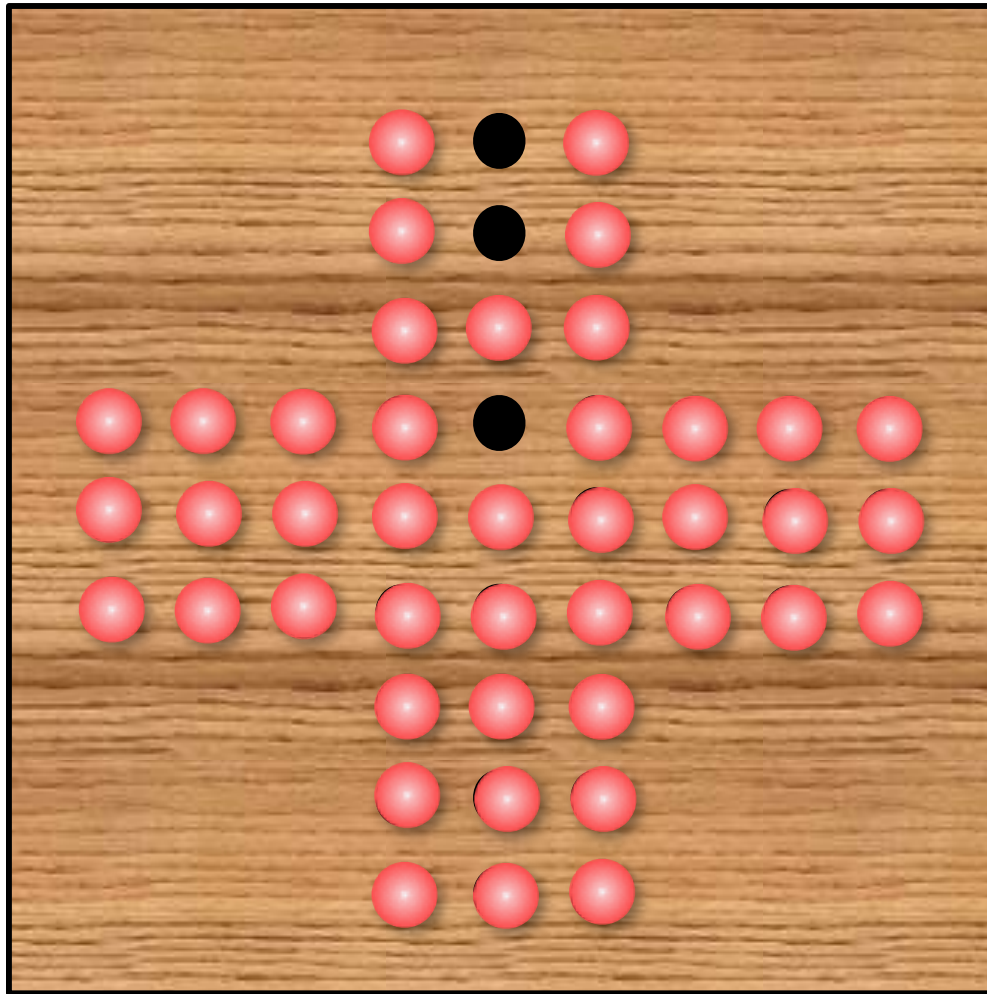
AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

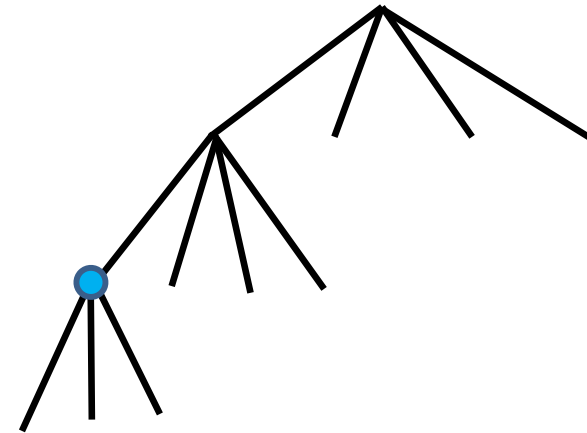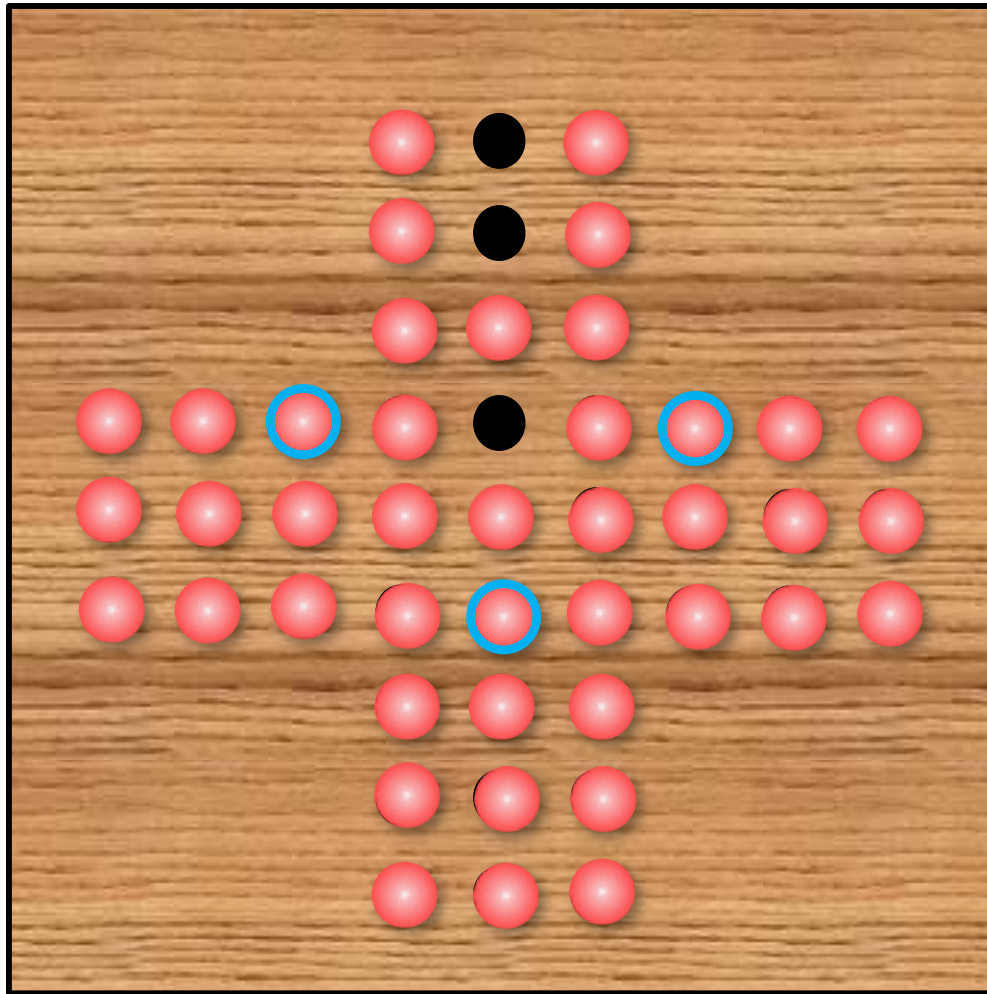# AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

AI Formulation of Peg Solitaire

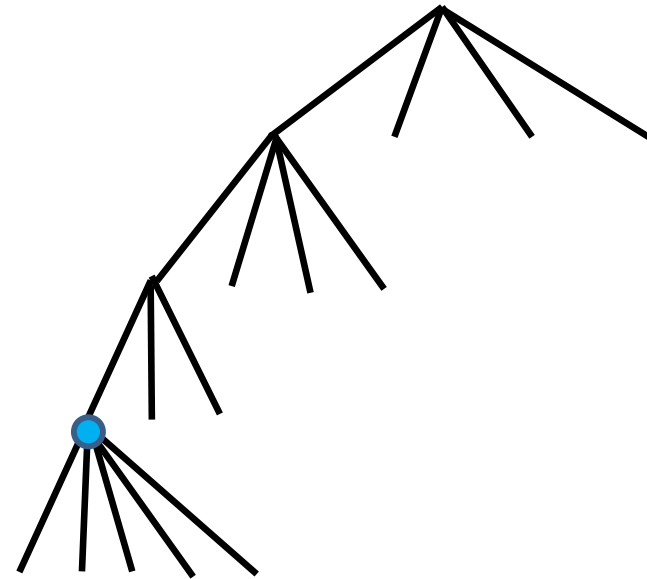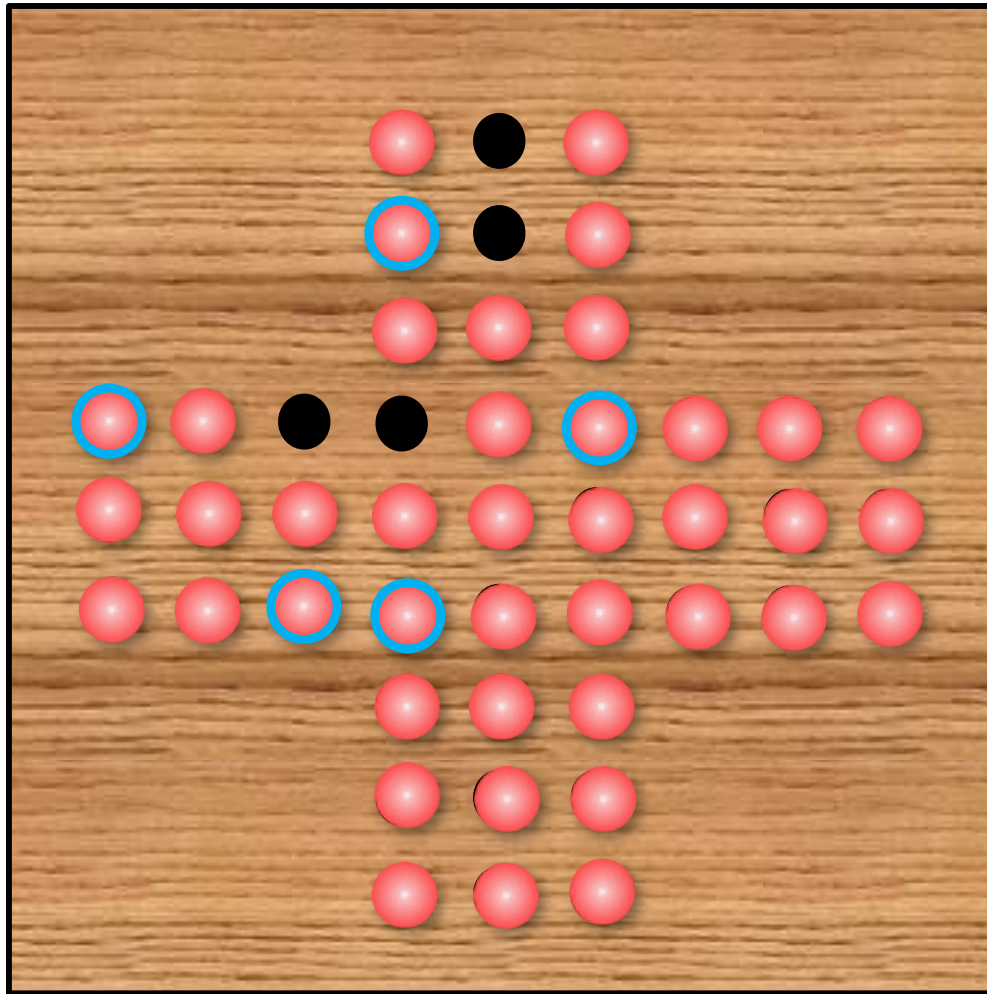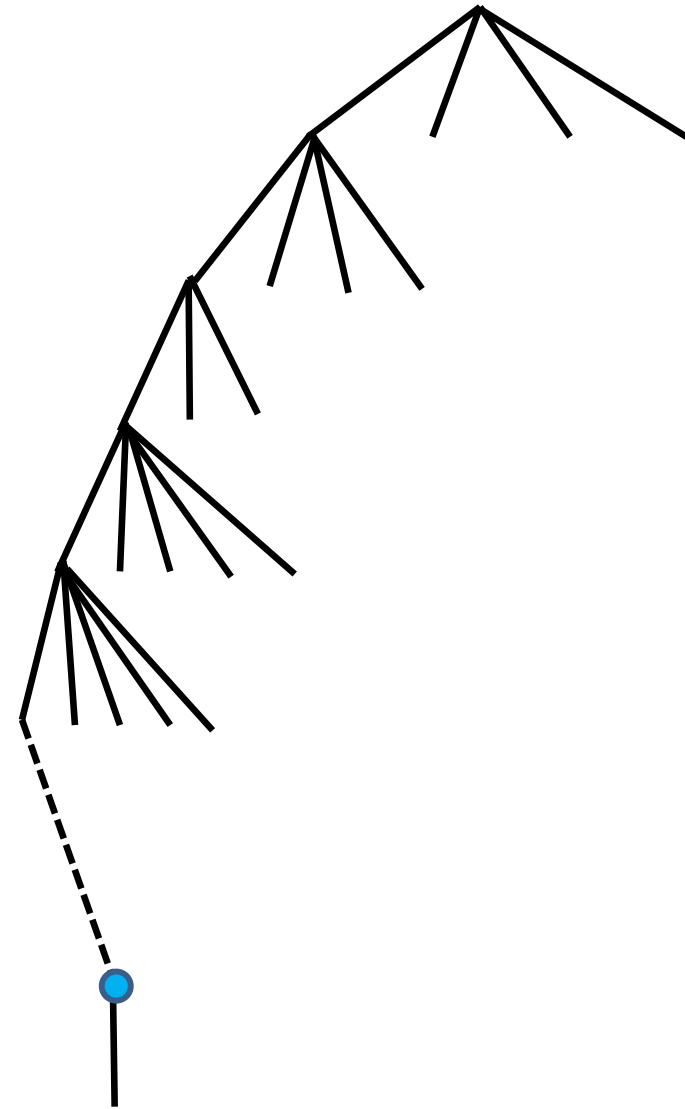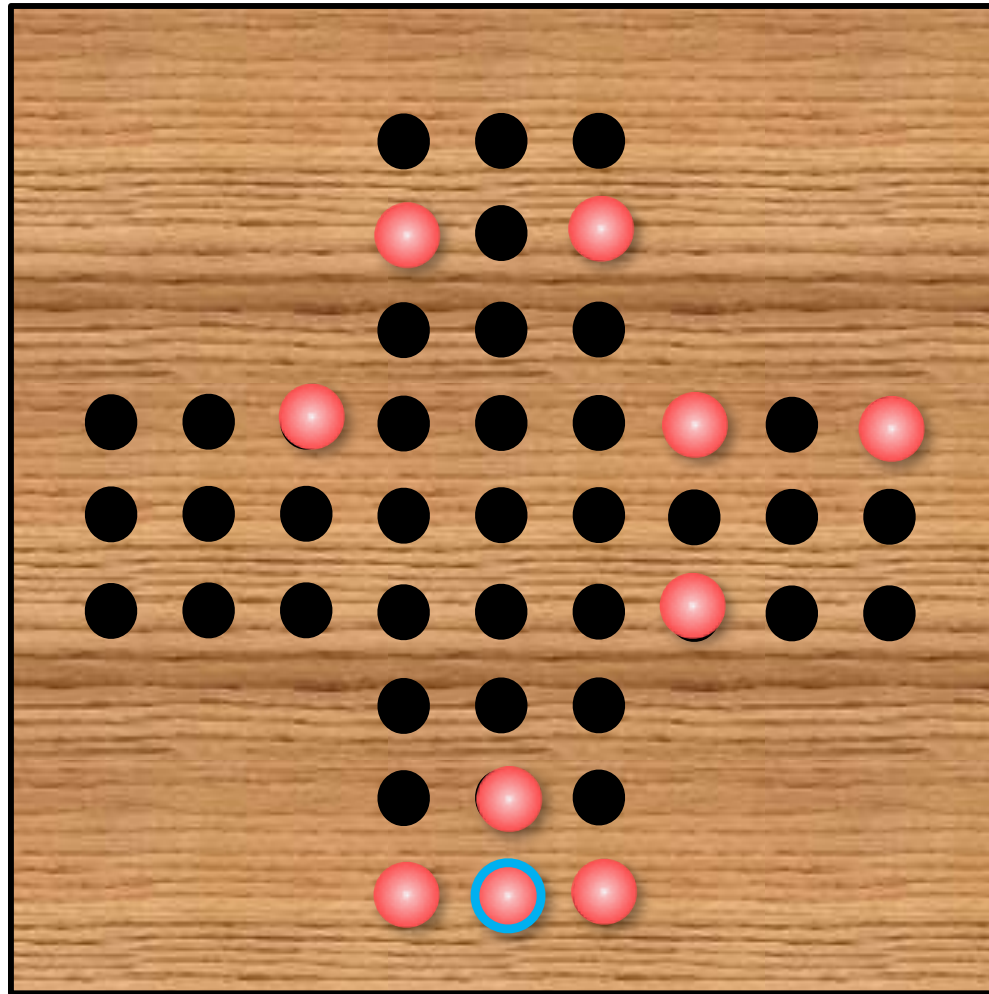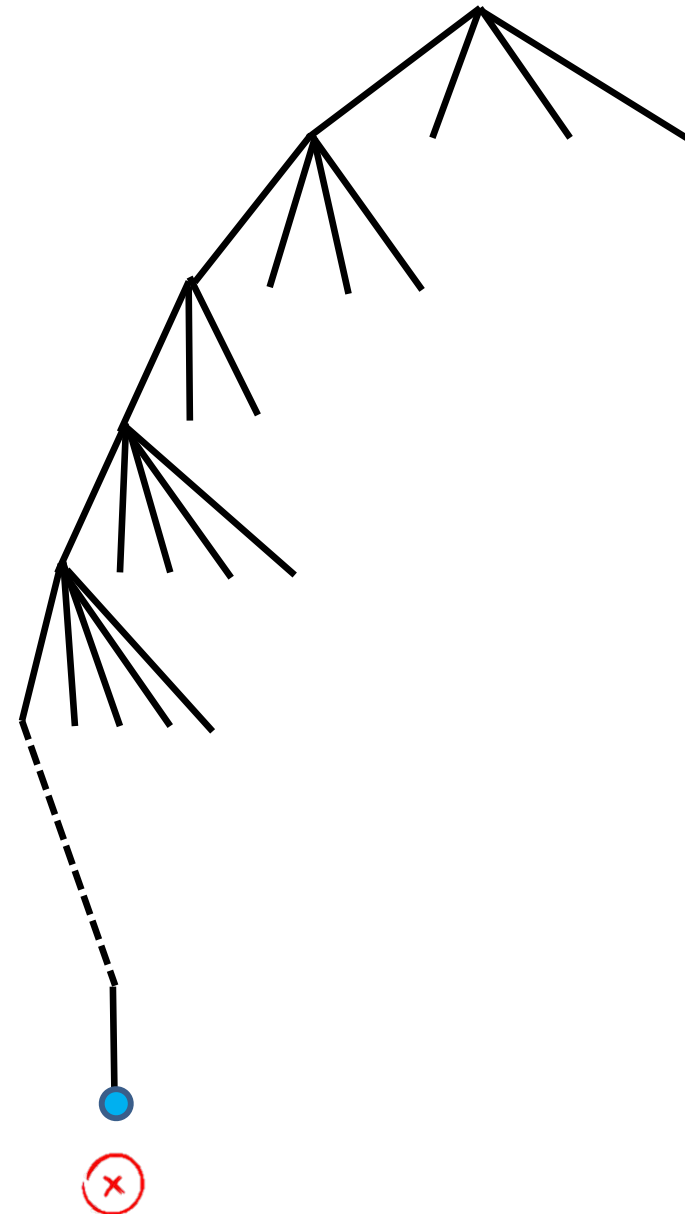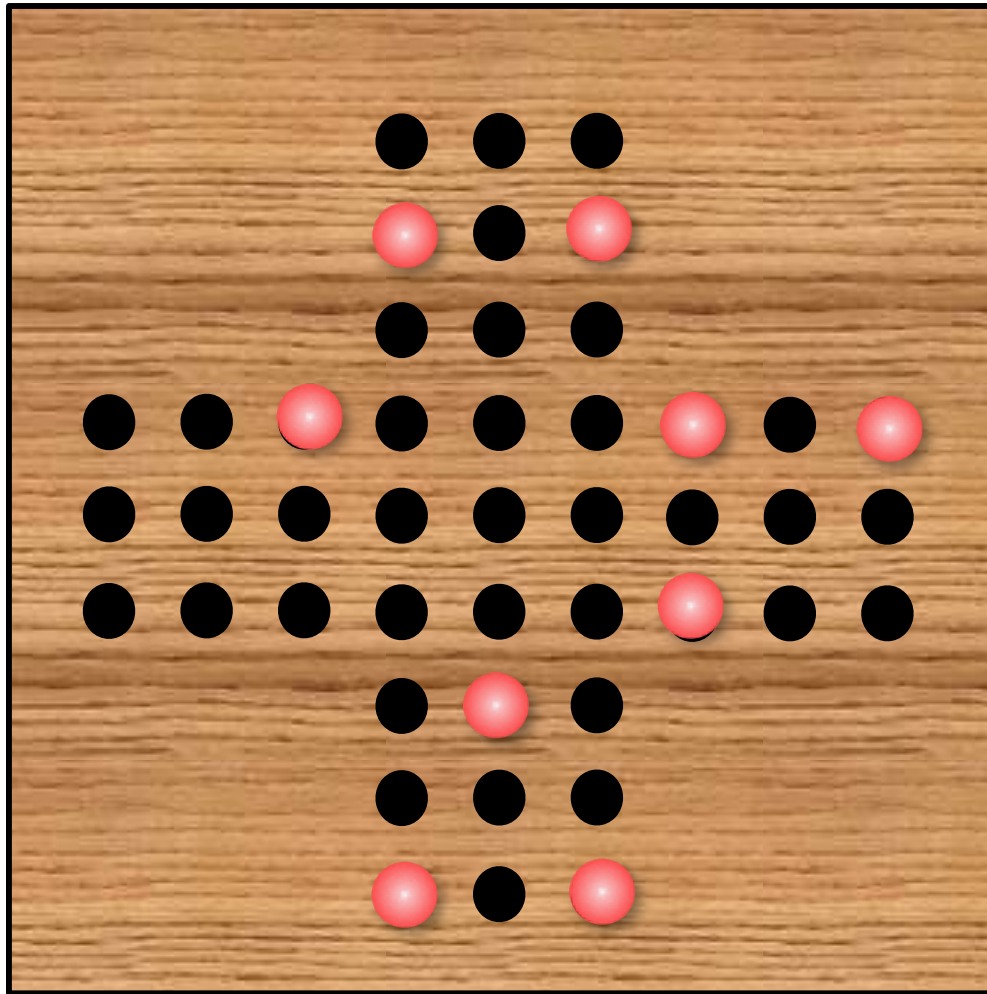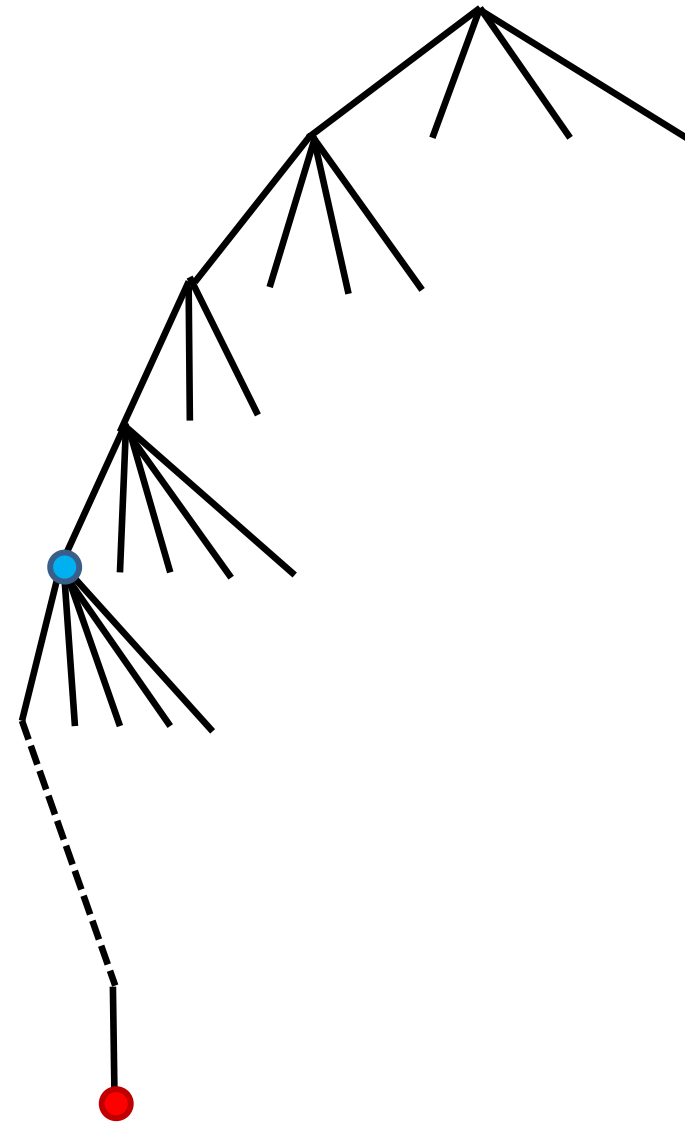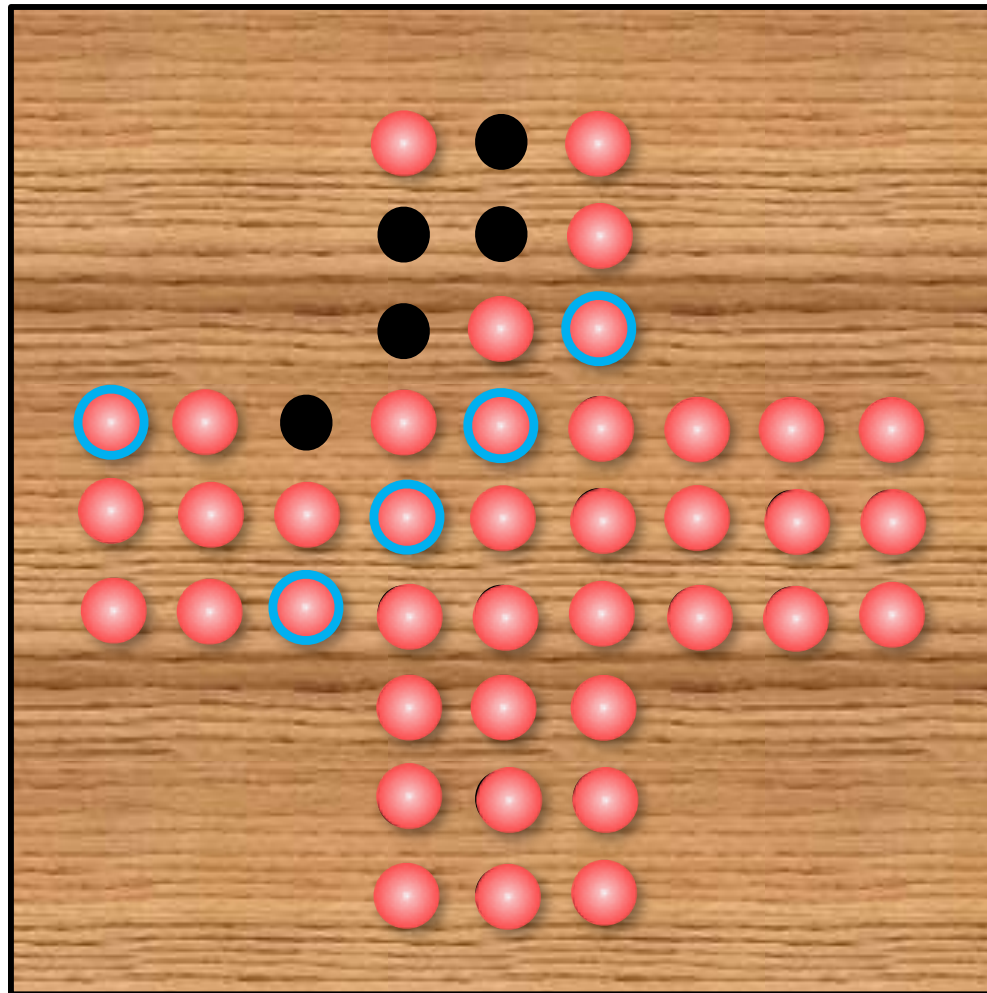# AI Formulation of Peg Solitaire
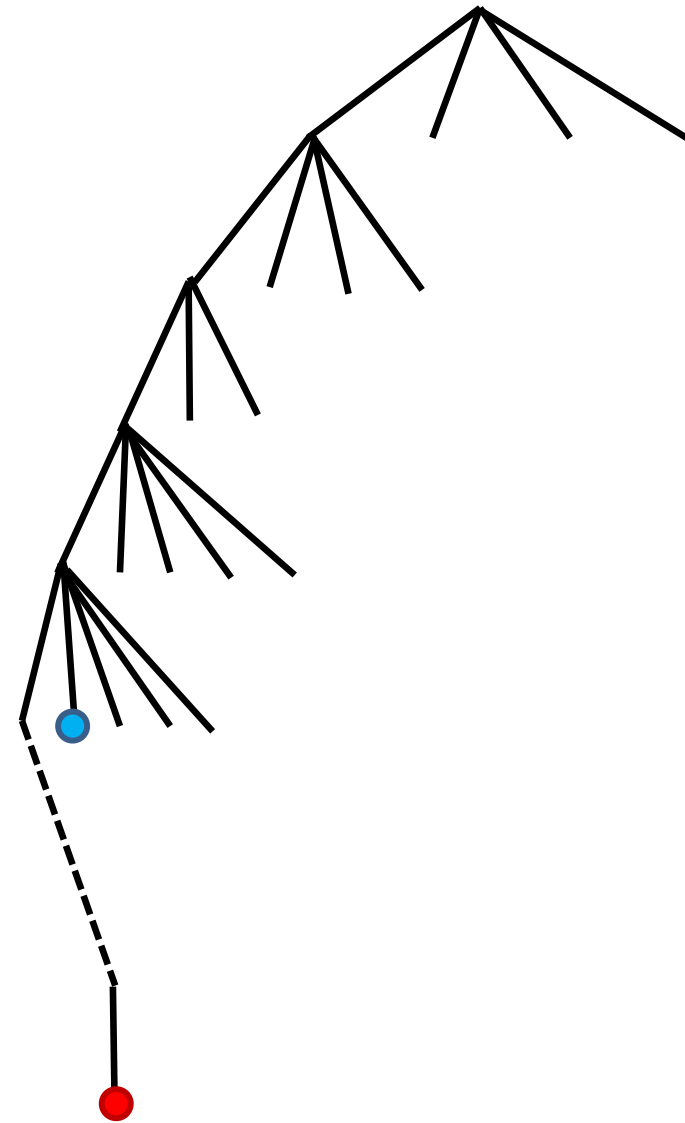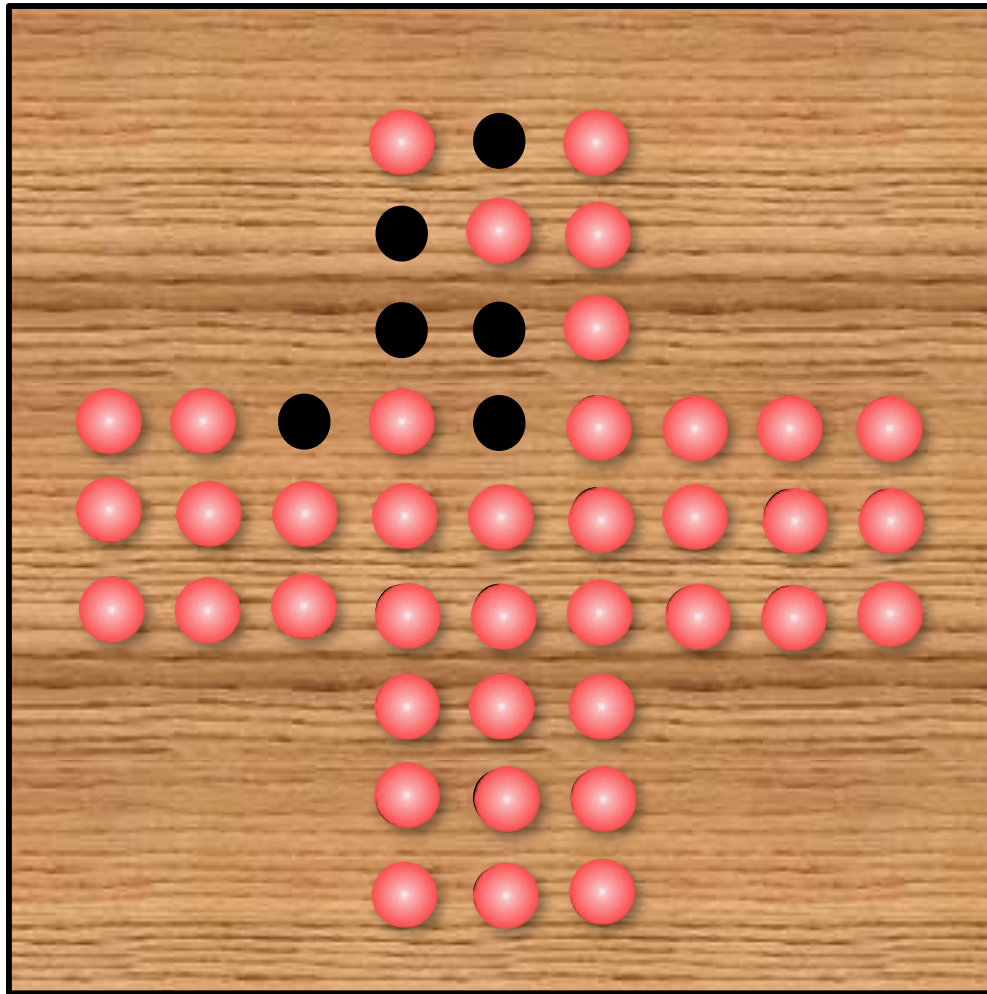
AI Formulation of Peg Solitaire
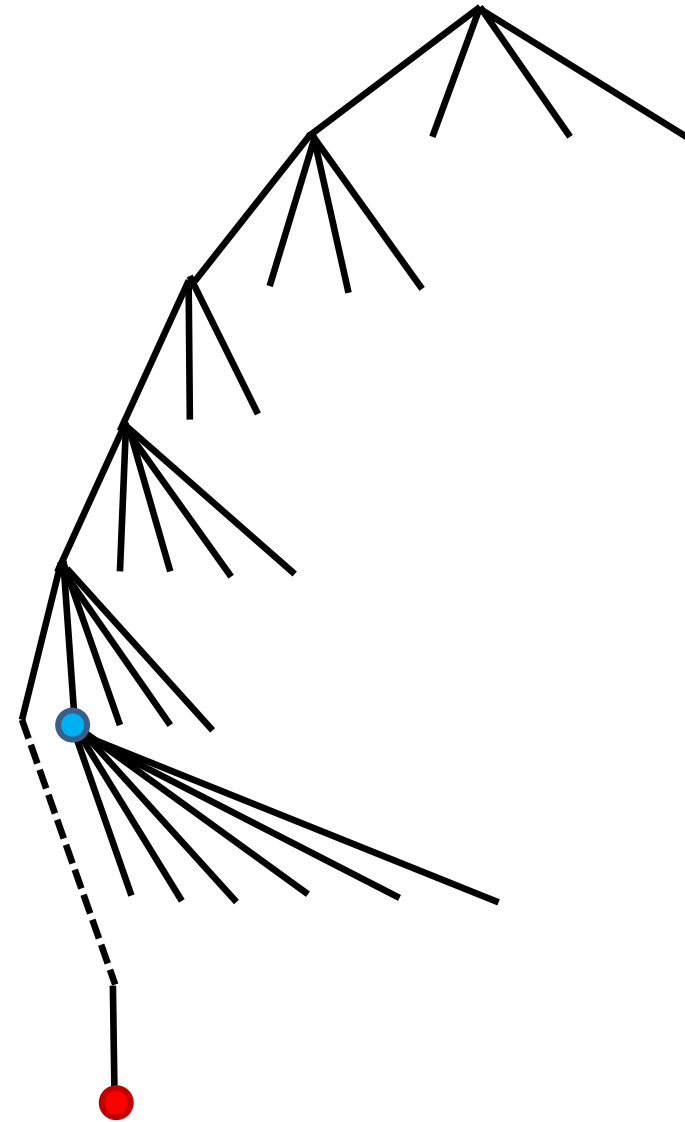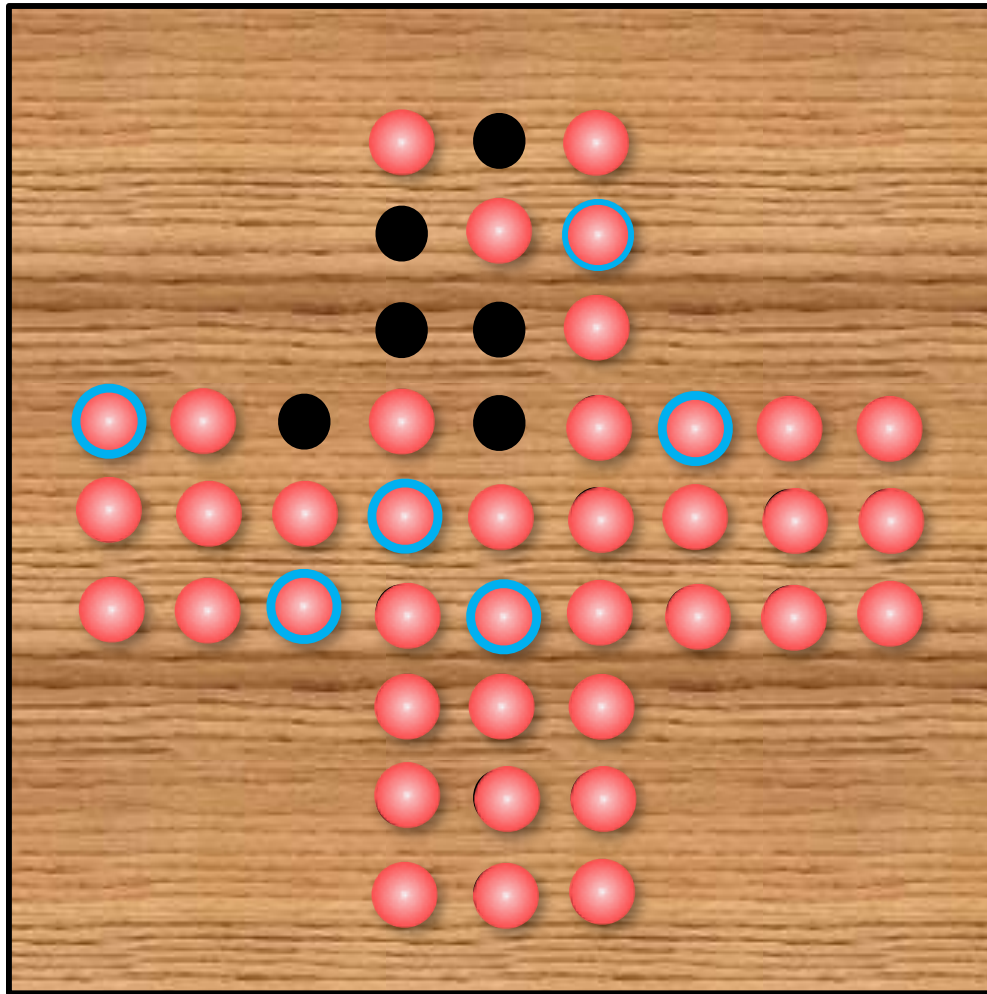
AI Formulation of Peg Solitaire
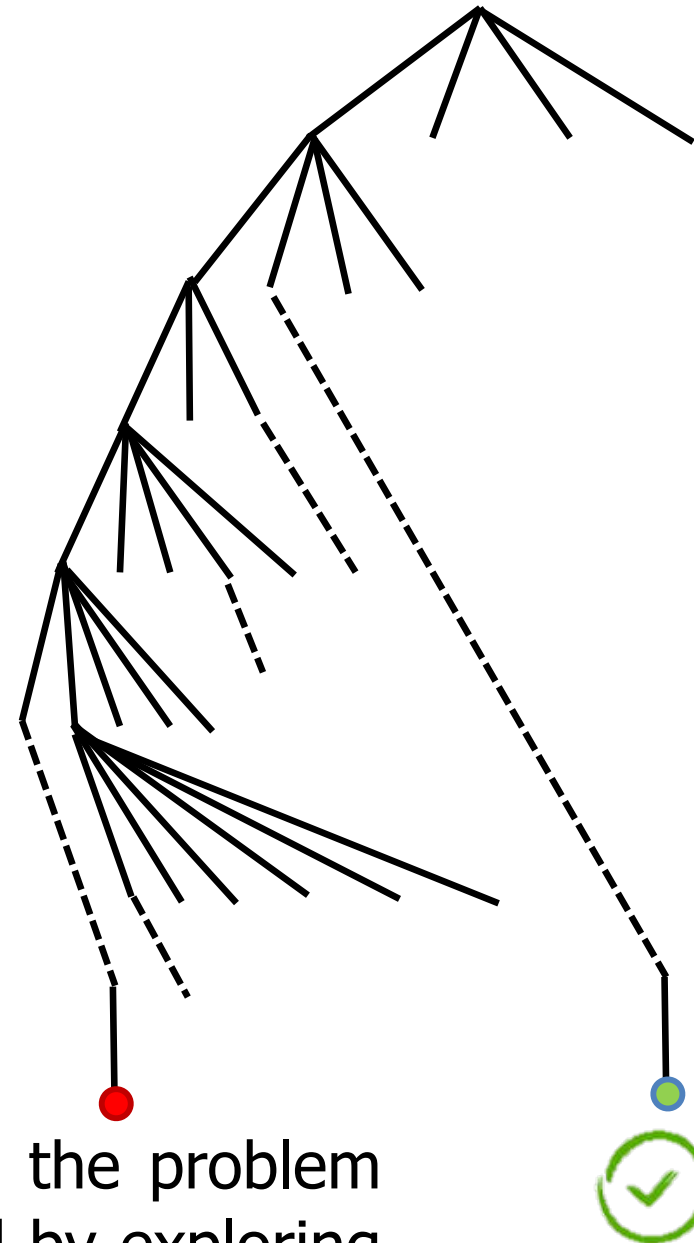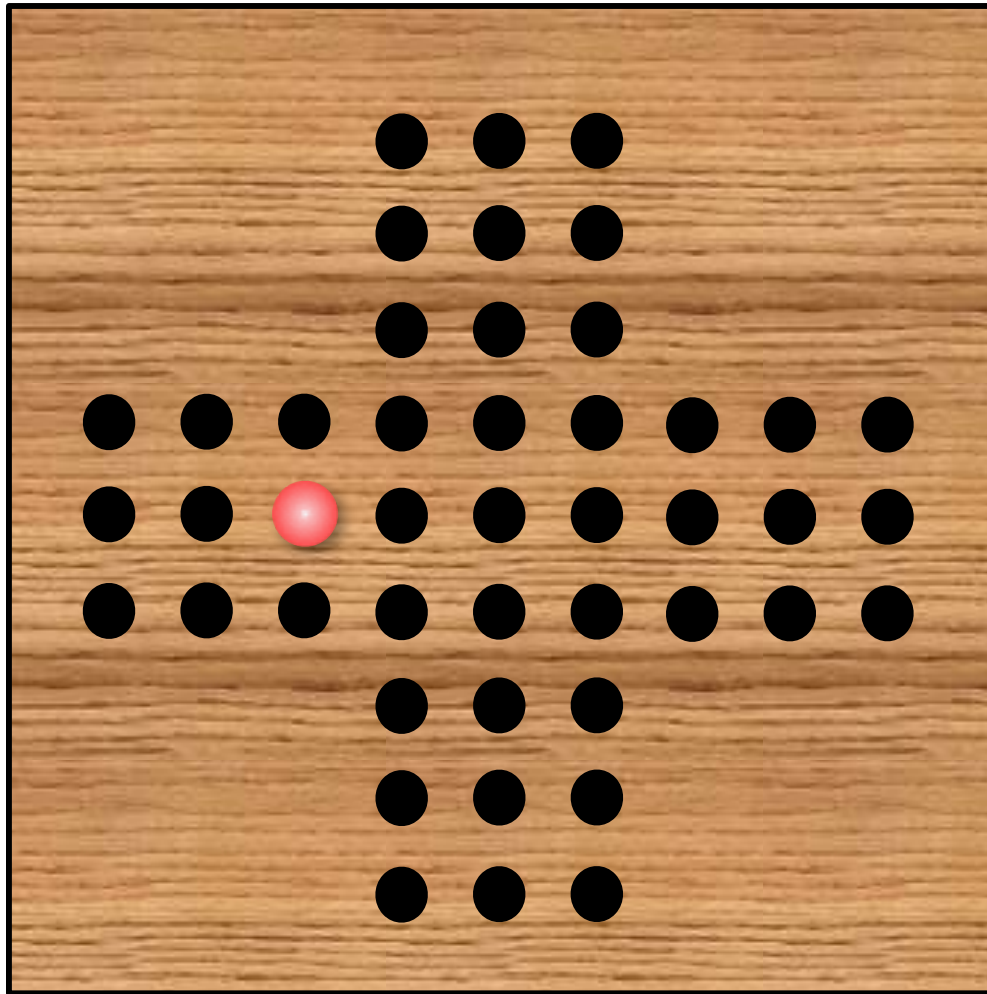
—>In theory, the problem can be solved by exploring a large tree of possibilities.

Chess

Chess

(20 possibilities)

…

Chess

(20 possibilities)

…

Chess

(20 possibilities)

(oracle)

Chess

(20 possibilities)

…

(oracle)

(28 possibilities)

…

École des Ponts
ParisTech

EPFL

Chess

(20 possibilities)

... (oracle)
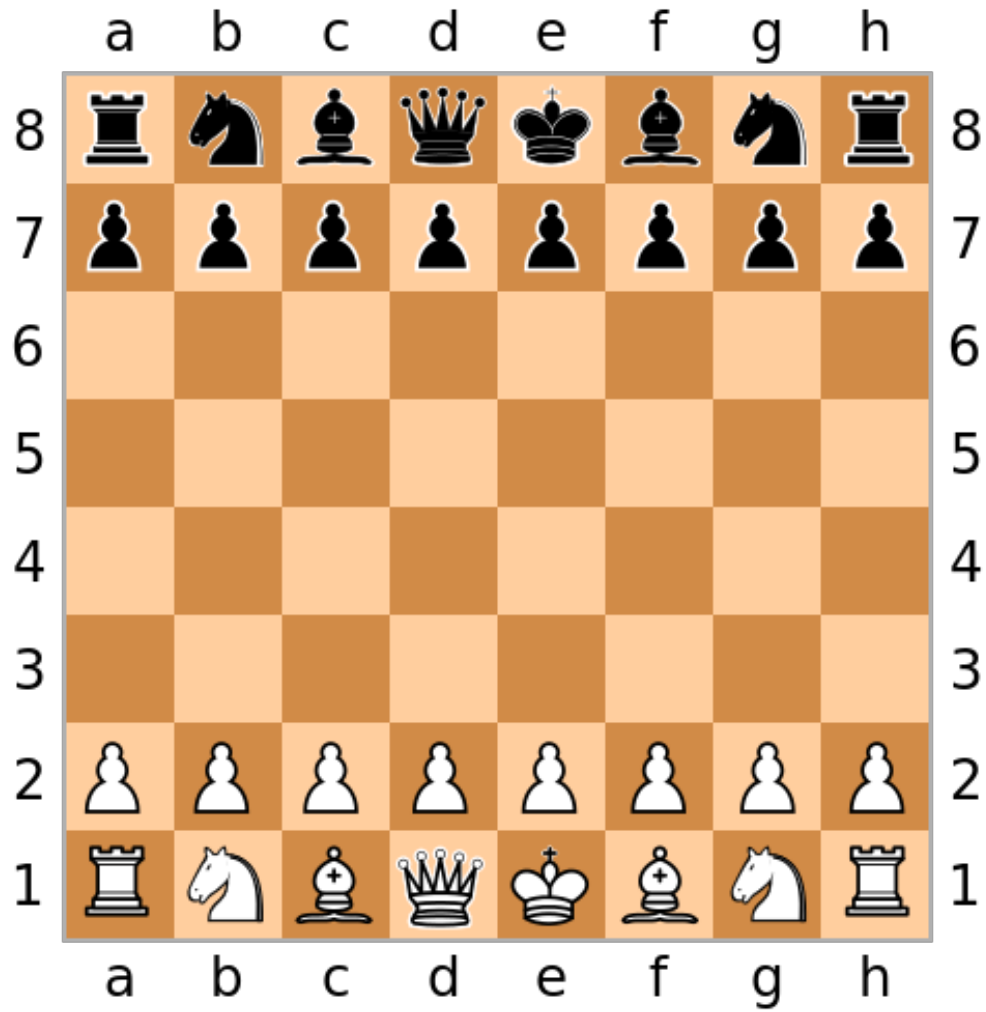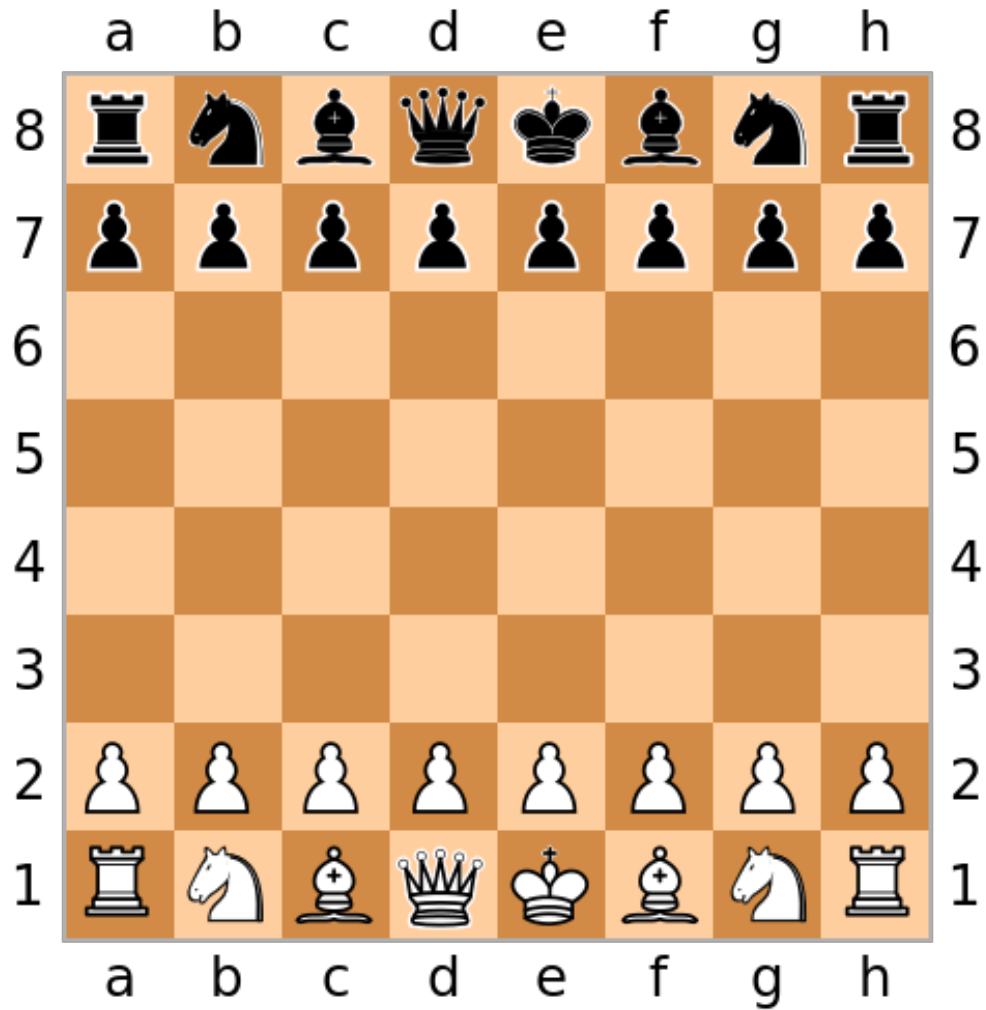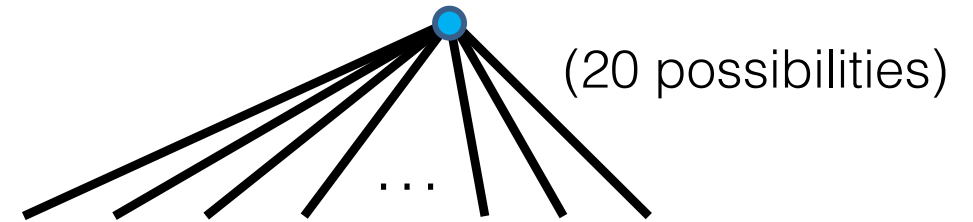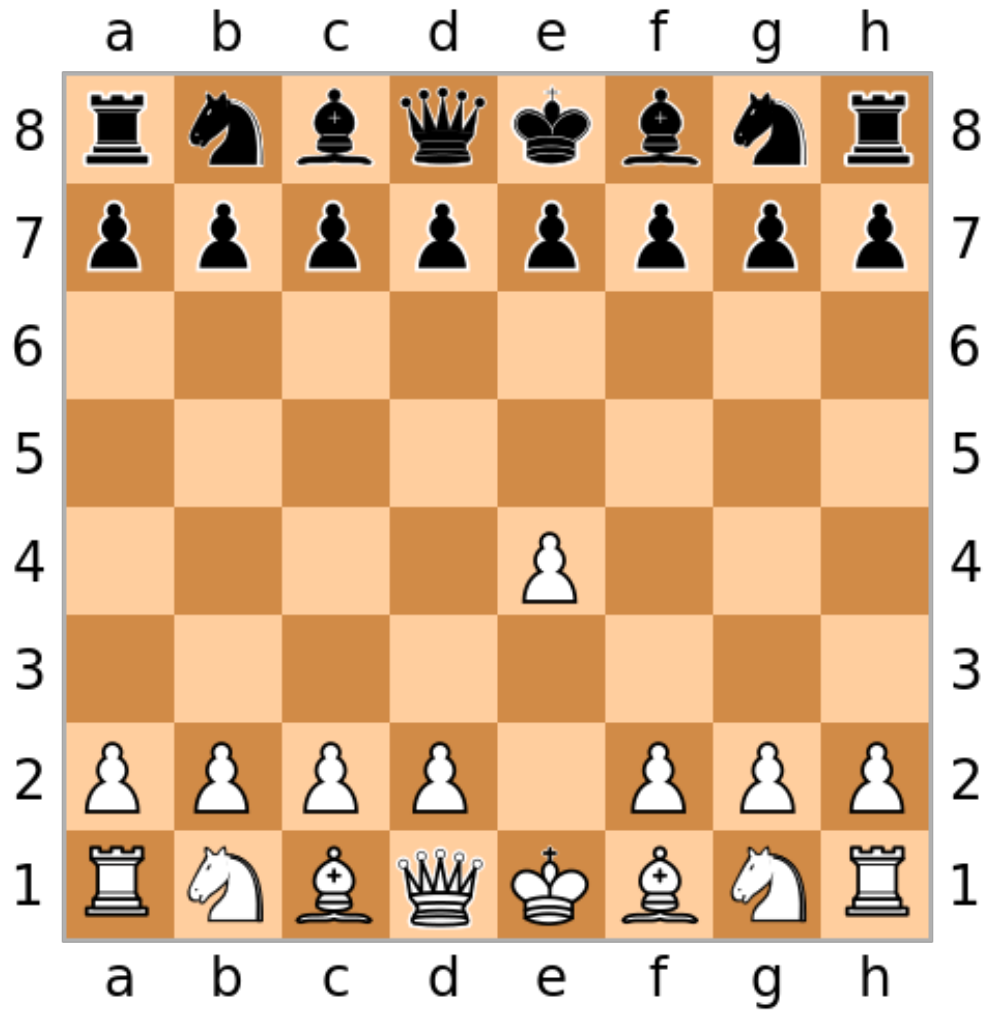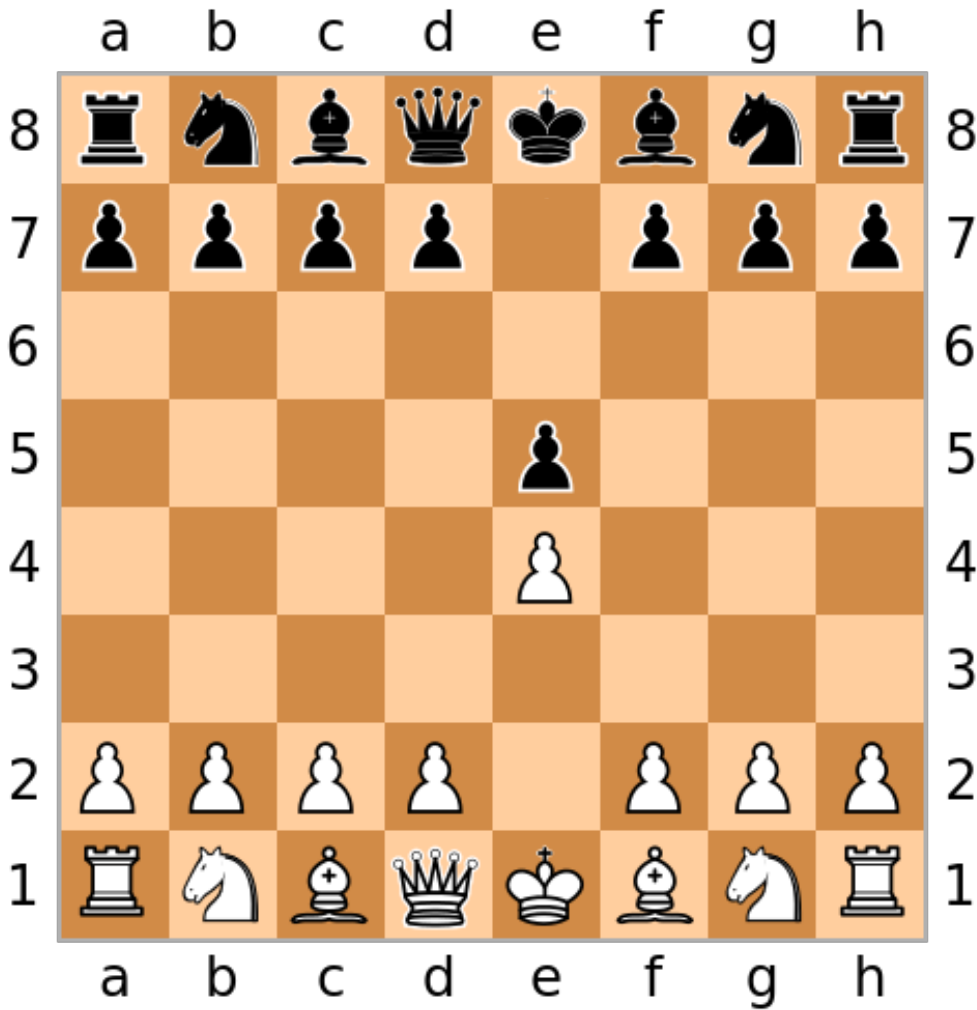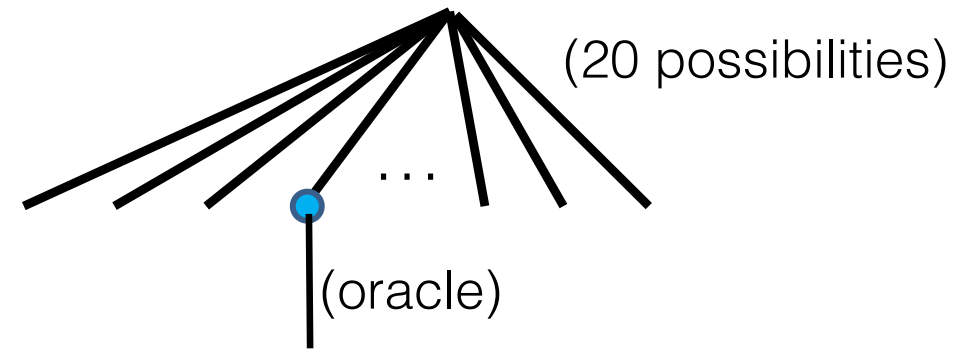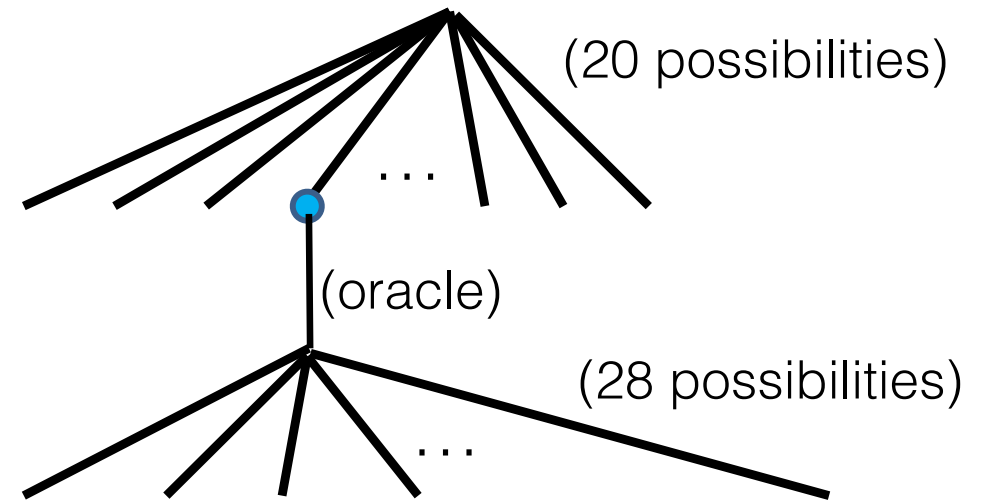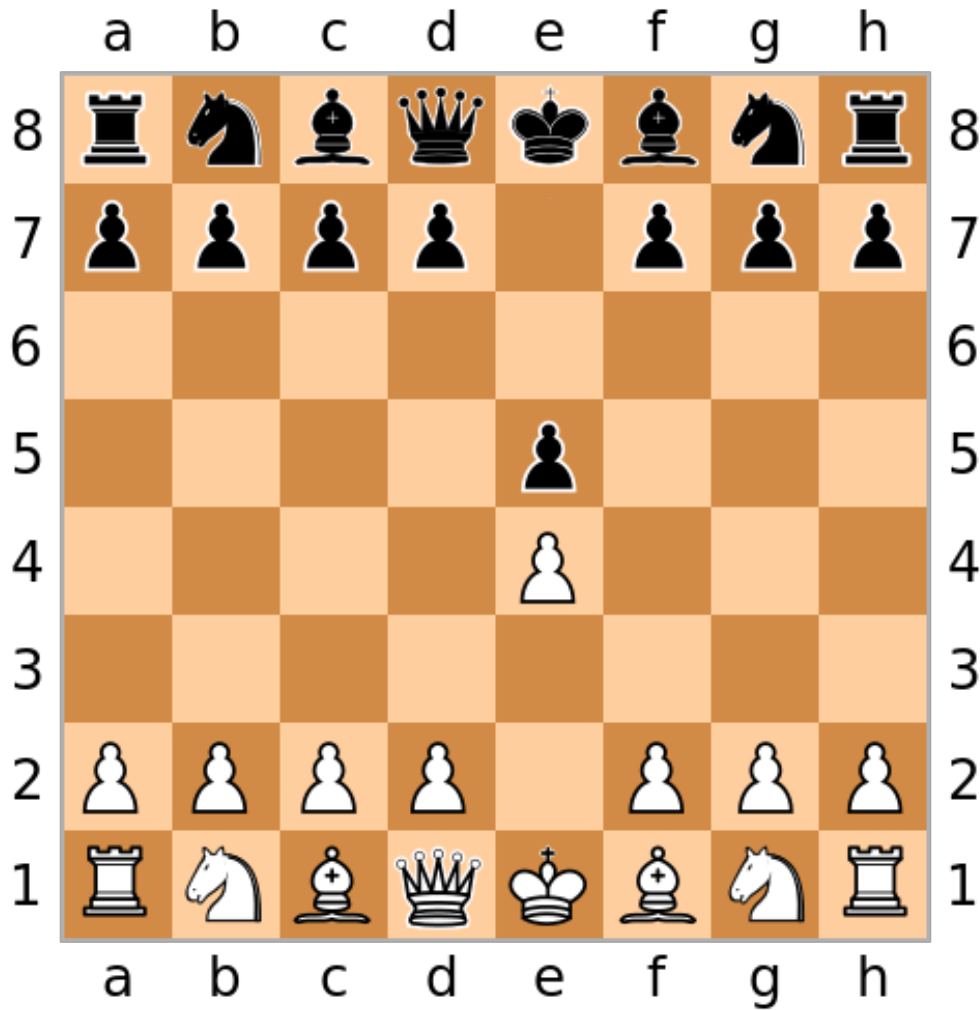
(28 possibilities)

...
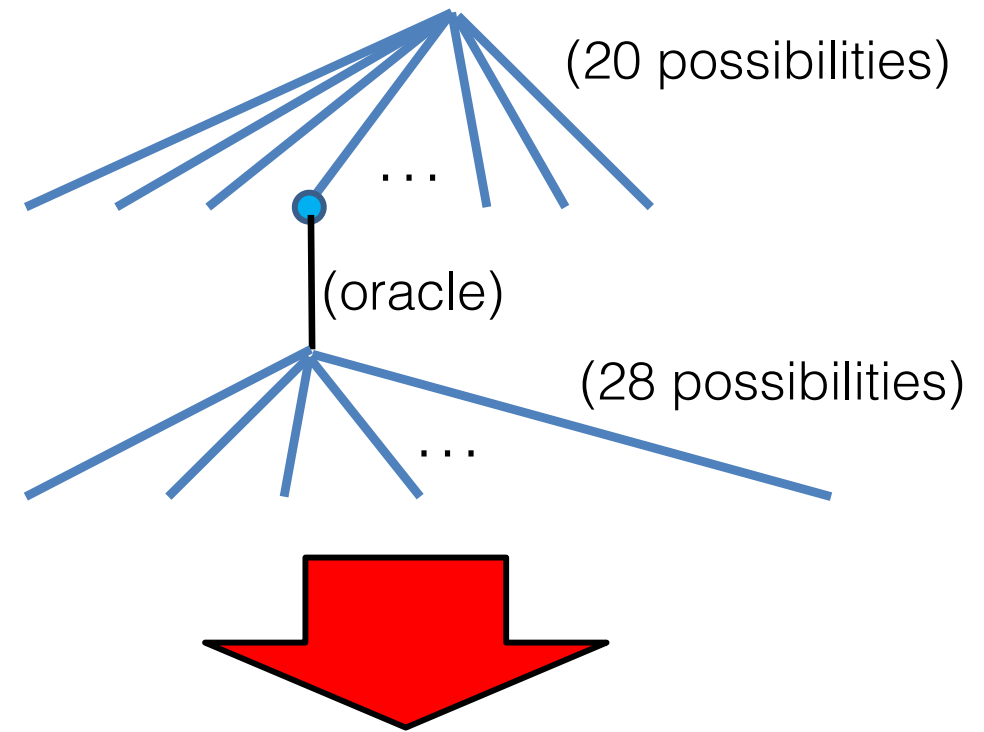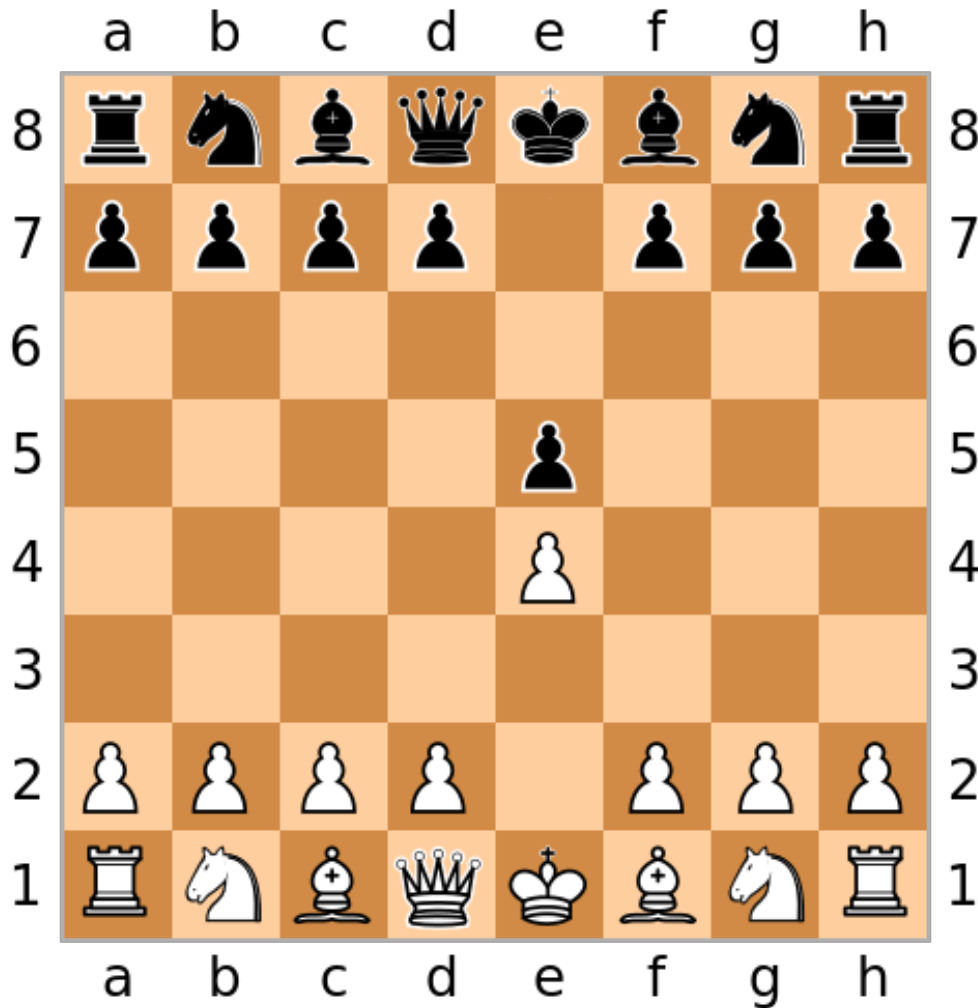
Exploring the whole tree game until we find a winning path is not doable: There are more than 10^{120} possible games!

Chess

(20 possibilities)

(oracle)

...
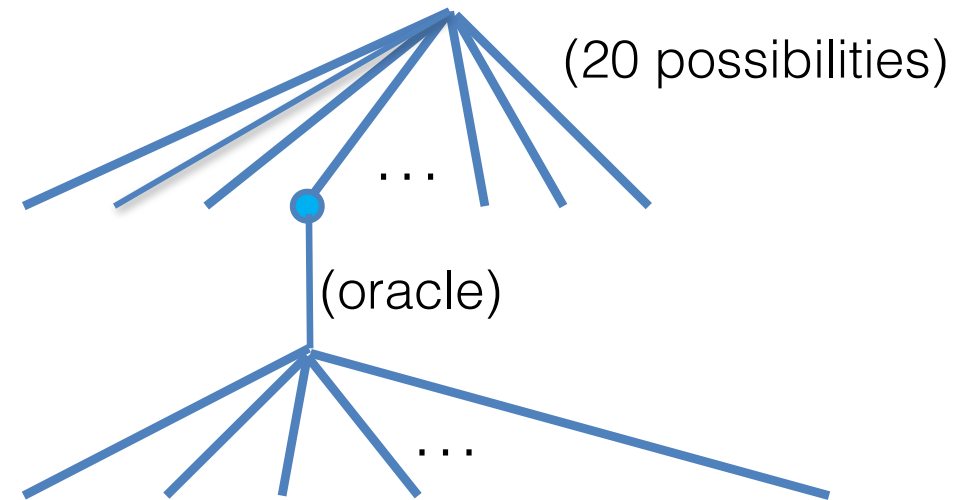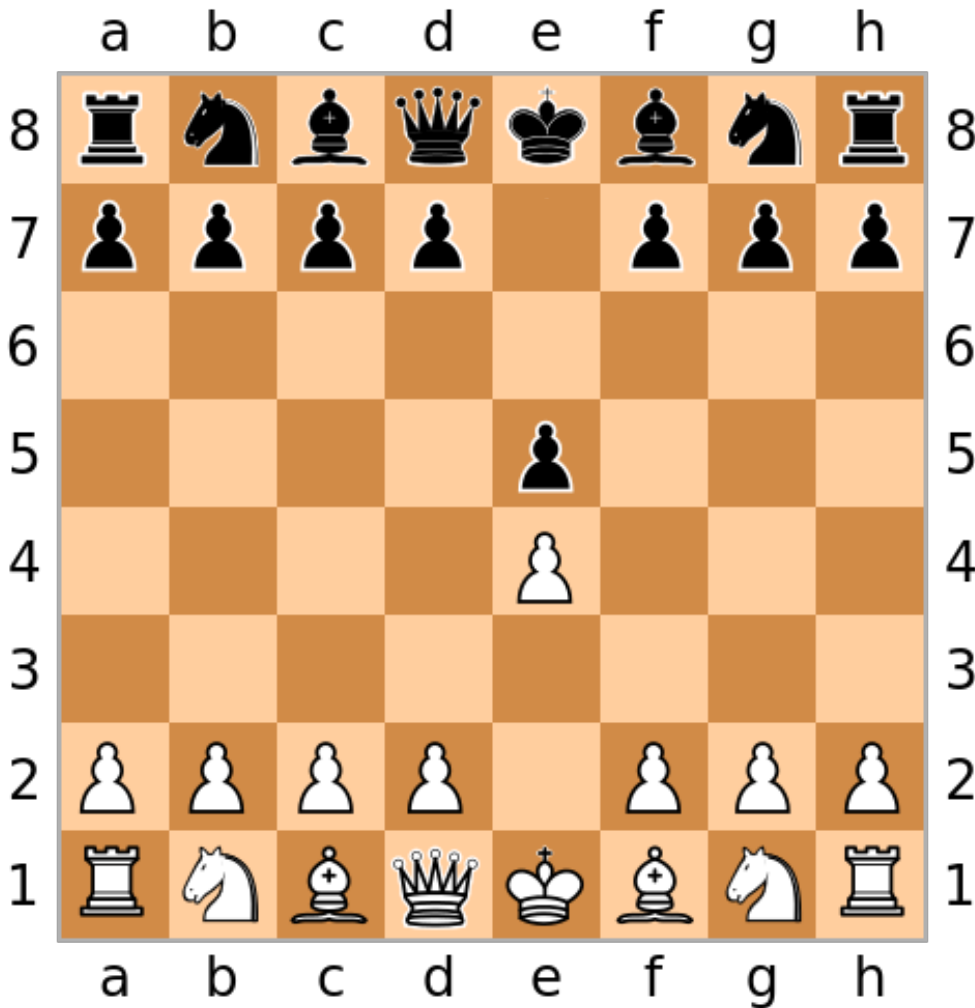
Classic solution:

1) Explore the tree of possible games down to some depth.

2) Use a 'proxy' function P(s) to predict who will win, given the state s of the board at that depth.

Proxy Function *P*(*s*)

Given the state s of the board, the proxy function P(s)

- should be positive if the Whites are likely to win;
- should be negative if the Blacks are likely to win;
- Its absolute value should increase with the confidence.

How to build such a function?

Building the Proxy Function (1)

First, introduce 'Features' $f_i(s)$ based on expert knowledge such as:

- $f_h(s) = 1$ if the White Queen is still alive, 0 otherwise;
- $f_i(s) = 1$ if the Black Queen is still alive, 0 otherwise;
- $f_j(s) = 1$ if the White Bishop #1 is still alive, 0 otherwise;
- $f_k(s) =$ number of possible moves for Black;
- $f_l(s) = +\infty$ if Black is mate, $-\infty$ if White is mate, 0 otherwise.
- etc…

Building the Proxy Function (2)

Second, write P as a linear combination of the features:

$$P(s) = \sum_i \alpha_i f_i(s)$$

How can we learn the α_i weights?

→ Use many games from the history of chess to optimize.

This approach thus requires human expertise:
1. To design the features;
2. To estimate good weights for the features.

MinMax Algorithm
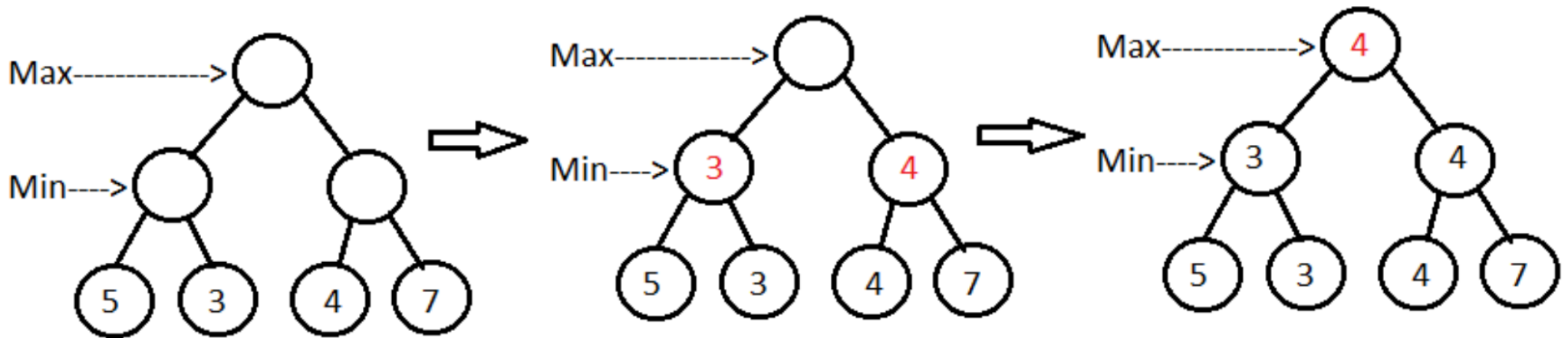
- The proxy function is evaluated only in the leaves of the game tree.
- To assign values and moves in the other nodes, one picks the min score for the adversary and the largest for oneself.
- This still requires a lot of computation even when building the tree only down to a certain level.
- Can be sped up using heuristics such as alpha-beta pruning.

Alpha-Beta Pruning
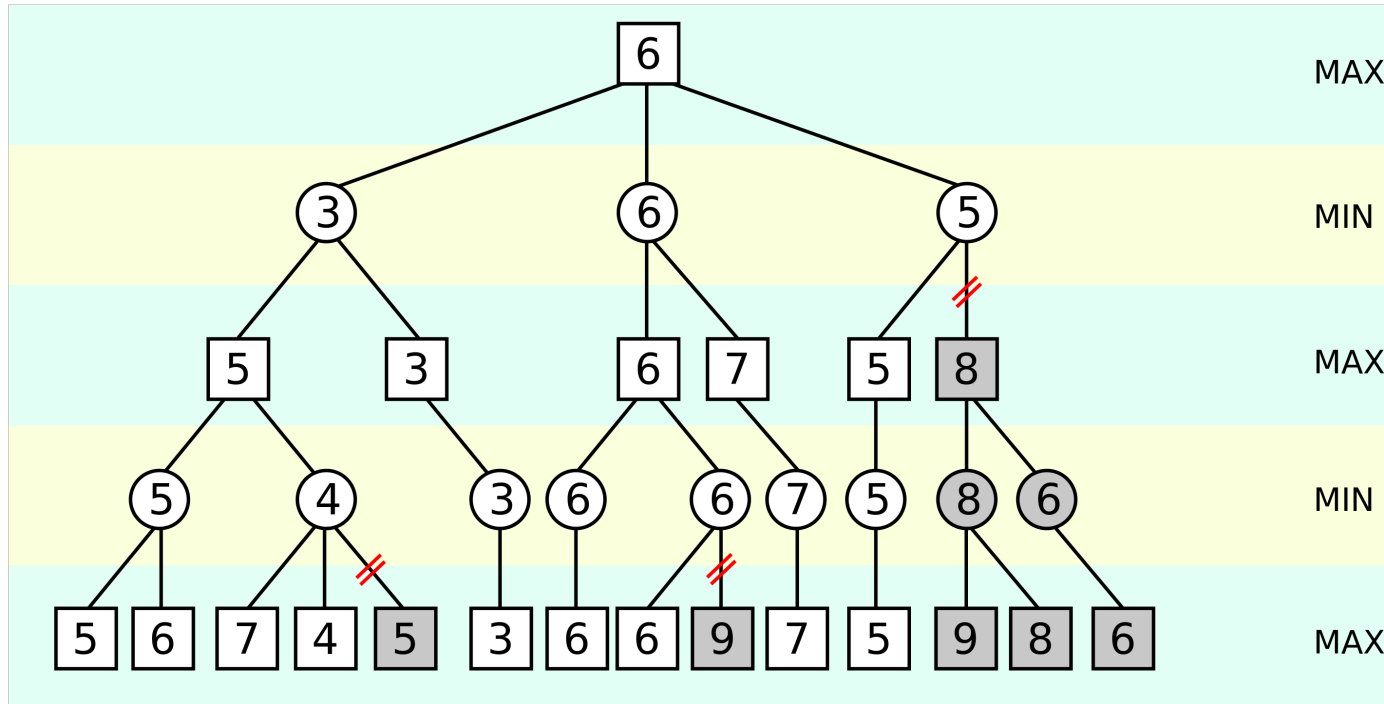
- The max and min levels represent the turn of the player and the adversary, respectively.
- When traversing the tree from left to right, the grayed-out subtrees need not be evaluated, as they correspond to states no better than those that have been explored.
- This helps and makes it possible to play chess on a cellphone.

Go

19x19=361
possible moves

. . .

360
possible moves
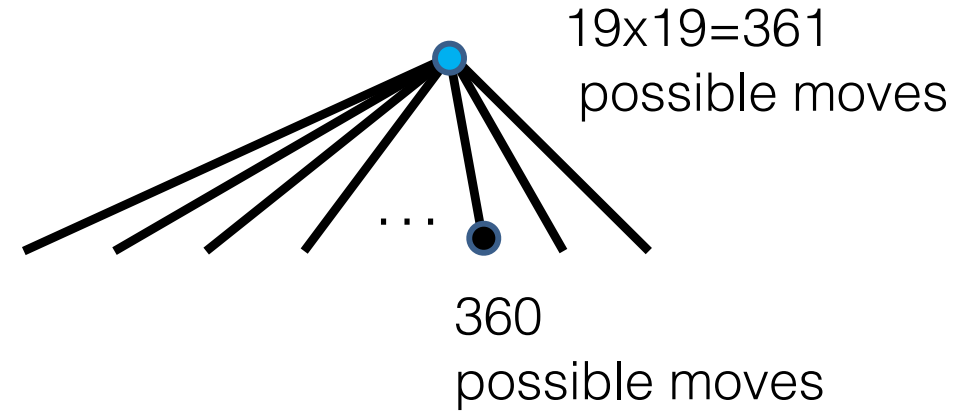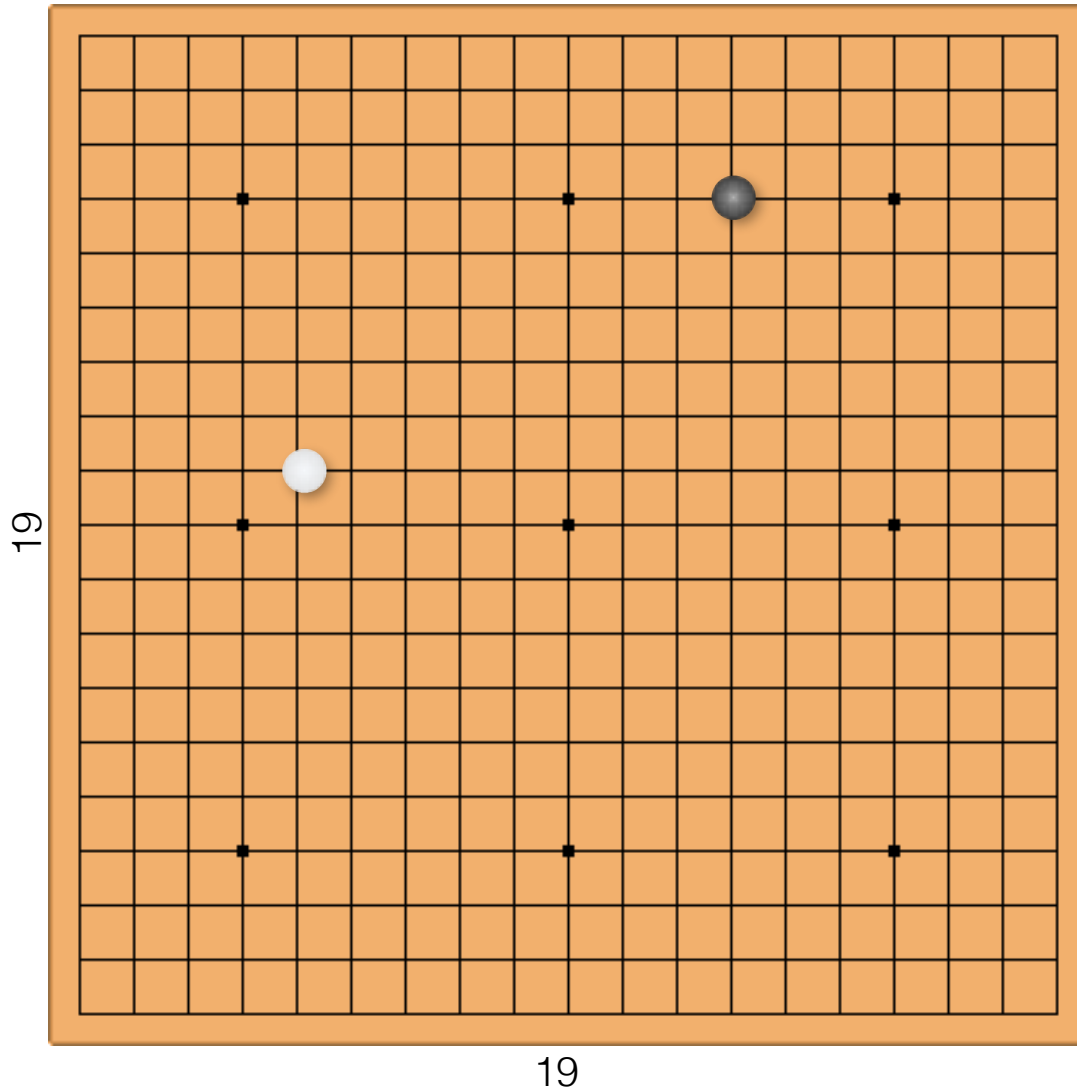
Estimated number of possible games: 10^{172} vs 10^{120} for chess.

—> Massively larger!

Goal: Building Territories

Proxy Function?

For this situation.
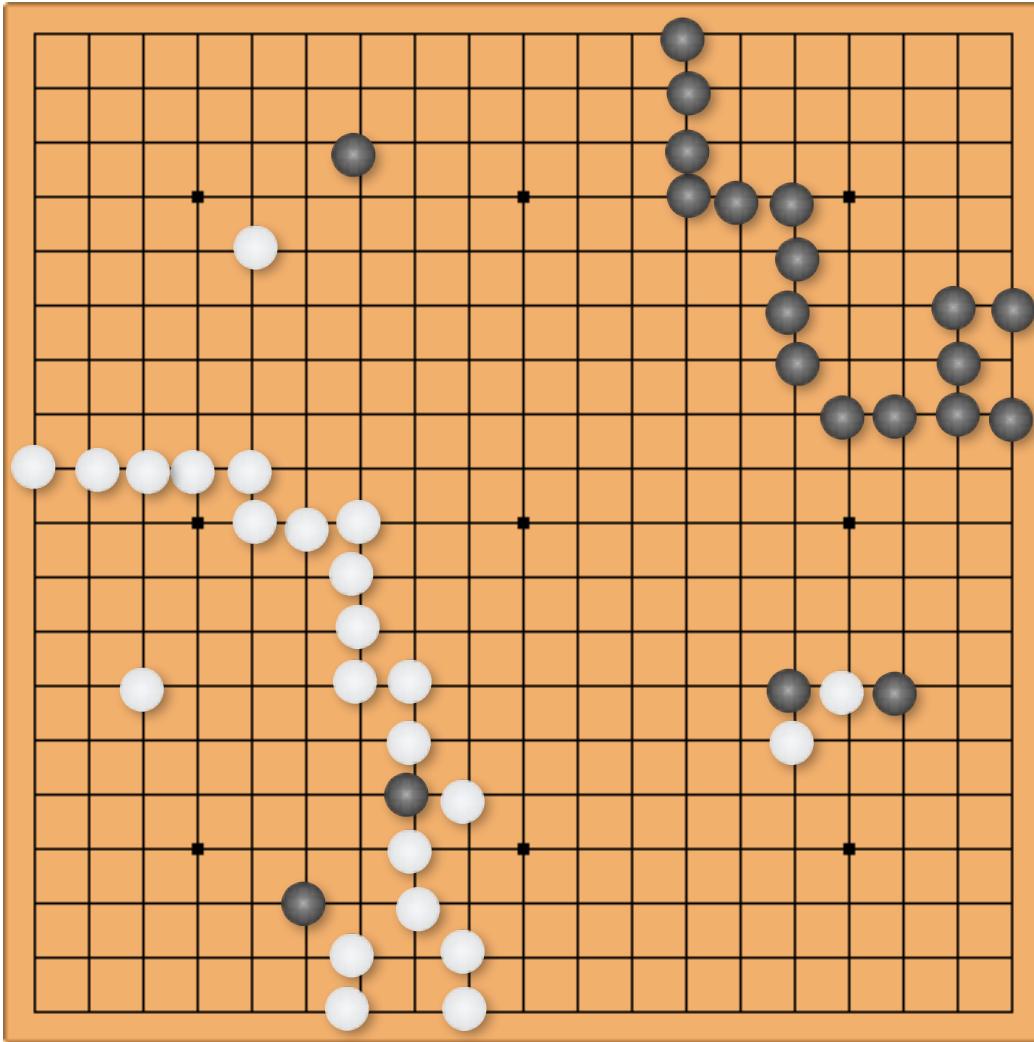
Proxy Function?

And this one.
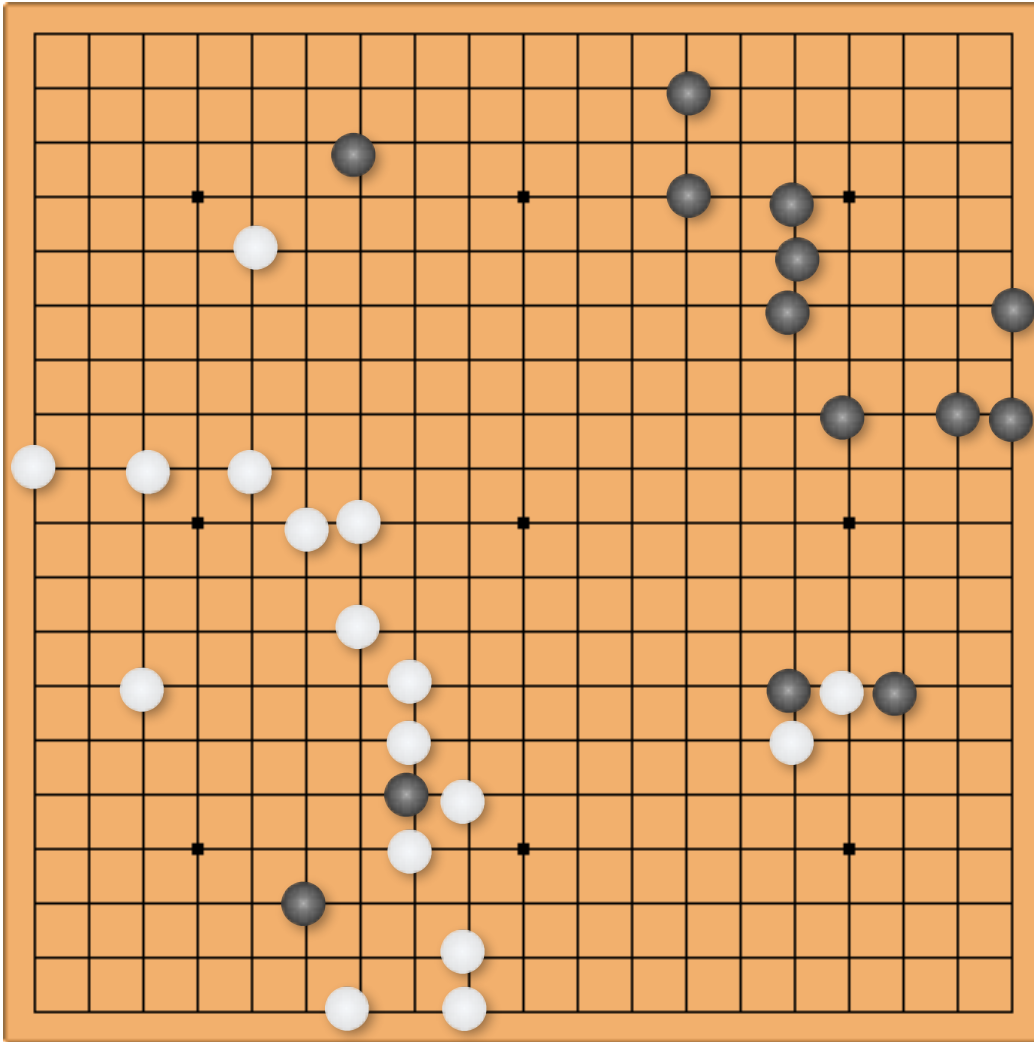
Proxy Function?

And that one.

- Very complex.

- Difficult to build a good proxy function.
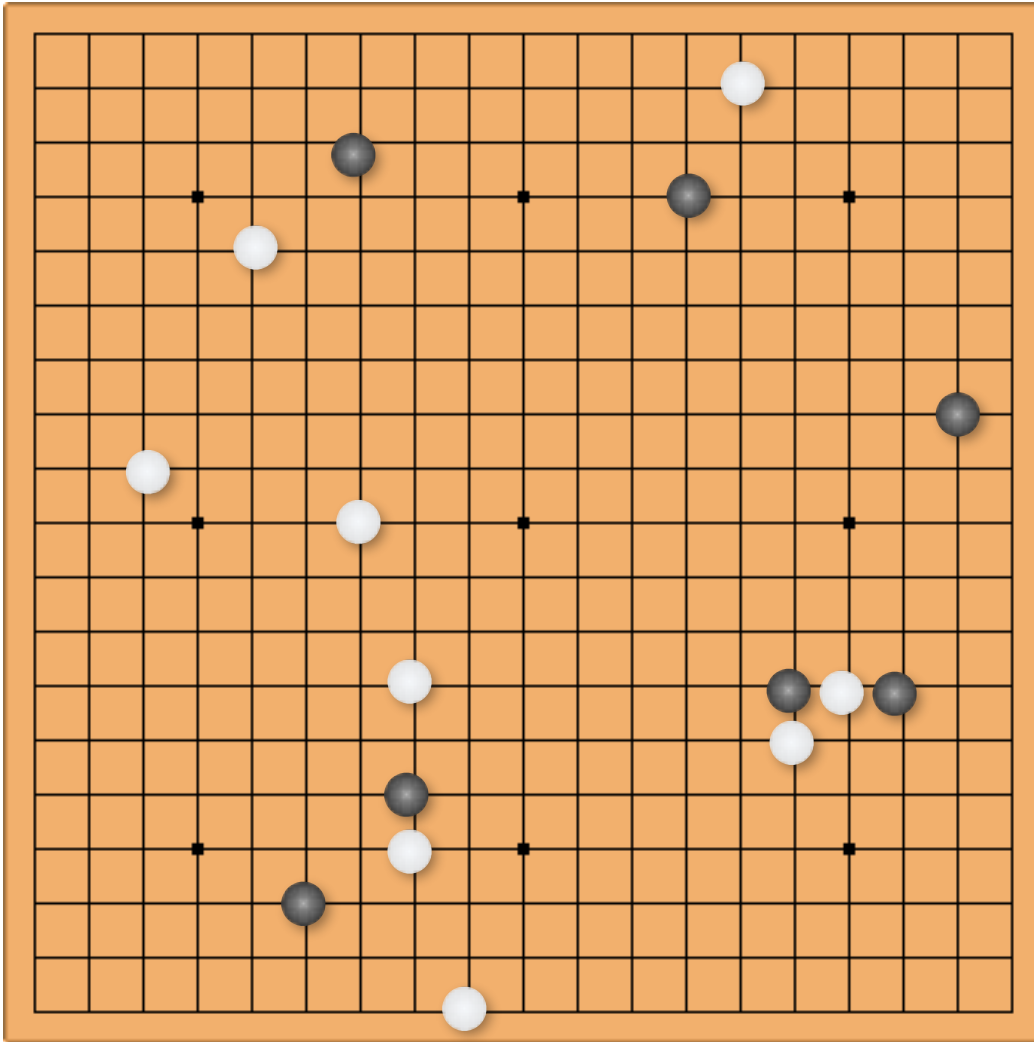
The AlphaGo Zero Algorithm
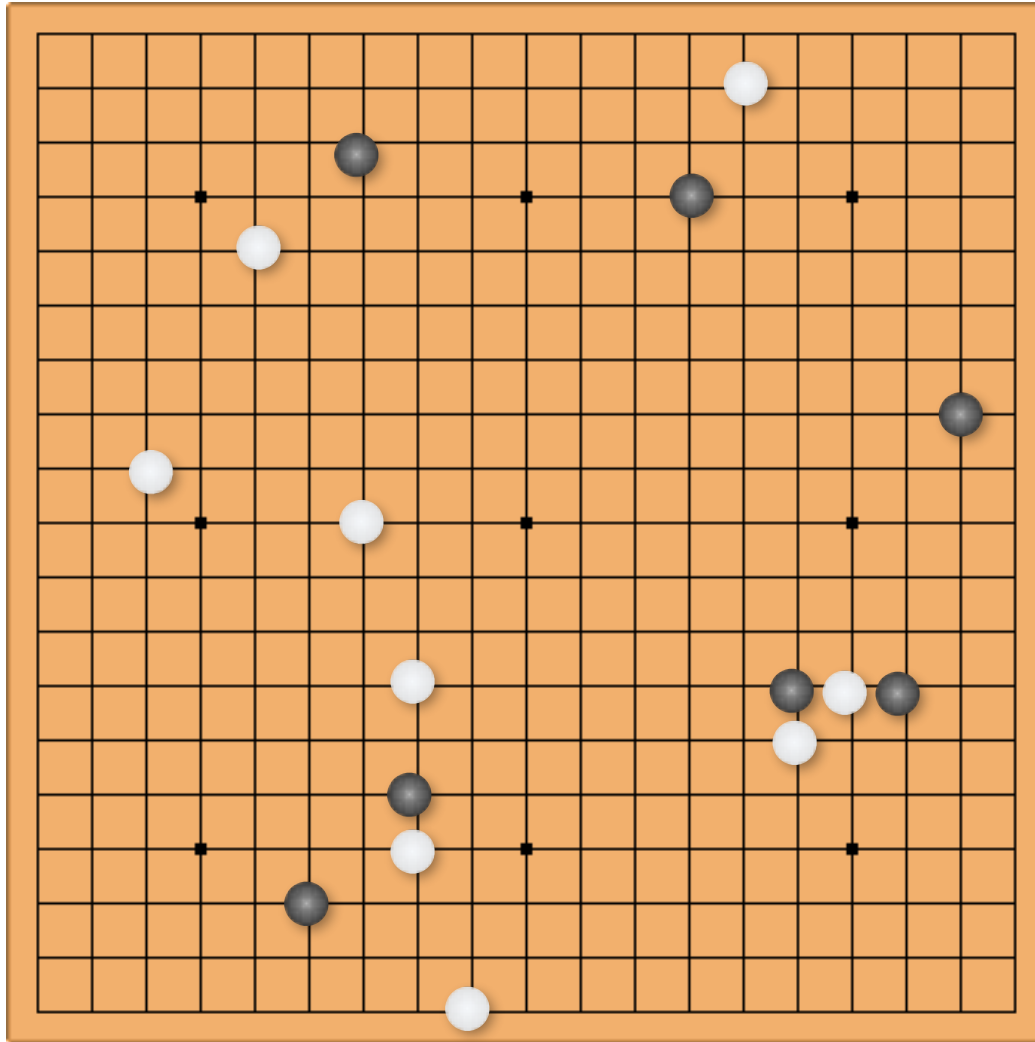
Mastering the Game of Go without Human Knowledge
David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, Demis Hassabis.
Nature, 2017.

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm
David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis
Science, 2018.

École des Ponts
ParisTech

EPFL

Policy and Value

Let us assume we are given the function

$$f: s \longrightarrow f(s) = (\mathbf{p}, v),$$

where

- s denotes the current state of the ban.
- $\mathbf{p}$ is the *policy,* that is, the vector of probabilities $\mathbf{p}_a$ that playing move $\mathbf{a}$ will lead to victory.
- v is the value of $\mathbf{s}$ (~ proxy function in chess).

How should we use it?
- In theory, we could choose to play action $\hat{a} = \arg \max_a p(a)$.
- In practice, f is never perfect, and it is often better to explore several possible moves to decide on the next action.

But how do we implement f in the first place?

École des Ponts
ParisTech

EPFL

From Go Ban to Images

Go Ban representation
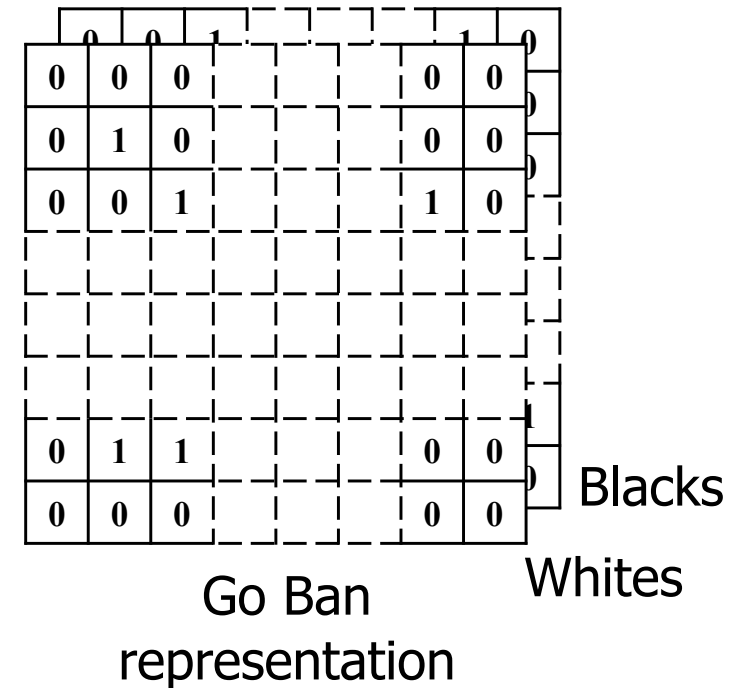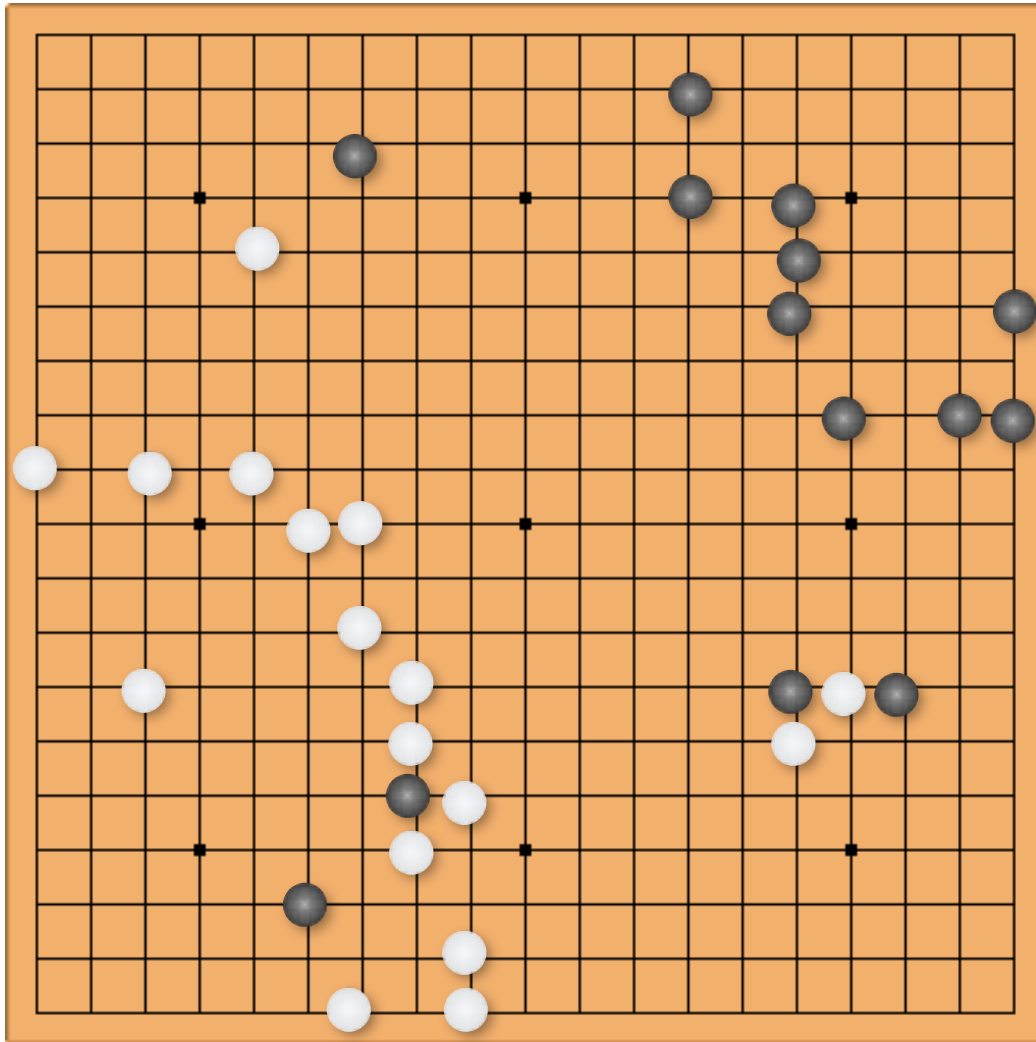
Blacks

Whites

- The state of the go ban can be represented by two binary images, one for white and one for black.

- They can be used as input to a CNN.

CNNs to the Rescue

$$f(s; \Theta) \rightarrow (\mathbf{p}, v)$$

Go Ban representation s

Blacks

Whites
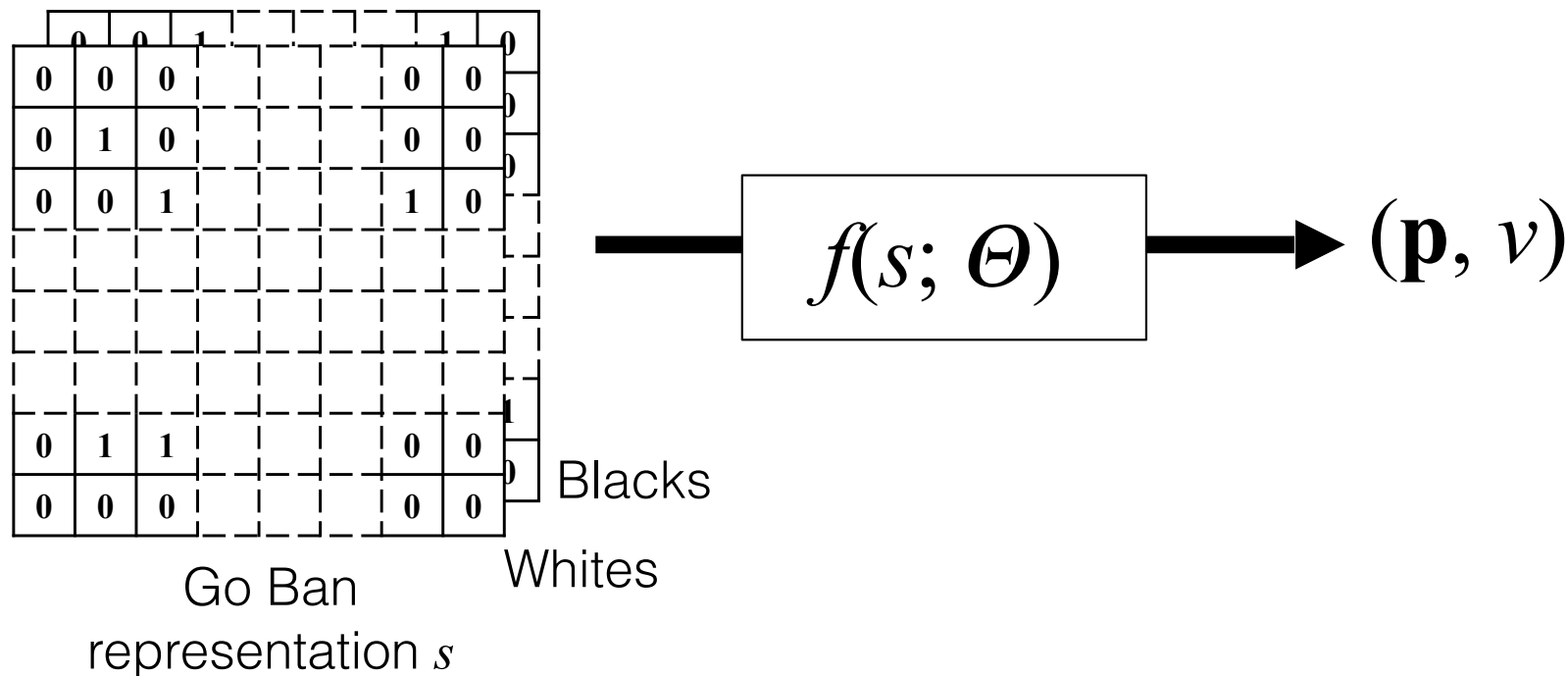
- f can be implemented by a CNN with weights Θ.
- We need a large training database to learn them.

Learning the Network Parameters

1. Initialize the parameters Θ of f randomly and play against itself.
2. For each move, play according to the probabilities in p = f(s).
3. Run each simulation until the end of the game.
4. For each board visited by the simulation, store s, **p**, and v in a set T:

    - If the game is won, set v = +1;
    - If the game is lost, set v = -1.

5. Update the policies.
6. Retrain f using a training set made of pairs (s, (**p**, v)).
7. Goto 2

Reinforcement Learning

- The algorithm described on the previous slides tries many different policies and learns from its mistakes.

- This requires large numbers of simulations to cover the space of possibilities well.

- It is a generic paradigm that works well, but mostly on simulations and computer games so far.

- Time will tell if it can be extended further.

See details in