Reinforcement Learning In GOMOKU

Cheng Zhong
2021-11-27

Outline

- ADP
- MCTS
- Tree-Search + Heuristic
- AlphaGo Zero

ADP for Gomoku

- Adaptive Dynamic Programming(ADP):

  ADP used in Gomoku is trained by temporal difference learning (TDL)

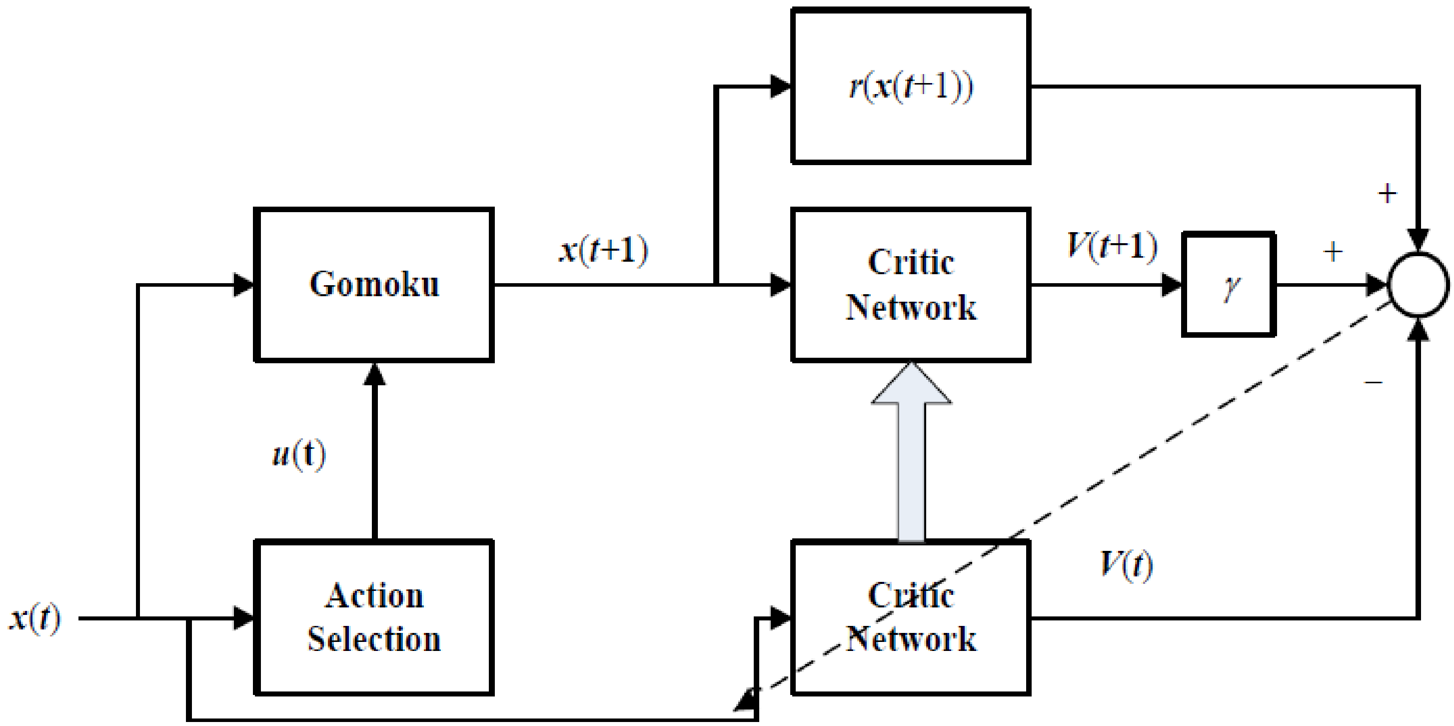
- Key idea of ADP

  In TD learning, the action decision or value function can be described in continuous form, approximated by nonlinear function such as neural network

Dongbin Zhao, Zhen Zhang, and Yujie Dai.
Self-teaching adaptive dynamic programming for Gomoku.
Neurocomputing, 78(1):23 - 29, 2012.

ADP for Gomoku

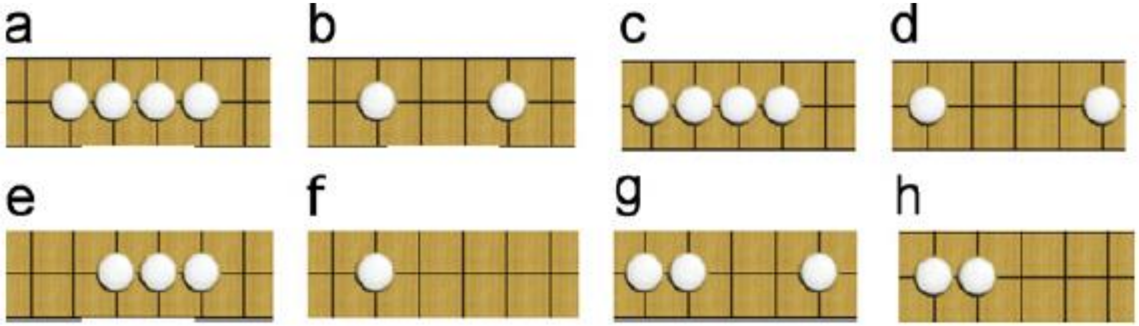
- The ADP structure



  - x(t): current board state;
  - u(t): action;
  - x(t+1): next step state;
  - r(x(t+1)): reward;
  - The critic network is used to eatimate value function V(t)

ADP for Gomoku

- The state to describe a board situation
  - 20 patterns for each of two players, totally 40 patterns



  - Whose turn to move
  - In the offensive/defensive(Who is first to move)

ADP for Gomoku

- The state to describe a board situation
  - Five input nodes indicate the number of every pattern except for five-in-a-row (n denotes the number of a pattern)

Value of n	Input 1	Input 2	Input 3	Input 4	Input 5
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
>4	1	1	1	1	$(n-4)/2$

  - The number of the special pattern five-in-a-row, is represented by 1 input node. If this pattern shows up, then its input is 1, otherwise 0

ADP for Gomoku

- The state to describe a board situation
  - For each pattern we assign two input nodes to represent the turn
  - Use two input nodes to indicate which player is the first to move
  - Totally 19*5*2+1*1*2+40*2+2 = 274 input nodes

ADP for Gomoku

- Critic Network in the ADP (The value function)
  - Train the function approximator
    - Define the prediction error

$$e(t) = \alpha[r(t+1) + \gamma V(t+1) - V(t)]$$

    - To minimize the objective error

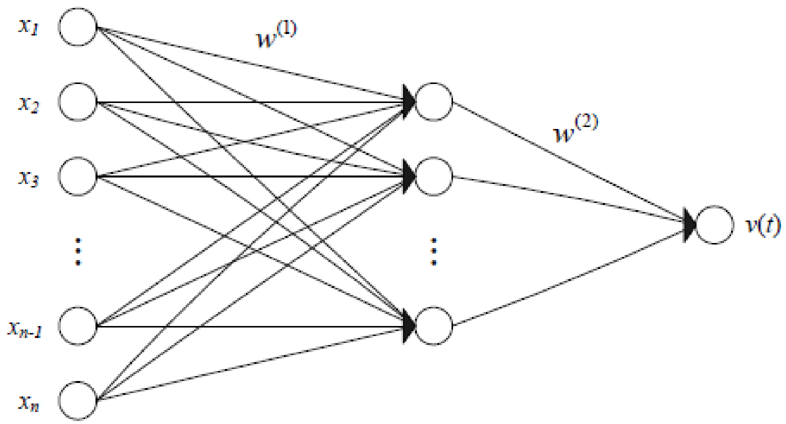
$$E(t) = \frac{1}{2}e^2(t)$$

ADP for Gomoku

- Reward
  - The reward is set to 0 during the game.
  - After a game, if player 1 wins, the final reward is 1, if he loses, the reward is 0, and if he draws, the reward is 0.5.

ADP for Gomoku

- Critic Network in the ADP (The value function)
  - Used to evaluate board situations(winning probability of player1)
  - A feed forward three-layer fully connected neural network



  - Unnecessary to be neural network, you can try other functions

ADP for Gomoku

- Action
  - Player 1 chooses the move that leads to the state with the maximal output value obtained from the critic network.
  - Player 2 selects the move that leads to the state with the minimal output value obtained by the same critic network.

ADP for Gomoku

- Action
  - Reduce the action space
    - Only considering the empty positions near the ones occupied
    - When there are several alternative actions which have equally high evaluation, we simply choose the one that is last found

ADP for Gomoku

- Action
  - Cope with the exploration and exploitation dilemma
    - Let player 2 randomly select his first move, meanwhile player 1 place his piece on the center of the board if he is in the offensive and select his first move randomly if he is in the defensive
    - Let both players select moves following ϵ-greedy policy

$$a(t) = \begin{cases} \arg\max_a V(t+1) & \text{with probability } 1-\varepsilon \\ \text{random action} & \text{with probability } \varepsilon \end{cases}$$

ADP for Gomoku

- Self-teaching
  - Playing against itself
  - On a platform called Pisvorky
  - Both player1 and player2 use the same critic network

→ ST-Gomoku

Case	Input (Turn)	Hidden	Training	Beginner	Diletante	Candidate
Case 1	274 (80)	100	60,000	30:0	22:8	13:17

ADP with MCTS for Gomoku

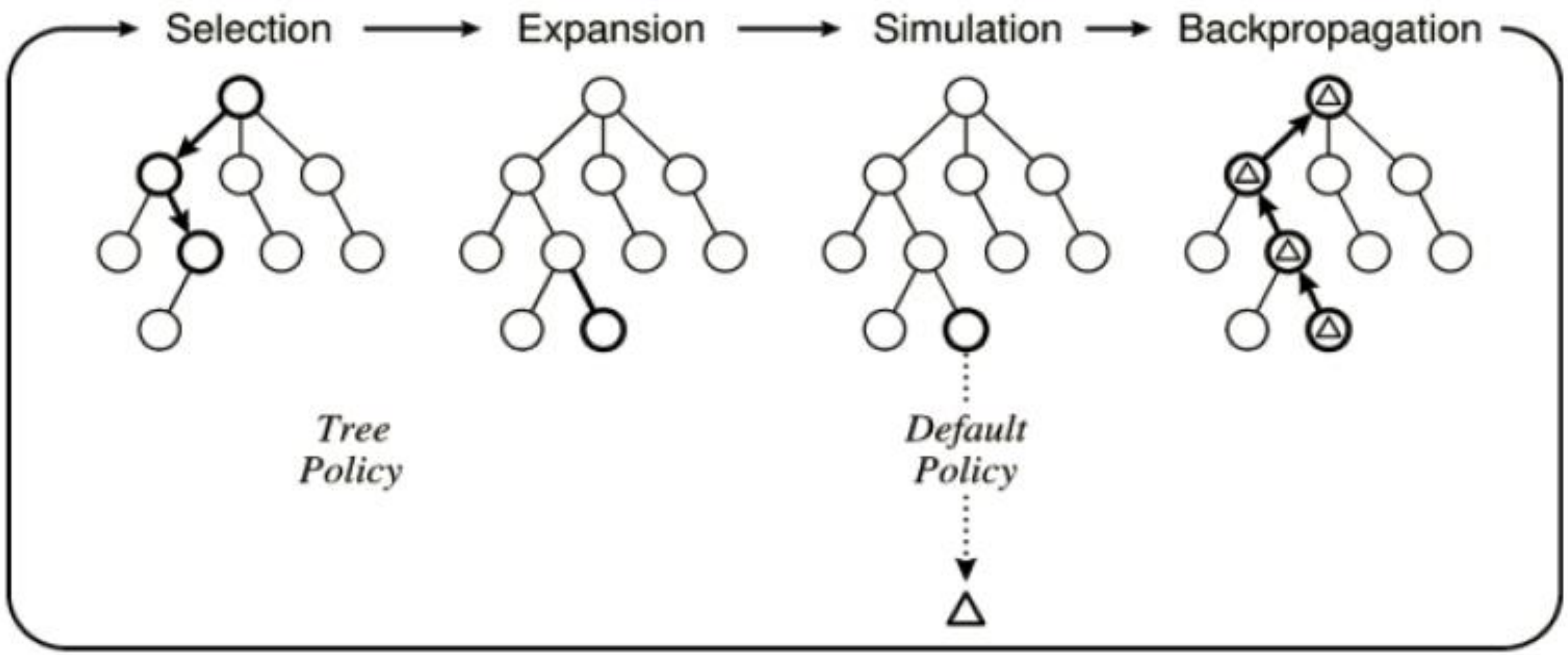
- Monte Carlo Tree Search(MCTS)
    - The Basic process of MCTS
    - HMCTS
    - UCT
- ADP with MCTS

Zhentao Tang, Dongbin Zhao, Kun Shao, and Le Lv.
ADP with MCTS algorithm for Gomoku.
2016 IEEE Symp. Ser. Comput. Intell. SSCI 2016, (61273136), 2017.

MCTS

- Requires a large number of simulation and builds up a large search tree according to the results.
- The estimated value will be more accurate with the increase of the simulation times and nodes accessed.

The Basic process of MCTS



HMCTS

- Heuristic Monte Carlo Tree Search
- Apply Heuristic Knowledge in Simulation Policy

HMCTS

- Heuristic rules
  - If four-in-a-row is occurred in my side, the player will be forced to move its piece to the position where it can emerge five-in-a-row in my side.
  - If four-in-a-row is occurred in opposite side, the player will be forced to move its piece to the position where it can block five-in-a-row in opposite side.
  - If three-in-a-row is occurred in my side, the player will be forced to move its piece to the position where it can emerge four-in-a-row in my side.
  - If three-in-a-row is occurred in opposite side, the player will be forced to move its piece to the position where it can block four-in-a-row in opposite side.

HMCTS

- Save more time in simulation than random sampling and get converge earlier
- Q-value function

$$Q(s,a)=\frac{1}{N(s,a)}\sum_{i=1}^{N(s)} l_i(s,a)z_i$$

Algorithm 1: HMCTS for Gomoku

```
input original state s0;
output action a corresponding to the highest value of MCTS;
add Heuristic Knowledge;
obtain possible action moves M from state s0;
for each move m in moves M do
    reward r_total ← 0;
    while simulation times < assigned times do
        reward r ← Simulation(s(m));
        r_total ← r_total + r;
        simulation times add one;
    end while
    add (m, r_total) into data;
    end for each
return action Best(data)

Simulation(state s_t)
    if (s_t is win and s_t is terminal)  then return 1.0;
                                          else return 0.0;
    end if
    if (s_t satisfied with Heuristic Knowledge)
        then obtain forced action a_f;
                new state s_t+1 ← f(s_t, a_f);
        else choose random action a_r ∈ untried actions;
                new state s_t+1 ← f(s_t, a_r);
    end if
    return Simulation(s_t+1)

Best(data)
    return action a   //the maximum r_total of m from data
```

UCT

- Upper Confidence bounds for Tree
  - Based on Upper Confidence Bounds(UCB)

$$\frac{Q(v')}{N(v')} + c\sqrt{\frac{2\ln N(v)}{N(v')}}$$

  - Balance the conflict between exploration and exploitation and find out the final result earlier
  - $\frac{Q(v')}{N(v')}$ is the average reward of node v', $N(v')$ and $N(v)$ is the visited count of node v' and v, v is the parent of v'

UCT

Algorithm 2: UCT for Gomoku

input create root node v_0 with state s_0;
output action a corresponding to the highest value of UCT;
while within computational budget **do**
 $v_l \leftarrow$ Tree Policy(v_0);
 Policy $\leftarrow$ Heuristic Knowledge;
 reward $r \leftarrow$ Policy($s(v_l)$);
 Back Update(v_l, r);
end while
return action a(Best Child(v_0))

$\boxed{\text{Tree Policy(node } v)}$
 while v is not in terminal state **do**
  if v not fully expanded **then** **return** Expand(v);
   else $v \leftarrow$ Best Child(v, $1/\sqrt{2}$);
  end if
 end while
 return v //this is the best child node

Expand(node v)
 choose random action $a \in$ untried actions from $A(s(v))$;
 add a new child v' to v
  with $s(v') \leftarrow f(s(v), a)$ and $a(v') \leftarrow a$;
 return v' //this is the expand node

$\boxed{\text{Best Child(node } v\text{, parameter } c)}$
 return $\arg\max_{v' \in child}((Q(v')/N(v')) + c\sqrt{2\ln N(v)/N(v')})$

$\boxed{\text{Policy(state } s)}$
 while s is not terminal **do**
  if s satisfied with heuristic knowledge **then**
   obtain forced action a;
  else choose random action $a \in A(s)$ uniformly;
  end if
  $s \leftarrow f(s, a)$;
 end while
 return reward for state s

Back Update(node v, reward r)
 while v is not null **do**
  $N(v) \leftarrow N(v) + 1$;
  $Q(v) \leftarrow Q(v) + r$;
  $v \leftarrow$ parent of v;
 end while

MCTS

- UCT compared to HMCTS
  - Be originated from HMCTS.
  - Can help to find out the suitable leaf nodes earlier.
  - Can save more time than HMCTS.

ADP with MCTS

- Use ADP to train critic network, get top-5 candidate moves and their ADP winning probabilities
- Take each of candidate moves as the root node of MCTS and simulate, get their MCTS winning probabilities
- Calculate the weighted sum of two winning probabilities:

$$w_p = \lambda w_1 + (1 - \lambda) w_2$$

ADP with MCTS

- ADP: ST-Gomoku

Algorithm 3: ADP with MCTS

input original state s_0;
output action *a correspond to ADP with MCTS*;
M_{ADP}, $W_{ADP} \leftarrow$ ADP Stage(s_0);
$W_{MCTS} \leftarrow$ MCTS Stage(M_{ADP});
for each w_1, w_2 in pairs(W_{ADP}, W_{MCTS}) **do**
    $w_p \leftarrow \lambda w_1 + (1\text{-}\lambda)w_2$;
    add p into P;
end for each
return action a correspond to max p in P

ADP Stage(state s)
    obtain top 5 winning probability W_{ADP} from ADP(s) ;
    obtain their moves M_{ADP} correspond to W_{ADP};
    return M_{ADP}, W_{ADP}

MCTS Stage(moves M_{ADP})
    for each move m in M_{ADP} **do**
        create m as root node with correspond state s
        obtain w_2 from *MTCS*(m, s)
        add w_2 into W_{MCTS}
    end for each
    return W_{MCTS}

ADP with MCTS

- Compared to ADP :
  - Eliminate the neural network evaluation function's “short sight” defect, ensure the accuracy of the search
- Compared to MCTS :
  - Save a large amount of time to find out the suitable action for Gomoku

Other Heuristic Functions

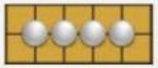
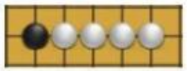
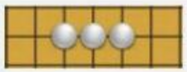
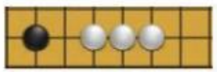
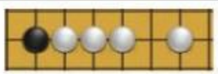
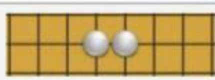
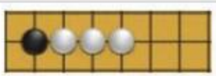
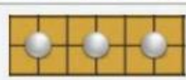
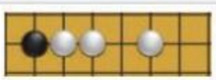
ID(Type)	Pattern	Value
1		10000
2		1000
3		1000
4		1000 * factor
5		1000 * factor
6		1000 * factor
7		100
8		100
9		100
10		100 * factor

Fig. 3. Example of the patterns and their heuristic value.

$$H_i = \sum \{10^{L_{open}} * factor^j + 10^{L_{hclose}-1} * factor^k\} \quad (3)$$

$Factor^{j,k} = 0.9$

$$UCB = v_i + k_1 * \sqrt{\frac{ln(N)}{n_i}} + k_2 * \frac{H_i}{max(H)}$$

Xu Cao and Yanghao Lin.
UCT-ADP Progressive Bias Algorithm for Solving Gomoku.
2019 IEEE Symposium Series on Computational Intelligence (SSCI) .

ADP with MCTS

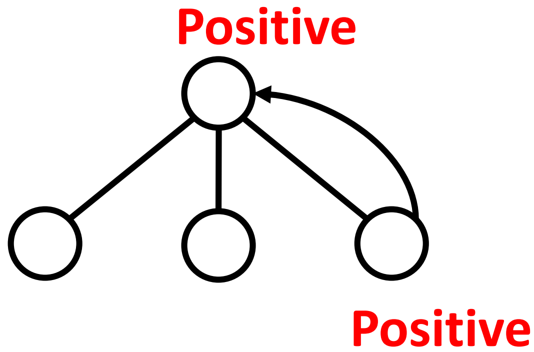
- Experimental Results

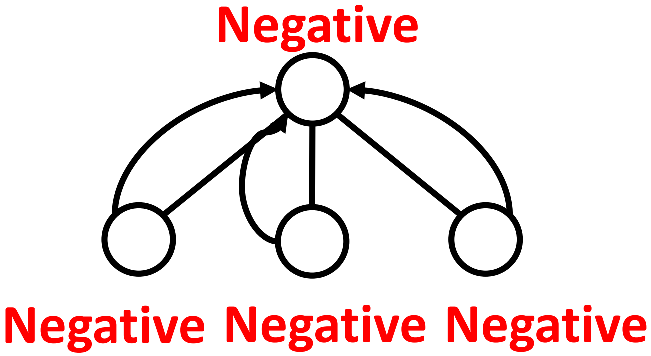
TABLE IV. COMPARISION AGAINST 5-STAR GOMOKU

Algorithm	Gomoku Level		
	Beginner	Dilettante	Candidate
ADP	100:0	73:27	43:57
HMCTS	46:54	13:87	0:100
ADP-HMCTS	100:0	89:11	71:29
ADP-UCT	100:0	82:18	64:36

Search Tree + Heuristic

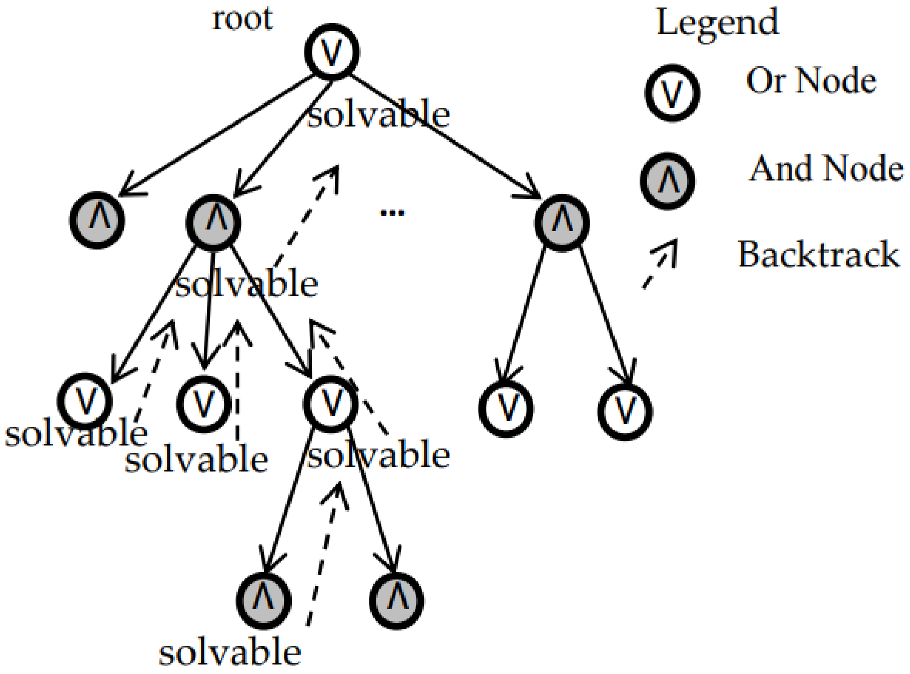
- AND/OR Tree





Search Tree + Heuristic

- AND/OR Tree



Zhikun Zhao, et al.
A VCT Discovery Algorithm of Renju.
2021 4th International Conference on Artificial Intelligence and Big Data

Search Tree + Heuristic

- AND/OR Tree
  - Board situation: Have, Not, Unknown
  - 2 Nodes:
    - Black Turn (OR)
      - Positive if there is an action (White take) leading to Black Positive
      - Not if all actions leading to Black Negative
    - White (AND)
      - Positive if all actions leading to Black Positive
      - Negative if there is an action leading to Black Negative

Search Tree + Heuristic

- Compute the decay value of parents
  - The situation produced by strong offensive moves should be considered prior to those produced by weak offensive moves.
  - If an AND node has solvable branches, then the value of other branches should be increased, AND node has greater possibility of solvable

Constant

$$H(n) = \begin{cases} H(parent) - \left(\dfrac{c}{attackScore}\right)^2, when\ parent\ is\ an\ OR\ node \\ H(parent) * \dfrac{total}{total - solvable}, when\ parent\ is\ an\ AND\ node \end{cases}$$

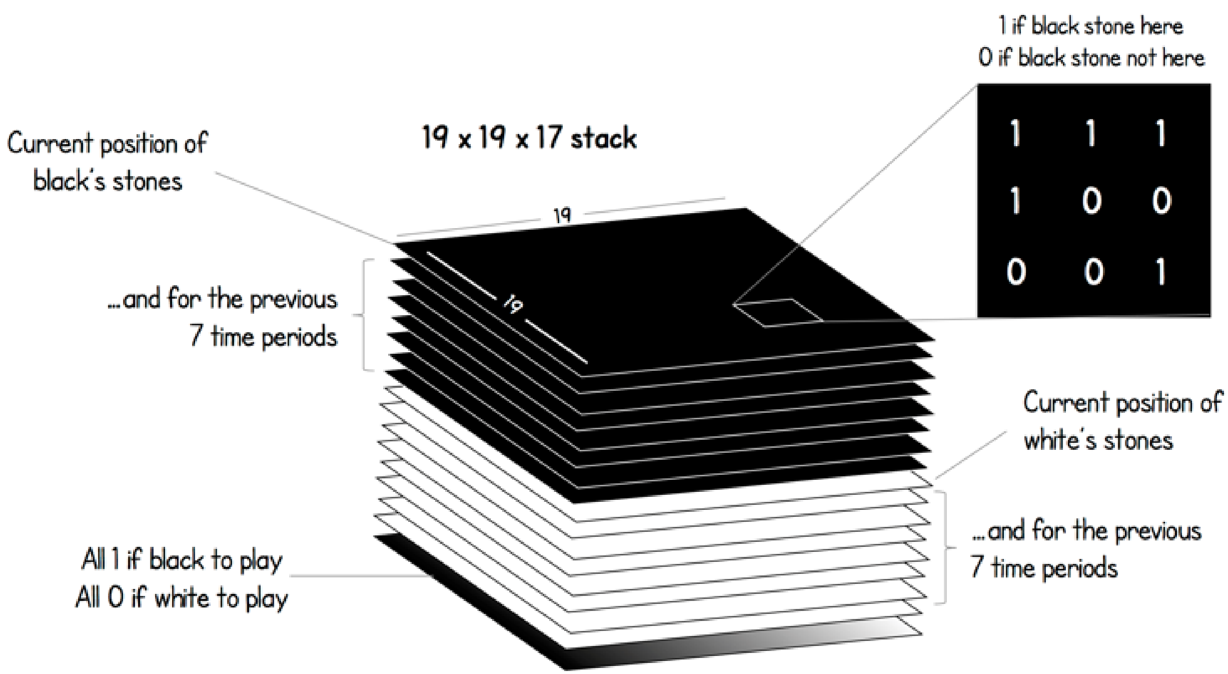
First

Last

Alpha-Zero Solution

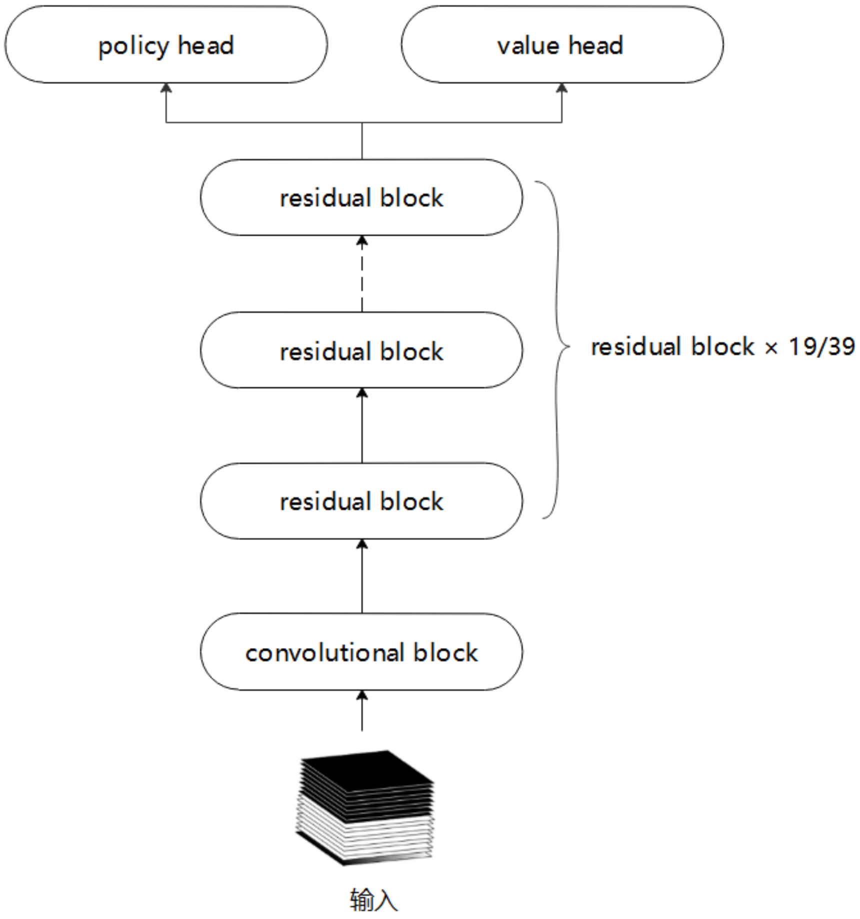
Alpha-Zero Solution

- Method
  - Feature Extractor

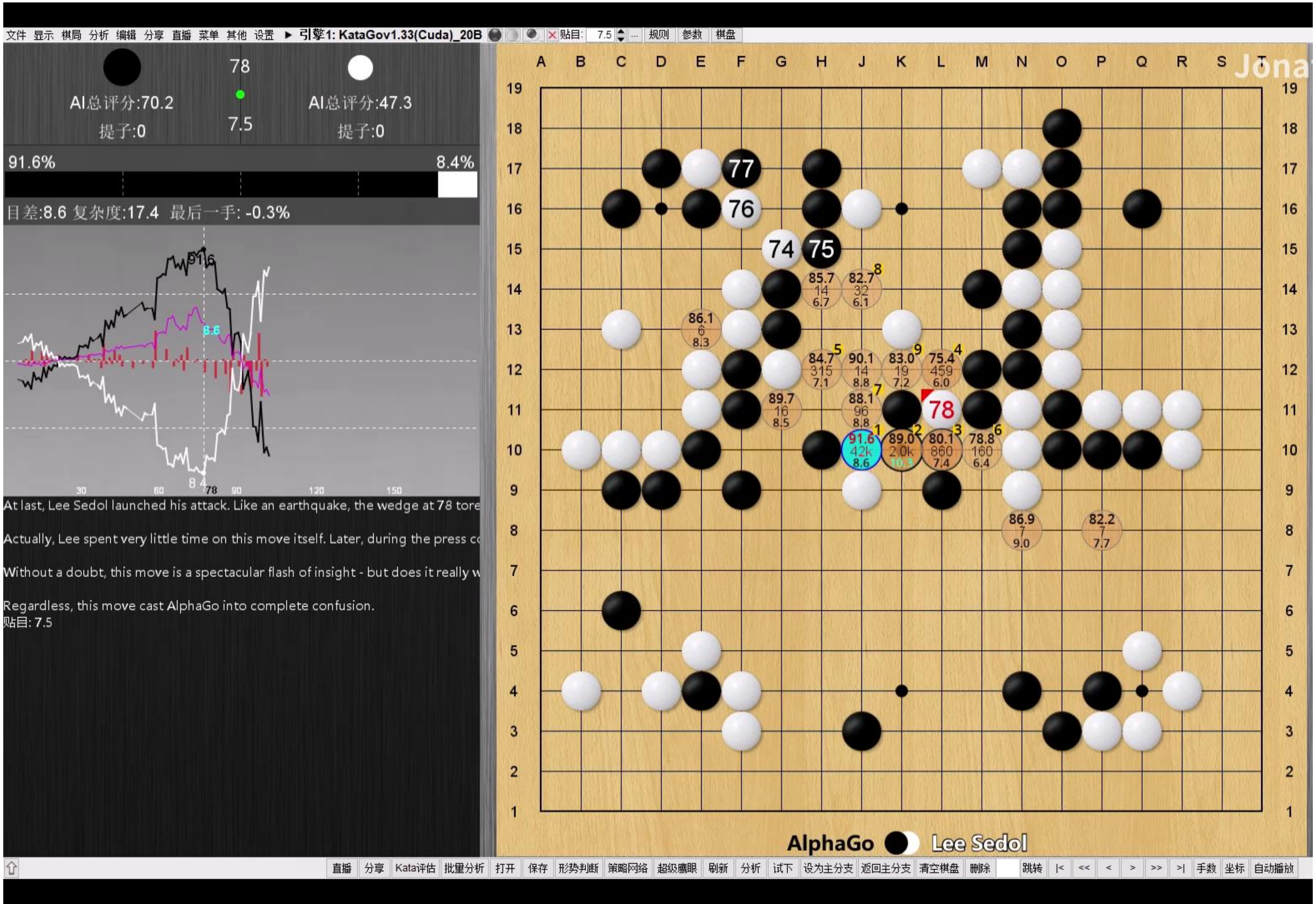


Alpha-Zero Solution

- Method
  - Feature Extractor
  - Policy & Value Head

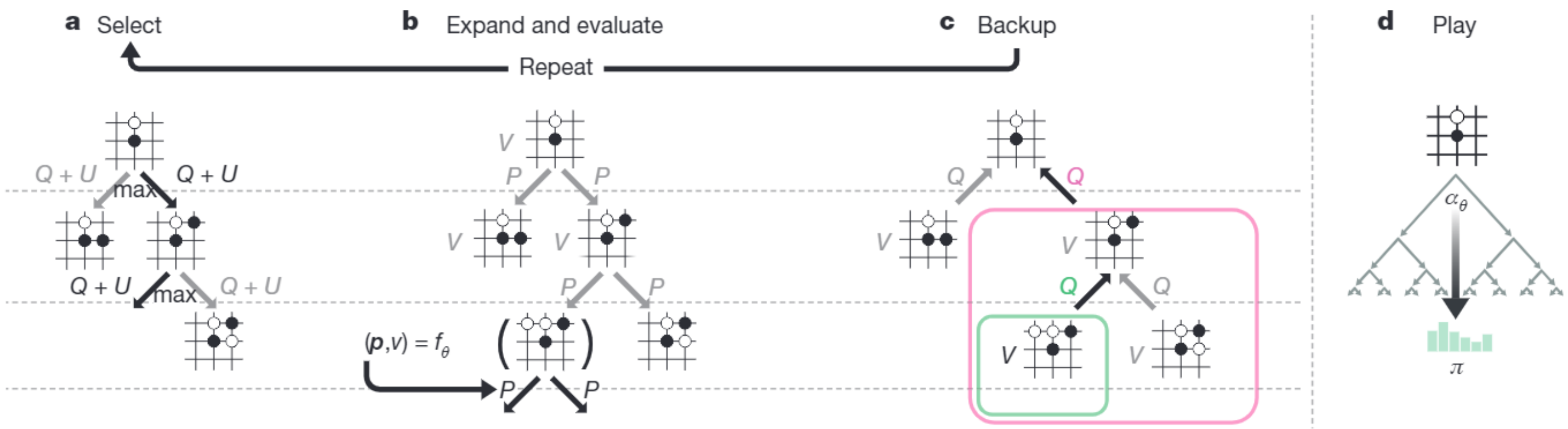


Alpha-Zero Solution



Alpha-Zero Solution

- Training
  - MCTS Generate

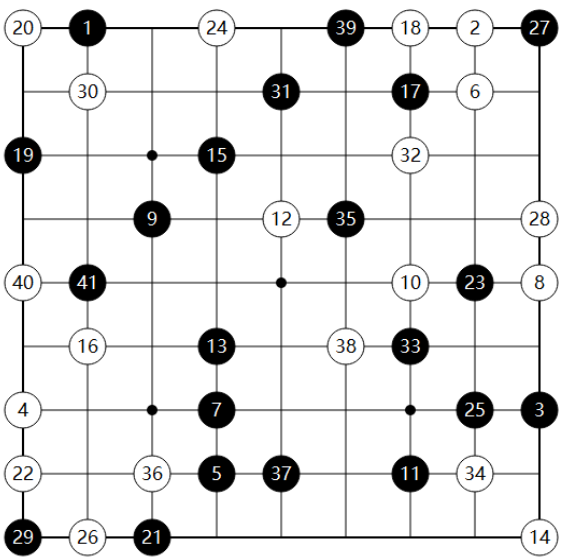


  - DL Training

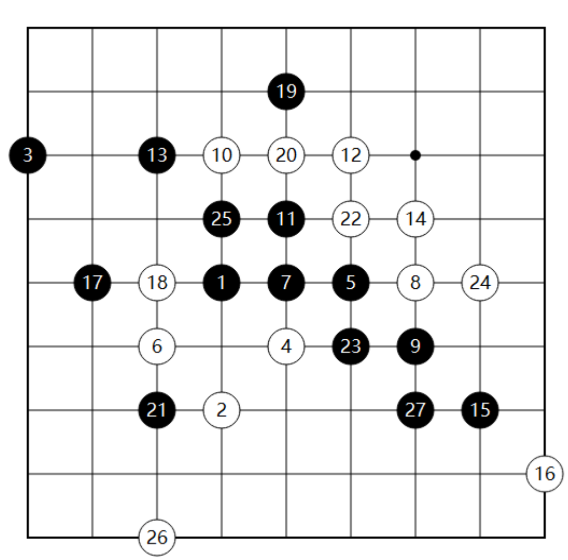
Alpha-Zero Solution

- Result

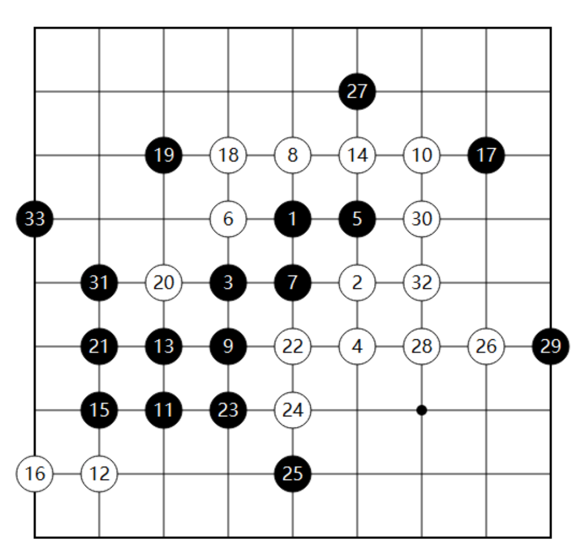
Iteration 0



Iteration 800



Iteration 4400



https://www.cnblogs.com/zhiyiYo/p/14683450.html