SOLVING A MATHEMATICAL PROBLEM IN SQUARE WAR: A GO-LIKE BOARD GAME

CHU LUO

Abstract. In this paper, we present a board game: Square War. The game definition of Square War is similar to the classic Chinese board game Go. Then we propose a mathematical problem of the game Square War. Finally, we show that the problem can be solved by using a method of mixed mathematics and computer science.

1. Introduction

Go, also known as Weiqi, is an old Chinese board game involving two sides, black and white. According the rule of Go, the board is a 19×19 two-dimensional grid of straight lines. The game starts with a vacant board which has 361 intersections. By convention, the black side first places a black stone, which is a game piece, on a vacant intersection. Then two sides alternately place stones with the side color on the remaining intersections. The complexity of Go is considered to be approximately 10^{171} legal states on the board, significantly more than the estimated number 10^{50} of chess [12]. Computational intelligence of Go still challenges computer scientists. Unlike chess, there currently is no computer program that can defeat professional human players.

Therefore, researchers attempt to develop invincible methods in computer Go, from various perspectives [6], [4], [2], [5], [9], [8] and [10]. The ability of computer Go is being improved gradually as the time advances.

Furthermore, there exist variants of the game Go, where artificial intelligence programs are also developed. For example, the game NoGo and Go-Moku are defined using similar game settings of Go. In [1], a computer program of Go-Moku shows that the player who moves first can always win the game through a specific strategy. The proof is similar to the mixed proof of mathematics and computer science of the famous Four-color theorem: contiguous regions in a map can be colored with at most four colors and without any two neighbors having the same color. A formal description of the proof of the Four-color theorem is detailed in [7]. For the game NoGo, a number of studies such as [3] and [11] introduce various methods to improve the ability of computer playing programs.

In this paper, we first present a board game: Square War. The game setting of Square War is very similar to Go. Then we propose a mathematical problem of the game Square War. Lastly, we use a method of mixed mathematics and computer science to solve the problem.

We thank people who gave comments to this work.

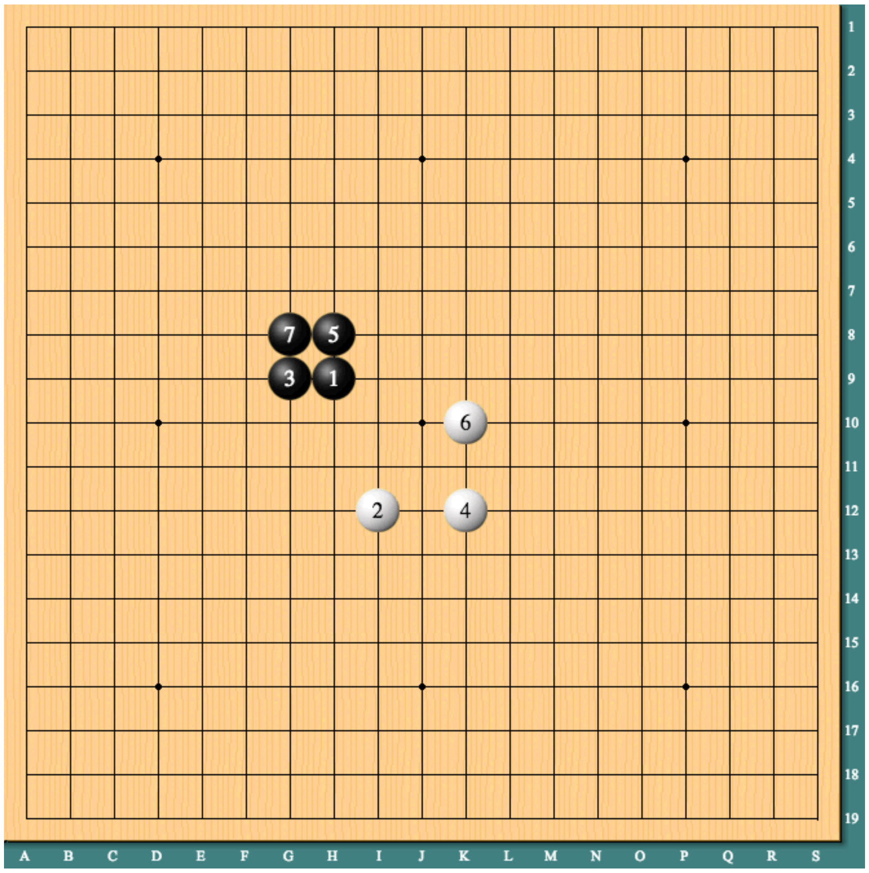

FIGURE 1. An example of the rules.

2. Definition of Square War

The game Square War comprises the following rules:

(1) The game is played on the same chess board as Go: 19×19 two-dimensional grid.
(2) The game is played by two sides. One side uses black stones (game pieces) and the other side uses white stones. This is also the same as Go.
(3) The black side first places a stone on an empty intersection and then two sides alternately place stones on empty intersections with their side colors. However, once a stone is placed, it cannot be removed.
(4) To add difficulty to the black side, the first two stones of the black side must be adjacent on a line of the grid.
(5) When four stones of a side become the four vertices of a square and the edges of this square overlap the grid lines, this side wins the game.

Figure 1 gives an example of the rules where the numbers on stones represent the step sequences. Black stones 1, 3, 5, and 7 become the four vertices of a square and the edges of this square cover the grid lines. As a result, the black side wins.

3. Mathematical Problem

Similar to other board games, the side moving first has advantage in Square War. This advantage can be crucial. For example, it is mathematically proven that the player who moves first can always win the game with a tactic in Go-Moku.

In this context, we propose a mathematical problem:

- Can the player who moves first always win the game with a tactic regardless of the opponent tactic?

4. Solution

In this section, we show the answer of the problem is positive. This is equal to the following theorem.

Theorem 4.1. *In the game Square War, let X be the tactic set of the black side. Similarly, let Y be the tactic set of the white side. If $x \in X$ and $y \in Y$, denote*

$$F(x,y) = \begin{cases} 1 & : \textit{when the black side wins} \\ 0 & : \textit{when the white side wins.} \end{cases} \tag{4.1}$$

$\forall y \in Y$, there always exists $x \in X$ such that $F(x,y) = 1$.

Proof. We prove with mixed mathematics and computer concepts. Given y, it is sure that the brute-force search can verify whether there is a related x, as the legal states on the board is finite. However, the computational complexity of the search may be extremely high. We use an algorithm T to give a X_0 where $X_0 \subset X$. And we show that $\forall y \in Y$, there always exists $x \in X_0$ such that $F(x,y) = 1$.

In T, $\forall x \in X_0$ starts with the position of $(J, 10)$ on the board, which is the centre of the board. After the centre is captured by the black side, we can assume that $\forall y \in Y$ starts with the position of (p,q) where p is from K to S and $1 \leq q \leq 10$, because the board is symmetric about the stone 1, the line J and the line 10. Then, T assigns the next black stone of $\forall x \in X_0$ to $(I, 10)$.

The fourth step is critical now. We define a set U by stating that $\forall u \in U$, the triple of stone 2, $(J, 11)$ and u can form an isosceles right triangle which has two edges overlap the grid lines of the board. Similarly, we define a set V by stating that $\forall v \in V$, the triple of stone 2, $(H, 11)$ and v can form an isosceles right triangle which has two edges overlap the grid lines of the board. We then define a set $W = \{(I, 11)\} \cup \{(J, 11)\} \cup \{(H, 10)\} \cup \{(H, 11)\} \cup \{(I, 9)\} \cup \{(J, 9)\} \cup \{(H, 9)\} \cup \{(H, 13)\} \cup \{(J, 13)\} \cup U \cup V$. If y assign the number 4 white stone to a position outside the set W, the algorithm T can help the black side to win the game by the following sequence:

- The black side captures $(I, 11)$ using stone 5. After this, the black side will win if it capture $(J, 11)$.
- The white side captures $(J, 11)$ using stone 6. Otherwise, the black side can win the game by capturing $(J, 11)$.
- The black side captures $(H, 10)$ using stone 7. After this, the black side will win if it capture $(H, 11)$. Because the white stones $2, 4$ and 6 cannot form an isosceles right triangle, the white side cannot win in the next step.
- The white side captures $(H, 11)$ using stone 8. Otherwise, the black side can win the game by capturing $(H, 11)$.

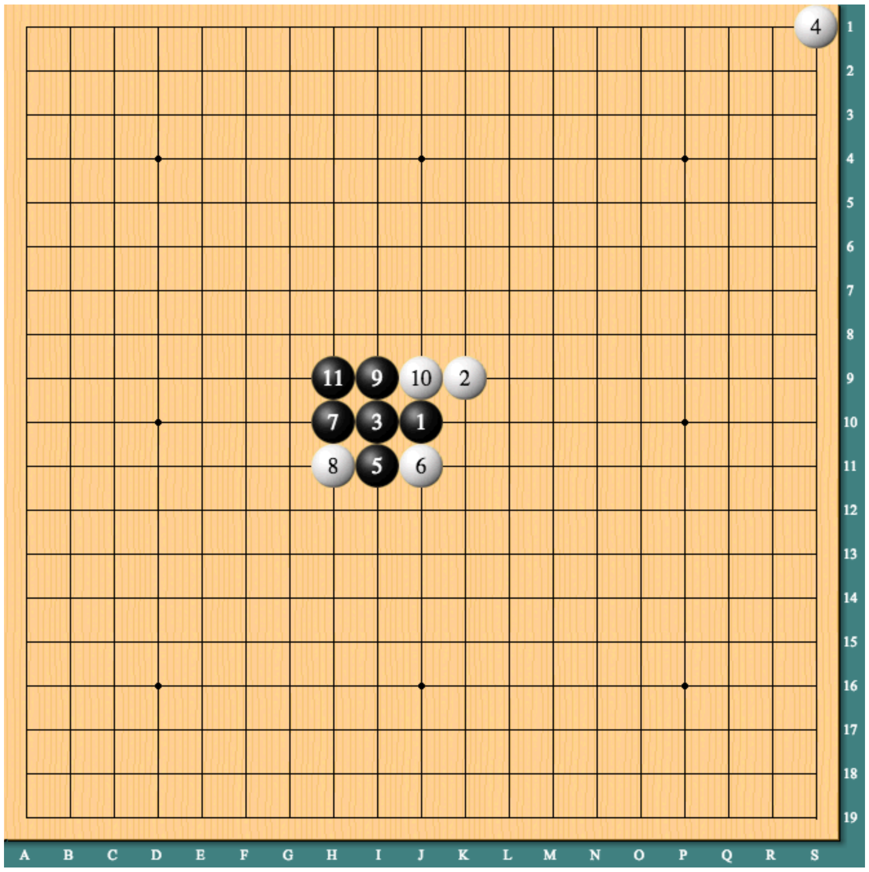

FIGURE 2. A possible situation of Square War

- The black side captures $(I, 9)$ using stone 9. After this, the black side will win if it capture $(J, 9)$ or $(H, 9)$. Because none of three white stones can form an isosceles right triangle, the white side cannot win in the next step.
- The white side captures a position using stone 10. After this, at least one of $(J, 9)$ or $(H, 9)$ is empty.
- The black side captures $(J, 9)$ or $(H, 9)$ using stone 11 and wins the game, because either stone $1, 3, 9, 11$ or $3, 7, 9, 11$ is a square to win. One case of the sequence is shown in Figure 2.

The main idea of the sequence is similar to checkmate in chess. The black side tries its best to form a square without giving chance to the white side to do so. To avoid failure, the white side has to stop the black side winning in the next step. When the black side has two different positions to form a square in the next step, the white side can only capture at most one of them. This leads to a win of the black side. However, if y assign the number 4 white stone to a position inside the set W, the black side cannot be ensured a win using the above sequence.

Hence, we integrate the algorithm T with the algorithm Z to search for a sequence that ensures a win of the black side in any possible situation of the number

Algorithm 1 Z

Input: W; black stone list S_b; black stones $1,3$; white stone list S_w; board intersection recording steps $\Phi_{i,j}, i = A, ..., S,\ j = 1, ..., 19$; number of maximal steps m;

Output: Whether a win of the black side can always be ensured using this search W_a; number of cases explored C_e; number of cases where the black side can always be ensured a win using this search C_w;

```
1:  $C_e := 0$;
2:  $C_w := 0$;
3:  for all possible white stone 2 do
4:    for $k = A, ..., S$ do
5:      for $l = 1, ..., 19$ do
6:        for all board intersection $\Phi_{i,j}, i = A, ..., S,\ j = 1, ..., 19$ do
7:          Assign $\Phi_{i,j}$ to empty;
8:        end for
9:        Initialise $\Phi$ with stones $1, 2, 3$;
10:       Initialise $S_b$ with stone 1, then add stone 3 from back;
11:       Initialise $S_w$ with stone 2;
12:       if the intersection $\Phi_{k,l}$ has a stone then
13:         Continue;
14:       end if
15:       $B_{case} := false$;
16:       if the intersection $\Phi_{k,l}$ is inside $W$ then
17:         $B_{case} := true$;
18:       end if
19:       if $B_{case}$ is $false$ then
20:         Continue;
21:       end if
22:       Create the stone 4 at $(k, l)$;
23:       Assign $\Phi_{k,l}$ to stone 4;
24:       Add stone 4 to $S_w$ from back;
25:       Create the array $S_n, n = 5, ..., m-1$;
26:       for all $S_n, n = 5, ..., m-1$ do
27:         Assign $S_n$ to 0;
28:       end for
29:       for $n = 5, ..., m-1$ do
30:         if $n$ is equal to $m-1$ then
31:           $n := n-3$;
32:           Remove the last stone from $S_b$ and assign the position of this stone to empty in $\Phi$;
33:           Remove the last stone from $S_w$ and assign the position of this stone to empty in $\Phi$;
34:           Continue;
35:         end if
```

```
36:       $B_{next} := false$;
37:       $S_n := S_n + 1$;
38:       for all $S_f, f = n+1, ..., m-1$ do
39:         Assign $S_f$ to 0;
40:       end for
41:       $P_{step} := 0$;
42:       if $n$ mod 2 is 1 then
43:         if the black side can win in this step by adding a stone to form a square with 3 existing stones then
44:           $C_w := C_w + 1$;
45:           Print $n + 1$;
46:           Break;
47:         end if
48:         for all combinations of two unique black stones $p, q$ in $S_b$ do
49:           if $p, q$ are on the same line of the board grid with a distance at most 3 then
50:             for all permutations of two empty positions $r, s$ that can form a square with $p, q$ do
51:               $P_{step} := P_{step} + 1$;
52:               if $P_{step}$ is equal to $S_n$ then
53:                 Add a stone at $r$ to $S_b$ from back and update $\Phi$;
54:                 $B_{next} := true$;
55:                 Break;
56:               end if
57:             end for
58:           end if
59:           if $B_{next}$ is $true$ then
60:             Break;
61:           end if
62:         end for
63:         if $B_{next}$ is $true$ then
64:           Continue;
65:         else
66:           $n := n - 3$;
67:           Remove the last stone from $S_b$ and assign the position of this stone to empty in $\Phi$;
68:           Remove the last stone from $S_w$ and assign the position of this stone to empty in $\Phi$;
69:           Continue;
70:         end if
71:       else
72:         if the white side can win in this step by adding a stone to form a square with 3 existing stones then
73:           $n := n - 2$;
```

```
74:                 Remove the last stone from $S_b$ and assign the position of this stone to empty in $\Phi$;
75:                 Continue;
76:             **end if**
77:             **if** the black side can win in the next step by adding a stone to form a square with 3 existing stones **then**
78:                 Use a white stone to stop the black side winning, add this stone to $S_w$ from back and update $\Phi$;
79:             **end if**
80:         **end if**
81:       **end for**
82:       $C_e := C_e + 1$;
83:     **end for**
84:   **end for**
85: **end for**
86: **if** $C_e$ is equal to $C_w$ **then**
87:   $W_a := true$;
88: **else**
89:   $W_a := false$;
90: **end if**
91: Return $W_a, C_e, C_w$;
```

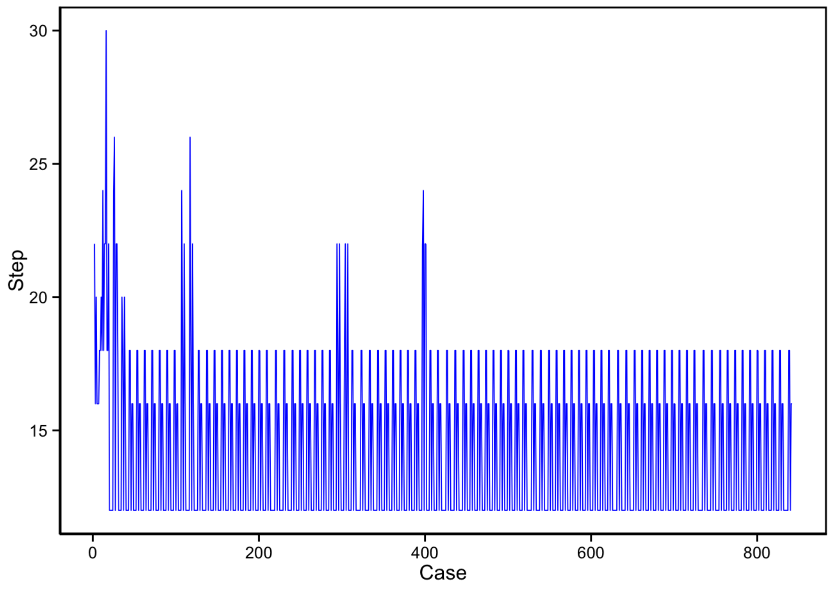

FIGURE 3. Results.

4 white stone inside the set W. The algorithm Z is a depth-first backtracking algorithm. It searches the tactics to ensure a win of the black side according to situations of the number 4 white stone inside W. Since W contains significantly fewer cases than the whole set Y, this algorithm is less complex than brute-force methods.

After the implementation using C++, we have obtained a positive answer from the algorithm Z. It has ended with 842 cases in a short time, giving all positive outputs, as shown in Figure 3. Although the numbers of steps for the black side to win are not optimized, they are at most 30.

In summary, $\forall y \in Y$, we can always find $x \in X_0$ such that $F(x, y) = 1$ using the algorithm T. This completes the proof. $\square$

References

[1] LV Alus, MPH Huntjens, et al. Go-moku solved by new search techniques. *Computational Intelligence*, 12(1):7–23, 1996.

[2] GMJB Chaslot, Jahn-Takeshi Saito, Bruno Bouzy, JWHM Uiterwijk, and H Jaap Van Den Herik. Monte-carlo strategies for computer go. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, pages 83–91. Citeseer, 2006.

[3] C-W Chou, Olivier Teytaud, and S-J Yen. Revisiting monte-carlo tree search on a normal form game: Nogo. In *Applications of Evolutionary Computation*, pages 73–82. Springer, 2011.

[4] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

[5] Sylvain Gelly, Yizao Wang, Rémi Munos, and Olivier Teytaud. Modification of uct with patterns in monte-carlo go. 2006.

[6] Bertrand Georgeot and Olivier Giraud. The game of go as a complex network. *EPL (Europhysics Letters)*, 97(6):68002, 2012.

[7] Georges Gonthier. Formal proof–the four-color theorem. *Notices of the AMS*, 55(11):1382–1393, 2008.

[8] David Silver, Richard S Sutton, and Martin Müller. Reinforcement learning of local shape in the game of go. In *IJCAI*, volume 7, pages 1053–1058, 2007.

[9] David Silver, Richard S Sutton, and Martin Müller. Temporal-difference search in computer go. *Machine learning*, 87(2):183–219, 2012.

[10] David Stern, Ralf Herbrich, and Thore Graepel. Bayesian pattern ranking for move prediction in the game of go. In *Proceedings of the 23rd international conference on Machine learning*, pages 873–880. ACM, 2006.

[11] Yuxia Sun, Cheng Liu, and Hongkun Qiu. The research on patterns and uct algorithm in nogo game. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 1178–1182. IEEE, 2013.

[12] John Tromp and Gunnar Farnebäck. Combinatorics of go. In *Computers and Games*, pages 84–99. Springer, 2007.

Department of Computer Science and Engineering, University of Oulu, Oulu, Finland

E-mail address: `chu.luo@ee.oulu.fi`