Appendix

In this Appendix, we include the following materials.

- Summarize how blocks on stones are defined to be *unconditionally alive (UCA)* for achieving safety in L&D problems by Benson (1976).
- Present zone dilation methods that expand zones to satisfy the three CR conditions and illustrate the cases where zone dilation is needed for L&D problems.
- Present a goal-achieving solver that combines RZS with depth-first search.
- Introduce the game of Slither and illustrate its zone dilation using an example.
- Provide experiment details, including training settings, the collection of L&D problems, and their run times. The L&D problems are attached in SGF format in the supplementary materials.

Unconditionally Alive

This subsection summarizes the definition of a block being unconditionally alive (UCA), as presented by Benson (1976). For a set of blocks to be UCA, each block needs to be associated with at least two *vital regions*, each of which is surrounded by blocks within that set. For a region to be considered vital for a block, all empty grids within the region must be adjacent to that block. For the position p_b in Figure 1, the (only) white block is UCA with the two vital regions at F2 and G1. UCA is also satisfied for the two white blocks in the position p_c, where the first vital region contains E2 and F2, and the other G1, with a similar situation in p_d. In contrast, UCA is not satisfied for the other four positions in Figure 1, e.g., for p_a, the white block at D2 (one single stone) has no vital regions. Intuitively, since each block is associated with at least two vital regions, and each of these regions must have at least one empty grid, each block has at least two liberties, like the cases of p_b, p_c and p_d in Figure 1. Namely, each of these blocks can maintain at least two liberties even if the opponent is allowed unlimited consecutive moves. Thus, these blocks are safe or unconditional alive.

Zone Dilation

For all replayable solution trees rooted at an internal node n (i.e. $p(n)$ is not UCA), their corresponding RZs z are derived by dilating (expanding) some zone z_u such that the resulting zone satisfies the three CR conditions so that replay is possible. Depending on whose turn it is to play at n, z_u is defined as follows:

OR-player ($\pi_\square$) to move z_u is the union of the winning move m and z', where z' is the RZ of the successor state of n after playing m.

AND-player ($\pi_\bigcirc$) to move z_u is the union of RZs of their children.

In practice, for most cases, z remains the same as z_u, so long as z_u already satisfies the CR conditions. For example, all cases in Figures 1 and 4 do not require additional dilation. Let z be initialized to z_u. Dilation needs to be performed to satisfy the CR conditions mainly in cases where suicide moves are prohibited and blocks are captured on the border of the RZ z, for the game of Go. Let us illustrate by the following three examples. The first is a dilation case for $\pi_\square$ (White) with capturing blocks in Figure 5, the second and the third are cases for $\pi_\bigcirc$ (Black) to play with capturing and suicides as in Figures 6 and 7.

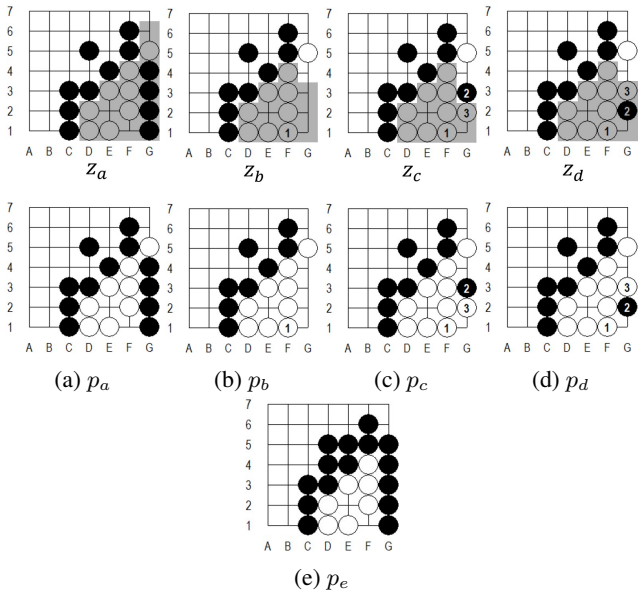


Figure 5: A dilation case for $\pi_\square$ (White) with capturing blocks. For the position p_a in (a), White is to play and wins as follows. White plays at F1 to capture four black stones, leading to the position p_b. Then, as in (c), for Black's move at G3, White replies G2 to achieve UCA and obtains an RZ z_c as shaded. Since G3 is outside of the RZ, it is a null move, which reduces the must-play region to G1 and G2 only. It is trivial for White to reply at G2 for Black's move at G1. For Black's move at G2, White replies at G3 to achieve UCA and obtains an RZ z_d as shaded in (d). Thus, we conclude that White wins for the position p_b (Black to play). The union of RZs of the children of p_b is z_b, as shaded in (b). This union satisfies the CR conditions, and thus is a valid RZ. Now, let us examine p_a with White to play. Let z_u be the union of F1 (the move that was searched) and z_b, which is still z_b. Unfortunately, in this case, z_u can not serve as an RZ for p_a. As a counter-example, consider p_e, which shares the same pattern within z_u as p_a. If z_u is an RZ for p_a, for any position that shares the same pattern, say, p_e, White should be able to replay their winning strategy at F1. This is clearly not the case in p_e. In order to replay the winning strategy (guaranteed by satisfying all the CR conditions), we need to dilate z_u to z_a.

To formulate a heuristic that dilates RZs while satisfying the CR conditions, let us review the following rules for Go. [4]

GR-1 For all positions, all blocks must have at least one liberty, and a block without any liberties is removed from

[4] Note that we don't consider ko and SSK in this paper.

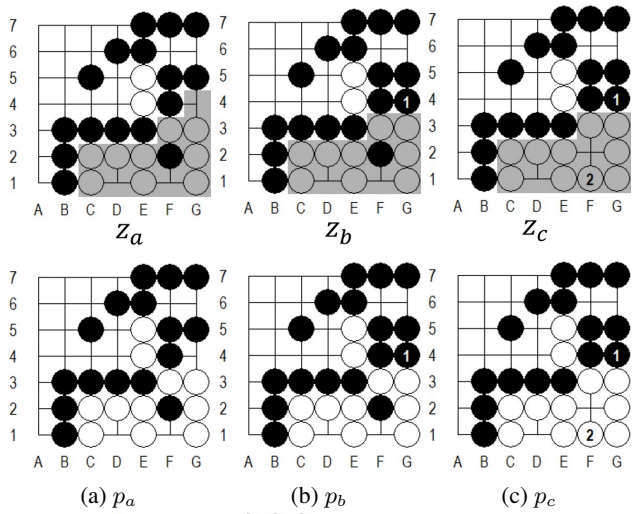


Figure 6: A dilation case for $\pi_{\bigcirc}$ (Black) to play. For p_a, where Black is to play, let Black play at G4 leading to p_b, and White reply at F1 leading to p_c. Thus, p_c achieves UCA, and the shaded grids in (c) is its RZ z_c. Then, for p_b where White is to play and win at F1, F1 is replayable in the same zone, so its RZ z_b is still z_c. Since Black's move at G4 is outside of $z_b (= z_c)$, it is a null move and thus all moves outside z_b are pruned. This leads to a win for White in p_a due to no more Black moves. To derive the RZ of p_a, we first consider the union of the RZs for all Black moves (actually just one Black move at G4 here), still the same as z_b. Unfortunately, z_b cannot serve as an RZ for p_a yet, since in p_d, which shares the same zone pattern, Black can directly capture the four White stones on the right by playing at F1, leading to p_e. The RZ therefore needs to be dilated from z_b to z_a with one extra grid at G4.

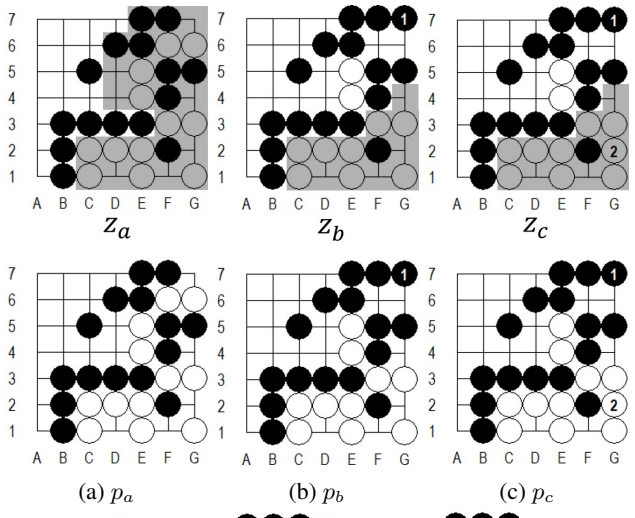


Figure 7: A dilation case for $\pi_{\bigcirc}$ (Black) with suicide. In the position p_a in (a), Black cannot play at G4 since suicide is prohibited. Instead, Black tries G7 to capture two white stones as in (b). White needs to respond at G2 to win in p_c. Since p_c is identical to that in Figure 6 (a), p_c wins with the same zone, z_c here. For p_b where it is White to play and win at G2, G2 is replayable in the same zone, so its RZ z_b is still z_c. Since Black's move at G7 is outside of $z_b (= z_c)$, all moves outside z_b are pruned, leading to those moves inside z_b for Black to move in (a). In (a), since G4 and D1 are illegal, Black can only play F1 and G2, both of which Black can win with the same RZs (we omit the details for simplicity). Consequently, the union of the RZs for all Black moves is still the same as z_b. Unfortunately, z_b cannot serve as an RZ for p_a, since there exists some position p_d with the same zone pattern, except in this case Black can play at G4 to kill white stones as in (e). So the zone needs to be dilated to z_a as in (a).

the board.

Placing a stone on an unoccupied grid g will lead to three kinds of situations.

GR-2 One or more opponent's blocks without any more liberties are captured. This is a legal move.

GR-3 No opponent blocks are captured, and one of the player's own blocks has no more liberties. This is a suicide, which is illegal in Go.

GR-4 All blocks have at least one liberty. This is a legal move.

Now, we give some new definitions related to z. Inside a zone z, a grid is called a z-grid, and a block is called a z-block. For a z-block, its liberties located inside z are called z-liberties of the block. A z-border is the set of z-grids which are adjacent to any of grids of $\bar{z}$. For example, in Figure 1 (a), the z-border includes D1, D2, E3, F3 and G3, and D1 and E2 are the z-liberties of the white z-block at D2.

In order to let z satisfy the three CR conditions on $p(n)$, we propose some rules for dilation from a zone z. The first is DL-1 below, used to ensure that all p_* with the same zone pattern, i.e., $\beta(p_*) \odot z = \beta(p(n)) \odot z$, are legal at least inside the zone.

DL-1 All z-blocks (both white and black) must have at least one z-liberty. Assume that a z-block b has no z-liberty.

Add one of b's liberty into z.

Internal OR-Nodes Now, suppose that $\pi_\square$ is to play and win at move m. In this case, the dilation only needs to satisfy CR-3. First, assume that $\pi_\square$ wins without capturing any blocks. Then, simply apply DL-1 to z. It is trivial to see the CR-3 condition satisfied in this case.

Second, assume that $\pi_\square$ wins by capturing one of $\pi_\bigcirc$'s block b. Then, add into z the captured block b and all $\pi_\square$'s blocks surrounding b, and subsequently apply DL-1 to the resulting zone. For example, in Figure 5, the captured black block (with four stones) and the white block at G5 surrounding the black block are included in z_a. In addition, DL-1 is then used to add the unoccupied grid at G6 into the RZ.

Internal AND-Nodes Now, suppose that $\pi_\bigcirc$ is to play and $\pi_\square$ still wins on $p(n)$. In this case, we need to ensure both CR-1 and CR-2 conditions are satisfied. Let p_* have the same zone pattern on z as $p(n)$.

For CR-1, all Black moves inside z of p_* must be legal for $p(n)$ with the same zone pattern. Equivalently, the following condition holds: For all illegal moves m on the RZ z in $p(n)$, m are illegal in p_*. For the game of Go, since suicides are illegal, let us consider all suicides on g inside the zone z of $p(n)$. Let g be adjacent to $\pi_\bigcirc$'s block b, if any. To ensure that g is also a suicide in p_*, add b and all $\pi_\square$'s blocks surrounding b and g into z. For example, in Figure 7 (a), Black move at G4 has the black block at (F4, F5, G5) have no liberty, so it is a suicide. In this case, this Black block and two extra surrounding white blocks at (E4, E5) and (F6, G6) are added into z.

For CR-2, all $\pi_\bigcirc$'s moves outside z of p_* cannot change the zone pattern of $p(n)$. The only way to change the zone inside is to capture white z-blocks b on the z-border. As GR-1, b has at least one liberty in $p(n)$. So, let us consider the following two cases. In the case that b has at least two liberties, add two of them into the zone z. For example, in Figure 7 (a), two liberties at D4 and D5 are added after the white block at (E4, E5) is added. Another example is in Figure 6 (a), where the two liberties of the right white block (with four stones) need to be added. In the case that b has one and only one liberty, we add all the black blocks surrounding b. For example, in Figure 7 (a), after the white block at (F6, G6) is added, the only liberty of the block is at G7 which is in turn added and the surrounding black block at F7 should be added.

Finally, we use DL-1 to dilate z. The above process is repeated until all are satisfied.

RZS Solver in Depth-First Search

In this subsection, we present a RZS solver implemented within a depth-first search algorithm, which we call AchieveGoal. It is similar to the must-play-based Hex solver by Hayward (2009). The algorithm AchieveGoal has inputs (p, G), where p is the given position and G the goal to achieve. G is global and does not change during the process. The output is (v, z), where v is either *win* if the goal is achieved, or *fail* otherwise. z is the RZ, if p is a win. A heuristic function $selectmove(p, M)$ is used to choose a move from a set of moves M on p. When the algorithm returns a win, the function $dilate(z)$ also dilates the returning RZ to ensure that the three CR conditions are satisfied for all winning nodes. From Lemma 1, the goal G can be achieved for position p with an RZ z, if AchieveGoal(p, G) returns (win, z).

Algorithm 1: AchieveGoal (p, G)

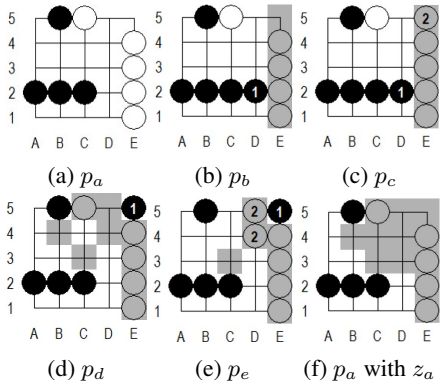
```
1:  if Achieve(p, G) then                          ▷ Immediately achieves G
2:      return (win, z), where z is the RZ of p.
3:  if Fail(p, G) then                             ▷ Fails to achieve G
4:      return (fail, ∅)
5:  if π(p) is π□ then                             ▷ Internal OR-node
6:      while M: more legal moves not searched yet do
7:          (m, p') ← nextmove(p,M)                ▷ p': next position
8:          (v, z') ← AchieveGoal(p', G)
9:          if v is win then return (win, dilate(z' ∪ {m}))
10:     return (fail, ∅)
11: else                                           ▷ Internal AND-node
12:     z ← ∅
13:     M ← all legal moves in p                   ▷ must-play region
14:     while more legal moves in M do
15:         (m, p') ← nextmove(p,M)
16:         (v, z') ← AchieveGoal(p', G)
17:         if v is win then
18:             z ← z ∪ z'
19:             if m ∈ z' then M ← M \ {m}
20:             else M ← M ∩ z'                    ▷ m is a null move
21:         else
22:             return (fail, ∅)
23:     return (win, dilate(z))                    ▷ M is empty
```

RZS for the Game of Slither

This subsection demonstrates that our RZS can even be applied to games where the stones move on the board as the game progresses.

Slither is also a connection game play on an $n \times n$ square board, in which stones can be placed on unoccupied grids and moved. It has been shown that the game is PSPACE-complete and cannot end in a draw (Bonnet, Jamain, and Saffidine 2015). Like the game of Hex, the first player who connects its own sides (e.g., up/down for White) wins. A move in Slither is composed of two phases. The first is an optional relocation which allows the player to shift an existing stone of their color to an adjacent or diagonal empty grid. The second is to place a stone on another empty grid. The resulting position cannot have any two diagonal grids occupied by the player's stones which do not have an orthogonally-adjacent stone, as illustrated as follows. For example, it is illegal for White to add a single stone at D5 into the position in Figure 8 (a), due to the diagonal (D5, E4). However, the diagonal (D5, E4) is legal in (e) since D4 connects to both via a direct adjacency. This restriction is called the diagonal rule (Bonnet, Jamain, and Saffidine 2015). Figure 8 (a) illustrates an winning case for White, and the zone dilation for the RZ. More zone dilation methods for Slither

are beyond the scope of this paper.



(a) p_a (b) p_b (c) p_c

(d) p_d (e) p_e (f) p_a with z_a

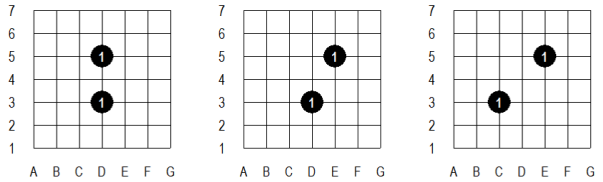
Figure 8: The position p_a in (a) is for Black to play first and a win for White. If Black plays at D2 as in (b), then White simply plays at E5 to connect the upper and lower sides to win with the RZ z_c, shaded in (c). Since Black D2 is outside the RZ in (b), which is also z_c, it is a null move, and thus the must-play region is reduced to E5. Therefore, Black must play at E5 (no matter how Black moves stones in the first phase) to prevent White from winning directly as in (d). For Black E5, White wins by moving the stone at C5 to D5, then placing a stone at D4 to win, leading to a win in (e). Thus, position p_a wins. The RZ for the position in (e) is dilated to include C3 due to the diagonal rule. The RZ in (d) is dilated to include B4 (due to the diagonal rule with C5) and C5 itself. The RZ for (a) is derived as shaded in (f) as follows. Take the union of the above two RZs in both (b) and (d). Then dilate it by including C3, C4, C5, and D3, since Black can move any one of them to D4 and D5 to prevent White's winning move as in (e).

Experiment Details

The following lists the training settings for two AlphaZero programs with and without the FTL method in 7x7 kill-all Go. The network architecture consists of 5 residual blocks and 64 filters. We generate a total of 2,500,000 self-play games in training with 400 MCTS simulations for each move. The learning rate is fixed to 0.02 for the first 1,500,000 self-play games, then reduced to 0.002 for the remainder; the komi is set to 48 during training.

We select 20 L&D problems for 7x7 kill-all Go from three different Black openings: the one-point jump opening, knight's move opening, and diagonal jump opening, as shown in Figure 9 (a), (b), and (c) respectively. Figure 11 lists these 20 problems in three groups, 12 from the one-point jump opening, 6 from the knight's move opening, and 2 from the diagonal jump opening. In each group, problems are sorted from simple to complex, in terms of simulation count for RZS with FTL. Particularly, the two problems 11 and 18, as in Figure 11 (k) and (r) respectively, are commonly played as joseki (openings) in 19x19.

Table 2 lists the simulation counts (or node counts) required to solve each problem for all methods. In the table,



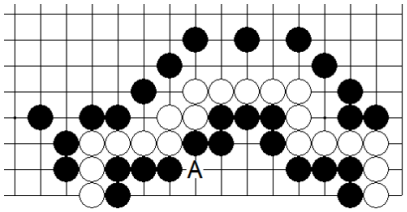
(a) One-point jump. (b) Knight's move. (c) Diagonal jump.

Figure 9: Openings for 7x7 kill-all Go.

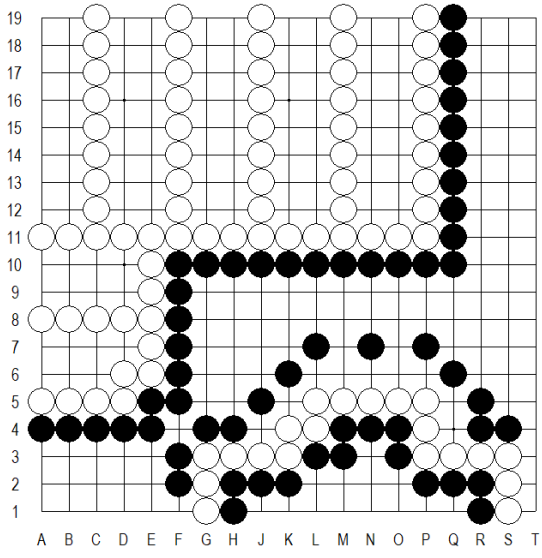
dashes indicate that the problem cannot be solved within the given computation budget.

Table 3 lists the experiment results for 19x19 Go L&D problems, where the columns of volume and page number indicate the problem corresponding to the problem in that page of the volume of the book (Cho 1987). To satisfy the requirements of each solver, we modified the problems by filling some stones. One of the modified L&D problems is shown in Figure 10 (b), where the original problem is shown in Figure 10 (a).

The above L&D problems are attached in SGF format in the supplementary material (in the directory "tsumego").

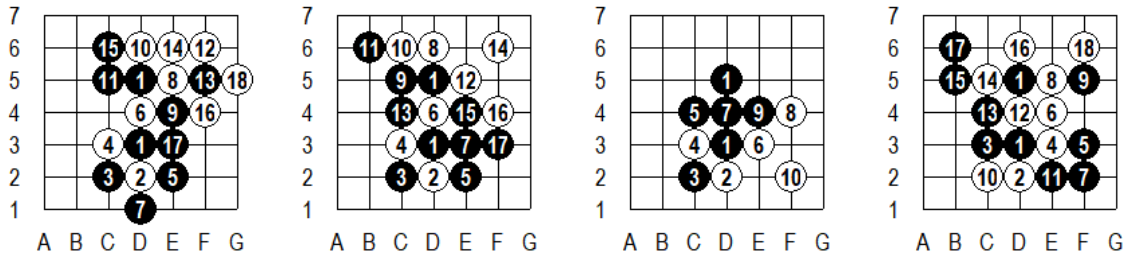


(a) Original L&D problem.

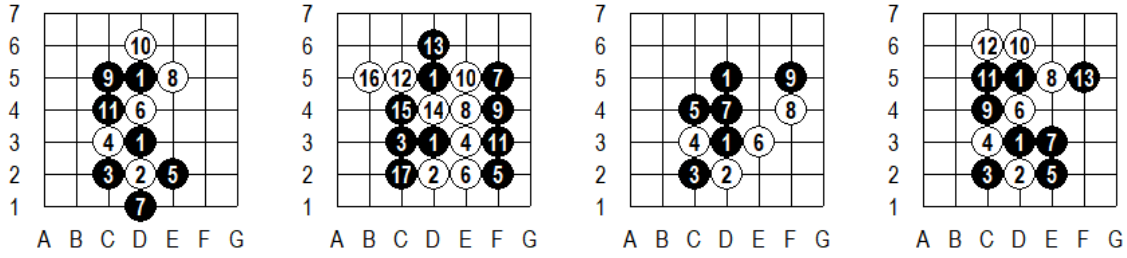


(b) A modified 19x19 L&D problem from (a).

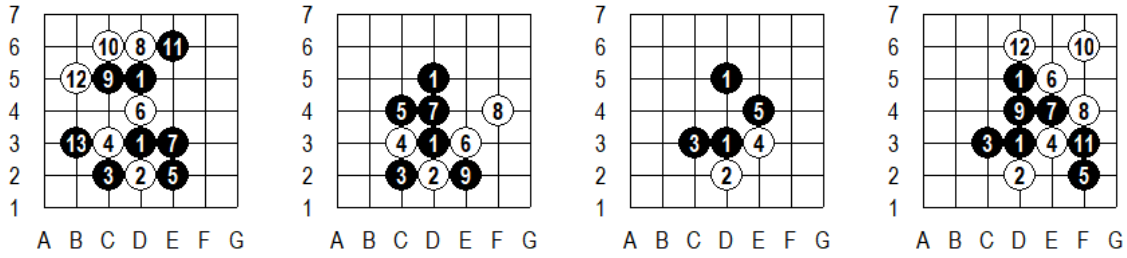
Figure 10: A 19x19 L&D problem from Cho's book, volume 2, page 139. The problem is White to play for life, and the correct answer is to play at A.



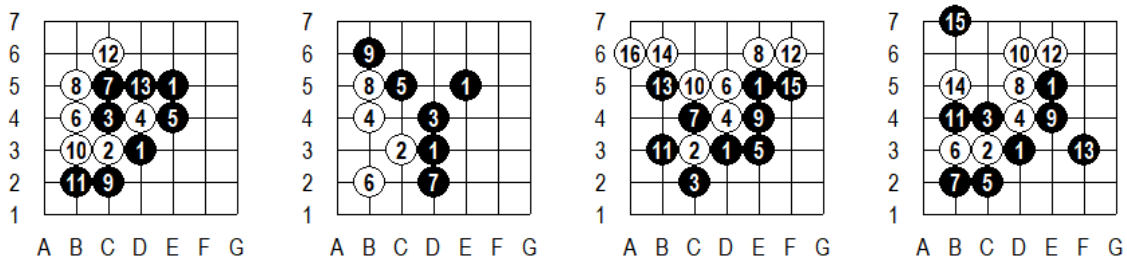
(a) 1. Black's turn. (b) 2. White's turn. (c) 3. Black's turn. (d) 4. Black's turn.



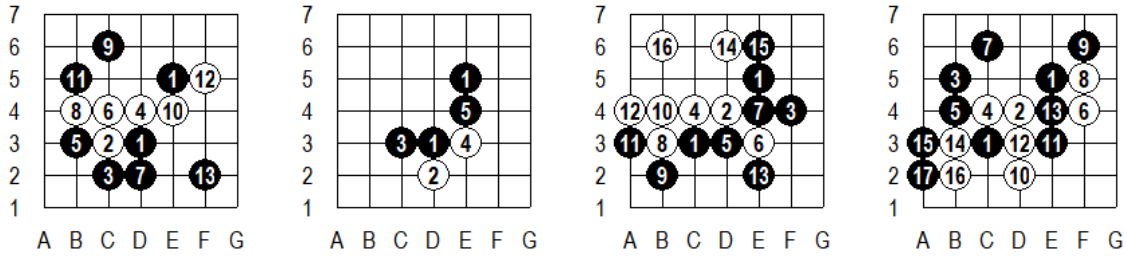
(e) 5. White's turn. (f) 6. White's turn. (g) 7. White's turn. (h) 8. White's turn.



(i) 9. White's turn. (j) 10. White's turn. (k) 11. White's turn. (l) 12. Black's turn.



(m) 13. White's turn. (n) 14. White's turn. (o) 15. Black's turn. (p) 16. White's turn.



(q) 17. White's turn. (r) 18. White's turn. (s) 19. Black's turn. (t) 20. White's turn.

Figure 11: 20 7x7 kill-all tsumego problems.

ID	No-RZS	RZS	FTL(w/o RZS)	FTL(w/ RZS)
1	-	255,314	76,780	**325**
2	-	204,355	428,239	**4,958**
3	-	-	-	**217,461**
4	-	-	-	**106,430**
5	-	-	-	**204,691**
6	-	-	-	**177,231**
7	-	-	-	**173,056**
8	-	-	-	**164,633**
9	-	-	-	**180,901**
10	-	-	-	**335,878**
11	-	-	-	**449,791**
12	-	-	-	**365,809**
13	-	23,998	127,910	**1,129**
14	-	220,988	-	**915**
15	-	245,360	-	**2,402**
16	-	-	-	**18,279**
17	-	-	-	**319,276**
18	-	-	-	**423,564**
19	-	-	-	**7,067**
20	-	-	-	**13,137**

Table 2: Node counts on solving 7x7 kill-all Go problems. The bold numbers show the lowest number of search nodes to prove the problem among all methods.

ID	Volume	Page No.	ELF(w/o RZS)	ELF(w/ RZS)	FTL(w/o RZS)	FTL(w/ RZS)	T-EXP
1	1	31	-	12,804	42,472	**2,596**	-
2	1	37	-	**32,518**	-	38,275	-
3	1	40	-	40,002	-	**9,375**	-
4	1	42	-	8,419	32,980	**416**	1,037
5	1	46	-	**2,434**	161,040	65,687	1,036,877
6	1	57	-	**3,499**	-	3,878	-
7	1	60	-	-	-	**27,974**	-
8	1	68	-	-	-	**1,960**	-
9	1	74	-	2,911	-	**1,547**	1,158,037
10	1	80	-	-	-	**39,758**	-
11	1	84	-	-	-	**11,422**	-
12	1	88	-	-	-	-	-
13	1	90	-	-	-	-	-
14	1	98	-	-	-	-	-
15	1	101	-	**17,165**	-	27,103	-
16	1	104	-	33,177	-	**6,071**	-
17	1	132	-	-	-	**65,713**	-
18	1	135	-	-	-	**212,209**	-
19	1	139	-	-	-	**15,964**	-
20	1	152	-	-	-	-	-
21	1	156	-	6,540	-	**2,031**	-
22	1	174	-	-	-	**30,897**	-
23	1	176	-	-	-	**26,562**	-
24	1	186	-	-	-	-	-
25	1	189	-	50,399	-	**25,147**	-
26	1	195	-	8,698	-	**7,055**	-
27	1	196	-	-	-	**23,738**	648,529
28	1	197	-	-	-	**44,385**	-
29	1	198	-	**8,080**	-	11,921	-
30	1	222	-	-	-	-	-
31	1	243	-	**776**	33,768	2,108	2,811
32	1	244	-	**8,223**	-	48,851	87,168
33	1	252	-	**4,053**	-	5,171	-
34	1	258	-	4,657	37,219	**434**	556
35	1	262	-	-	89,912	**739**	1,040
36	1	264	-	16,178	-	**6,062**	-
37	1	268	-	**5,466**	-	16,438	-
38	1	269	-	**3,693**	-	5,625	-
39	1	270	-	**4,956**	-	22,998	-
40	1	272	-	**1,539**	-	22,058	131,050
41	1	279	-	**8,463**	-	13,240	345,351
42	1	288	-	-	-	**35,664**	-
43	1	298	-	-	-	**10,804**	-
44	1	299	-	45,906	-	**32,555**	-
45	1	300	-	**25,915**	-	46,271	-
46	1	301	-	-	-	**18,811**	-
47	1	302	-	-	-	-	-
48	1	318	-	-	-	**190,704**	-
49	1	323	-	-	-	**12,544**	-
50	1	324	-	34,065	-	**26,933**	-

ID	Volume	Page No.	ELF(w/o RZ)	ELF(w/ RZ)	FTL(w/o RZ)	FTL(w/ RZ)	T-EXP
51	2	120	-	75,016	-	**28,647**	-
52	2	124	-	-	-	-	-
53	2	125	-	-	-	**180,507**	-
54	2	126	-	-	-	-	-
55	2	127	-	-	-	**58,900**	-
56	2	129	-	-	-	-	-
57	2	132	-	-	-	-	-
58	2	133	-	7,424	-	**5,287**	-
59	2	138	-	-	-	-	-
60	2	139	-	-	-	**195,445**	-
61	2	142	-	-	-	-	-
62	2	144	-	-	-	-	-
63	2	146	-	-	-	-	-
64	2	147	-	-	-	**251,136**	-
65	2	152	-	-	-	-	-
66	2	156	-	-	-	**59,898**	-
67	2	158	-	-	-	-	-
68	2	162	-	**95,163**	-	-	-
69	2	164	-	-	-	-	-
70	2	166	-	-	-	-	-
71	2	245	-	-	-	-	-
72	2	246	-	-	-	**5,448**	-
73	2	248	-	-	-	-	-
74	2	252	-	90,000	-	**7,483**	-
75	2	253	-	-	-	-	-
76	2	254	-	-	-	-	-
77	2	257	-	-	-	-	-
78	2	260	-	-	-	-	-
79	2	262	-	-	-	-	-
80	2	274	-	-	-	-	-
81	2	326	-	-	-	-	-
82	2	328	-	-	-	**43,247**	-
83	2	329	-	13,668	-	**2,009**	-
84	2	330	-	41,389	-	**22,957**	-
85	2	331	-	-	-	-	-
86	2	332	-	-	-	-	-
87	2	333	-	33,092	-	**9,701**	-
88	2	334	-	-	-	-	-
89	2	336	-	-	-	**15,996**	-
90	2	337	-	12,582	-	**10,544**	-
91	2	338	-	-	-	**33,088**	-
92	2	340	-	-	-	-	-
93	2	341	-	-	-	**16,056**	-
94	2	342	-	-	-	**173,118**	-
95	2	343	-	103,057	-	**12,210**	-
96	2	344	-	-	-	**37,943**	-
97	2	345	-	**7,424**	-	27,276	8,385
98	2	351	-	-	-	-	-
99	2	352	-	-	-	**23,855**	-
100	2	353	-	-	-	-	-

ID	Volume	Page No.	ELF(w/o RZ)	ELF(w/ RZ)	FTL(w/o RZ)	FTL(w/ RZ)	T-EXP
101	2	355	-	-	-	-	-
102	2	359	-	-	-	-	-
103	2	363	-	-	-	-	-
104	2	364	-	-	-	**17,458**	-
105	2	366	-	-	-	**59,295**	-
106	2	372	-	-	-	**72,562**	-

Table 3: Node counts on solving 19x19 L&D problems. The bold numbers indicate the lowest number of search nodes to prove the problem among all methods.