Ladders are PSPACE-Complete

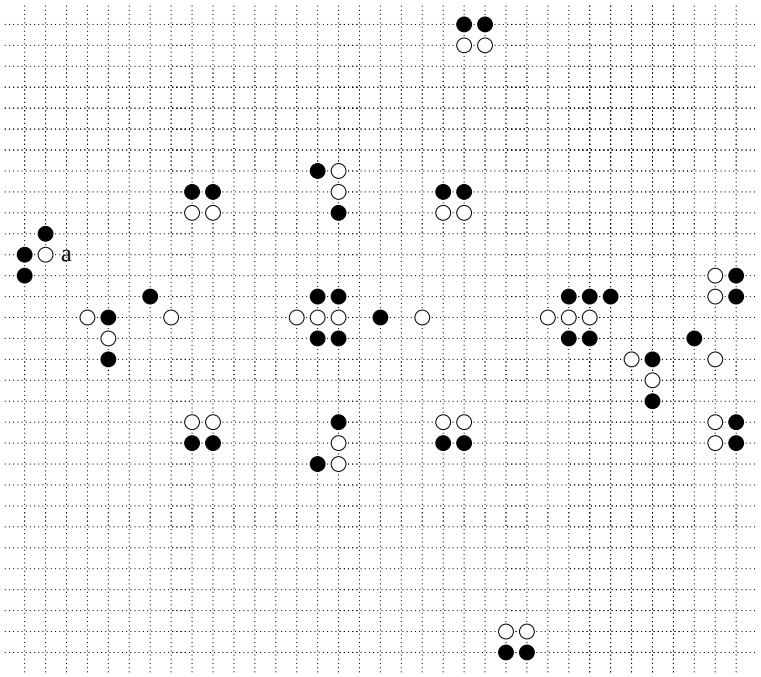
Marcel Crâşmaru[1] and John Tromp[2]

[1] Department of Mathematical and Computing Science,
Tokyo Institute of Technology,
2-12-1 Oo-okayama, Meguro-ku, Tokyo, Japan, 152,
marcel@is.titech.ac.jp
[2] CWI
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
tromp@cwi.nl

Abstract. In the game of Go, the question of whether a ladder—a method of capturing stones—works, is shown to be PSPACE-complete. Our reduction closely follows that of Lichtenstein and Sipser [LS80], who first showed PSPACE-hardness of Go by letting the outcome of a game depend on the capture of a large group of stones. We achieve greater simplicity by avoiding the need for pipes and crossovers.

1 Introduction

Consider the following Go[1] problem: Black to capture the marked white stone



[1] details of the rules may be found at http://www.cwi.nl/~tromp/go.html

We will show how this position encodes the Quantified Boolean Formula (**QBF**) $\forall x \exists y (x \vee y) \wedge (\neg x \vee \neg y)$. Since this formula is true, the ladder should work, as the reader may verify.

We formalize ladders in Go as follows:
LADDERS: Given a position on an arbitrarily-sized Go board, and a white group with 2 liberties, can Black keep putting white in atari—that is, reduce white to 1 liberty—until capture?

Theorem 1 LADDERS *is PSPACE-complete.*

The game of Go can be formalized as:
GO: Given a position on an arbitrarily-sized Go board, does Black have a winning strategy?

As shown by Lichtenstein and Sipser [LS80], one can construct positions in which black victory hinges upon the survival of a very large eyeless white group, that Black has almost entirely surrounded. To survive, it needs to connect to a 2-eyed group through a structure of pipes and junctions that can be modeled after a Quantified Boolean Formula. This proved **GO** to be PSPACE hard. Robson [R83] used the same idea but introduced a collection of *ko*'s into the structure, so that the large group could connect out only if its owner held an appropriate subset of all the ko's. Such ko-games were shown to be EXPTIME-complete. But even though the owner of the large group might not be able to obtain an appropriate subset of ko's, he might be able to keep cycling through the ko's, so that the outcome of the game depends on the exact rule dealing with whole-board-repetition. Robson assumed a null-ruling so he established EXPTIME-completeness of the question whether an arbitrary position is a forced win for Black or a null result. An interesting open question about Robson's construction is what side would first have to yield with the superko rule that simply forbids the whole-board position from repeating.

Both constructions employ *pipes*, a pipe being a line of white stones sandwiched between 2 lines of black stones. Pipes are essential in containing the flow of play between *junctions* and *joins*. A disadvantage of pipes is that they take up space, and thus cannot simply cross on a Go board. Both Robson (directly), and Lichtenstein and Sipser (indirectly; at the conceptually higher level of graphs), constructed ingenious but somewhat complicated pipe-intersections. How much easier it would be to model play not as a flow to be contained but as light that travels unaided through empty space; bent by mirrors where need be.

2 Enter the ladder

One of the first aspects of the game that beginners familiarize themselves with, the ladder (Figure 2) is a straightforward method of capturing stones by repeated atari on alternate sides. As shown in diagram L2, the ladder travels diagonally across the board and its fate will depend on what meets its path. For a ladder to *work*, i.e. result in capture, it must either hit the edge of the board, or an existing black stone, as in diagram W2. It fails if it hits or borders on a white

stone, as in diagram F2. In that case White's move at 12 puts the black stone at 9 in atari, and if Black persists at 13, White captures her way to freedom.

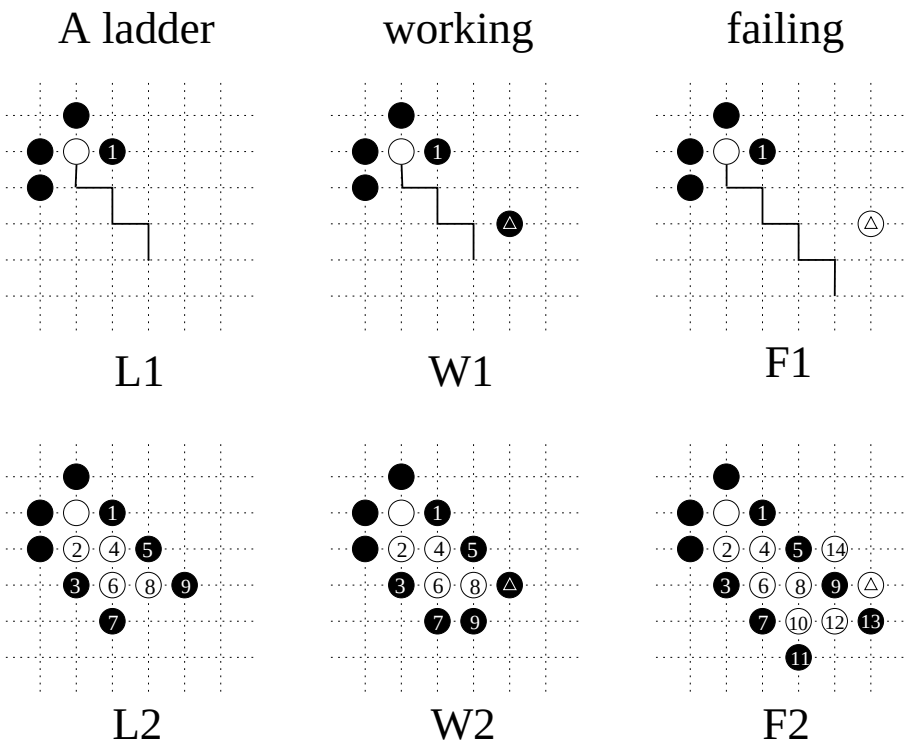


Fig. 1. various ladders

Ladders are forced sequences that can run all across the board, causing plays in one area of the board to affect other, remote areas. Ladders are also ubiquitous in Go; they come up many times per game, if not in actual play then at least in the variations that a player considers to decide on his next move.

We show how ladders can take the place of pipes in constructing hard capture problems.

3 Of Forks, Joins, and Mirrors

Our introductory ladder problem features the four different *gadgets* listed in Figure 3: the black choice (B), the white choice (W), the join (J) and the mirror (M) (for conciseness a black choice is partially merged with the join to its right in the centre of the problem).

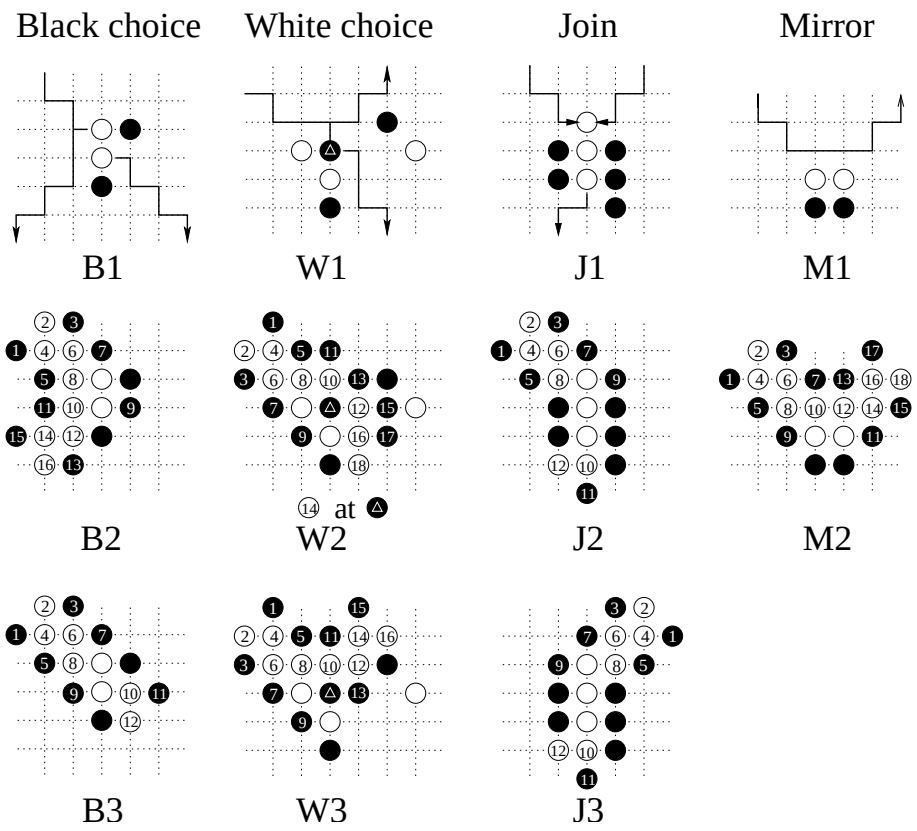


Fig. 2. ladder gadgets

In diagram B1, we see a Black choice gadget with a projected ladder approaching from the top left. When Black plays the ladder, he'll have a choice of playing move 9 on the right or the left of White, leading respectively to diagram B2 or B3. From Black's viewpoint, the top-left ladder works if either the bottom-left ladder in B2, or bottom-right ladder in B3, work.

In diagram W1, we see a White choice gadget with a projected ladder approaching from the top left. When Black plays the ladder, white's move 10 puts the marked black stone in atari, and Black must play above it to prevent White from getting too many liberties. Now White can choose to either capture the marked stone, or extend to the right, leading respectively to diagram W2 or W3. Black's moves 15 and 17 in diagram W2 are needed to route the ladder around the rightmost white stone, which would otherwise interfere. ¿From Black's viewpoint, the top-left ladder works if both the right-down ladder in W2, and right-up ladder in W3, work.

In diagram J1, we see a Join gadget with projected ladders approaching from the top left and top right. Diagram J2 shows what happens with a ladder from the top left. The forced sequence ends with the ladder continuing to the bottom left. Diagram J3 shows the symmetrical case of a ladder from the top right. ¿From Black's viewpoint, either top ladder works if the bottom-left ladder does.

Finally, in diagram M1, we see a Mirror gadget with a projected ladder approaching from the top left. When Black plays the ladder, he is forced to send it back up with move 11. Mirrors allow us to direct ladders from one gadget to the next.

3.1 Problem Analysis

Figure 3 shows a line of play in our original problem.

This line of play is entirely forced except for White's choice of playing 'a' and Black's choice of playing 'b'. If we let boolean variable x represent whether White chose to send the ladder up, and let y represent whether Black chose to send the ladder up, then the current line of play corresponds to setting $(x, y) = (\text{true}, \text{false})$. In general we have for each variable a choice gadget, an upper and lower mirror, and a join gadget, positioned at the corners of an imaginary diamond shape. The setting of the variable determines which (upper or lower) edges of the diamond get covered and which get exposed.

Now consider the top black choice gadget. If play arrives here and the ladder leaves to the bottom-left (T), then it works if and only if x is true. If it leaves to the bottom-right, then it works if and only if y is true. It follows that the ladder going up from White's choice at 'u'—which after bouncing off 2 mirrors enters the top black choice gadget—works if $x \vee y$ holds. Similarly, the ladder going down from White's choice at 'd' works if $\neg x \vee \neg y$ holds. Hence, after both variables have been set, the ladder works if $(x \vee y) \wedge (\neg x \vee \neg y)$. This shows that our original problem indeed encodes the truth of the formula $\forall x \exists y (x \vee y) \wedge (\neg x \vee \neg y)$,

To prove Theorem 1, i.e. PSPACE-completeness, we must show two things: first, that **LADDERS** belongs to PSPACE, and second, that **QBF** (known to be PSPACE-complete) reduces to **LADDERS**.

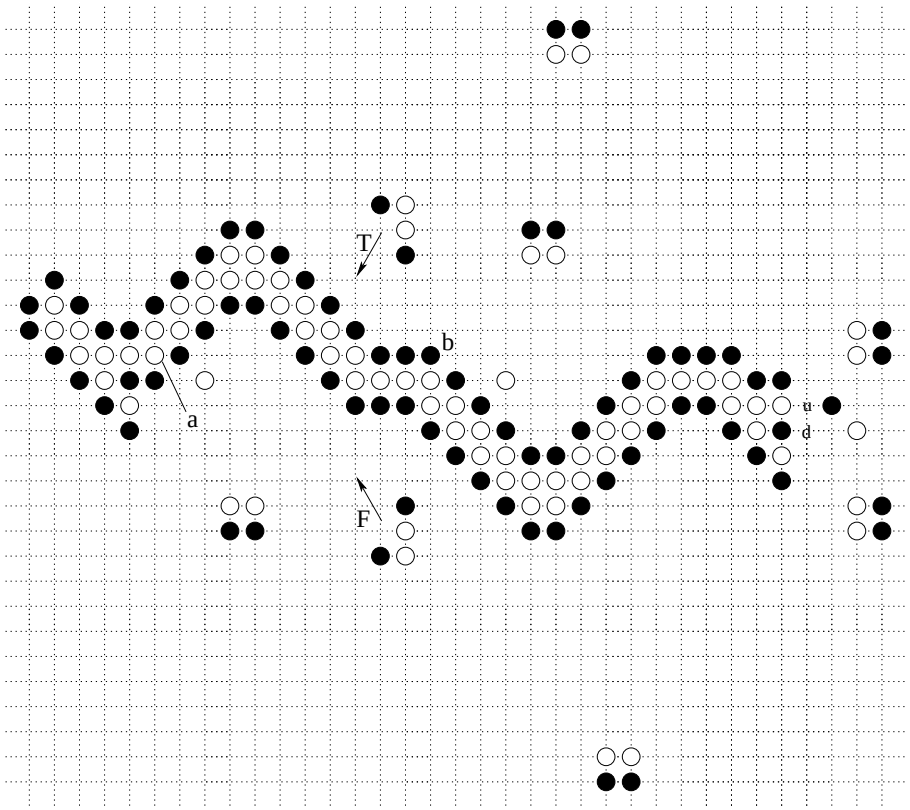


Fig. 3. a forced line apart from choices ‘a’ and ‘b’

3.2 LADDERS ∈ PSPACE

Membership in PSPACE would follow if capturability can be determined by a polynomial-depth-limited search. If the white group grows larger every c moves, for some constant c, then since it cannot grow larger than the whole board, a linear search depth would suffice. The only way for the white group to stop growing for many moves is for White and Black to capture single stones back and forth, a situation known as ko. The rules of go forbid taking back immediately in a ko, since this recreates the position of 2 moves back. The stronger "superko" rule forbids repetition of *any* earlier position, but this rule is not universally accepted as opposed to the *basic* ko rule above.

Now, if there are at least 4 kos adjacent to the ladder, then White, in atari, has at least 3 choices of where to capture, while Black has only 2 choices of capture. Under the basic ko rule, this allows both players to cycle forever, while the superko rule forbids Black first. In both cases the search ends in failure for Black.

With at most 3 kos, there are at most 6 configurations (001, 010, 011, 100, 101, 110, according to what player holds which kos). Hence if the players are allowed to capture for 6 turns, then a repetition must have occured and again the search ends in failure for Black. Figure 3.2 shows an example where the ladder runs into a such a "triple ko". With superko, White will be forbidden to cycle and the ladder works, but without superko, it will cycle forever and Black fails. In conclusion, if the search is pruned when the possibility of cycling arises, then at least once every 6 moves, White must get out of atari by adding a stone to his group, and this limits search depth to 6 times the boardsize, showing that **LADDERS** is in PSPACE.

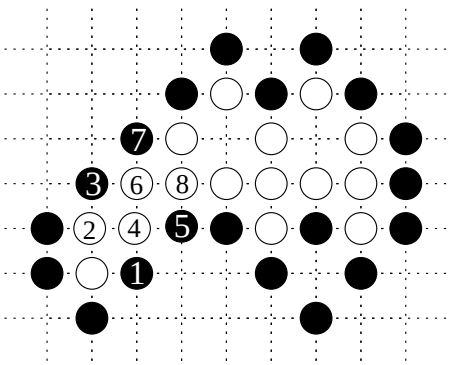


Fig. 4. A ladder depending on a triple ko

3.3 QBF reduces to LADDERS

Consider the standard PSPACE-complete problem
QBF: Given a quantified boolean formula $F = Q_1x_1Q_2x_2 \ldots Q_nx_nE$, where E is a Boolean expression involving $x_1, \ldots, x_n$ and each Q_i is either "$\forall$" or "$\exists$", determine if F is true.

We show how to reduce **QBF** to **LADDERS** by way of the example

$$\exists x \forall y \exists z (\neg x \wedge \neg y) \vee (\neg x \wedge y) \vee (\neg z \wedge (\neg x \vee z)).$$

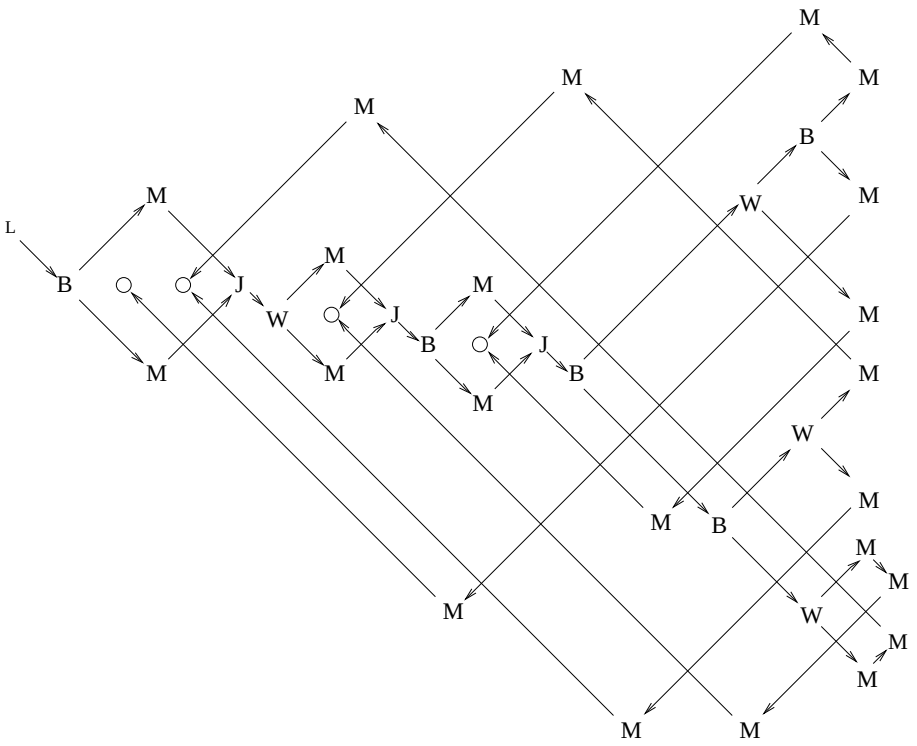


Fig. 5. schematic of ladder instance $\exists x \forall y \exists z (\neg x \wedge \neg y) \vee (\neg x \wedge y) \vee (\neg z \wedge (\neg x \vee z))$

The construction for this example is illustrated in Figure 5. The sequence of diamonds is similar to that in the opening problem, with a Black choice gadget for each $\exists$ quantifier, and a White choice gadget for each $\forall$. The size of each diamond is made proportional to the maximum of the number of positive and the number of negative occurances of the corresponding variable, so that we can have a disjoint incoming ladder for each occurance. Inside each diamond we place extra white stones to act as ladderbreakers. This ensures the failing of any

incoming ladder an the uncovered side of the diamond. Next, to the right of the last diamond, the boolean expression is laid out. Each $\vee$ is mapped to a Black choice gadget, and each $\wedge$ to a White choice gadget. The two subexpressions are then recursively laid out to the upper right, and lower right, spaced sufficiently apart to allow for disjoint ladders. At the leaves we place mirrors directing the ladder to the appropriate diamond. Ladder-paths are free to intersect since the actual line of play can only follow one path back to a diamond. It should be obvious how to apply this method to any formula in **QBF**.

As explained in section 3, the ladder thus constructed works if and only if the formula is true.

4 Conclusions

For the first time, we have identified a natural aspect of the game of Go—the ladder—which is not only PSPACE hard, but PSPACE-complete. This may surprise many Go players who think reading out ladders is an elementary exercise in visualization.

Our reduction improves on that of Lichtenstein and Sipser [LS80] in simplicity (by avoiding the need for intersection gadgets), economy (using a number of stones only linear in formula size), and aesthetic appeal (the opening problem would not look out of place in a go magazine).

References

[GJ79] Garey, M., R., Johnson, D., S., Computers and Intractability, Bell Telephone Laboratories, (1979)

[LS80] Lichtenstein, D. and Sipser, M., GO is Polynomial-Space Hard, Journal of the ACM, Vol. **27**, No. 2, (April 1980) 393-401.

[R83] Robson, J., The Complexity of Go, Proc. IFIP (International Federation of Information Processing), (1983) 413-417.

[P94] Papadimitriou, H., Computational complexity, Addison-Wesley, (1994)