BoxOff is NP-complete

Ryan Hayward[1], Robert Hearn[2], and Mahya Jamshidian[3]

[1] University of Alberta, Edmonton AB T6G 2R3, Canada
hayward@ualberta.ca
[2] bob.hearn@gmail.com
[3] University of Alberta, Edmonton AB T6G 2R3, Canada
mjamshidian@ualberta.ca

Abstract. BoxOff is a one-player game invented by Steven Meyers: a rectangular board is covered with colored stones; a legal move is to remove two same-colored stones from opposite corners of an otherwise empty rectangle. Removing all stones wins the game. We show that it can be hard to determine whether a BoxOff puzzle is winnable: by reducing from Boolean Satisfiability, we show that BoxOff is NP-complete, even when only four colors are used.

Keywords: BoxOff puzzle · NP-Complete · Satisfiability.

1 Introduction

In 2013 in *Games Magazine*, Steven Meyers introduced his new solitaire game BoxOff: see the article by Kerry Handscomb in *Abstract Games Magazine* for a colorful introduction [6, 3]. The board has a rectangular grid; each board cell is empty or has a colored stone. On a move, the player removes two stones of the same color that lie on opposite corners of an otherwise empty rectangle. The player wins by clearing the board. We are interested in this decision question: given a BoxOff puzzle, is it solvable, i.e. can the player win? Consider Figure 1. The left puzzle is solvable, e.g. remove {a1,b1}, then {b2,c2}, then {a2,c1}. The right puzzle is not solvable: each of {b2,c1}, {b1,c2} must be removed before the other, which is impossible. To learn the basics of BoxOff strategy, see Handscomb's article.

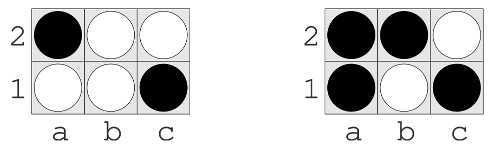


Fig. 1: Two 2-color BoxOff puzzles (left is solvable, right is not).

Browne and Maire investigated the complexity of BoxOff [1]. They gave a Monte Carlo analysis of random play, described a polytime algorithm to solve one-column k-color BoxOff, and noted that otherwise the puzzle's complexity was unknown. We will show that 4-color BoxOff is NP-complete, resolving this open question.

2 Reduction overview

In the usual way [2, 5], we will show that 4-color BoxOff is NP-hard by reducing from 3-CNF-Satisfiability (3-SAT). Given a 3-SAT formula, we construct a BoxOff puzzle that is solvable if and only if the formula is satisfiable. The BoxOff puzzle models a Boolean circuit, formed from gadgets (independent sub-puzzles) that simulate Boolean variables, And and Or gadgets, fanout gadgets, color-switch gadgets, and turn gadgets. In order to ensure that gadgets influence each other only with respect to the flow of the circuit, we insulate the gadgets by placing each inside one cell of a larger overlay grid.

We will show that the input formula is satisfiable if and only if the BoxOff puzzle can be cleared by a two-phase process, starting with a multiple-source/single-sink flow from each variable to a single cell indicating that the complete formula is satisfied, followed by a cleanup phase that erases all overlay stones and any remaining gadget stones.

3 4-color BoxOff is NP-complete

Here we give our reduction, and also show an example: the BoxOff puzzle corresponding to $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$.

We transform the input formula into a puzzle by connecting the literal gadgets to appropriate Or gadgets and thence And gadgets, using the wiring (turn, fanout, and color-change) gadgets. We do not need a crossover gadget: there is only empty space between the stones that are paired, and the overlay grid keeps any gadgets from interacting that are not directly paired in a row or column.[4]

3.1 Overlay Stones and Gadget Stones

We use two stone colors (black and white) for the overlay grid. Within each row or column that includes an overlay stone, the stones alternate colors. We use two other colors (red and blue) for our gadgets. Each gadget fits in a bounded grid called a *container*, defined as the empty rectangular regions within the overlay grid. We align gadgets so that one gadget's output is on the same line (horizontal

[4] Reducing instead from Planar 3-SAT would not help us avoid the need for a crossover, if crossing signals were not trivial in this setting. The reason is that we need to connect the clauses together to produce a single output signal, a situation that is common in SAT reductions. But Planar 3-SAT only applies when the graph connecting the variables to their clauses is planar; it does not allow us to further connect the clauses. Instead, we would then have reduced from Bounded One-Player Constraint Logic [4], which solves this problem.

or vertical) as the next gadget's input. At least one overlay stone separates any pair of stones not in gadgets in the same row or column of the overlay grid, so they can never interact. We only put two gadgets in the same row or column when they are connected. Figure 2 is an example of the overlay grid.

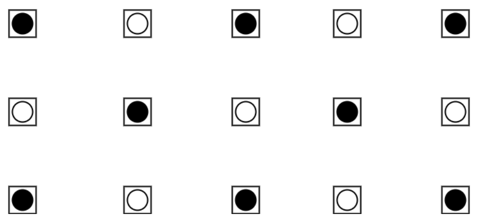


Fig. 2: The overlay black and white stones, demarking the containers within. The gadget stones are located in these containers, to prevent unwanted interactions.

3.2 Signals

Signals are propagated whenever an *output* stone from one gadget is paired with an *input* stone in another gadget, removing both. We then say that the output and corresponding input have been *activated*. The activation of its inputs (if any) is what allows a gadget to activate its output(s).

3.3 Variable Gadget

A *variable* gadget consists of a single output stone, connecting to inputs in two other gadgets. See Figure 3(a). Each variable corresponds to a switch, where the player can set the output signal to be true or false. So each gadget has two possible output directions, indicated in the figure by arrows. A satisfying assignment corresponds to setting the appropriate output signal (true or false) for each variable gadget.

3.4 Wiring Gadgets

Lemma 1. *The* turn *gadget rotates its input signal by 90 degrees.*

Proof. When the input signal is available, the red stone can be matched and removed, allowing the blue stone to propagate the output signal.

Lemma 2. *The* fanout *gadget splits one input signal into two.*

Proof. An active input signal matches the middle blue stone, allowing the lower and the upper red stone to be removed. The remaining blue stones can leave the gadget as active signals.

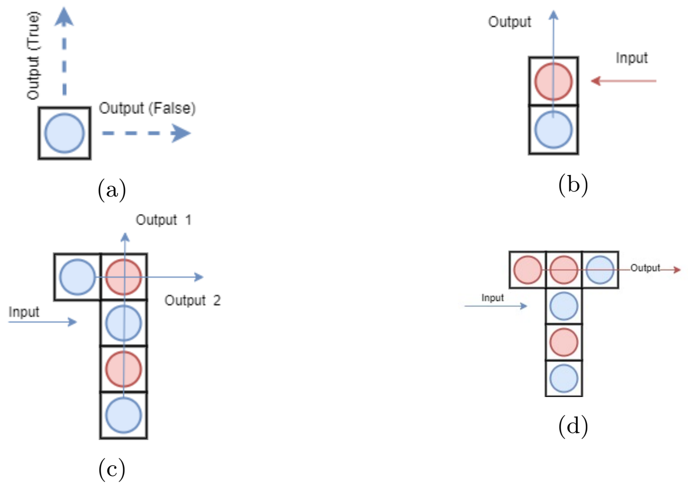


Fig. 3: Gadgets: (a) Variable (b) Turn (c) Fanout (d) Color-change.

Lemma 3. *The* color-change *gadget turns a blue input signal into a red output signal.*

Proof. An active input signal matches the middle blue stone, allowing the upper-middle and lower red stone to match and be removed. The remaining blue stones match and the upper-left red stone serves as an output signal.

There are two ways to change colors: to maintain the direction of the input signal, use the color-change gadget; to change the direction of the signal, use the turn gadget. (The color-change gadget is actually not necessary: we could just combine a fanout gadget and turn gadget instead. However, the color-change gadget makes our transformations slightly more compact.)

3.5 Logical Gadgets

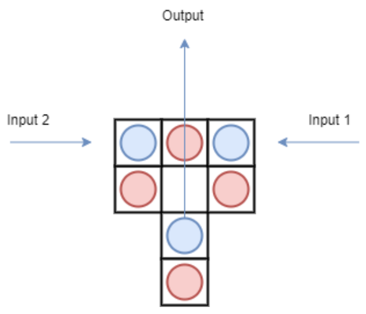
Lemma 4. *The* Or *gadget functions as a logical OR operator.*

Proof. Case 1) Both inputs are active: Both upper-level blue stones are matched with input signals. Then the upper red stone is matched with either of the lower red stones. Then the lower blue stone can leave the gadget as a active signal. The two remaining red stones will match once the output signal is activated, i.e. once the lower blue stone disappears.

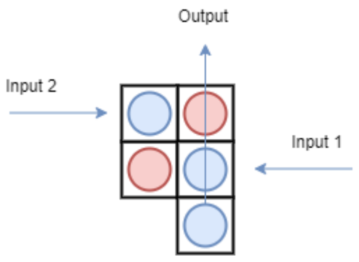
Case 2) Only input 1 is active: The upper-right blue stone matches input signal 1. Then the upper red stone matches the right red stone, allowing the output stone to leave the gadget as an active signal.

Case 3) Only input 2 is active: Similar to the previous case.

Case 4) Neither input is active: No stones are matched: the output signal cannot leave the gadget, so is inactive.



(a)



(b)

Fig. 4: Logical gadgets: (a) OR, (b) AND.

Lemma 5. *The* AND *gadget functions as a logical AND operator.*

Proof. *Case 1) Both inputs are active:* The upper and middle blue stones match the inputs. Then the red stones match, so the output blue stone is an active signal.

Case 2) At most one input is active: At least one blue stone remains, preventing the red stones from matching, so the output signal is inactive.

3.6 Gadget Interactions

We have already shown that gadgets not in the same row or column cannot interact. But we also need to show that gadgets that are in the same row or column can only interact as intended. Ideally this would mean that no pairs of stones can be removed other than in the valid propagation of signals. In fact, a slightly weaker condition suffices. First, we only connect non-turn gadgets together via intervening turns, never directly. Then we only need to analyze connections involving turns. But a turn has only two stones: the input stone is the one intended to be matched, and can only be paired with its intended partner; the output stone should instead be used to propagate the signal onward. If instead the output stone is matched with a stone in another gadget, then the signal simply doesn't propagate, and no incorrect solution of the puzzle is thus enabled. But what about the loss of the stone that inappropriately paired with the turn's output stone? This can only be a stone in the turn's input gadget. If that gadget has a single output, then any change in its properties is irrelevant, because the output can't propagate further. The only other case is that the input gadget is a fanout. We need to verify that if a fanout stone inappropriately pairs with an output turn gadget's output stone, this does not then enable the other output to activate. By inspection, this is the case.

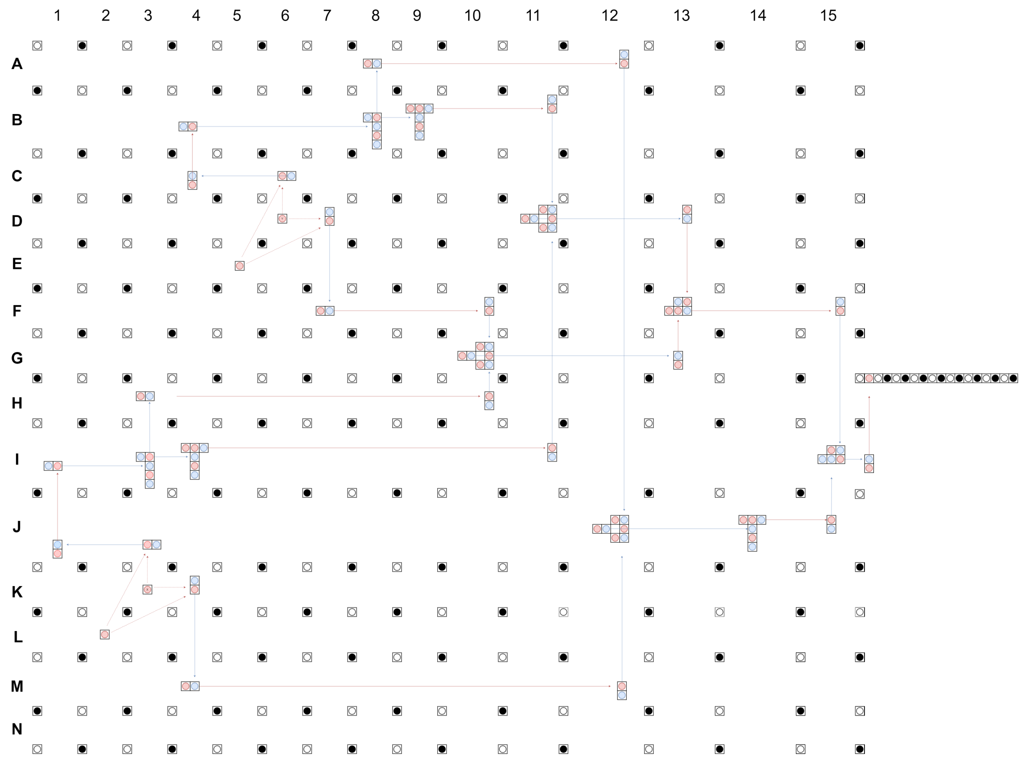


Fig. 5: Transformation of $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$. Gadgets for x,y in cells K3,D6, auxiliary stones at L2,E5. Or at G10 is $(x \vee \bar{y})$, Or at D11 is $(x \vee y)$, Or at J12 is $(\bar{x} \vee y)$. And at F13 is $((x \vee y) \wedge (x \vee \bar{y}))$, And at I15 is $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$.

3.7 The Reduction

The main result of our paper is the following theorem. The example transformation shown in Figure 5 may serve as a reference here; it is analyzed explicitly in Section 3.8. (Note: for compactness, in the example we relax the rule about only connecting non-turn gadgets via turns; in this instance no unwanted interactions arise.)

Theorem 1. *Deciding whether a BoxOff game with 4 colors is solvable is NP-Complete.*

Proof. Let ϕ be a 3-CNF formula with variables $x_1, \cdots, x_n$. Then we construct a corresponding BoxOff configuration as follows.

Variable gadgets. There is one gadget for each variable x_j.

Or *gadgets.* Our Or gadgets take two inputs. Each clause has three literals, so we use two chained Or gadgets for each clause.

Fanout gadgets. A variable can appear in more than one clause, so we might need to duplicate a variable's output signal to all instances of the corresponding literals. For this, we use fanout gadgets, possibly sending signals far around the grid, so we also need turn (and if necessary color-change) gadgets. A variable gadget's True (resp. False) output signal flows to Or gadgets in which the variable is a positive (resp. negated) literal.

And *gadgets:* We use the And gadgets to logically combine the value of all clauses; again, this operator takes only two inputs, so we need to chain $k - 1$ And gadgets to resolve a formula with k clauses. We can activate the final And if and only if the formula is satisfiable. Notice that when at least one input to the And gadgets is inactive, at least one And gadget will have uncleared stones, and the BoxOff puzzle will not be solvable.

In the rest of this proof, we want to show that the BoxOff puzzle is solvable when the input formula is satisfiable. Hence, assume the selected assignment is satisfying; then, the construction described allows us to activate the final And gadget, by activating the variable gadgets appropriately and propagating all signals where possible.

Cleaning up overlay grid: In order to be able to clean up the board, we modify our construction so that when the final And gadget emits an active signal, we can remove the overlay grid. We route this active signal to a special middle row of our construction, where to the right of the final overlay-grid stone, say black, we add a red stone and then another black stone, and then alternating white and black stones, enough to match the middle row of the overlay grid. See Figure 6(a).

In Figure 6(b), notice that the red stone in the middle row matches the final And active signal, so the middle-row black stones adjacent to the red stone will match, allowing the middle row to disappear completely, which will then allow each column to disappear, as we construct the overlay grid so that within each row and column, the black and white stones alternate, and the total number of overlay stones in each column is odd.

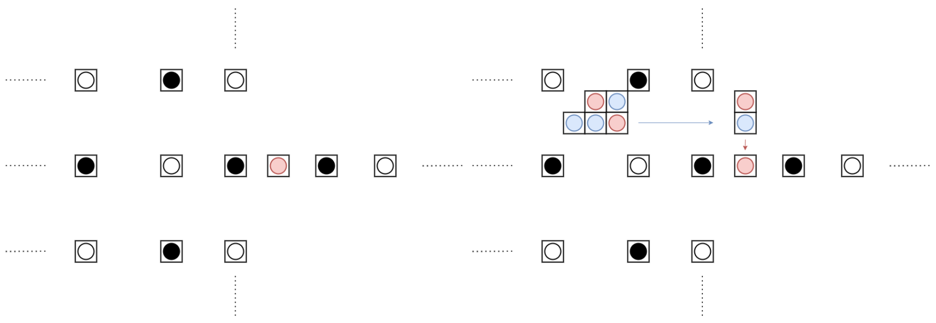

(a) (b)

Fig. 6: (a) Rightmost edge of the grid, showing the middle row. (b) Rightmost edge of the grid, with middle row and final AND gadget.

Cleaning up remaining stones: Depending on the particular Boolean assignment selected for our formula, each gadget (except for the AND gadgets) might have remaining stones, either because the signal never reached the gadget, or the signal reached the gadget but not all stones were removed when the signal flowed through. We now show how to add extra stones to variable gadgets so that a final cleanup is possible whenever the formula is satisfied.

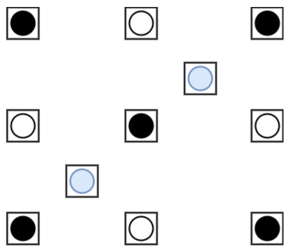


Fig. 7: The modified variable gadget has an extra stone located in a diagonally adjacent cell.

The idea is that the extra stone can activate the True/False signal not assigned to the variable. This auxiliary stone must not interact with any gadget until the overlay grid has vanished; we place it in a diagonally adjacent cell, as shown in Figure 7.

We constructed our gadgets so that they normally receive inputs in a straight line, but the auxiliary stone will not be in a line with the variable's output turn gadgets. To address this, we orient those turn gadgets such that the auxiliary stone can still pair with them, as shown in Figure 8.

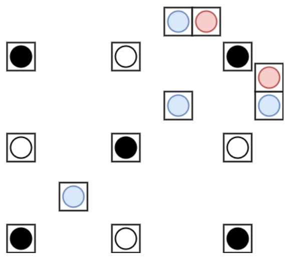


Fig. 8: The modified variable gadget, showing the connecting turn gadgets and the auxiliary stone to be used during the clean-up phase.

Once all overlay stones and the satisfied parts of the gadgets are gone, the auxiliary stone can be matched with the turn at the output of the unchosen path and clear the following stones.

Once the auxiliary stones for each unactivated literal path have cleared, each instance of each gadget has disappeared: variable and auxiliary gadgets have cleared; ANDs have cleared; ORs all have both inputs activated (because each traces back to a literal, or to another OR that has activated); fanouts split variable outputs, and since all variable outputs are active, fanouts have cleared; turns occur only on pathways connecting the above and so have cleared. So, with a satisfying assignment, we can clear the entire board. If there is no satisfying assignment, then the final AND gadget is not cleared by signal propagation, and overlay grid ensures that this gadget cannot be cleared in any other way.

Our reduction is polynomial: we have one AND or OR gadget per logical operator in the formula, and one variable or fanout per literal. We can place each variable, auxiliary variable, AND, OR, and fanout in its own column and row. If we allow an equal number of rows and columns for routing, we can connect any gadget output to any other gadget input with at most four turns and one color-change gadget. Thus the number of rows and columns needed is polynomial in the input size. The size of each container (i.e., the spacing of the overlay stones) also need only be polynomial in the input size: the gadgets might need to be positioned at various offsets within the containers to align inputs and outputs, but at worst the required space would be polynomial in the total number of gadgets, and all we actually need is that the logarithm of this spacing is polynomial.

Finally, BoxOff is clearly in NP: a solution is a list of pairings, which is of polynomial length, and can be easily verified. This completes the proof.

3.8 Example transformation

Here we describe our example transformation of $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$ shown in Figure 5. For the sake of simplicity, this example is in 2-CNF.

To begin, consider a trial assignment, say x and y both false. Since x is false, we follow the horizontal-right output signal from variable gadget x (at K3), which then matches the red stone at $K4$. The signal then proceeds to the lower input of the OR gadget at $J12$, then to the color-change gadget at $J14$, through a turn gadget at $J15$, and reaches the lower-level input of the AND gadget at $I15$.

Meanwhile, starting from variable gadget y at $D6$, output proceeds to the turn at $D7$, follows arrows to the upper input of the OR gadget at $G10$, and can leave the gadget to reach the lower input of the AND gadget at $F13$.

Notice that these two signals cannot move further, since the D11 OR gadget has no active input. Thus neither AND gadget at F13,I15 has an active upper input, so each is inactive. So our current assignment fails.

Next consider the satisfying assignment with both x, y true. Now the x variable stone at K3 activates the turn at J3 and the signal continues to the I3 fanout gadget, whose vertical output continues to the lower input of the G10 OR. The horizontal fanout output activates the I4 color-change gadget at $I4$ and then the D11 OR lower input.

Also, the y variable at $D6$ matches the red turn stone at $C6$. The output of the turn gadget matches the input of the B8 fanout, whose vertical output activates the upper input of the J12 Or. The vertical fanout output matches the upper input of the D11 Or.

Now all Or gadgets have active outputs, so the two And gadgets have both inputs active, so both have active outputs. The output of the I15 And matches the red stone outside of the overlay grid, so the middle black-white row clears, and each black-white column then clears.

After all black and white stones are gone, we are left with six turn gadgets, two blue stones from the OR gadgets J12 and G10, and the auxiliary stone for both x and y. The auxiliary x-stone at $L2$ matches the red stone at $K4$ and eventually clears out the remaining lower blue stone for the OR gadget at $J11$, and all turn gadgets in between. The auxiliary y-stone at $E5$ matches the D7 turn gadget and—the last step—the upper blue stone for the G10 OR, and all turn gadgets in between. Finally, the board is clear.

4 Conclusion

We showed that 4-color BoxOff is NP-complete, resolving an open question of Browne and Maire [1]. Our 3SAT reduction is straightforward, except for the extra machinery needed to clean up the board once the final And activates. Our reduction requires at least four colors: two for the overlay grid and two for the gadgets. Might a different approach show hardness for three, or even two colors? (One color is trivial.) We conjecture that with three colors BoxOff is still hard, but with two it is polynomial. We encourage further work to clarify this fascinating boundary.

Acknowledgment

We would like to show our gratitude to Steven Meyers, the designer of the BoxOff game, for his careful review of the paper.

References

1. Browne, C., Maire, F.: Monte carlo analysis of a puzzle game. In: Pfahringer, B., Renz, J. (eds.) AI 2015: Advances in Artificial Intelligence - 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30 - December 4, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9457, pp. 83–95. Springer (2015), https://eprints.qut.edu.au/89493
2. Cook, S.A.: The complexity of theorem-proving procedures. In: Harrison, M.A., Banerji, R.B., Ullman, J.D. (eds.) Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA. pp. 151–158. ACM (1971)
3. Handscomb, K.: Steven Meyers' BoxOff. Abstract Games Magazine **19**, 35–36 (2020)
4. Hearn, R.A., Demaine, E.D.: Games, Puzzles and Computation. A K Peters (2009)

5. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972)
6. Meyers, S.: Boxoff game. Games Magazine (Aug 2013)