BoxOff is NP-complete

No Author Given

No Institute Given

Abstract. BoxOff is the one-player puzzle 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; to win, find a sequence of moves that clears the board. In k-color BoxOff there are at most k stone colors. We show that it can be hard to determine whether a BoxOff puzzle is winnable: by transforming from Conjunctive-Normal-Form Satisfiability, 4-color BoxOff is NP-complete.

Keywords: BoxOff puzzle · NP-Complete · satisfiability.

1 Introduction

In 2013 in an article in *Games Magazine*, Kerry Handscomb introduced the new one-player puzzle BoxOff, created by Steven Meyers [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 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.

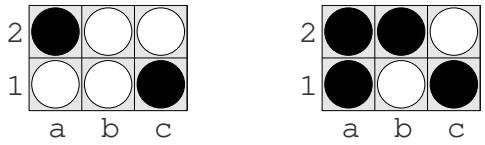

Fig. 1: Two 2-color BoxOff puzzles (left is solvable, right is not).

Browne and Maire investigated the complexity of BoxOff [1]. They gave a Monte Carlo analysis of random play, described a polytime algorithm to solve (determine winnability) 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.

2 Transformation overview

In the usual way [2, 4], we will show that 4-color BoxOff is NP-complete by transforming from 3-CNF-Satisfiability. Given a 3-CNF-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 literals (one gadget per literal), and- and or-operators (one gadget for each operator in each clause), multiple fanout gadgets (at most $\log m$ per variable) that duplicates a logical results as necessary, a color-switch gadget (to maintain parity of each set of colored stones), and a turning gadget (that changes the direction a signal flows within the BoxOff circuit simulation). (We will not need any crossover gadgets, commonly used in circuit transformations.) 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 an overlay grid.

We will show that the input CNF 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 literal satisfied by a particular Boolean assignment) to a single cell indicating that the complete formula is satisfied, and then a cleanup phase that erases all overlay stones and any remaining gadget stones, including unsatisfied-literal gadgets.

3 4-color BoxOff is NP-complete

Here are the details of our transformation. We also show an example: the BoxOff puzzle from the Boolean formula $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$.

We transform the input formula into a puzzle by connecting gadgets to the OR gadgets that make up each clause and then AND them together, 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 keep any gadgets from interacting that are not directly paired in a row or column.

3.1 Overlay stones and gadget stones

We use two stone colors (black and white) for the orthogonal overlay grid. Within each row or column that includes an overlay stone, the overlay stones alternate colors black-white-black-white etc. We use two other colors (red and blue) for gadgets. Each gadget fits in a bounded grid we call a *container*, with at most five rows and at most five columns: we place overlay stones to separate the containers. We align gadgets so that one gadget's output is on the same line (horizontal or vertical) as the next gadget's input. Within a container, we might have to shift a gadget at most three cells, so a 9×9 grid suffices for each container. The overlay grid separates gadgets, so we put the black and white stones for the overlay grid at locations $(10x, 10y)$ for integers $1 \leq x \leq X$ and $1 \leq y \leq Y$. After we have

placed all gadgets, we add extra black-white barrier stones along each container edge. We will show that for an input 3-CNF-sat formula with n variables and m clauses, X and Y are both at most $(15 + \log m) \times n + 10 \times \log m + 18m$, so our transformation is polynomial.

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 2(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 of literals to variables corresponds to setting the appropriate output signal (true or false) for each variable gadget.

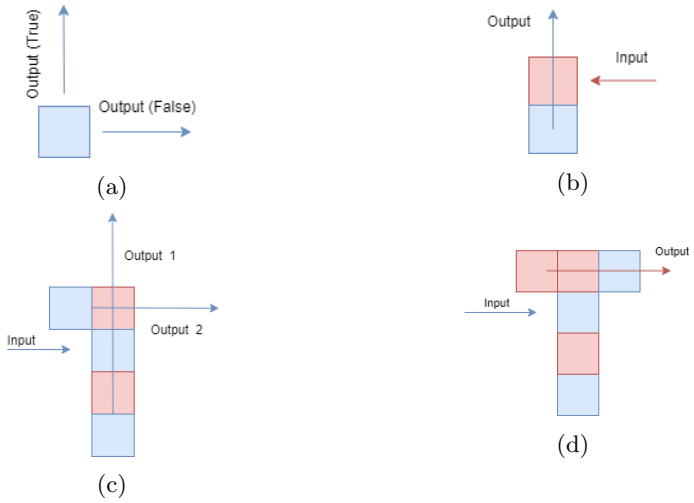

Fig. 2: Gadgets: (a) variable (b) turn (c) fanout (d) color-change.

3.4 Wiring gadgets

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

Proof. When the input signal is active, the red stone is matched and removed, then the blue stone propagates the output signal.

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

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

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 stone match and the left-upper 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

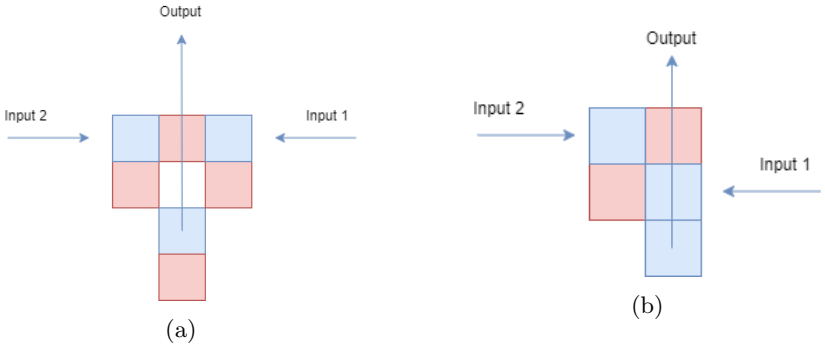

Fig. 3: Logical gadgets: (a) OR, (b) AND.

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 an input signal. 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 activate, i.e. once the lower blue stone disappears.

Case 2) Only input 1 is active: The left upper-level blue stone matches input signal 1. Then the upper red stone matches the left 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.

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.

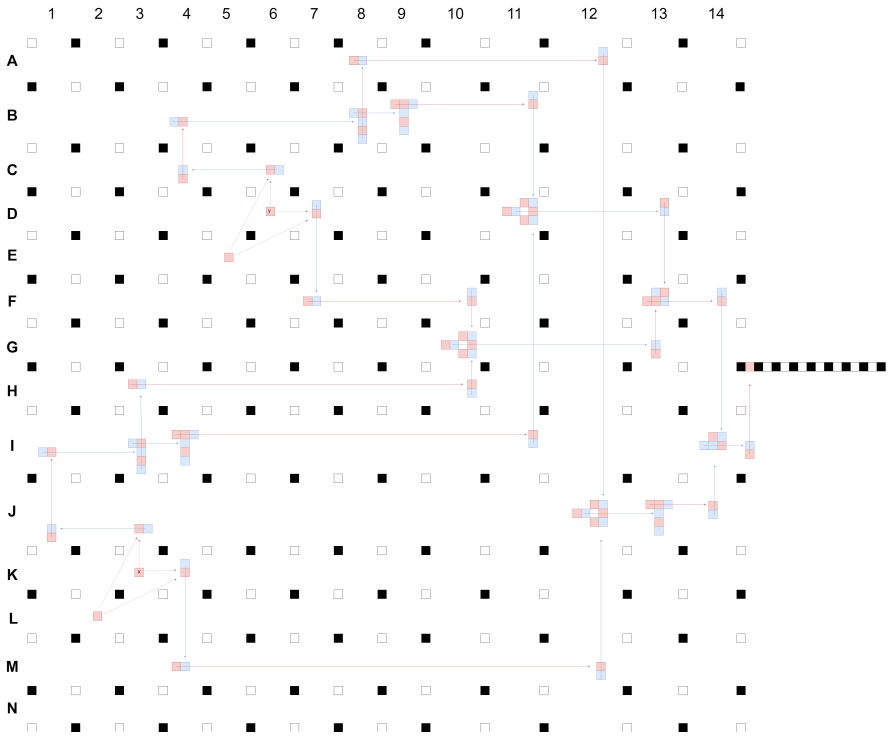

Fig. 4: Transformation of $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$. Gadgets for x,y in cells K3,D6, auxiliary stones at L2,E5. OR at G10 is $(x \vee \bar{y})$, OR at D11 is $(x \vee y)$, OR at J12 is $(\bar{x} \vee y)$. AND at F13 is $((x \vee y) \wedge (x \vee \bar{y}))$, AND at I14 is $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$. We have not yet added the barrier stones.

3.6 The Transformation

Figure 4 shows the BoxOff puzzle we get by transforming the Boolean formula $(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y)$. Notice how signals move in straight lines among gadgets. The main result of our paper is the following theorem and proof.

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

Proof. This proof has five parts. ϕ is a formula with variables $x_1, \cdots, x_n$.

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

OR gadgets. Our OR gadgets take two inputs. We assume the input Boolean formula is in exact 3-CNF, so each clause has exactly three literals, so we use two 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. As mentioned before, a variable gadget's True (resp. False) output signal flows only 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 $k - 1$ AND gadgets to resolve a formula with k clause. 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 the BoxOff puzzle is solvable when the input formula is satisfiable. Hence, assume the selected assignment is satisfying and that the last AND gadget has active output. So the board is as shown in Figure 5.

Cleaning up black and white stones: 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, removal of the overlaying black-white grid begins. So 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. See Figure 6.

As in Figure 7, notice that the red stone in the middle row matches the last-AND active signal, so the middle-row white stones adjacent to the red stone will match, allowing the middle row to disappear, 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 row, and column, is odd.

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

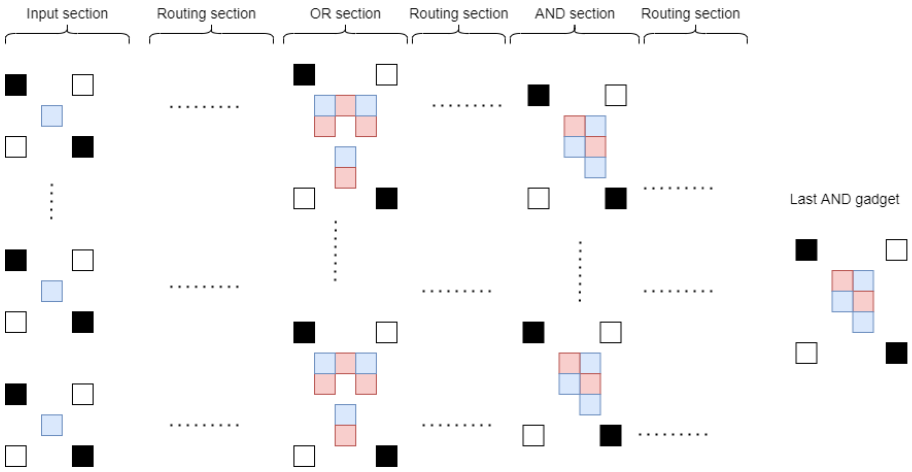

Fig. 5: A representation of the constructed BoxOff puzzle.

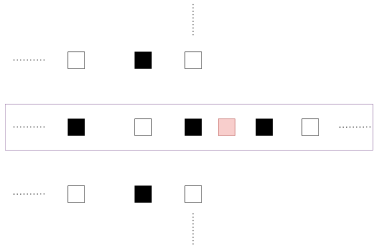

Fig. 6: Rightmost edge of the grid, showing the middle row.

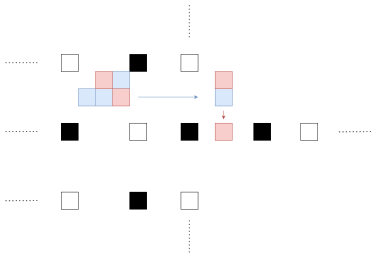

Fig. 7: Rightmost edge of the grid, with middle row and final AND gadget.

flowed through. We now show as in Figure 8 how to add extra stones to variable gadgets so that a final cleanup is possible whenever the formula is satisfied.

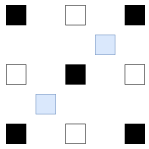

Fig. 8: The modified variable gadget has an extra stone located in a diagonally adjacent cell.

The idea is that the extra variable can activate the True/False signal not assigned to the variable. For our idea to work, this auxiliary stone must not interact with any gadget until the black-white overlay grid has vanished.

Reacll that we constructed gadgets so that they receive inputs in a straight line. For activating the other path by auxiliary stone, it may not be aligned with the next gadget. In order to fix that, we feed the output signal of the main variable gadget to a turn gadget. Then, we feed the output of the gadget to the next gadgets as in Figure 9.

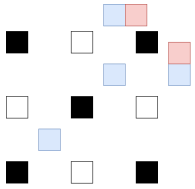

Fig. 9: The modified final variable gadget. It has the turn gadgets at the output of its signal, and the auxiliary stone to be used at the clean-up phase.

Once all black-white 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 has 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 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 board. Also, if there is no satisfying assignment, then the final AND gadget is not cleared by the signal, and barrier stones ensure that this gadget cannot be cleared in any other way. See Figure 10.

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.

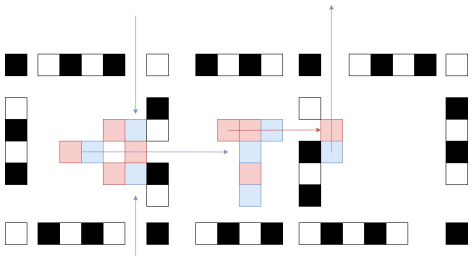

Fig. 10: Extra barrier stones added around three containers.

Now show that the board size is polynomial in n and m (number of variables and clauses respectively). Each gadget fits in a 4×4 cell. Each variable is present in at most m clauses. We need at most $\lfloor \log m \rfloor + 1$ fanout gadgets to duplicate the signals. There are two OR gadgets per clause. There will be $m-1$ AND gadgets in total. How many turn and color-change gadgets are there? Each output signal from fanout, variable, OR, and AND gadgets uses at most one color-change gadget. Thus there are at most $2 \times n + 2 \times (\lfloor \log m \rfloor + 1) + 2 \times m + m - 1$ color-change gadgets. Finally each output signal uses the turn gadget at most four times (allowing a turn of up to 270 degrees and possibly a final straightening). So the size of the grid is in $O(n \times \log m + m)$.

3.7 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 4. 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 $J13$, through a turn gadget at $J14$, and reaches the lower-level input of the AND gadget at $I14$.

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,I14 has an active upper, 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 I14 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 transformation is straightforward, except for the extra machinery needed to clean up the board once the final AND activates. Our transformation requires at least four colors: two for the overlay grid and two for the gadgets. Might a different transformation show hardness for three, or even two colors? (One color is trivial.) We guess that three colors is still hard; two colors might be polynomial. We encourage further work to clarify this fascinating boundary.

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. 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)