Single-Player and Two-Player Buttons & Scissors Games (Extended Abstract)

Kyle Burke[1], Erik D. Demaine[2], Harrison Gregg[3], Robert A. Hearn[4], Adam Hesterberg[2], Michael Hoffmann[5], Hiro Ito[6], Irina Kostitsyna[7], Jody Leonard[3], Maarten Löffler[8], Aaron Santiago[3], Christiane Schmidt[9], Ryuhei Uehara[10], Yushi Uno[11], and Aaron Williams[3]

Abstract. We study the computational complexity of the Buttons & Scissors game and obtain sharp thresholds with respect to several parameters. Specifically we show that the game is NP-complete for $C = 2$ colors but polytime solvable for $C = 1$. Similarly the game is NP-complete if every color is used by at most $F = 4$ buttons but polytime solvable for $F \leq 3$. We also consider restrictions on the board size, cut directions, and cut sizes. Finally, we introduce several natural two-player versions of the game and show that they are PSPACE-complete.

1 Introduction

Buttons & Scissors is a single-player puzzle by KyWorks. The goal of each level is to remove every button by a sequence of horizontal, vertical, and diagonal cuts, as illustrated by Fig. 1. It is NP-complete to decide if a given level is solvable [2]. We study several restricted versions of the game and show that some remain hard, whereas others can be solved in polynomial time. We also consider natural extensions to two player games which turn out to be PSPACE-complete.

Section 2 begins with preliminaries, then we discuss one-player puzzles in Section 3 and two-player games in Section 4. Open problems appear in Section 5. Due to space restrictions, some proofs are sketched or omitted. A full version of this article can be found on arXiv.

[1] Plymouth State University, `kgburke@plymouth.edu`
[2] Massachusetts Institute of Technology, {`edemaine,achester`}`@mit.edu`
[3] Bard College at Simon's Rock, {`hgregg11,jleonard11,asantiago11,awilliams`}`@simons-rock.edu`
[4] `bob@hearn.to`
[5] ETH Zürich, `hoffmann@inf.ethz.ch`
[6] The University of Electro-Communications, `itohiro@uec.ac.jp`
[7] Technische Universiteit Eindhoven, `i.kostitsyna@tue.nl`. Supported in part by NWO project no. 639.023.208.
[8] Universiteit Utrecht, `m.loffler@uu.nl`
[9] Linköping University, `christiane.schmidt@liu.se`. Supported in part by grant 2014-03476 from Sweden's innovation agency VINNOVA.
[10] Japan Advanced Institute of Science and Technology, `uehara@jaist.ac.jp`
[11] Osaka Prefecture University, `uno@mi.s.osakafu-u.ac.jp`

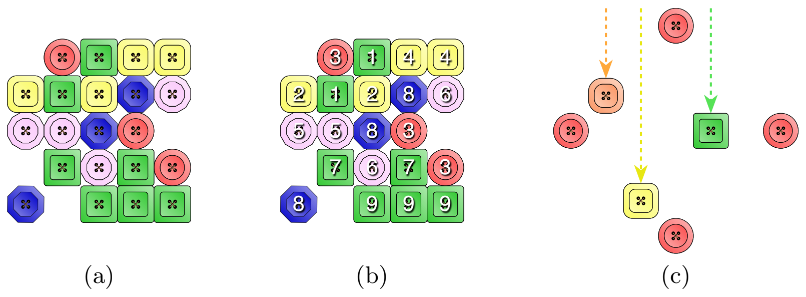

Fig. 1. (a) Level 7 in the Buttons & Scissors app is an $m \times n = 5 \times 5$ grid with $C = 5$ colors, each used at most $F = 7$ times; (b) a solution using nine cuts with sizes in $S = \{2,3\}$ and directions $d = ⋇$ (no vertical cut is used); (c) a gadget used in Theorem 5.

2 Preliminaries

A Buttons & Scissors *board* B is an $m \times n$ grid, where each grid position is either empty or occupied by a button with one of C different colors. A *cut* is given by two distinct buttons b_1, b_2 of the same color c that share either the x-coordinate, the y-coordinate, or are located on the same diagonal (45° and -45°). The *size* s of a cut is the number of buttons on the line segment $\overline{b_1 b_2}$ and so $s \geq 2$. A cut is *feasible* for B if $\overline{b_1 b_2}$ only contains buttons of a single color.

When a feasible cut is applied to a board B, the resulting board B' is obtained by substituting the buttons of color c on $\overline{b_1 b_2}$ with empty grid entries. A *solution* to board B is a sequence of boards and feasible cuts $B_1, x_1, B_2, x_2, \ldots, B_t, x_t, B_{t+1}$, where B_{t+1} is empty, and each cut x_i is feasible for B_i and creates B_{i+1}.

Each instance can be parameterized as follows (see Fig. 1 for an example):

1. The *board size* $m \times n$.
2. The *number of colors* C.
3. The *maximum frequency* F of an individual color.
4. The *cut directions* d can be limited to $d \in \{✳, ⋇, +, -\}$.
5. The *cut size set* S limits feasible cuts to having size $s \in S$.

Each $d \in \{✳, ⋇, +, -\}$ is a set of cut directions (i.e. $+$ for horizontal and vertical). We limit ourselves to these options because an $m \times n$ board can be rotated 90° to an equivalent $n \times m$ board, or 45° to an equivalent $k \times k$ board for $k = m + n - 1$ with blank squares. Similarly, we can shear the grid by padding row i with $i - 1$ blanks on the left and $m - i$ blanks on the right which converts $d = +$ to $d = \not-$. We obtain the family of games below ($B\&S[n \times n, \infty, \infty, ✳, \{2,3\}](B)$ is the original):

Decision Problem: $B\&S[m \times n, C, F, d, S](B)$.
Input: An $m \times n$ board B with buttons of C colors, each used at most F times.
Output: True $\iff$ B is solvable with cuts of size $s \in S$ and directions d.

Now we provide three observations for later use. First note that a single cut of size s can be accomplished by cuts of size $s_1, s_2, \ldots, s_k$ so long as $s = s_1 + s_2 + \cdots + s_k$ and $s_i \geq 2$ for all i. Second note that removing all buttons of a single color from a solvable instance cannot result in an unsolvable instance.

Remark 1. A board can be solved with cut sizes $S = \{2,3,\ldots\}$ if and only if it can be solved with cut sizes $S' = \{2,3\}$. Also, $\{3,4,\ldots\}$ and $\{3,4,5\}$ are equivalent.

Remark 2. If board B' is obtained from board B by removing every button of a single color, then $B\&S[m \times n, C, F, d, S](B) \implies B\&S[m \times n, C, F, d, S](B')$.

3 Single-Player Puzzle

3.1 Board Size

We solve one row problems below, and give a conjecture for two rows in Section 5.

Theorem 1. *$B\&S[1 \times n, \infty, \infty, -, \{2,3\}](B)$ is polytime solvable.*

Proof. Consider the following context-free grammar,

$$S \to \varepsilon \mid \square \mid SS \mid xSx \mid xSxSx$$

where $\square$ is an empty square and $x \in \{1,2,\ldots,C\}$. By Remark 1, the solvable $1 \times n$ boards are in one-to-one correspondence with the strings in this language. $\square$

3.2 Number of Colors

Hardness for 2 colors. We begin with a straightforward reduction from 3SAT. The result will be strengthened later by Theorem 7 using a more difficult proof.

Theorem 2. *$B\&S[n \times n, 2, \infty, +, \{2,3\}](B)$ is NP-complete.*

Proof Sketch: A *variable gadget* has its own row with exactly three buttons. The middle button is alone in its column, and must be matched with at least one of the other two in the variable row. If the left button is not used in this match, we consider the variable set to *true*. If the right button is not used, we consider the variable set to *false*. A button not used in a variable is an *available output*, and can then serve as an *available input* to be used in other gadgets.

Every *clause gadget* has its own column, with exactly four buttons. The topmost button (*clause button*) is alone in its row; the others are inputs. If at least one of these is an available input, then we can match the clause button with all available inputs. We construct one clause gadget per formula clause, connecting its inputs to the appropriate variable outputs. Then, we can clear all the clauses just when we have made variable selections that satisfy the formula.

The variables are connected to the clauses via a multi-purpose *split gadget* (Fig. 2(a)). Unlike the variable and the clause, this gadget uses buttons of two colors. The bottom button is an input; the top two are outputs. If the input button is available, we can match the middle row of the gadget as shown in Fig. 2(b), leaving the output buttons available. But if the input is not available, then the only way the middle row can be cleared is to first clear the red buttons in vertical pairs, as shown in Fig. 2(c); then the output buttons are not available.

We provide a further description of the split gadget and complete the proof in the full version of this article.

$\square$

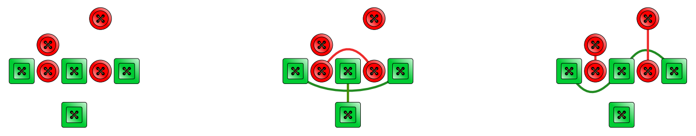

Fig. 2. Split gadget (a) and the two possible ways to clear it (b) and (c).

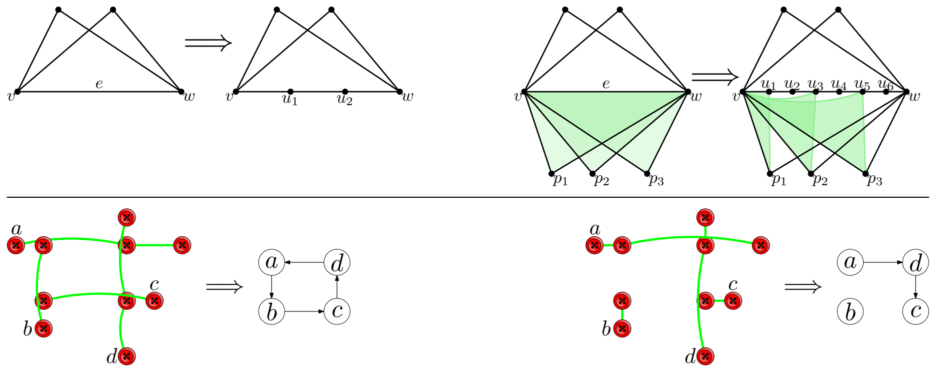

Fig. 3. Top-left: splitting 3-cycles when there are no adjacent triangles to edge e; top-right: splitting 3-cycles when e has adjacent triangles (shaded). Bottom-left: constructing G_c from four cuts blocking each other in a cycle; bottom-right: constructing G_c from the same cuts after reassigning the blocking buttons

Polynomial-Time Algorithm for 1-color and any cut directions. Given an instance B with $C = 1$ color and cut directions $d \in \{✳, ✶, +, -\}$, we construct a hypergraph G that has one node per button in B. A set of nodes is connected with a hyperedge if the corresponding buttons lie on the same horizontal, vertical, or diagonal line whose direction is in d, i.e., they can potentially be removed by the same cut. By Remark 1 it is sufficient to consider a hypergraph with only 2- and 3-edges. A solution to B corresponds to a perfect matching in G. For clarity, we shall call a 3-edge in G a *triangle*, and a 2-edge simply an *edge*.

Cornuéjols et al. [1] showed how to compute a perfect K_2 and K_3 matching in a graph in polynomial time. However, their result is not directly applicable to our graph G yet, as we need to find a matching that consists only of edges and proper triangles, and avoids K_3's formed by cycles of three edges.

To apply [1] we construct graph G' by adding vertices to eliminate all cycles of three edges as follows (see top of Fig. 3). Start with $G' = G$. Consider an $e = (v, w) \in G'$ in a 3-cycle (a cycle of three edges). There are two cases: e is not adjacent to any triangle in G', or e is adjacent to some triangles in G'. In the first case we add vertices u_1 and u_2 that split e into three edges (v, u_1), (u_1, u_2), and (u_2, w). In the second case, when e is adjacent to k triangles, we add $2k$ vertices $u_1, u_2 \ldots, u_{2k}$ along e, and replace every $\triangle p_i v w$ with $\triangle p_i v u_{2i-1}$.

Lemma 1. *There exists a perfect edge- and triangle-matching in G' iff there exists perfect edge- and triangle-matching in G.*

Proof. Given a perfect matching M in G, we construct a perfect matching M' in G'. Consider $e = (v, w)$ in G. If e is not adjacent to any triangles in G, then

- if $e \in M$ then add edges (v, u_1) and (u_2, w) of G' to M' (both v and w are covered by e, and all v, w, u_1, and u_2 are covered by M');
- if $e \notin M$ then add edge (u_1, u_2) of G' to M' (v and w are not covered by e, and u_1 and u_2 are covered by M').

In both cases above the extra nodes in G' are covered by edges in M', and if v and w in G are covered by e in M then v and w are covered by (v, u_1) and (u_2, w) in G'. If e is adjacent to some triangles in G,

- if $e \in M$ then in G' add edges (v, u_1), (u_{2k}, w), and (u_{2j}, u_{2j+1}) to M', for $1 \le j < k$;
- if $\triangle p_i v w \in M$ for some i then add $\triangle p_i v u_{2i-1}$, edges (u_{2j-1}, u_{2j}) for $1 \le j < i$, (u_{2j}, u_{2j+1}) for $i \le j < k$, and (u_{2k}, w) of G' to M';
- if neither e nor any triangle adjacent to e is in M then add edges (u_{2j-1}, u_{2j}) of G' to M', for $1 \le j \le k$.

In all the above cases the extra nodes in G' are covered by edges in M', and if v and w in G are covered by e or a triangle in M then v and w are also covered by (v, u_1) and (u_2, w) or by a corresponding triangle in G'.

Refer to the full version of this article for the details on how to create a perfect matching in G from one in G'. □

Thus, a perfect edge- and triangle-matching in G that does not use a 3-cycle (if it exists) can be found by first converting G to G' and applying the result in [1] to G'. A solution of B consisting of 2- and 3-cuts can be reduced to a perfect edge- and triangle-matching in G; however, the opposite is not a trivial task. A perfect matching in G can correspond to a set of cuts C_M in B that are blocking each other (see bottom of Fig. 3). To extract a proper order of the cuts we build another graph G_c that has a node per cut in C_M and a directed edge between two nodes if the cut corresponding to the second node is blocking the cut corresponding to the first node. If G_c does not have cycles, then there is a partial order on the cuts. The cuts that correspond to the nodes with no outgoing edges can be applied first, and the corresponding nodes can be removed from G_c. However, if G_c contains cycles, there is no order in which the cuts can be applied to clear up board B. In this case we will need to modify some of the cuts in order to remove cycles from G_c. We provide the details in the full version of this article.

By Lemma 1 and by the construction above we obtain the following theorem.

Theorem 3. *$B\&S[n \times n, 1, \infty, d, \{2,3\}](B)$ is polytime solvable for all $d \in \{\ast, \star, +, -\}$.*

3.3 Frequency of Colors

Theorem 4. *$B\&S[n \times n, \infty, 3, \ast, \{2,3\}](B)$ is polytime solvable.*

Proof. A single cut in any solution removes a color. By Remark 2, these cuts do not make a solvable board unsolvable. Thus, a greedy algorithm suffices. □

Hardness was established for maximum frequency $F = 7$ in [2]. We strengthen this to $F = 4$ via the modified clause gadget in Fig. 1 (c). In this gadget the leftmost circular button can be removed if and only if at least one of the three non-circular buttons is removed by a vertical cut. Thus, it can replace the clause gadget in Section 4.1 of [2]. Theorem 5 is proven in the full version of this article.

Theorem 5. *$B\&S[n \times n, \infty, 4, *, \{2,3\}](B)$ is NP-complete.*

3.4 Cut Sizes

Section 3.2 provided a polytime algorithm for 1-color. However, if we reduce the cut size set from $\{2,3,4\}$ to $\{3,4\}$ then it is NP-complete. We also strengthen Theorem 2 by showing that 2-color puzzles are hard with cut size set $\{2\}$.

Hardness for Cut Sizes $\{3, 4\}$ and 1-Color

Theorem 6. *$B\&S[n \times n, 1, \infty, *, \{3,4\}](B)$ is NP-complete.*

Proof. We show $B\&S[n \times n, 1, \infty, *, \{3,4\}](B)$ to be NP-hard by a reduction from PLANAR 3-SAT, which was shown to be NP-complete by Lichtenstein [3].

An instance F of the PLANAR 3-SAT problem is a Boolean formula in 3-CNF consisting of a set $\mathcal{C}$ of m clauses over n variables $\mathcal{V}$. The variable-clause incidence graph $G = (\mathcal{C} \cup \mathcal{V}, E)$ is planar, and all variables are connected in a

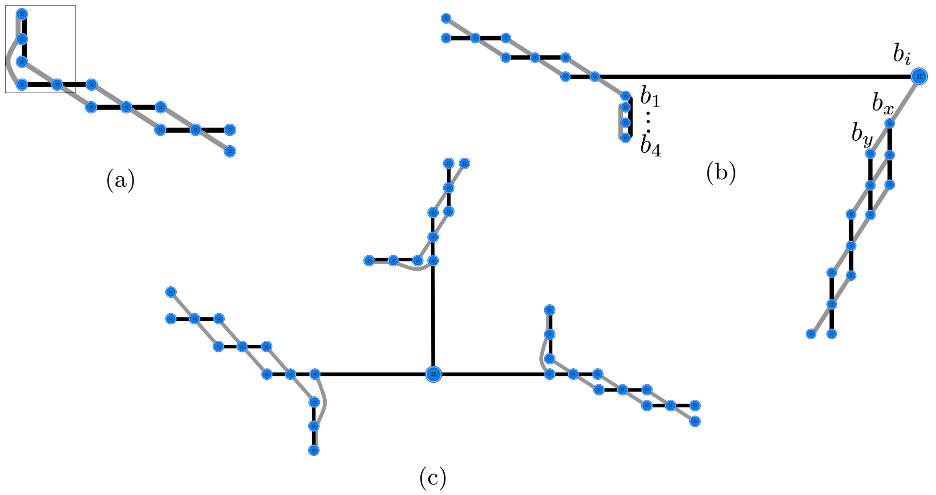

Fig. 4. (a) The only two cut possibilities in the variable gadget (shown in black and gray), corresponding to truth assignments of "true" and "false", respectively. (b) The bend gadget for the 1-color case. (c) The clause gadget for the 1-color case.

cycle. The PLANAR 3-SAT problem is to decide whether there exists a truth assignment to the variables such that at least one literal per clause is true.

We turn the planar embedding of G into a Buttons & Scissors board, i.e., we present variables, clauses and edges by single-color buttons that need to be cut. We provide detailed descriptions of each gadget in the full version of this article.

The **variable gadget**, shown in Fig. 4(a), enables us to associate horizontal and diagonal cut patterns with "true" and "false" values, respectively.

The **bend gadget**, shown in Fig. 4(b), enables us to bend a wire to match the bends in G's embedding while enforcing that the same values are propagated through the bent wire.

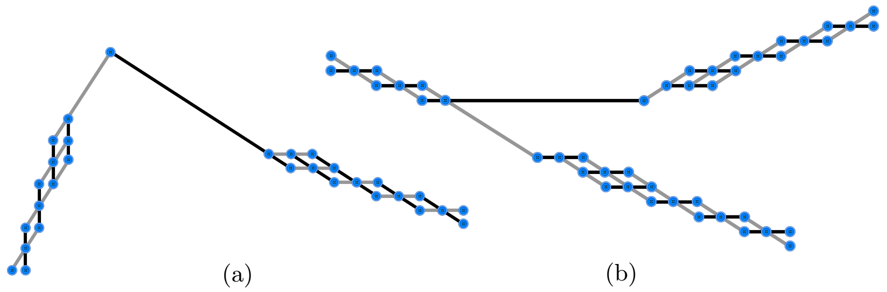

Fig. 5. (a) The not gadget, negating the input truth assignment, for the 1-color case. (b) The split gadget for the 1-color case.

The **split gadget**, shown in Fig. 5(b), enables us to increase the number of wires leaving a variable and propagating its truth assignment.

The **not gadget**, shown in Fig. 5(a), enables us to reverse the truth assignment in a variable wire.

The **clause gadget** is shown in Fig. 4(c). This gadget simulates a conjunction of literals.

Thus, the resulting Buttons & Scissors board has a solution if and only if at least one of the literals per clause is set to true, that is, if and only if the original PLANAR 3-SAT formula F is satisfiable. It is easy to see that this reduction is possible in polynomial time. In addition, given a Buttons & Scissors board and a sequence of cuts, it is easy to check whether those constitute a solution, i.e., whether all cuts are feasible and result in a board with only empty grid entries. Hence, $B\&S[n \times n, 1, \infty, *, \{3,4\}](B)$ is in the class NP. Consequently, $B\&S[n \times n, 1, \infty, *, \{3,4\}](B)$ is NP-complete. $\square$

Hardness for Cut Size {2} and 2-Colors An intermediate problem is below.
Decision Problem: Graph Decycling on (G, S).
Input: Directed graph $G = (V, E)$ and a set of disjoint pairs of vertices $S \subseteq V \times V$.
Output: True, if we can make G acyclic by removing either s or s' from G for every pair $(s, s') \in S$. Otherwise, False.

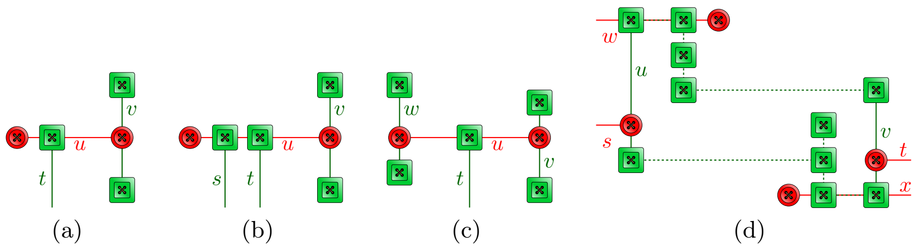

Fig. 6. Three types of nodes: (a) in-degree 1 (tu) and out-degree 1 (uv); (b) in-degree 2 (su and tu) and out-degree 1 (uv); (c) in-degree 1 (tu) and out-degree 2 (uv and uw). In (d) the nodes u and v are linked in S and we can choose to remove u or v.

Lemma 2. *Graph Decycling reduces to Buttons & Scissors with* 2 *colors.*

Proof. Consider an instance (G, S) to graph decycling. First, we observe that we can assume that every vertex in G has degree 2 or 3, and more specifically, in-degree 1 or 2, and out-degree 1 or 2. Indeed, we can safely remove any vertices with in- or out-degree 0 without changing the outcome of the problem. Also, we can replace a node with out-degree k by a binary tree of nodes with out-degree 2. The same applies to nodes with in-degree k.

Furthermore, we can assume that every vertex that appears in S has degree 2. Indeed, we can replace any degree 3 vertex by two vertices of degree 2 and 3, and use the degree 2 vertex in S without changing the outcome. Similarly, we can assume that no two vertices of degree 3 are adjacent. Finally, we can assume that G is bipartite, and furthermore, that all vertices that occur in S are in the same half of V, since we can replace any edge by a path of two edges.

Now, we discuss how to model such a graph in a Buttons & Scissors instance. Each node will correspond to a pair of buttons, either a red or a green pair according to a bipartition of V. These pairs of buttons will be mapped to locations in the plane on a common (horizontal for red, vertical for green) line, and such that any two buttons of the same color that are not a pair are not on a common (horizontal, vertical, or diagonal) line (unless otherwise specified). If two nodes of opposite colors u and v are connected by an edge in G, we say that u *blocks* v. In this case, one of the buttons of u will be on the same line as the buttons of v, and more specifically, it will be between the two buttons of v. That is, v can only be cut if u is cut first. Buttons of opposite colors that are not connected by an edge will not be on any common lines either.

As discussed above, we can assume we have only three possible types of nodes. Fig. 6(a) illustrates the simplest case, of a node u with one incoming edge tu and one outgoing edge uv. Clearly, t blocks u and u blocks v. To model a node with in-degree 2, we need to put two buttons of different same-colored nodes on the same line (see Fig. 6(b)). As long as the other endpoints of these two edges are not on a common line this is no problem: we never want to create a cut that removes one button of s and one of t, since that would create an unsolvable instance. Finally, to model a node with out-degree 2, we simply place a vertical edge on both ends of u (see Fig. 6(c)). Note that is it important here that we do not connect two nodes with out-degree 2 to the same two nodes with in-degree 2, since then we would have both pairs of endpoints on a common line; however, we assumed that nodes of degree 3 are never adjacent so this does not occur.

It remains to create a mechanism to remove vertices from G as dictated by S. This is illustrated in Fig. 6(d) with details in the full version of this article. □

A proof of Lemma 3 is in the full version of this article. Lemmas 2 and 3 give Theorem 7.

Lemma 3. *SAT reduces to Graph Decycling.*

Theorem 7. *$B\&S[n \times n, 2, \infty, +, \{2\}](B)$ is NP-complete.*

4 Two-Player Games

We consider three two-player Buttons & Scissors variants. First we consider color restricted games where (a) each player can only cut specific colors, and (b) players are not restricted to specific colors. For (a) player blue may only cut Blue buttons, while the red player may only cut Red buttons. For (b) we distinguish by winning criterion: for (Impartial) the last player who makes a feasible cut wins; for (Scoring) players keep track of the total number of buttons they've cut. When no cuts can be made, the player with the most buttons cut wins.

In the following sections, we show that all variants are PSPACE-complete.

4.1 Cut-By-Color Games

In this section the first player can only cut blue buttons, the second player can only cut red buttons, and the last player to make a cut wins.

Theorem 8. *The partisan LAST two-player Buttons & Scissors game, where one player cuts blue buttons, the other red buttons, is PSPACE-complete.*

Proof. The proof is by reduction from $G_{\%free}$(CNF) [5]: given a boolean formula $\Phi(x_1, \ldots, x_n)$ in CNF and a partition of the variables into two disjoint subsets of equal size V_b and V_r, two players take turns in setting values of the variables, the first (Blue) player sets the values of variables in V_b, and the second (Red) player sets the values of variables in V_r. Blue wins if, after all variables have been assigned some values, formula Φ is satisfied, and loses otherwise.

For a given instance of formula Φ we construct a Buttons & Scissors board B, such that Blue can win the game on B if and only if he can satisfy formula Φ. We will prove this statement in different formulation: Red wins the game on B if and only if formula Φ cannot be satisfied. For a complete example see the full version of this article.

The **red variable gadget** is shown in Fig. 7(a). Red "sets the value" of the corresponding variable by choosing the first cut to be a (false) or b (true), and

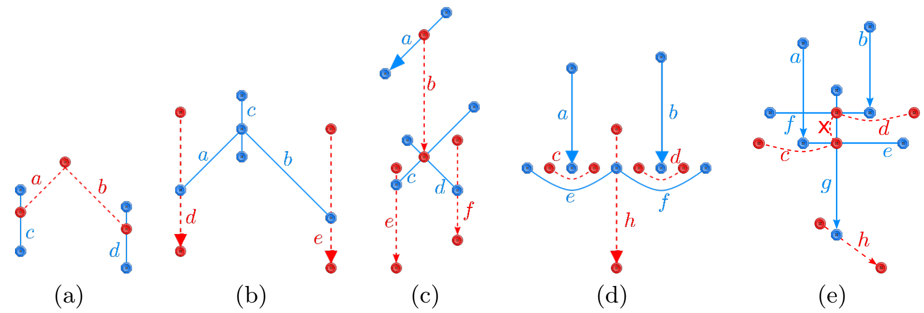

Fig. 7. (a) The red (dashed) variable gadget, (b) the blue (solid) variable gadget, (c) the split gadget, (d) the OR gadget, and (e) the AND gadget. Lines (or arcs used for clarity) indicate which buttons are aligned.

thus unlocking one of the two cuts, c or d, respectively, for Blue to follow up (and to propagate the value of the variable).

The **blue variable gadget** is shown in Fig. 7(b). Blue "sets the value" of the corresponding variable by choosing the first cut to be a (false) or b (true), and thus unlocking one of the two cuts, d or e, respectively, for the red player to follow up. Blue has one extra cut c that is used to pass the turn to Red. Alternatively, Blue can choose to start with the 3-button cut c and disallow Red from making any cuts in the gadget. In that case the corresponding variable cannot be used to satisfy Φ.

Fig. 7(d) depicts the **OR** gadget: if Blue cuts a or b (or both), Red can leave the gadget with cut h. Cuts a and b unblock cuts c and d, respectively, which in turn unblock e and f, respectively.

Fig. 7(e) depicts the **AND** gadget for two inputs. The proper way of passing the gadget: Blue makes both cuts a and b, and Red makes cuts c and d when they get unblocked, thus enabling Blue to make cut g and exit the gadget. However, Red could also take an "illegal" cut x, thus, unblocking two extra cuts, e and f, for the blue player, and, hence, putting Red at a disadvantage. Thus, if at any point in the game Red chooses (or is forced to) make cut x in any of the AND gadgets, the game result is predetermined, and Red cannot win on B.

Fig. 7(c) shows the **split** gadget; it enables us to increase the number of cuts leaving a variable and propagating its truth assignment. Blue's cut a unblocks Red's cut b, which unblocks both c and d. If Blue cuts c and d this enables Red to cut e and f, respectively. The gadget also exists with Blue and Red reversed.

A variant of the split gadget evaluates the formula Φ: cuts e and f are deleted. If the variable values are propagated to this gadget and Red is forced to make the cut b, Blue then gets extra cuts which Red will not be able to follow up.

The game progresses as follows: Blue selects an assignment to a blue variable. This unlocks a path of red-blue cuts that goes through some AND and OR gadgets and leads to the final gadget. As the order of the cuts in such a path is deterministic, and does not affect the choice of values of other variables, w.l.o.g., we assume that Red and Blue make all the cuts in this path (until it gets "stuck") before setting the next variable. The path gets stuck when it reaches some AND gadget for which the other input has not been cleared. The last cut in such a path was made by Red, thus afterwards it will be Blue's turn, and he may choose to make the leftover cut c from the variable gadget to pass the turn to Red.

If the final gadget is not unblocked yet, Red always has a cut to make after Blue makes a move, as there is the same number of blue and red variables. However, if Blue can force Red to make moves until the final gadget is reached, then Blue gets extra cuts; thus, Red will run out of moves and lose the game. Otherwise, if Blue cannot fulfill some AND or OR gadgets, the Red player will make the last move and win. Therefore, if Φ cannot be satisfied, Red wins. □

4.2 Any Color Games

Theorem 9. IMPARTIAL *two-player Buttons & Scissors is PSPACE-complete.*

Theorem 10. SCORING *two-player Buttons & Scissors is PSPACE-complete.*

We show that IMPARTIAL is PSPACE-complete, then use one more gadget to show SCORING is PSPACE-complete. We reduce from GEOGRAPHY[1], (PSPACE-complete [4]). We use Lemma 4 to start with low-degree GEOGRAPHY instances.

Lemma 4. GEOGRAPHY *is* PSPACE-*complete even when vertices have max degree 3 and the max in-degree and out-degree of each vertex is 2.*

The full version of this article proves Lemma 4 and Theorem 9 with these gadgets:

- **In-degree 1, out-degree 1**: The gadget for this is a pair of buttons such that removing the first pair frees up the second, as in Fig. 8(a).
- **In-degree 1, out-degree 2**: See Fig. 8(b) and the full version of this article.
- **In-degree 2, out-degree 1**: See Fig. 8(c) and the full version of this article.
- **In-degree 0**: The gadgets for this look just like the gadgets for the analagous in-degree 1 gadgets, but without the button pair for the incoming edge.
- **Out-degree 0**: Each edge is a button pair that won't free up other buttons.

To show SCORING is hard, we create a reduction where after each turn, that player will have cut the most buttons; the last player to move wins. This alternating-advantage situation is caused by an initial gadget. The optimal play sequence begins by cutting two buttons, then three, then three, then three a final time. After these four moves, the first player will have five points and the second player six. Each subsequent cut removes two buttons so each turn ends with the current player ahead.

Fig. 8(d) shows the starting gadget that sets up this initial back-and-forth. The color-f buttons will be the last two cut; the right-hand f button must be blocking the next gadget. Lemma 5 postulates that f will be last.

[1] Specifically, DIRECTED VERTEX GEOGRAPHY, usually called GEOGRAPHY.

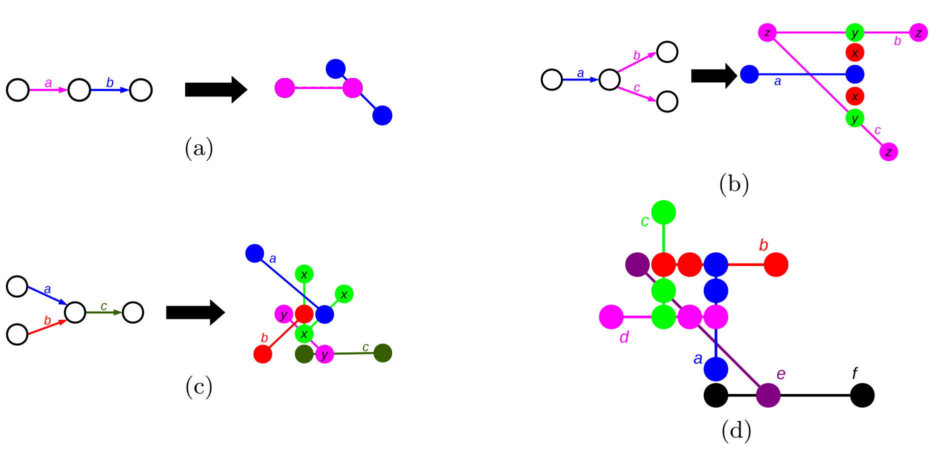

Fig. 8. Reduction gadgets for vertex with (a) one incoming arc and one outgoing arc, (b) one incoming arc and two outgoing arcs, and (c) two incoming arcs and one outgoing arcs. (d) The starting gadget for Scoring.

Lemma 5. *If a player has a winning strategy, then part of that winning strategy includes cutting all possible buttons of colors a, b, c, d, and e before cutting f.*

The full version of this article proves Lemma 5, and also shows how these lemmas provide Theorem 10.

5 Open Problems

Interesting problems for boards with a constant number of rows are still open. A conjecture for $m = 2$ rows appears below.

Conjecture 1. There is a polynomial time algorithm that removes all but s buttons from any full $2 \times n$ board with $C = 2$ colors for some constant s.

References

1. G. Cornuéjols, D. Hartvigsen, and W. Pulleyblank. Packing subgraphs in a graph. *Operations Research Letters*, 1(4):139–143, 1982.
2. H. Gregg, J. Leonard, A. Santiago, and A. Williams. Buttons & Scissors is NP-complete. In *Proc. 27th Canad. Conf. Comput. Geom.*, 2015.
3. D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.
4. D. Lichtenstein and M. Sipser. Go is polynomial-space hard. *J. ACM*, 27(2):393–401, 1980.
5. T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.