MIT Libraries | DSpace@MIT

MIT Open Access Articles

Single-Player and Two-Player Buttons & Scissors Games

Massachusetts Institute of Technology

Single-Player and Two-Player Buttons & Scissors Games

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.

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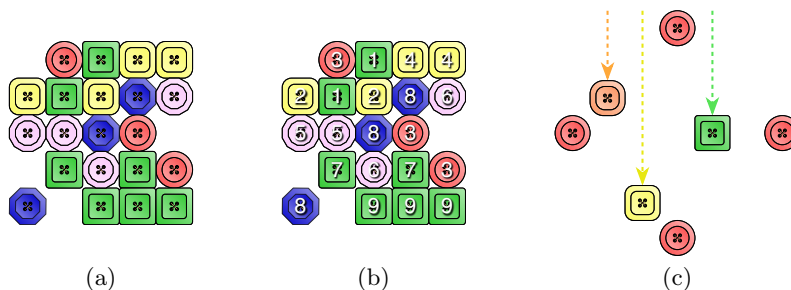
Fig. 1. (a) Level 7 in the Buttons & Scissors app is an $m \times n = 5 \times 5$ grid with $C = 5$ colors, each used at most $F = 7$ times; (b) a solution using nine cuts with sizes in $S = \{2,3\}$ and directions $d = *$ (no vertical cut is used); (c) a gadget used in Theorem 5.

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 $d \in \{*, \times, +, -\}$.
 5. The *cut size set* S limits feasible cuts to having size $s \in S$.

Each $d \in \{*, \times, +, -\}$ 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 $d = \times$. 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.   □

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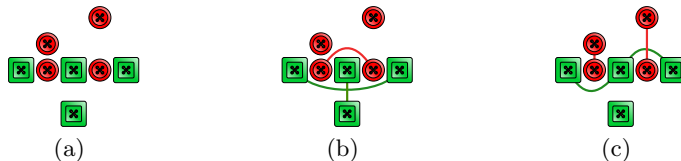
Fig. 2. Split gadget (a) and the two possible ways to clear it (b) and (c).

The split gadget has two additional effects beyond splitting. First, it changes the "signal color" from one of the two colors to the other. Second, it rotates the "signal direction" from vertical to horizontal: if the input is available to be used vertically by the split, then the outputs are available to be used horizontally as inputs to other gadgets. Therefore, we can restore the original signal color and direction by attaching further splits to the original split outputs.

We split each variable output as often as needed to reach the clause inputs, each of which will also be the output of a split rotated 90°from Fig. 2(a). If we obtain more split outputs than clause inputs for a variable, then we add a red button in the same column as a red pair on an unused split branch. Then all buttons in this column can be cleared regardless of how the split is cleared. □

Polynomial-Time Algorithm for 1-color and any cut directions. Given an instance B with $C = 1$ color and cut directions $d \in \{*, \star, +, -\}$, 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 $\triangle p_i vw$ 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.*
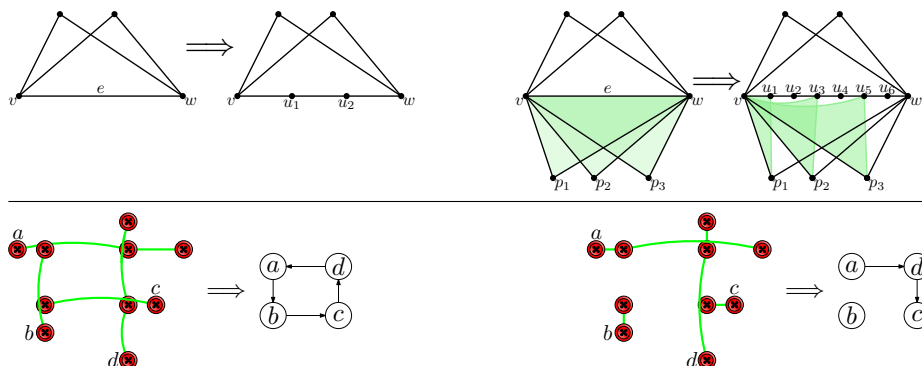
Fig. 3. Top-left: splitting 3-cycles when there are no adjacent triangles to edge e; top-right: splitting 3-cycles when e has adjacent triangles (shaded). Bottom-left: constructing G_c from four cuts blocking each other in a cycle; bottom-right: constructing G_c from the same cuts after reassigning the blocking buttons

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 $1 \leq j < k$;
- 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 $1 \leq j < i$, (u_{2j}, u_{2j+1}) for $i \leq j < k$, 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 $1 \leq j \leq k$.

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'.

Now, given a perfect matching M' in G' we show how to construct a perfect matching M in G. Again, consider an edge e in G that is replaced by several edges in G'. If e is not adjacent to any triangles in G, then
- if u_1 is covered by edge (v, u_1) in M' then u_2 has to be covered by edge (u_2, w), therefore we can add edge e to M (all v, w, u_1, and u_2 are covered by M', and both v and w are covered by e in M);
- if u_1 is covered by edge (u_1, u_2) in M' then v and w have to be covered by other edges or triangles in M', and v and w will be covered by the corresponding edges or triangles in M.

In the second case, when e is adjacent to some triangles in G,

- if $\triangle p_i v u_{2i-1} \in M'$ for some i then add $\triangle p_i v w$ to M. 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) are forced in M', and thus both nodes v and w are covered. Both nodes v and w are covered by the corresponding triangle $\triangle p_i v w$ in M;
- if none of the triangles $\triangle p_i v u_{2i-1}$ is in M' then consider node u_1 in G':
  - if u_1 is covered by edge (v, u_1) in M' then edges (u_{2j}, u_{2j+1}) for $1 \le j < k$ and (u_{2k}, w) are forced in M'. Therefore we can add edge e to M (all v, w, u_j are covered by M', and both v and w are covered by e in M);
  - if u_1 is covered by edge (u_1, u_2) in M' then edges (u_{2j-1}, u_{2j}) for $2 \le j \le k$ are forced in M', therefore v and w have to be covered by other edges or triangles in M', and v and w will be covered by the corresponding edges or triangles in M.

$\square$

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.

To do so we perform the following two steps:

- Repeat: choose a cycle in G_c and for every edge (c_1, c_2) in it reassign the button of c_2 that is blocking cut c_1 to c_1. After one step
  - the total number of buttons in all the cuts stays the same,
  - if cut c_2 consisted of two buttons, or if cut c_2 consisted of three buttons and the button blocking cut c_1 was not in the middle of c_2, then after reassigning the buttons the length of c_2 decreases,
  - if cut c_2 consisted of three buttons and the button blocking cut c_1 was in the middle of c_2, then after reassigning the buttons the direction of the edge (c_1, c_2) changes to (c_2, c_1).
- After the previous step can no longer be applied to G_c such that the total length of the cuts decreases, there can only be cycles left in G_c that consist of 3-cuts with the blocking buttons being the middle ones. Then for any edge (c_1, c_2), if we reassign the middle button of c_2 to c_1, c_2 will become a 2-cut, and c_1 will have four buttons, and therefore can be split into two 2-cuts. The direction of the corresponding edge will also change its direction. In this way the rest of the cycles can be removed from G_c.

To summarize, a solution to 1-color Buttons & Scissors level B can be found, if it exists, with following algorithm:

1. Convert B to hypergraph G that encodes all possible cuts of length two and three buttons using the allowed cut directions.
2. Convert G to G' that contains no 3-cycles that are not triangles, and find a matching in G'.
3. Construct the directed graph G_c that encodes which cuts from the matching are blocking each other and remove all the cycles from G_c by reassigning some buttons to other cuts.
4. Extract a partial order from G_c that will give a proper order in which the cuts can be applied to solve B.

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

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

3.3 Frequency of Colors

Theorem 4. *Buttons & Scissors with maximum color frequency $F = 3$, i.e. $B\&S[n \times n, \infty, 3, *, \{2,3\}](B)$, 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. *Buttons & Scissors with maximum color frequency $F = 4$, i.e. $B\&S[n \times n, \infty, 4, *, \{2,3\}](B)$, is NP-complete.*

Proof. The puzzle with $F = 7$, i.e. $B\&S[n \times n, \infty, 7, *, 2](B)$, was proven NP-complete in [2] via 3-SAT whose clauses have literals of distinct variables. (The construction created boards that do not have buttons of the same color on any diagonal, so the NP-completeness was also established for these parameters and both $d = *$ and $d = +$ in [2] .)

We obtain hardness for $F = 4$ by modifying the original proof's clause gadget. The top of Fig. 4 illustrates the original gadget for clause $C_x = L_i \vee L_j \vee L_k$, where L_i is either the positive literal for variable V_i or the negative literal $\neg V_i$, and similarly $L_j \in \{V_j, \neg V_j\}$ and $L_k \in \{V_k, \neg V_k\}$. Included in this gadget are the following buttons:

– Two clause buttons $\circledx_L$ and $\circledx_R$. The L and R subscripts denote $\underline{L}$eft and $\underline{R}$ight, respectively.
– Literal instance buttons $\boxed{\text{i,x}}_M$, $\boxed{\text{j,x}}_M$, and $\boxed{\text{k,x}}_M$. The parameter $\boxed{\text{,x}}$ refers to the clause C_x under consideration, and we omit it from Fig. 4 and the discussion below. The M subscripts denote $\underline{M}$iddle.

By convention different shapes and interior labels different colored buttons, whereas subscripts do not alter the color. Thus, $\circledx_L$ and $\circledx_R$ have the same color, whereas $\underline{\boxed{\mathrm{i}}}_M$, $\underline{\boxed{\mathrm{j}}}_M$, and $\underline{\boxed{\mathrm{k}}}_M$ are all distinct. Each of the middle buttons can be removed by a vertical cut as denoted by the downward arrows. This original clause gadget has the following property: $\circledx_L$ can be removed if and only if at least one of $\underline{\boxed{\mathrm{i}}}_M$, $\underline{\boxed{\mathrm{j}}}_M$, or $\underline{\boxed{\mathrm{k}}}_M$ is removed by vertical cut.

The bottom of Fig. 4 illustrates the new gadget for the same clause $C_x = L_i \vee L_j \vee L_k$. If n is the number of variables in the 3-SAT instance, then original gadget was contained in one row and $4n + 12$ columns, including $n + 1$ blank columns to clarify the presentation. The new gadget uses $8n + 5$ rows and $8n + 5$ columns (see the bottom of Figure 2 in [2]). In this gadget we add two buttons to those discussed above:

– Two additional clause buttons $\circledx_T$ and $\circledx_B$. The T and B subscripts denote $\underline{T}$op and $\underline{B}$ottom, respectively.

We now prove that the new gadget has the same property as the old gadget. We consider four cases that cover all possibilities:

– If $\underline{\boxed{\mathrm{i}}}_M$ is cut vertically, then cut $\circledx_L$ and $\circledx_T$ (and then $\circledx_B$ and $\circledx_R$);
– If $\underline{\boxed{\mathrm{j}}}_M$ is cut vertically, then cut $\circledx_L$ and $\circledx_B$ (and then $\circledx_T$ and $\circledx_R$);
– If $\underline{\boxed{\mathrm{k}}}_M$ is cut vertically, then cut $\circledx_L$ and $\circledx_R$ (and then $\circledx_T$ and $\circledx_B$);
– If none of these buttons is removed by a vertical cut, then $\circledx_L$ cannot be removed since the only other buttons of the same color are blocked.

Therefore, the new gadget is a perfect replacement for the previous clause gadget.

This substitution increases the frequency of each clause button from 2 to 4, and decreases the frequency of each instance literal button color to 3. Thus, the maximum frequency is reduced from $F = 7$ to $F = 4$ by Remark 6 in [2]. □

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. *Buttons & Scissors for 1-color with cut sizes* $\{3, 4\}$, *i.e.* $B\&S[n \times n, 1, \infty, *, \{3, 4\}](B)$, *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 [4].

An instance F of the PLANAR 3-SAT problem is a Boolean formula in 3-CNF consisting of a set $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of m clauses over n variables $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$. Clauses in F contain variables and negated variables, denoted as *literals*. The variable-clause incidence graph $G = (\mathcal{C} \cup \mathcal{V}, E)$ is planar, where $\{C_i, x_j\} \in E \Leftrightarrow x_j$ or $\neg x_j$ is in C_i, and with all variables connected in a cycle. It is sufficient to consider formulae where G has a rectilinear embedding, see Knuth
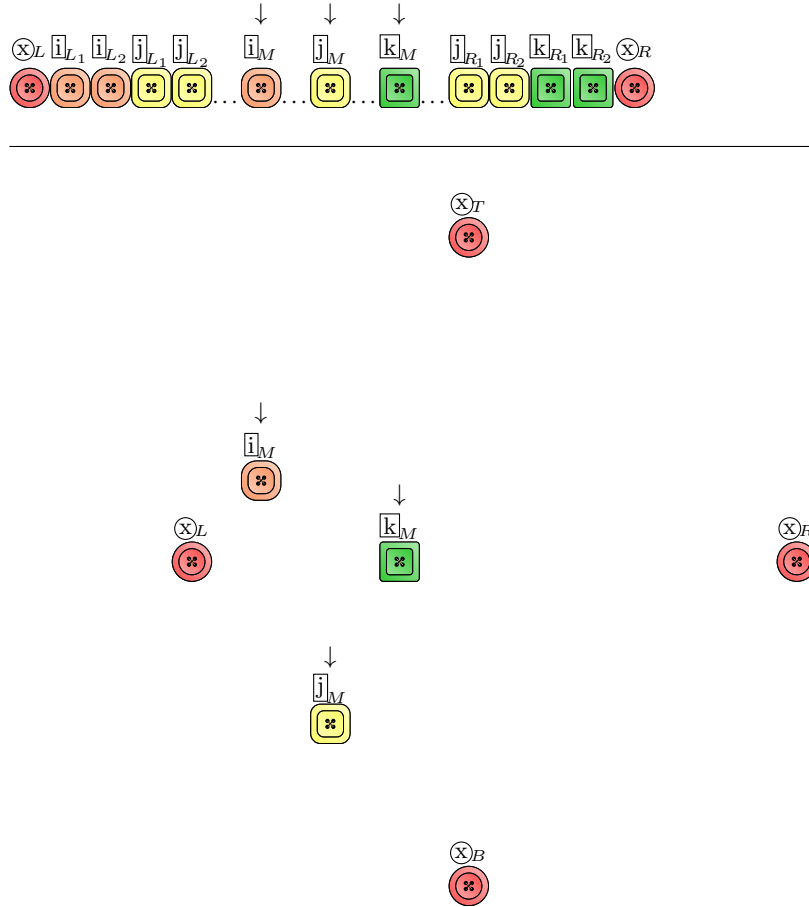
Fig. 4. The original (top) and new (bottom) gadget for clause $C_x = L_1 \lor L_2 \lor L_3$. In both cases the leftmost circular button $\circledx_L$ can be removed if and only if at least one of the three middle buttons $\boxed{i}_M$, $\boxed{j}_M$, $\boxed{k}_M$ is removed by a vertical cut, as denoted by downward arrows. Labels $\boxed{,x}$ are omitted from all of the square buttons to save space, and the $\cdots$ denote empty squares between the three middle buttons.

and Raghunathan [3]. 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 continue by describing each of the gadgets necessary.

The **variable gadget** is shown in Fig. 5(a). The buttons to the right are positioned such that they can only be cut with either horizontal cuts of size 3 (black) or diagonal cuts of size 3 (gray). , see Fig. 5(b). To obtain a feasible cutting pattern for the rest, this leaves only two cut possibilities for the four vertically aligned buttons of the variable gadget: cutting the topmost three buttons, in which case further cuts in the wire are enforced to be horizontal, or cutting the two topmost and the bottom button (which is possible by executing the first diagonal cut first), in which case further cuts in the wire are enforced to be diagonal. These exactly two feasible solutions of the variable gadget correspond to a truth setting of "true" (black) and "false" (gray) of the variable.

The **bend gadget**, shown in Fig. 5(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. Note that the three lower of the four vertically aligned buttons, , $b_1, \ldots, b_4$,, have to be removed with the same vertical cut, as none of them is aligned with any other button pair on the board.

The **split gadget**, shown in Fig. 6(b), enables us to increase the number of wires leaving a variable and propagating its truth assignment. The last diagonal cut (gray) and the last horizontal cut (black) of the input wire from the left only feature two buttons in this wire, i.e., if we want to make a feasible cut, we need to pick up another button, a single button that starts the two output wires. If we enter with the black cut pattern, the upper output wire's single button is cut,

(a)

(b)
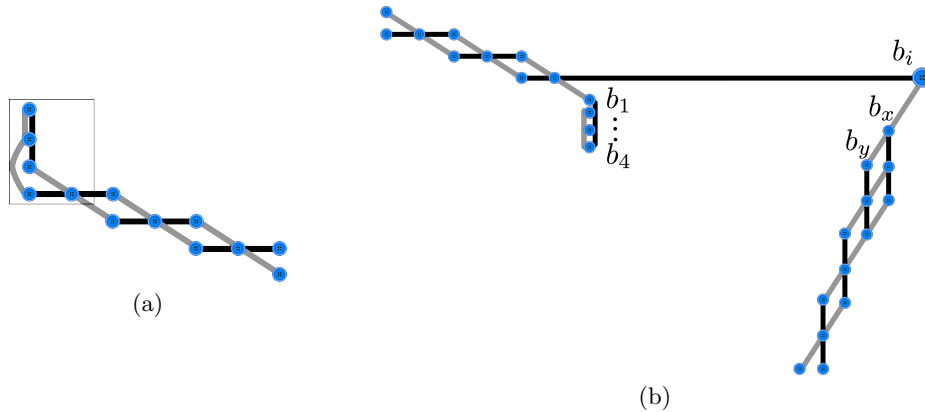
Fig. 5. (a) The only two cut possibilities in the variable gadget (shown in black and gray), corresponding to truth assignments of "true" and "false", respectively. (b) The bend gadget for the 1-color case.
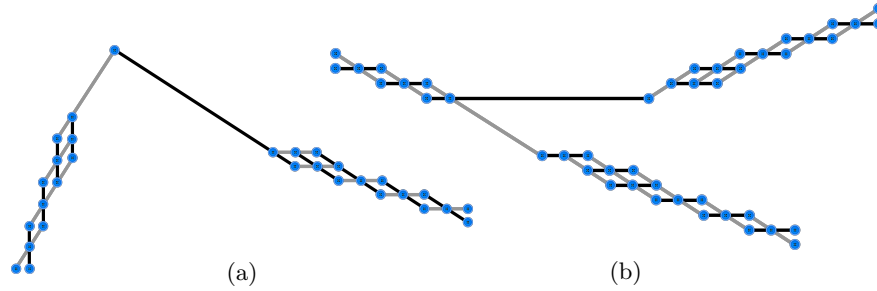
Fig. 6. (a) The not gadget, negating the input truth assignment, for the 1-color case. (b) The split gadget for the 1-color case.

enforcing black, horizontal cuts in the rest. In the lower output wire, the single button still needs to be cut. Except for the two buttons of the last diagonal of the input wire, it is only aligned with the other uppermost buttons of this output wire, thus, the horizontal cuts need to be made, enforcing black, horizontal cuts in the rest of the wire. An analogous argument is used for the gray cut pattern.

The **not gadget**, shown in Fig. 6(a), enables us to reverse the truth assignment in a variable wire. (That is, the cut pattern switches from vertical/horizontal to diagonal and vice versa.) Again, the last diagonal cut (gray) and the last horizontal cut (black) of the input wire from the left only feature two buttons in this wire. Moreover, there is again a single isolated button. If the gadget is entered with the black, vertical cut pattern, the isolated button cannot be cut with any cut from the input wire. The only remaining possibility is to cut the isolated button with the two leftmost diagonal buttons of the output wire, enforcing diagonal cuts in the rest of the wire. If the gadget is entered with the gray, diagonal cut pattern, the isolated button needs to be cut with the last diagonal wire cut (as otherwise its size of two would render it infeasible). In the output wire the first diagonal cut would be to short, enforcing horizontal cuts in the rest of the wire.

The **clause gadget** is shown in Fig. 7. Wires from literals approach the gadget along three diagonals. The clause has one isolated button (shown enlarged), and two additional buttons for each wire that build a row of four aligned buttons with the last two buttons of the wire. When a gray cut direction is used, the two lower plus the topmost of this row of four combine for a feasible cut. If a black cut direction is used, the three lower buttons of this row allow a feasible cut. But this still leaves the isolated button to cut: Only if in at least one of the input wires the black cuts, corresponding to a truth setting of "true" for the literal, are used, this button can be added to the last black cut of the wire (extending its size from 3 to 4). If all literals are set to "false", i.e., along all three input wires the gray cut pattern is used, cutting the isolated button with two others along one of the horizontal/vertical axis, leaves at least one isolated button, rendering the complete board infeasible, as we are not left with an empty board.
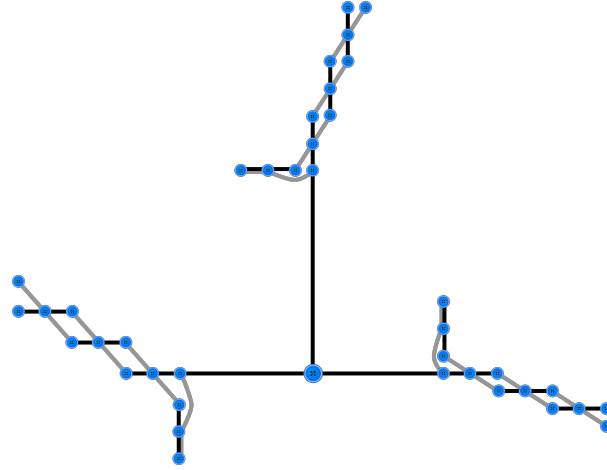
Fig. 7. The clause gadget for the 1-color case.

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. □

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.

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
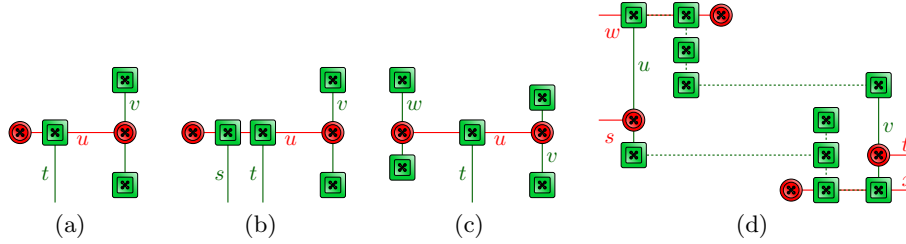
Fig. 8. Three types of nodes: (a) in-degree 1 (tu) and out-degree 1 (uv); (b) in-degree 2 (su and tu) and out-degree 1 (uv); (c) in-degree 1 (tu) and out-degree 2 (uv and uw). In (d) the nodes u and v are linked in S and we can choose to remove u or v.

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. 8(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. 8(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. 8(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.

We embed G in this fashion. What is left is to create a mechanism to remove vertices from G as dictated by S. We assume all vertices in S correspond to green button pairs. Suppose we have edges su, uw, tv, and vx, and u and v are linked in S. We place six additional green buttons on two vertical lines as illustrated in Fig. 8(d). Three buttons are placed such that one is on the same horizontal line as the top end of u and the other is on the same horizontal line as the top end of

v, and the third is not on a common line with any other node. The third button assures we must use this vertical cut, but we can choose whether to remove the top of u or the top of v. Similarly, we create a construction that allows us to remove the bottom of u or the bottom of v. Since u and v are ordinary degree 2 nodes, we never want to remove the top of one and the bottom of the other, since this would yield an unsolvable situation. Note that, in case we are allows to make cuts of size 3, we can also remove all six extra nodes completely. This does not affect the reduction, since it never pays to do so. □

Lemma 3. *SAT reduces to Graph Decycling.*

Proof. Create a cycle for each clause, and mark a vertex in it for each literal in the clause: one of these needs to be removed to make the graph acyclic. Create a pair of removable vertices for each variable. Duplicate literals with little bow-tie gadgets: two cycles that share a vertex. If shared vertex is *not* removed, we must remove two more vertices, one from each cycle. □

Lemmas 2 and 3 give Theorem 7.

Theorem 7. *Buttons & Scissors for 2-colors with only cuts of two directions of size 2, i.e., $B\&S[n \times n, 2, \infty, +, \{2\}](B)$ is NP-complete.*

4 Two-Player Games

We consider four different types of two-player Buttons & Scissors variants, in two different types:
 – Games where each player may only make cuts on specific colors:
     1. One player may only cut Blue buttons, while the other player may only cut Red buttons.
     2. Either player may cut Green buttons.
  – Games where players are not restricted to the colors they may cut:
     1. The last player who makes a feasible cut wins. (IMPARTIAL)
     2. Players keep track of the total number of buttons they've cut. After no more cuts can be made, the player with the highest number of buttons cut wins. (SCORING)
   In the following sections, we show that all four 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.*
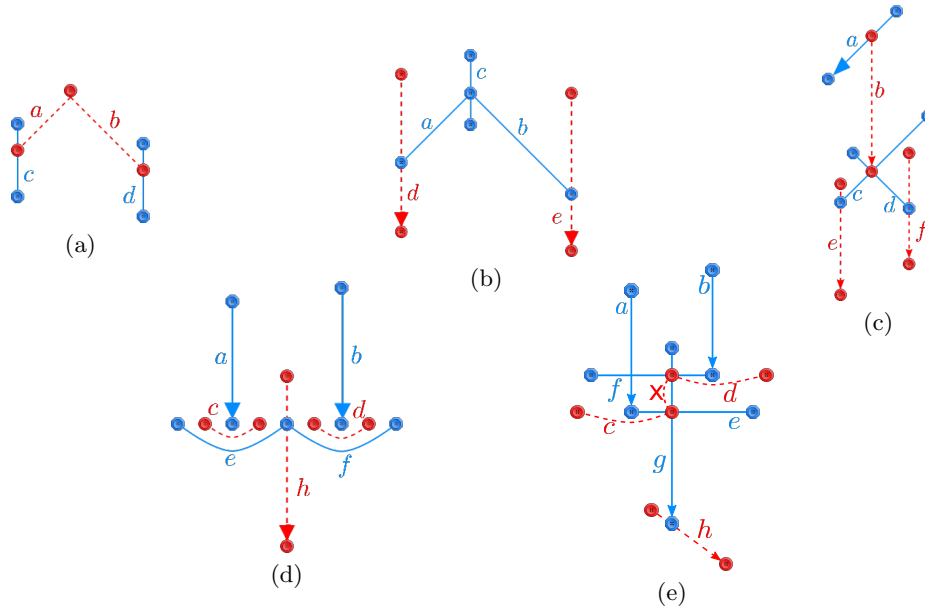
Fig. 9. (a) The red (dashed) variable gadget, (b) the blue (solid) variable gadget, (c) the split gadget, (d) the OR gadget, and (e) the AND gadget. Lines (or arcs used for clarity) indicate which buttons are aligned.

Proof. The proof is by reduction from $G_{\%free}$(CNF) [6]: 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. See Fig. 10 for a full example.

The **red variable gadget** is shown in Fig. 9 (a). Red "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, 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. 9 (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.
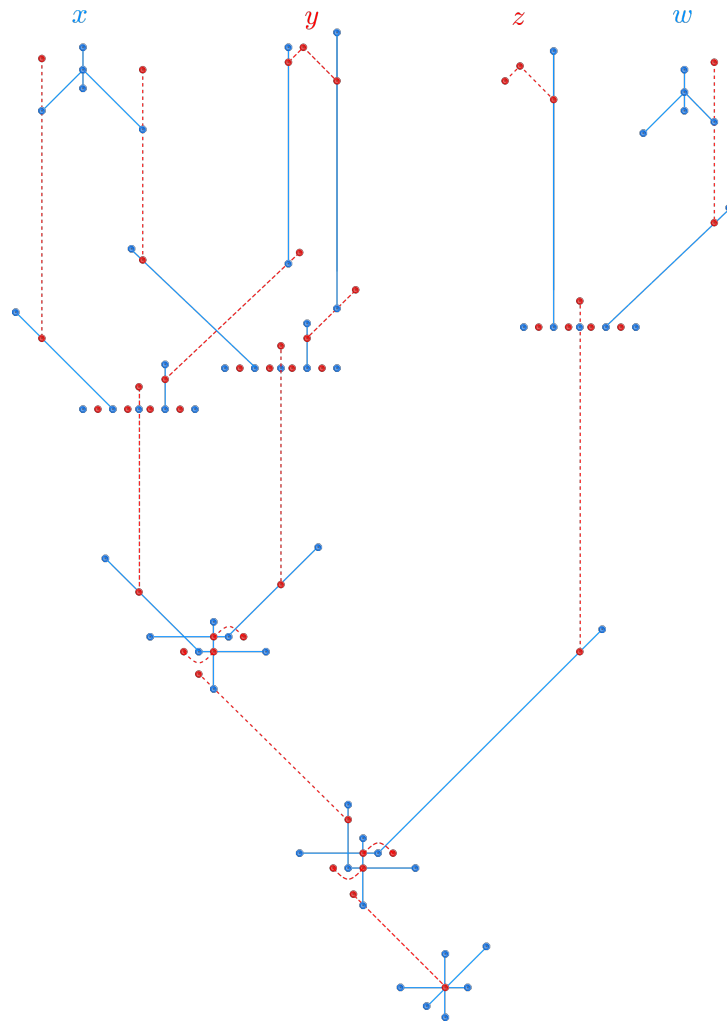
Fig. 10. Example for the construction from the proof of Theorem 8: we consider $(x \vee y) \wedge (\neg x \vee \neg y) \wedge (z \vee w)$.

Fig. 9 (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. 9 (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. 9 (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 (*Impartial*) is PSPACE-complete.*

Theorem 10. *Scoring two-player Buttons & Scissors (*Scoring*) 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 [5]). We use Lemma 4 to start with low-degree Geography instances.

[1] Specifically, Directed Vertex Geography, usually called Geography.

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.*

Proof. The proof is simply a reduction that adds some vertices to reduce the degree. To reduce the in-degree, we use the gadget shown in Fig. 11 to iteratively reduce the number of edges entering any vertex, v, with in-degree above 3. Notice that the player who moves to v will be the same in both games.
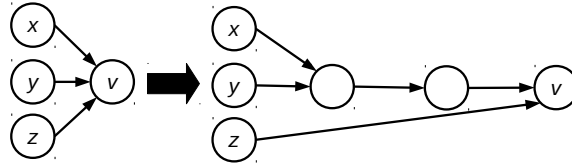

Fig. 11. Reducing the in-degree to v by 1 in GEOGRAPHY.

To reduce the out-degree, use use an analagous construction, as shown in Fig. 12. Just as in the previous gadget, we ensure that the same player will arrive at x, y, and z as in the original game. Additionally, we ensure that the player that leaves v still makes the choice between x and y in the reduced gadget.
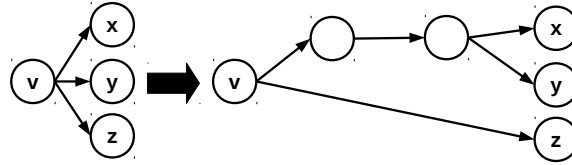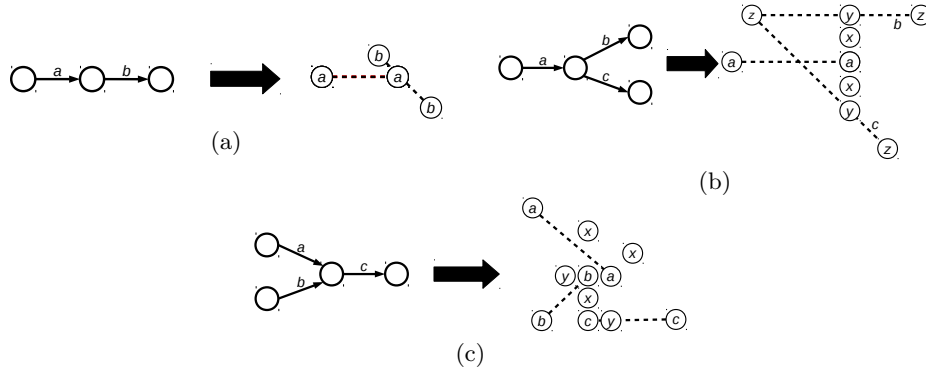

Fig. 12. Reducing the out-degree of v by 1 in GEOGRAPHY.

After repeatedly reducing the in and out degree to at most 2, some vertices may still have both in and out degree 2. To fix this, we simply add two vertices as shown in Fig. 13. □
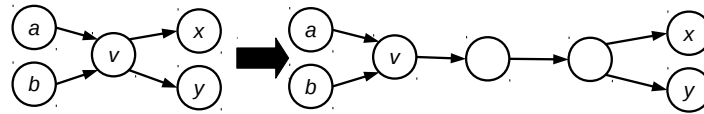
Fig. 13. Final step for vertices, v, with both in and out-degree of 2.

Fig. 14. 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.

The following gadgets prove Theorem 9.
- **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. 14(a).
- **In-degree 1, out-degree 2**: The gadget for this is shown in Fig. 14(b). Incoming edge a is represented by the button pair colored a. The outgoing edges b and c are represented by three non-colinear buttons colored z. After the first player removes the button pair a, the second player will cut the pair colored x, and the first player will cut the y pair. Just as the second player would choose between moving to b or c in the GEOGRAPHY position, so do they choose between cutting the z pair labelled b or the z pair labelled c. Once one of those pairs is chosen, the other z-colored button is stranded.
- **In-degree 2, out-degree 1**: The gadget for this is shown in Fig. 14(c). Incoming GEOGRAPHY arcs a and b are represented by button pairs a and b, respectively. No matter which of those two are cut, it allows the x pair to be cut, followed by the y pair, then the c pair, representing the c arc in the GEOGRAPHY position. The player that cuts the c pair will be the opposite of the player that cuts the a or b pair, just as in the GEOGRAPHY position.
- **In-degree 0**: The gadgets for this look just like the gadgets for the analagous in-degree 1 gadgets, but with the button pair missing that represents the incoming edge.
- **Out-degree 0**: Each edge is a button pair that doesn't free up other buttons upon removal.

Proof. (For Theorem 9.) We reduce from our restricted-degree version of GE-OGRAPHY. The reduction replaces each GEOGRAPHY vertex, v, with the coore-sponding gadget described above. Each arc between two vertices becomes a button pair (as described in the reduction) used in the gadgets for both vertices. ☐

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 first 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.
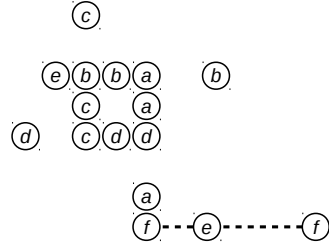
Fig. 15. The starting gadget for SCORING.

Fig. 15 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.

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.*

Proof. The first player only earns two points by cutting any of a, b, c, or d. After this first move, there is always another three-point move to take until all of a, b, c, and d are gone. Thus there are three chances to earn three points. This gadget is the only place for a player to earn three points in one turn, so the second player must take two of those three triple button cuts. If they don't, they will never have more points than the first player. The second player must spend each of their first two moves cutting three buttons.

Similarly, the first player cannot spend the first three turns earning only two points each time. If they do, then the second player can take three on each of their first three turns, making the score 6 to 9 in favor of the second player. Since all moves thereafter only earn 2 points, the second player will always have more points. The first player must cut three buttons on their second or third turn.

Thus, after the first five turns (first-second-first-second-first) the score must be 7 to 6 in favor of the first player. The only way for this to happen is if f is cut after a, b, c, d, and e. □

With the lemmas in place, we can prove Theorem 10.

Proof. (Proof of Theorem 10.) We reduce again from the low-degree GEOGRA-PHY game considered in Lemma 4. Use the reduction to IMPARTIAL described earlier to generate a Buttons & Scissors board B. Add the starting gadget for

Scoring (shown in Fig. 15) to create B' so that the buttons marked 'f' block the first move that can be made on B.

After the buttons 'f' are cut, the second player will have exactly one more point than the first. Throughout all the gadgets used in the PSPACE-hard reduction for Impartial, each cut removes two buttons. Thus the last player who can win playing Impartial on B can win playing Scoring on B'. □

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. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal of Discrete Math.*, 5(3):422–427, 1992.
4. D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.
5. D. Lichtenstein and M. Sipser. Go is polynomial-space hard. *J. ACM*, 27(2):393–401, 1980.
6. T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.