Computational Complexity of Buttons & Scissors

Harrison Gregg, Jody Leonard, Aaron Santiago, and Aaron Williams
hgregg11@simons-rock.edu jleonard11@simons-rock.edu asantiago11@simons-rock.edu
haron@uvic.ca
Division of Science, Mathematics and Computing, Bard College at Simon’s Rock

Abstract. Buttons & Scissors is a popular single-player puzzle game by KyWorks. The level is played on an \( n \times n \) grid, where each position is empty or has a single coloured button sewn onto it. The player’s goal is to remove all of the buttons using a sequence of horizontal, vertical, and diagonal scissor cuts. Each cut begins and ends at distinct buttons of the same colour, and cannot contain any buttons of another colour. We prove that deciding whether a given level can be completed is NP-complete. In fact, we prove that it is NP-hard to determine the number of removable buttons within any constant factor. On the other hand, we formulate the game as an integer program and use it to solve the hardest levels of the game in under one minute per level. We also provide a positive result for a colour-restricted version of the game.

Keywords: computational complexity, NP-complete, inapproximability, integer programming, pencil and paper game.

1 Introduction

Buttons & Scissors is a single-player puzzle game by KyWorks that was introduced for mobile devices in 2013. The goal is to remove all of the buttons from an \( n \times n \) grid using horizontal, vertical, and diagonal scissor cuts, where each cut removes at least two buttons of an individual colour and no buttons of any other colour; see Figure 1 for an illustration and Section 2 for a formal description. The game is both popular and widely played, having a 4.5-star rating on Amazon.com (based on 350+ reviews) and over 170,000 downloads on Google Play. It has also spawned copycat games like Scissors & Buttons Cutter by Tuyen Hoang.

Buttons & Scissors is reminiscent of many grid-based pencil and paper games that have been popularized by the Japanese company Nikoli. To analyze the computational complexity of an individual game, it is necessary to generalize certain aspects of the game, including the grid size. For example, the generalized version of Sudoku involves an \( n^2 \times n^2 \) grid with blocks of size \( n \) and integers from 1 to \( n^2 \) (with the standard version having \( n = 3 \)). The book Games, Puzzles, and Computation by Hearn and Demaine [3] recounts known results for many grid games including the NP-completeness of Sudoku, Kakuro, Light Up, Slitherlink, Nonogram, Fillomino, Lits, Masyu, Hitori, Nurikabe, Tentai Show, Hiroimono, Heyawake, and Morpion Solitaire. More recently, the NP-completeness of Shakashaka was established by Demaine, Okamoto, Uehara, and Uno [1].
Some of these grid puzzles have been reborn as mobile applications. For example, *Flow Free* by Big Duck Games is a functionally equivalent remake of the Numberlink puzzle. The computational complexity of some other mobile grid puzzles have been analyzed, including the popular *Candy Crush Saga* (see Walsh\cite{5}, and Guala and Leucci and Natale\cite{2}).

The computational complexity of these grid puzzles can often be determined through reductions from another grid puzzle. In this paper we instead reduce 3-SAT directly to Buttons & Scissors. We also provide an inapproximability result, which was not done in the aforementioned articles. More specifically, we prove that it is NP-hard to approximate the number of removable buttons within any constant factor.

Although Buttons & Scissors is intractable in theory, we provide an integer programming model that solves small levels quickly. In particular, we solve the final 25 levels in the Buttons & Scissors application in under one minute each on average\footnote{This test was performed with a Macbook Air 2013 model with a dual-core Intel i5 processor and 4GB of RAM.}. The integer program also verified that one of the levels on the Scissors & Buttons Cutter clone is impossible\footnote{An iTunes review from DebateCate states “...some of the puzzles are impossible to complete. Not because they’re difficult, but because they are impossible. For instance, level 45 is impossible to solve.”}.

Section \ref{buttons-scissors-definitions} formalizes our generalized Buttons & Scissors game. Section \ref{np-complete} proves that the problem is NP-complete, and then Section \ref{in approximability} provides our inapproximability result. Section \ref{integer-programming} provides our integer programming model. Section \ref{conclusion} ends concludes open problems and a small positive result.

\section{Buttons & Scissors Definitions}

In this section we formalize the Buttons & Scissors game, our terminology, and our decision problem and optimization problem.

\renewcommand{\thefigure}{1}

\begin{figure}[h]
\centering
\subfloat[Fig. 1. (a) The first level in Buttons & Scissors and (b) its unique solution showing the buttons removed during each of four scissor cuts.]{
\begin{minipage}{0.45\textwidth}
\includegraphics[width=\linewidth]{buttons-scissors.png}
\end{minipage}\hfill
\begin{minipage}{0.45\textwidth}
\includegraphics[width=\linewidth]{buttons-scissors-solution.png}
\end{minipage}}
\end{figure}
A board $B$ is an $n$-by-$n$ matrix with entries in $\{0, 1, \ldots, c\}$, where each 0 is empty and each $i > 0$ is a button of colour $i$. By convention we omit the 0 entries when we display the matrices. For example, the level in Figure 1(a) is represented by the following matrix on the left, which we display as the matrix on the right:

\[
\begin{bmatrix}
0 & 4 & 0 & 3 & 0 \\
0 & 5 & 3 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 \\
0 & 5 & 0 & 0 & 0
\end{bmatrix}
\quad \begin{bmatrix}
4 & 3 \\
5 & 3 \\
5 & 1 & 4 & 1 \\
3 & 5
\end{bmatrix}.
\]

A cut is a sequence of pairs

$$(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$$

with $k \geq 2$ which satisfy one of the following directional properties

1. $y_i = y_{i+1}$ and $x_i = x_{i+1} + 1$ for $1 \leq i < k$ (a vertical cut);
2. $x_i = x_{i+1}$ and $y_i = y_{i+1} + 1$ for $1 \leq i < k$ (a horizontal cut);
3. $x_i = x_{i+1} + 1$ and $y_i = y_{i+1} + 1$ for all $1 \leq i < k$ (a diagonal cut);
4. $x_i = x_{i+1} + 1$ and $y_i = y_{i+1} - 1$ for all $1 \leq i < k$ (a diagonal cut).

Notice that we only consider diagonal cuts that run at $45^\circ$ or $-45^\circ$. Any pair $(x_i, y_i)$ and $(x_j, y_j)$ that lies on the same vertical, horizontal, or diagonal line is co-linear and we extend this terminology to larger sets if they are pairwise co-linear. For notational convenience, we sometimes refer to a cut using the reverse sequence $(x_k, y_k), (x_{k-1}, y_{k-1}), \ldots, (x_1, y_1)$.

With respect to an $n$-by-$n$ board $B$, a cut is valid if

a) $1 \leq x_i \leq n$ and $1 \leq y_i \leq n$ for all $1 \leq i \leq k$, and
b) $B[x_1, y_1], B[x_2, y_2], \ldots, B[x_k, y_k]$ contains at least two non-empty entries, and all non-empty entries are buttons with the same colour.

When a valid cut is applied to a board $B$, the result is a board $B'$ that is obtained from $B$ by setting $B'[x_i, y_i] = 0$ for all $1 \leq i \leq k$. In other words, the cut removes all buttons along its path.

A solution to a given board $B_1$ is a sequence

$$(B_1, c_1, B_2, c_2, \ldots, B_t, c_t, B_{t+1})$$

where each $c_i$ is a valid cut for the board $B_i$, and $B_{t+1}$ is the empty $n$-by-$n$ matrix of all zeroes. The length of this solution is $t$.

Given these formal definitions, we will refer to matrix values and buttons interchangeably throughout the text. Now we state our decision and optimization problems.

---

3 We encode the Buttons & Scissors palette of red, orange, yellow, green, blue, violet, and white buttons as 1 through 7 respectively.
**Decision Problem**  \( \text{B\&S}(B) \)  
Input: An \( n \)-by-\( n \) board \( B \).  
Output: True if \( B \) has a solution, and False otherwise.

**Optimization Problem**  \( \text{maxB\&S}(B) \)  
Input: An \( n \)-by-\( n \) board \( B \).  
Output: The maximum number of buttons \( m \) that can be removed from \( B \) by a sequence of valid cuts.

**Remark 1.** \( \text{B\&S}(B) \) is in NP since a sequence of valid cuts \( c_1, c_2, \ldots, c_t \) serves as a certificate that can be verified in \( O(n^2) \)-time when \( B \) is \( n \)-by-\( n \).

### 3 NP-Completeness

In this section, we prove that \( \text{B\&S}(B) \) is NP-Complete by a reduction from 3-SAT \[4\]. In particular, we refer to 3-SAT as the NP-complete version in which each clause contains exactly three literals, and the literals in each clause are distinct. We introduce gadgets for variables, literals, disjunction, and conjunction. Then we stitch the gadgets together at the end of the section. Figure 2 illustrates the layout of the entire reduction.

#### 3.1 Decision

Our binary decision gadget involves four buttons of the same colour:

![Decision Gadget Image]

Observe that these buttons can be removed in one of two ways. In particular, \( A \) can be cut either with \( T \) or \( F \). This dichotomy will correspond to a variable assignment in 3-SAT. Our reduction uses distinct button colours for each decision gadget, so they are self-contained, except for buttons of other colours that will separate the button on the right from the others. Furthermore, the space between the button on the right and the others does not affect these observations. We define this gadget below and use it vertically in Figure 2.

**Definition 1.** A decision gadget is the following matrix

\[
\begin{bmatrix}
A & T & F & \cdots & C \\
\end{bmatrix}
\]

where \( A = T = F = C > 0 \).

With respect to this gadget, button \( A \) is the assignment button, button \( T \) is the true button, button \( F \) is the false button, and button \( C \) is the cleanup button.
Fig. 2. The Buttons & Scissors level above can be solved if and only if there is a satisfying assignment to the 3-SAT instance \((B \lor \neg B \lor C) \land (A \lor B \lor \neg D) \land (\neg B \lor \neg C \lor \neg D) \land (A \lor B \lor C) \land (\neg A \lor \neg C \lor \neg D) \land (B \lor \neg B \lor \neg C) \land (B \lor C \lor \neg C)\). This image comes from an online implementation of the reduction that can be viewed at [http://jabdownsmash.com/button3sat/index.html](http://jabdownsmash.com/button3sat/index.html). (The colours can be cycled in this online implementation to avoid ambiguity between similar but distinct shades.)
Remark 2. A decision gadget’s buttons are clearable in exactly two ways:

1. A cut containing $A$ and $T$, and a cut containing $C$ and $F$;
2. A cut containing $A$ and $F$, and a cut containing $C$ and $T$.

3.2 Literal

The decision gadget allows a 3-SAT variable $V$ to be set to true or false. In other words, the decision gadget ensures that either $V$ or $\neg V$ is true. The following gadget allows us to replicate the removal of the corresponding $T$ or $F$ button $k$ times, where $k$ is the number of clauses that contain the literal $V$ or $\neg V$. We illustrate the gadget for $k = 3$ below:

![Diagram of the literal gadget](image)

Notice that none of the buttons above can be removed. However, if the decision button is removed, then all of the buttons can be removed. Furthermore, this holds even when any subset of the clause buttons on the right are not present. We formalize this gadget below, where the presence of the decision and clause buttons have their own boolean parameters. In particular, we use true (i.e. $T$) to represent that a button is removed from the gadget, and we follow this convention in our gadgets as well.

**Definition 2.** $\text{LITERAL}(D, V_1, V_2, \ldots, V_k)$ is the following matrix

$$\begin{bmatrix} k & k & \ldots & 2 & 2 & 1 & B & C_1 & C_2 & \ldots & C_k \end{bmatrix} \text{ where } C_i = \begin{cases} i & \text{if } V_i = F \\ 0 & \text{if } V_i = T \end{cases}$$

and the decision button is not present (i.e. $B = 0$) if $D = T$, or has its own distinct value (i.e. color) if $D = F$.

The gadget is used as its own Buttons & Scissors level in the next Lemma.

**Lemma 1.** $B&S(\text{LITERAL}(D, V_1, V_2, \ldots, V_k)) \iff D$.

**Proof.** If $D = T$, then $B = 0$ and the buttons can be removed by cutting in turn all 1s, 2s, 3s, and finally $k$s. If $D = F$, then $B$ cannot be removed by any cut. \qed

We note that the functionality of this gadget is not changed if empty positions are inserted between any of the buttons. Furthermore, when we use this gadget in our reduction, each literal uses its own set of $k$ colours.
### 3.3 3-Input Disjunction

We model a 3-input disjunction $V_1 \lor V_2 \lor V_3$ with the following gadget that appears vertically in our reduction:

![Gadget](image)

Notice that the buttons cannot all be cleared from the illustration above. However, if any non-empty subset of $B_1$, $B_2$, $B_3$ is removed from the illustration, then we claim that all of the buttons can be removed. In other words, the buttons in this gadget can all be removed if and only if $B_1$ or $B_2$ or $B_3$ have been previously removed. Furthermore, the empty spacing to the left and right of $B_1$, $B_2$, and $B_3$ does not affect whether all of the buttons can be removed. We present this result formally below where empty spaces are again omitted from the definitions and arguments.

**Definition 3.** $\text{OR}(V_1, V_2, V_3)$ is the following 1-by-11 matrix

\[
\begin{bmatrix}
1 & 1 & 2 & 2 & B_1 & B_2 & B_3 & 2 & 2 & 3 & 3
\end{bmatrix}
\]

where $B_i = \begin{cases} 
  i & \text{if } V_i = F \\
  0 & \text{if } V_i = T.
\end{cases}$

**Lemma 2.** $B&S(\text{OR}(V_1, V_2, V_3)) \iff V_1 \lor V_2 \lor V_3$.

**Proof.** Label the entries of the gadget’s matrix as follows:

\[
\begin{array}{cccc|cccccccc}
1 & 1 & 2 & 2 & B_1 & B_2 & B_3 & 2 & 2 & 3 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11
\end{array}
\]

We now prove that if $V_1 \lor V_2 \lor V_3$ is true, then $B&S(\text{OR}(V_1, V_2, V_3))$ is true. Each row of the following truth table contains a sequence of cuts that will remove all buttons, where each cut is described using its two endpoints according to the labeling above.

<table>
<thead>
<tr>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
<th>Cut Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>1-2, 3-9, 10-11</td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>3-4, 1-5, 8-9, 7-11</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>1-2, 3-6, 8-9, 10-11</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>1-2, 3-6, 8-9, 7-11</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>3-4, 1-5, 8-9, 10-11</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>3-4, 1-5, 8-9, 7-11</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>3-4, 1-5, 6-9, 10-11</td>
</tr>
</tbody>
</table>

Finally, we consider $V_1 = V_2 = V_3 = F$ as redrawn below.

\[
\begin{array}{cccc|cccccccc}
1 & 1 & 2 & 2 & 1 & 2 & 3 & 2 & 2 & 3 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11
\end{array}
\]

In this case it is impossible to remove the button in position 6. $\square$
In our reduction the colours of $B_1$, $B_2$, $B_3$ are obtained from the literals that make up the clause in the associated instance of 3-SAT. We also use a modified version of this gadget that has two buttons of the same distinct color on either side of the original gadget.

Definition 4. $\text{OR}'(V_1, V_2, V_3)$ is the following 1-by-13 matrix

$$\begin{bmatrix} 4 & 1 & 1 & 2 & 2 & B_1 & B_2 & B_3 & 2 & 2 & 3 & 3 & 4 \end{bmatrix}$$

where $B_i = \begin{cases} i & \text{if } V_i = F \\ 0 & \text{if } V_i = T. \end{cases}$

Lemma 3. $B&S(\text{OR}'(V_1, V_2, V_3)) \iff V_1 \lor V_2 \lor V_3.$

3.4 $k$-Input Conjunction

We model a $k$-input conjunction $V_1 \land V_2 \land \cdots \land V_k$ as follows:

The outer two buttons have the same colour and we refer to these buttons as success buttons. These success buttons are separated by $k = 6$ inner buttons of distinct colours. Notice that the buttons cannot be cleared from the illustration above. Furthermore, the outer success buttons cannot be cleared unless all of $B_1, B_2, \ldots, B_k$ are removed. Again the empty spacing to the left and right of each $B_i$ does not affect whether all of the buttons can be removed, and we will use alternate individual colours for the gadget in our reduction. We present this result formally below with empty spaces omitted from the definitions and arguments.

Definition 5. $\text{AND}(V_1, V_2, \ldots, V_k)$ is the 1-by-$(k+2)$ matrix

$$\begin{bmatrix} k+1 & B_1 & B_2 & \ldots & B_k & k+1 \end{bmatrix}$$

where $B_i = \begin{cases} i & \text{if } V_i = F \\ 0 & \text{if } V_i = T. \end{cases}$

The outer $k+1$ values are known as success buttons.

Lemma 4. $B&S(\text{AND}(V_1, V_2, \ldots, V_k)) \iff V_1 \land V_2 \land \cdots \land V_k.$

Proof. If $V_1 \land V_2 \land \cdots \land V_k$ is true, then the only non-zero values in $\text{AND}(V_1, V_2, \ldots, V_k)$ are the two copies of $k+1$, and these two values can be removed with a single cut. Otherwise, if $V_1 \land V_2 \land \cdots \land V_k$ is false, then $\text{AND}(V_1, V_2, \ldots, V_k)$ contains a single copy of button $i$ for some $i$, which cannot be removed by any cut. $\square$
3.5 Modeling 3-SAT

We now demonstrate the NP-completeness of the Buttons & Scissors decision problem $B\&S(B)$ by transforming a given instance of 3-SAT $P$ into a board $B$. The organization of the various gadgets is illustrated by Figure 2 and is discussed informally in Sections 3.1–3.4.

**Theorem 1.** $3$–$SAT(P) \iff B\&S(B)$.

**Proof.** Given a satisfying assignment of variables for $P$, the following procedure clears all of the buttons on $B$ using a sequence of valid cuts:

1. For each decision gadget, use the satisfying assignment to choose the cut containing $T$ or the cut containing $F$.
2. Clear the LITERAL gadget that the now cleared $T$ or $F$ was contained in with a sequence of cuts, as described in Lemma 1.
3. Clear each $OR'$ that each now cleared LITERAL was contained in, as described in Lemma 3.
4. Clear the AND that each 3-input disjunction was contained in, as described in Lemma 4. This includes clearing the success buttons as illustrated in Figure 2.
5. Clear the remaining buttons in each decision gadget with a sequence of cuts involving each $C$, as described in Remark 2.
6. Clear the LITERAL gadgets that were not cleared in step 2.

Now suppose that $B\&S(B)$ is true. Our goal is to prove that $P$ is satisfiable. Before starting our proof of this direction, we note that there is limited potential for possible cuts that do not follow the sequence of cuts given in the previous direction of the proof. In particular, the only distinct gadgets that share any colors are the LITERAL gadgets and the OR gadgets. There is potential for cuts between the auxiliary buttons of a LITERAL gadget and the top and bottom parts of the OR gadgets, but cutting any of these buttons does not create any way to eliminate any of the 3 input buttons to the OR gadget, leaving it so that $OR(A,B,C)$ is clearable only when at least one of $A$, $B$, or $C$ is $T$.

Now consider a sequence of valid cuts $c_1, c_2, \ldots, c_t$ that removes all of the buttons on $B$. Since all of the decision gadget buttons are removed, it must be that each $A$ button is removed in a cut that includes either $T$ of $F$ as per Remark 2. We use this binary choice to provide truth assignments to the variables in the 3-SAT instance as in the other direction of the proof. We now work backwards to show that this truth assignment is satisfying. Since the two success buttons on the outsides of the conjunction gadget are the only buttons of their particular colour, it must be that they are removed by a single cut $c_k$. In order to remove these buttons it must be that all of the outside buttons in the $OR'$ gadgets are removed by earlier cuts due to Lemma 4. These outer buttons also come in unique colour pairs, so this implies that all of the buttons in the $OR'$ gadgets are removed by earlier cuts by Lemma 3.
Now consider the decision gadgets prior to cut $c_k$. Because the success buttons are still in place prior to cut $c_k$, it is also true that the decision cleanup buttons are in place prior to cut $c_k$. Therefore, by our earlier reasoning, there is a $T$ or $F$ button present from each decision gadget prior to cut $c_k$. Therefore, the clause buttons of each literal gadget are removed after removing the opposite $T$ or $F$ button from each decision gadget. However, this is only possible if the aforementioned assignment to the variables of 3-SAT is satisfying. □

4 Inapproximability

In this section we describe how Theorem 1 implies an inapproximability result. Consider the success buttons from the example in Figure 2.

From the proof of Theorem 1, the success buttons can be removed if and only if $B&S(B)$ is solvable and the corresponding 3-SAT instance $P$ is satisfiable. Now consider the addition of button pairs to the outside of the success buttons. More specifically, we add buttons $x y x y \cdots x y$ to the left of the left success button, and $y x y x \cdots y x$ to the right of the right success button, where $x$ and $y$ are colours that are unused by the rest of the reduction, and we allow the board $B$ to expand to accommodate the addition of the new buttons. Notice that these additional buttons can be removed if and only if the 3-SAT instance $P$ is satisfiable. Furthermore, if we add enough of these buttons, then an approximation algorithm for $\max B&S(B)$ will be able to determine if one of these additional buttons can be removed, which in turn implies that the success buttons can be removed and $P$ is satisfiable. This informal argument leads to the following theorem, which we state without formal proof due to space constraints.

**Corollary 1.** If there is a $\frac{1}{c}$-approximation algorithm for $\max B&S(B)$ for some constant $c$, then $P=NP$.

5 Integer Programming

Here we give an integer programming model that solves Buttons & Scissors levels. Given an $n$-by-$n$ board $B$ containing the buttons $B_1, B_2, \ldots, B_m$ we consider a sequence of cuts in which each cut is valid or is empty. We make two observations:

- The maximum number of cuts required in any solution for $B$ is $\left\lfloor \frac{m}{3} \right\rfloor$;
- If $B$ has a solution, then it has a solution that only uses cuts containing two or three buttons. This is because cuts of more than three buttons can be accomplished by successive cuts of two or three buttons.
The variables of the integer program correspond to the subset of buttons that are removed during each of these cuts, which we refer to by their time or timestep. More specifically, a feasible subset of buttons is a subset of two or three buttons that could be the buttons removed by a single cut. In other words, a feasible subset contains two or three co-linear buttons of the same colour.

Let \( \{S_1, S_2, \cdots, S_m\} \) be the set of all feasible cuts. We define the set of variables in the integer program as follows,

\[
V = \left\{ v_{i,t} \mid 1 \leq i \leq m, 1 \leq t \leq \left\lfloor \frac{m}{2} \right\rfloor \right\}.
\]

We will add constraints to our integer program so that the following holds,

\[
v_{i,t} = \begin{cases} 
1 & \text{if } S_i \text{ is the feasible set of buttons removed during the } t \text{th cut;} \\
0 & \text{otherwise.}
\end{cases}
\]

Figure 3 illustrates the solution from Figure 1 using these variables.

\[
S_1 = \{1, 7\} \quad S_2 = \{2, 4, 9\} \\
S_3 = \{2, 4\} \quad S_4 = \{2, 9\} \\
S_5 = \{4, 9\} \quad S_6 = \{6, 8\} \\
S_7 = \{3, 5, 10\} \quad S_8 = \{3, 5\} \\
S_9 = \{3, 10\} \quad S_{10} = \{5, 10\}.
\]

Now we describe the constraints that complete the integer program:

1. At most one feasible subset is removed during each cut.
2. Each button \( B_1, B_2, \cdots, B_m \) must be removed by exactly one cut.
3. Consider a feasible subset \( S_i \). If \( S_i \) is removed by a cut, then every button blocking this cut must be removed by an earlier cut.

The first constraint translates to the following,

\[
\sum_i v_{i,t} \leq 1 \text{ for all timesteps } 1 \leq t \leq \left\lfloor \frac{m}{2} \right\rfloor.
\]

The second constraint translates to the following,

\[
\sum_{i \text{ if } B \in S_i} \sum_{t=1}^{\left\lfloor \frac{m}{2} \right\rfloor} v_{i,t} = 1 \text{ for every button } B \text{ in } \mathcal{B}.
\]
The third constraint requires the comparison of the times at which different cuts may occur. It can be conceptualized more easily using a function that gives the timestep of a cut, formulated as follows,

\[ T(i) = \sum_{t=1}^{[\frac{m}{2}]} t \cdot v_{i,k} = \begin{cases} 0 & \text{if } S_i \text{ is never removed as a cut;} \\ t & \text{the time, } t, \text{ when subset } S_i \text{ is removed} \end{cases} \]

The constraint also depends on the idea of a blocking button. A button \( B \) is a blocking button of \( S_i \) if \( B \) is co-linear with and located between the buttons in \( S_i \). This constraint then translates to the following, for all feasible subsets \( S_i \) and for all buttons \( B \) blocking \( S_i \),

\[ T(i) + m \left( 1 - \sum_{t=1}^{[\frac{m}{2}]} v_{i,t} \right) > \sum_{i \text{ if } B \in S_i} T(i). \]

The second term on the left side will be equal to zero if \( S_i \) is ever used as a cut, but will equal \( m \) if \( S_i \) is never used, thus ensuring that the left side is greater than the right side in this case.

Two authors implemented this integer program and an alternate integer program, and tested them against each other to verify correctness. Sample solutions and run-times appear in the appendix.

Finally we mention that this integer program has a polynomial number of variables and constraints. We save the counting details for the full version of the article.

6 Additional Results and Open Problems

Buttons & Scissors allows has a number of natural variations. In particular, the authors are interested in the following restrictions of the game:

1. Only horizontal and vertical scissor cuts are allowed.
2. No empty spaces are allowed on the initial board.
3. There is a constant number of distinct colours \( c \) for the buttons.
4. Each colour can be used by at most \( b \) buttons.

We have proven that the first restriction is difficult to approximate by modifying our decision gadget. We have also proven that the second restriction is difficult to decide by spacing out the gadgets and adding a 'dummy' colour to the empty spaces. We had hoped find a polynomial-time algorithm for the \( c = 1 \) colour restriction, but were unable to find one, and are not certain if it is possible.

Finally, we mention that Buttons & Scissors is solvable in polynomial-time when \( b = 3 \) for the fourth restriction. More specifically, a greedy algorithm that tries to remove all buttons of a single colour will clear the board in polynomial-time whenever it is possible. On the other hand, our NP-completeness reduction uses each colour at most \( b = 7 \) times. Determining the exact transition from poly-time solvability to (presumed) intractability in terms of \( b \) is an interesting open question.
References

1. E. Demaine, Y. Okamoto, R. Uehara, and Y. Uno. Computational complexity and an inte-
ger programming model of shakashaka. *IEICE Transactions on Fundamentals of Electronics, 
2. L. Guala, S. Leucci, and E. Natale. Bejeweled, candy crush and other match-three games are (np-
)hard. In *Proceedings of the 10th IEEE Conference on Computational Intelligence and Games*, 
page total 8, 2014.
2009.

A Solutions

Solutions for the final 25 levels in Buttons & Scissors is given on the next page. These solutions were computed using the integer programming formulation found in Section 5. Each solution is listed by button co-ordinates that are removed by successive cuts. The time required to solve each level is also given in seconds, as computed with a Macbook Air 2013 model with a dual-core Intel i5 processor and 4GB of RAM.
<table>
<thead>
<tr>
<th>Level</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>175</td>
<td>(1.5, 2.6)</td>
</tr>
<tr>
<td>180</td>
<td>(5.1, 6.2)</td>
</tr>
<tr>
<td>185</td>
<td>(2.0, 4.4)</td>
</tr>
<tr>
<td>190</td>
<td>(4.2, 4.5)</td>
</tr>
<tr>
<td>195</td>
<td>(0.3, 0.4)</td>
</tr>
<tr>
<td>200</td>
<td>(1.5, 0.6)</td>
</tr>
<tr>
<td>205</td>
<td>(0.1, 0.0)</td>
</tr>
<tr>
<td>210</td>
<td>(0.0, 0.0)</td>
</tr>
</tbody>
</table>

Note: The authors found a problem with the script that created this table during our final proofreading. It will be corrected before publication.