Clickomania is Hard, Even with Two Colors and Columns

Aviv Adler, Erik D. Demaine, Adam Hesterberg,
Quanquan Liu, Mikhail Rudoy

Clickomania is a classic computer puzzle game (also known as *SameGame, Chain-Shot!, and Swell-Foop,* among other names). Originally developed by Kuniaki "Morisuke" Moribe under the name *Chain-Shot!* for the Fujitsu FM-8, and announced in the November 1985 issue of *ASCII Monthly* magazine, it has since been made available for a variety of digital platforms [Mor03]. Figure 1 shows some examples. Although rules and objectives vary slightly among different versions, the basic premise is always the same: you are presented with a two-dimensional grid of colored square tiles and asked to clear those tiles by removing a contiguous like-colored group at each step (or *click*). After each click, any tiles suspended above empty space will fall, and empty columns will contract, so that the remaining tiles always form a contiguous group. Though the number of clicks is unlimited, if a contiguous like-colored group has only one tile (which we refer to as a *singleton*), then it cannot be clicked. To remove it, one must first connect it to at least one other tile of the same color. The game ends when no further clicks are possible, either because all tiles have been eliminated or because all remaining tiles are singletons. See Figure 2 for a sample Clickomania board and the results of a few clicks.

There are two main variants of the game, which differ in the objective of the player:

1. *Elimination variant:* The player wins by removing all the tiles.
2. *Score variant:* The player gains *points* from each click based on the number of tiles removed with that click (typically, this relation is quadratic: removing x tiles awards approximately x^2 points) and wins by achieving at least a specified score.

For each variant, we study how hard it is for a computer to determine whether it is possible to win from a given initial configuration of the board. Specifically, we show that Clickomania is computationally hard ("NP-complete") even for two-color patterns occupying just two columns (and thus also hard for more than two colors or columns). This result is best possible: one-color Clickomania puzzles are trivial to solve with just a single click, and one-column Clickomania puzzles are computationally easy to solve [BDD$^+$02]. Indeed, this paper completes a quest started 15 years ago when Biedl et al. [BDD$^+$02] proved Clickomania computationally hard for two columns and five colors, or five columns and three colors.

Figure 1: Examples of Clickomania games on a few platforms. **Top left:** the original, *Chain-Shot!*, released in 1985 on Fujitsu FM-8 [Mor03]. **Bottom left:** a variant available from Google Play [stf15]. **Right:** a Mario-themed variant, *Undake 30*, released on the SNES in 1995. All images used under fair use, from cited source or, in the case of the right image, a screen capture from an SNES emulator.

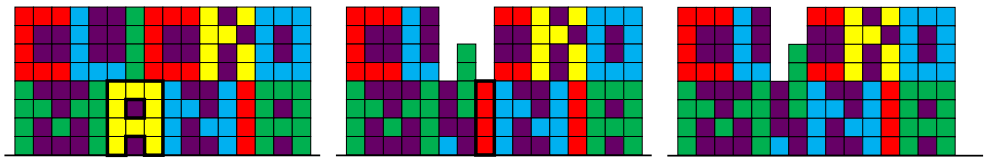

Figure 2: An example of a starting Clickomania board (**left**) and the result of two clicks. **Middle:** After clicking the first 'A'. **Right:** After clicking the left part of the second 'C' (which fell onto the original 'A'). Note the contraction of the emptied column.

1 A Crash Course in Complexity Theory

In this section, we provide the necessary background for understanding our proof that Clickomania is "computationally hard". Those familiar with the basics of complexity theory, in particular the notion of a reduction, can safely skip this section. For those curious to learn more, Garey and Johnson's classic *Computers and Intractability: A Guide to the Theory of NP-Completeness* [GJ79] and the notes and lecture videos from the MIT course on the topic[1] are both excellent sources.

"Given a Clickomania board, is it possible to win?" is an example of a *computational problem* with clearly defined input and desired output. We distinguish problems from *instances* of problems which are defined by specific inputs. Thus, elimination-variant and score-variant Clickomania are examples of problems, while an instance of Clickomania is a

[1]MIT 6.890, Algorithmic Lower Bounds: Fun with Hardness Proofs (Erik Demaine, Fall 2014), http://courses.csail.mit.edu/6.890/fall14/

specific board filled with colored squares, and in the case of the score variant, a target score. Clickomania is a special type of computational problem called a *decision problem*, because the answer (desired output) to any instance is either 'yes' or 'no'.

Given such a computational problem, the standard goal in theoretical computer science is to determine how hard the problem is for a computer to solve. In the case of Clickomania, is there an efficient algorithm which, given a starting configuration of the board, can determine whether it is possible to win (e.g., remove all of the tiles)? The study of such questions is known as *complexity theory*, and dates back to at least the 1970s (with the work of Richard Karp, Stephen Cook, and Leonid Levin) and more generally to the 1930s (with the work of Alan Turing). There is in fact a very rich body of work applying these concepts to the study of games and puzzles; see [HD09, Vig12]. Here we provide an informal introduction to this theory; in the next section, we will be more formal.

Classically, an algorithm is considered *efficient* if it is guaranteed to solve the problem in time polynomial in the size of the input. A problem is considered *easy*, or *in P*, if it has such an efficient algorithm. To prove that a problem is easy, we simply exhibit an efficient algorithm.

Unfortunately, it is generally difficult to show that there does not exist an efficient algorithm for a problem. In fact, we do not know whether most commonly studied computational problems have polynomial-time solutions. Nevertheless, it is often possible to compare the difficulty of solving two problems, using the technique of *reduction* pioneered by Cook, Karp, and Levin. Based on these comparisons, we organize problems into a *complexity hierarchy*.

Most famously and most relevant to Clickomania is the class of problems known as NP-COMPLETE. These problems all have equivalent complexity, and are widely believed (though not yet proved) to be hard. In fact, proving these problems hard is the most famous unsolved problem in computer science, also known as "does P = NP?".

1.1 Reductions

First, we define a polynomial-time reduction (which we refer to more simply as a *reduction*) from a problem A (e.g., a known NP-complete problem that we believe to be hard to solve) to a problem B (e.g., Clickomania). In essence, given an instance of a problem A, a reduction is a polynomial-time procedure to transform it into an instance of problem B (in this case, a starting Clickomania board), such that the two instances have the same yes/no answer.

Reductions can be used to solve problems: if we want to solve A, and know how to solve B, then a reduction from A to B gives us an algorithm for solving A. Thus a reduction from A to B shows that A is at least as easy to solve as B. Conversely, the same reduction shows that B is at least as hard as A: if problem A is hard to solve, and we want to study problem B, then a reduction from A to B shows that B is hard to solve—otherwise the reduction plus the solution to B would give a solution to A.

1.2 NP-completeness

NP stands for "Non-Deterministic Polynomial-time." It is the class of decision problems where 'yes' answers have polynomial-length *solutions* (certificates) that make it easy to check the answer is indeed 'yes' (but whose solutions may be hard to *find* and for which it may be hard to verify a 'no' solution).

For example, Clickomania is in NP [BDD+02]. Clearing a Clickomania board with n tiles requires at most n clicks, so a solution to a Clickomania board can be described as a length-n sequence of clicks, and such a solution can be checked in polynomial time by simulating the game play resulting from the described clicks. On the other hand, if it is impossible to win from a given Clickomania board, then there is no known succinct general way to convince someone of this fact.

A problem is *NP-hard* if *any* problem in NP can be reduced to it, that is, it is at least as hard as all problems in NP. A problem is *NP-complete* if it is both NP-hard and in NP. These problems are the hardest problems in NP.

The standard way to prove that a problem such as Clickomania is NP-hard (and thus NP-complete) is to construct a reduction from a known NP-hard problem to Clickomania: as depicted in the bottom of Figure 3, any NP problem can be reduced to the known NP-hard problem, so concatenating that reduction with the constructed reduction gives a way to reduce any problem in NP to Clickomania.

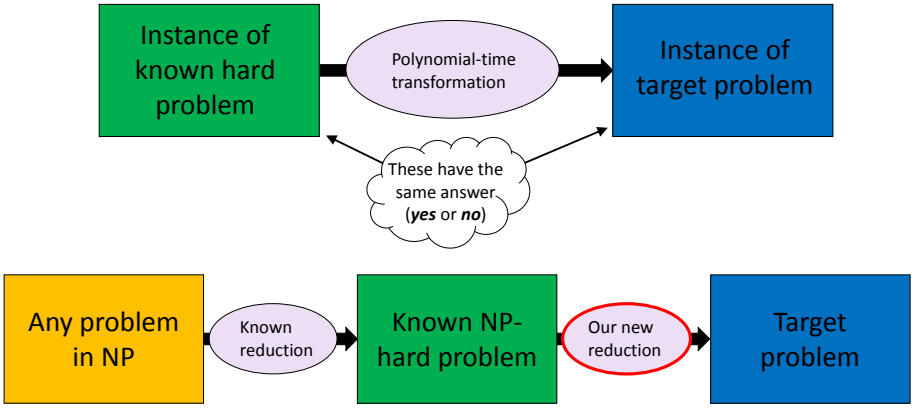

Figure 3: A summary of how reductions work (top), and how they are used to prove NP-hardness (bottom).

1.3 Approximation and Parameterization

In addition to the basic Clickomania decision problem, we consider two relaxations of the problem that are potentially easier. The first is *approximation*, where the goal is to come within a reasonably small factor of the optimal solution, as measured by some scoring function. For example, is there an efficient algorithm which, given a Clickomania instance, finds a solution whose score is at least 2/3 of the maximum? (Spoilers: there isn't.)

The second relaxation is *parameterization*. A parameterized problem is one where the *solution* is constrained to a fixed size, denoted by the parameter k, while n denotes the size of the *input*. A natural parameter for Clickomania is the number of clicks in the solution: the *k-Click Clickomania* problem asks to win a given Clickomania instance using at most k clicks. Many parameterized problems can be solved in polynomial time for fixed k, typically using a brute-force $n^{O(k)}$-time algorithm: for k-Click Clickomania, we can simply try all length-k sequences of possible clicks (with at most n choices for each click). But this time bound gets unwieldy even for relatively small k.

For parameterized problems, an "efficient" algorithm is one whose running time grows polynomially in n *where the exponent on n does not depend on k*, that is, an algorithm whose running time is $O(f(k) \cdot n^{\alpha})$, where f can be any function, but α is a constant not dependent on k. There are analogous classes to NP for "hard" parameterized problems; we will use a class called W[1], described in Section 7.

2 Classification of Clickomania Problems

To better understand what makes Clickomania hard, we consider restricting the instances according to two different measures: colors and columns. For example, if there is only one color, then the problem is easy to solve: unless the initial board has only one tile, one click will eliminate all the tiles. Actual Clickomania video games such as those in Figure 1 use few colors (usually a single-digit number), and relatively few columns as well. To capture this idea, we define *subclasses* of Clickomania instances and ask which of these subclasses are still hard:

Clickomania(w, c). *Given a grid of tiles with at most w columns and c different colors, is it possible to remove all of them under the rules of Clickomania?*

As a convention, w (and/or c) can be ∞ if we want to consider the set of Clickomania instances where the number of columns (and/or colors) is unrestricted.

Note that, if a class allows at most as many columns and at most as many colors as another class, then all instances of the first are also contained by the second; formally:

$$\text{Clickomania}(w_1, c_1) \subseteq \text{Clickomania}(w_2, c_2) \text{ if } w_1 \leq w_2 \text{ and } c_1 \leq c_2.$$

Classes containing more problem instances cannot be easier to solve. Thus, for example, if we show that Clickomania$(3, 4)$ is NP-complete, then so is Clickomania$(8, 4)$; whereas, if Clickomania$(1, 10)$ is polynomial-time solvable, then so is Clickomania$(1, 5)$.

The primary goal of this paper is to determine the precise boundary between *easy* Clickomania classes and *hard* Clickomania classes. In the first mathematical analysis of (elimination-variant) Clickomania, Biedl et al. [BDD+02] showed that Clickomania$(1, \infty)$ has a polynomial-time algorithm (as does Clickomania$(\infty, 1)$), while Clickomania$(2, 5)$ and Clickomania$(5, 3)$ are NP-complete. Thus, if we can prove that Clickomania$(2, 2)$ is NP-complete, we will have shown our main theorem:

Theorem 1. *Elimination-variant* Clickomania(w, c) *is in P if* $w = 1$ *or* $c = 1$*, and NP-complete otherwise.*

Figure 4 illustrates this theorem in comparison to prior work.

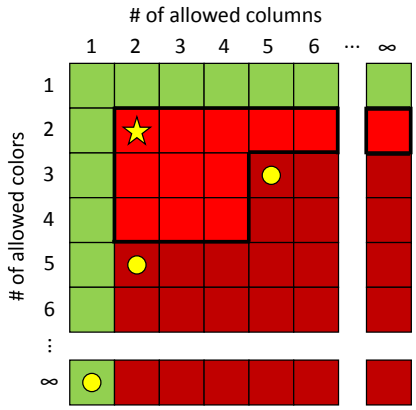

Figure 4: Depiction of the complexity boundary for Clickomania. Green cells represent subclasses in P; red cells (both dark and bright) represent subclasses which are NP-complete. The circles denote previously shown results; the yellow star denotes the result shown here. The brighter red denotes classes whose complexity were shown by this work.

Our hardness result also extends to the score variant:

Corollary 1. *Score-variant* Clickomania(w, c) *is NP-complete if* $w \geq 2$ *and* $c \geq 2$ *(and* Clickomania$(w, 1)$ *is trivially in P for all values of* w*).*

A key difference here is that the one-column case with a non-linear scoring function remains unresolved, while for the elimination variant the one-column case is in P.

We also show various corollaries concerning the hardness of approximating the solution to Clickomania with a variety of different parameters; see Theorem 3 for details. These results all build on the elimination-variant reduction—effectively requiring the player to solve the elimination variant of Clickomania as a subproblem in order achieve the desired score.

For parameterized complexity, we present the following results:

Theorem 2. Clickomania$(\infty, 2)$*, parameterized by the number of clicks, is W[1]-hard in the score variant.* Clickomania$(\infty, 4)$*, parameterized by the number of clicks, is W[1]-hard in the elimination variant.*

3 Starting Points

Let $\mathbb{Z}^+$ denote the positive integers. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. A *multiset* is a set which is allowed to contain multiple copies of an item.

We now introduce the NP-complete problems we reduce from. In full context, we are reducing from the well-known *3-Partition* problem; however, for a cleaner reduction we reduce from a new (to our knowledge) problem which we call *Sum-Ordering*. In order to demonstrate that Sum-Ordering is NP-hard, we reduce from 3-Partition. We give these two problems as follows:

3-Partition [GJ79]. *Given a multiset of $3n$ input numbers $A = \{a_1, a_2 \ldots, a_{3n}\}$ (where $a_i \in \mathbb{Z}^+$ for all i), output 'yes' if and only if there is a partition of A into n subsets (which by definition are mutually exclusive) $A_1, A_2, \ldots, A_n$ such that for each subset, the sum of its numbers is the same as all the others. Formally, a solution is a partitioning such that*

$$\sum_{a \in A_j} a = \frac{\sum_{a \in A} a}{n} \text{ for all } j \in [n]$$

Note that in this variant of the 3-Partition problem the inputs are all positive, and the subsets are allowed to have any size.

Sum-Ordering. *Given a list of n input numbers $a_1, a_2, \ldots, a_n \in \mathbb{Z}^+$ and m target numbers $t_1, t_2, \ldots, t_m \in \mathbb{Z}^+$, output 'yes' if and only if there is a way to order the inputs such that all the target numbers can be represented as partial sums of the inputs in that order. Formally, a solution is a permutation $\sigma : [n] \to [n]$ such that for all $j \in m$ there exists an $k_j \in n$ where*

$$\sum_{i=1}^{k_j} a_{\sigma(i)} = t_j$$

3-Partition is *strongly* NP-complete—and therefore Sum-Ordering itself is strongly NP-complete (it is trivially in NP because there is an easy way of verifying the solution). This is important, as our reduction will represent the numbers a_i from the input set as stacks of red tiles, so the input size is proportional to the sum of the inputs (and targets).

Lemma 1. *Sum-Ordering is strongly NP-complete.*

For reasons of space, we omit the lengthy, technical proof. Interested readers are invited to contact the authors directly.

Proof. First, we note that Sum Ordering is in NP, because a valid solution to the problem can be checked in polynomial time. This therefore leaves the task of showing that it is strongly NP-hard. We accomplish this by a reduction from 3-Partition.

Suppose we have a 3-Partition problem with input set $A = \{a_1, a_2, \ldots, a_{3n}\}$ (where the inputs are all natural numbers); we therefore wish to partition this into n subsets each summing to

$$t = \frac{\sum_{i=1}^{3n} a_i}{n}$$

which is itself a natural number (otherwise there is trivially no solution). We claim that this is equivalent to the Sum-Ordering problem with $3n$ inputs $a_1, a_2, \ldots, a_{3n}$ and n target numbers $t, 2t, 3t, \ldots, nt$. In this problem, note that for all j, $t_j = jt$.

First, we show that if there is a solution to the Sum-Ordering instance, then the original 3-Partition problem also has a solution. A solution to the Sum-Ordering problem is a permutation $\sigma : [3n] \to [3n]$ such that for all $j \in n$, there exists a $k_j \in 3n$ such that

$$\sum_{i=1}^{k_j} a_{\sigma(i)} = t_j = jt$$

Thus (where we define $k_0 = 0$ and $\sum_{i=1}^{k_0} a_{\sigma(i)} = 0$) we get,

$$\sum_{i=k_{j-1}+1}^{k_j} a_{\sigma(i)} = \sum_{i=1}^{k_j} a_{\sigma(i)} - \sum_{i=1}^{k_{j-1}} a_{\sigma(i)} = jt - (j-1)t = t$$

Thus, if we partition the inputs $A = a_1, a_2, \ldots, a_{3n}$ into the subsets $A_1, A_2, \ldots, A_n$ where

$$A_j = \{a_{\sigma(i)}\}_{i=k_{(j-1)}+1}^{k_j}$$

we can see that $\sum_{a \in A_j} a = t$ for all $j = 1, 2, \ldots, n$, thus showing that the 3-Partition instance also has a solution. Note that $k_{j-1} < k_j$ for all j because $t > 0$. Therefore, we are also guaranteed that $A_i \cap A_j = \emptyset$ for all $i, j \in [n]$.

We then show that if the original 3-Partition problem has a solution, so does the Sum-Ordering problem. This is exactly the argument above, but in reverse. Let $A_1, A_2, \ldots, A_n \subseteq A$ be the partition solving the 3-Partition instance, and for each $j \in [n]$ we define

$$k_j = \sum_{\ell=1}^{j} |A_\ell|.$$

We then order the inputs $a_1, a_2, \ldots, a_n$ in the following way: the elements of A_1 come first, then the elements of A_2, etc., until the elements of A_n have come last. The ordering of the elements within these groups is not important and can be set arbitrarily. Let this ordering be called σ. Then, we note that

$$\sum_{i=1}^{k_j} a_{\sigma(i)} = \sum_{\ell=1}^{j} \sum_{a \in A_\ell} a = \sum_{\ell=1}^{j} t = jt;$$

thus, the Sum-Ordering instance has a solution.

Thus, we have shown that the Sum-Ordering instance has a solution if and only if the original 3-Partition instance has a solution. It then only remains to note that the Sum-Ordering instance can be trivially generated from the 3-Partition instance in polynomial time. Therefore, because 3-Partition is strongly NP-hard, so is Sum-Ordering. □

We also make the following assumptions about Sum-Ordering instances (without loss of generality):

1. The set T is ordered (from t_1 being the smallest to t_m being the largest).

2. The set T contains no repeated elements.

3. $t_m = \sum_{i=1}^{n} a_i$.

The last one is without loss of generality because if T contains an element *larger* than the sum of all elements of A, there is trivially no solution, and if it doesn't contain an element equal to the sum of all elements of A, we can add that as an 'extra' element and any solutions are unchanged (because $t_m = \sum_{i=1}^{n} a_i$ is a prefix sum no matter what order A is in).

4 NP-Hardness Reduction

To reduce from Sum-Ordering to Clickomania, we need to convert an instance of Sum-Ordering ($a_1, a_2, \ldots, a_n, t_1, t_2, \ldots, t_m \in \mathbb{Z}^+$) into a Clickomania puzzle. We begin with a blueprint for the reduction:

1. Each input element a_i is represented as a stack of $4a_i$ red tiles in the right-hand column. We multiply by four to make the proof of correctness easier, as well as to allow the group to be clicked if a_i happens to be equal to 1. These red tiles are separated by (single) white tiles. We call these *input stacks*.

2. The order in which these are clicked represents the order of the solution to the Sum-Ordering problem; note that the right-hand column falls past the left-hand by four times the sum of the input elements corresponding to the stacks which have been clicked so far.

3. Above the input stacks sit a number of *verification* gadgets, each representing a target sum t_j from the Sum-Ordering problem. This 'activates' (allowing for a particular action) when the inputs represented by the previously-clicked input stacks sum to t_j.

4. At the bottom is a *final verifier*, which can only be cleared out if *all* the verification gadgets have been activated.

Thus, there should be a solution to our Clickomania construction if and only if there is a solution to the original Sum-Ordering instance. Our reduction decomposes into a collection of "gadgets"—pieces of the Clickomania construction that represent these different aspects of the Sum-Ordering problem. Specifically, we use four types of gadgets:

1. The *input* gadget, which contains the representations of the set $A = \{a_1, a_2, \ldots, a_n\}$. There is only one copy of this gadget, and it contains a representation of each a_i.

2. The *target verification* gadgets, each associated with a target sum t_j, that check whether the target sums are indeed prefix sums of the inputs in the given order. We refer to the target verification gadget associated with target t_j as V_j.

3. The *final verifier* or *base* (because it sits at the bottom of the construction), a gadget whose purpose is to be hard to clear and checks whether *all* the t_j-verifier gadgets have been correctly eliminated.

4. A *separator* gadget made of only white tiles, and whose purpose is to ensure that the other gadgets do not interfere with each other, and to provide correct relative heights of the two columns for the other gadgets (because the gadgets are not, in general, perfect rectangles). The separator also has another purpose, having to do with how the base is removed. We will describe this later. These gadgets are interleaved with the verification gadget; we thus refer to the separation gadget sitting above verification gadget V_j as S_j.

The full construction for the simple Sum-Ordering problem $A = \{1, 1, 2\}$; $T = \{1, 3, 4\}$ is shown in Figure 6. As depicted at left on the diagram, the ordering of the gadgets (from bottom to top) is: (1) Base; (2) Input; (3) an alternating sequence of t_j-verification and separation gadgets.

4.1 The Gadgets

To describe these gadgets, we use a value L which we define as $L = 8t_m + 2$. L is simply a number that needs to be 'large enough' so that the various gadgets do not get split up as a result of player actions (but not so large as to make the reduction non-polynomial time). Note that L is polynomial (in fact, linear) in the size of the Sum-Ordering problem. Hence this is a polynomial-time reduction because there are only $2m + 2$ gadgets in total (m verification gadgets, m separation gadgets, the base, and the input gadget) and none of them have a height of more than $6L$. It is trivial to verify that the construction can be built in polynomial time given the Sum-Ordering instance we wish to encode.

Note that in Figure 6, there is an unbroken string of white tiles starting from the top of the Base gadget and going all the way to the top of the construction. We refer to this string of white tiles as the *ribbon*. We note that as the base has a checkerboard pattern, it can only be destroyed by clicking on groups above it, and that it has to be destroyed row-by-row from top to bottom. However, for this to work, we must have many separate red and white groups above the base. The fact that the ribbon is one contiguous group of white tiles means that to solve the Clickomania instance, we have to cut the ribbon into many smaller groups. For illustrations of the various gadgets, see Figures 5 and 6.

4.1.1 The Base

In order for the reduction to work, there must be something preventing a solution to the Clickomania instance should the Sum-Ordering problem turn out not to have a solution. In the case of our reduction, this is done by having a gadget whose sole purpose is to be difficult to remove; specifically, a gadget at the bottom of the construction which is made up of a checkerboard pattern of red and white tiles (so every tile is a singleton). We define the base as having *layers*: each layer consists of two diagonally-connected tiles, with the tile on the

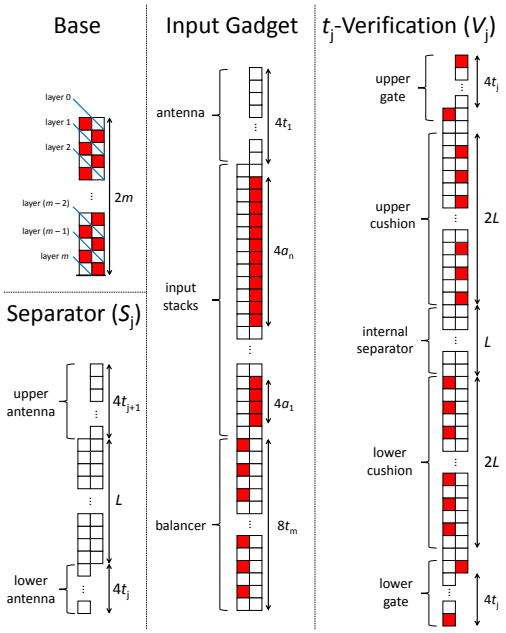

Figure 5: Schematics for all the gadget types, with internal constructions labeled.

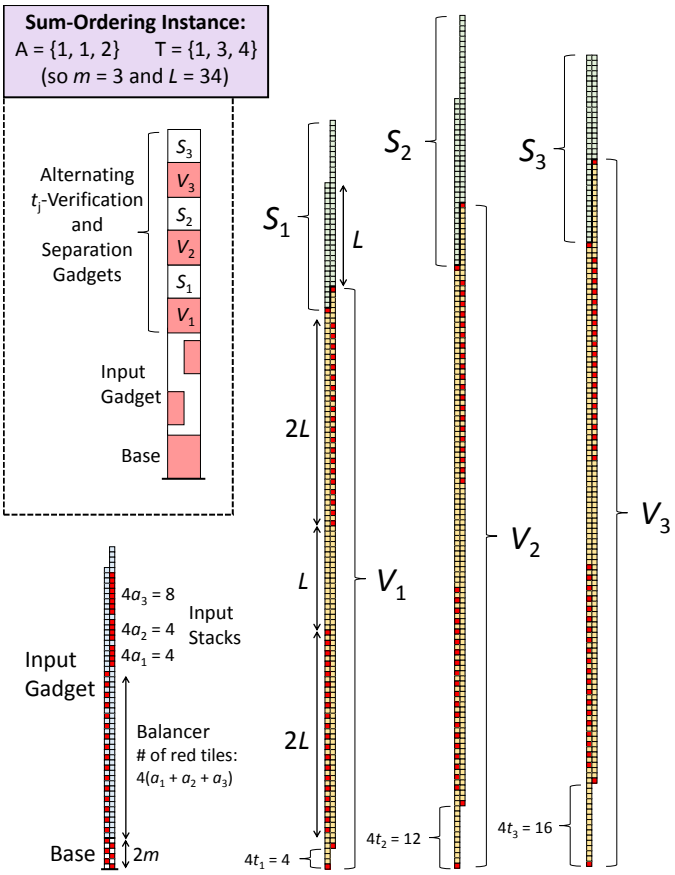

Figure 6: An example of the full construction for a simple Sum-Ordering Instance (given in top left, along with an overall schematic for the reduction). The white tiles within the construction are tinted so as to better distinguish the gadgets from each other (thicker black lines also separate them). In order to make the construction fit the page, 'L' in the diagram is only 20 (while it should be 34, as indicated on the top left).

right-hand column lower than on the left-hand column. The single white tile at the bottom of the left-hand column is a layer by itself, as is the single white tile at the top (within the base) of the right column. The height of the base is $2m$ (where m is the number of target sums in the Sum-Ordering problem). Thus, there are $m + 1$ white layers (each taking two rows, except for the first and last layers). We label the white layers from top-to-bottom as layer $0, 1, 2, \ldots, m$. This labeling gives the nice result that in the correct solution, layer j is always destroyed by the white tiles of the separator gadget S_j (with layer 0 destroyed by the white tiles previously in the input gadget). As mentioned above, the only way to remove the base is to hit it from above with alternating groups of red and white tiles. We now make this observation more precise by noting that each group can remove at most one 'layer' of

the base. We will later see that layer 0 is to be removed with the white tiles from the input gadget, while layer j (for $j > 0$) is to be removed using the tiles from the jth separator gadget.

4.1.2 The Input Gadget

The input gadget is meant to encode the input set $A = \{a_1, a_2, \ldots, a_n\}$ from the Sum-Ordering instance. The input gadget is composed of three sections, placed on top of one another (in order from bottom-to-top):

1. The *balancer*, meant to ensure that the gadget does not have many more white tiles on one side than on the other (which simplifies the correctness proof);
2. The *input stacks* section, which does the actual job of representing the input set A from the Sum-Ordering instance;
3. The *antenna*, which is simply there to allow the t_1-verification gadget to sit on top of the input gadget.

We begin by discussing the input stacks section. The elements a_i of A are each represented by a stack of red tiles in the right-hand column (separated by single white tiles), while the left-hand column contains only white tiles. However, we quadruple the size of the stacks (so element a_i is represented by a stack of $4a_i$ red tiles) in order to ensure that even if $a_i = 1$, its stack is clickable. The input stacks section is perfectly rectangular, with total height of $(4\sum_{i=1}^{n} a_i) + n + 1$ (each input stack contributes $4a_i$ for some i, there are $n - 1$ white tiles to separate the input stacks, and two final white tiles to separate the input stacks group from the two other sections in the input gadget). As discussed above, the solution to the Sum-Ordering instance will be expressed in the solution to the Clickomania instance by the order in which the input stacks are removed.

The balancer sub-gadget sits beneath the input stacks. It is a rectangle of height $8\sum_{i=1}^{n} a_i$, whose right-hand column consists entirely of white tiles, and whose left hand column alternates between white and red tiles. This means that all red tiles contained in this are singletons and cannot be clicked. Note that the balancer has precisely the same number of red tiles in its left-hand column as the input stacks have red tiles in its right-column. Thus, because the two sub-gadgets taken together are a rectangle (and thus have the same total number of tiles in each column), the connected set of white tiles in the two sub-gadgets is equally distributed between the two.

The final sub-gadget is an 'antenna' of white tiles on the top-left column; this antenna is $4t_1$ tiles tall, and is there so that the t_1-verification gadget can sit on top of the input gadget.

4.1.3 The t_j-Verification Gadget V_j

Though the layout of this gadget is depicted in Figure 5, we give it again in Figure 7, along with the details of the *gate mechanism*. The basic function of the t_j-verification gadget is to allow the cutting of the 'ribbon' of white tiles if and only if the previously-clicked input

stacks come to the correct sum. To accomplish this, we use the gate mechanism: two red tiles, one in each column, with the one in the right-hand column placed exactly $4t_j$ tiles higher than the one in the left-hand column. The idea is that the gate *closes* if the input stacks which have been clicked so far sum to exactly $4t_j$—so the input values they represent sum to exactly t_j (because the input stacks have height four times the size of the inputs). If the input stacks clicked so far sum to *more* than $4t_j$, we say that the gate has been re-opened. This is bad news, as we will see, because it means that the Clickomania instance no longer has a solution.

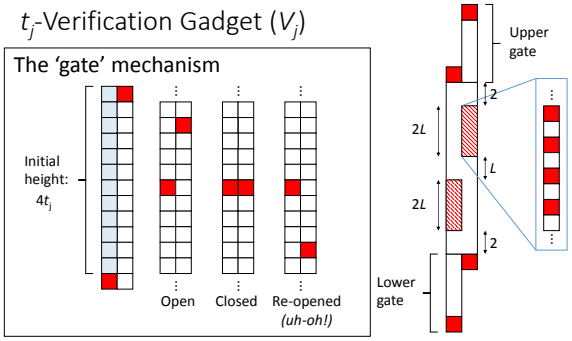

Figure 7: Anatomy of the t_j-verification gadget. **Left:** A diagram of the 'gate' mechanism; these consist of two red tiles (one per column), with the tile on the right-hand column exactly $4t_j$ higher than the one on the left. Note the whitespace above and below the gate remains connected unless the gate is closed. **Right:** The t_j-verification gadget V_j.

Note that the gate can never have an odd height difference between its two red tiles (as long as only input stacks have been clicked), as the difference starts out even and at every step can only change by an even amount. This is because the height of the input stacks are four times the input values. Thus, the whitespace inside the gadget is only separated from the whitespace above and below when the gate is closed. If the whitespace within the gadget is clicked when the gates are closed, then the gadget collapses to be a solid red block of height $L + 2$, thanks to the "cushions" indicated in Figure 5. Because an equal number of blocks are destroyed within the two columns, the configurations of all the gadgets above are unaffected when this happens. We refer to this event (the gates closing, and then the whitespace within being clicked) as the *activation* of the gadget. The closing gates serve to cut the "ribbon" of white tiles; the activation step makes this permanent (so that further clicking on the input stacks does not re-connect the ribbon).

Because the base has $m + 1$ white layers (one of which will be removed with the white tiles of the input gadget), we can only solve the Clickomania instance by cutting the ribbon m times (into $m + 1$ pieces—the input gadget, and the m separation gadgets, which we describe next).

4.1.4 The Separation Gadget S_j

Finally, for every $j \in [m]$, we include a separation gadget S_j. The separators are made of only white tiles, and serve three purposes: (1) they allow the t_j-verification gadgets to sit in the construction; (2) they separate the t_j-verification gadgets so that they don't interfere with each other; and (3) they serve as the white groups that in the 'canonical' solution to the Clickomania instance will remove the white layers of the base (after the t_j-verification gadgets have all had their interior white tiles removed).

The jth separator gadget consists of a lower 'antenna' coming out of the bottom of the gadget in the left column with height $4t_j$ (so it can sit on top of gadget V_j), and an upper 'antenna' on the right column with height $4t_{j+1}$ (if $j = m$ this antenna is omitted, so that the final construction is perfectly rectangular in shape). The middle of the gadget consists of a height-L block.

5 Proving the Reduction Works

In order to prove that the reduction works, we must show two things:

1. If the Sum-Ordering instance has a solution, then the Clickomania instance also has a solution.

2. If the Sum-Ordering instance does not have a solution, neither does the Clickomania instance.

5.1 Sum-Ordering to Clickomania

We first show that if the Sum-Ordering instance has a solution, so does the Clickomania instance. Recall that a Sum-Ordering instance is a pair of multisets

$$A = \{a_1, a_2, \ldots, a_n\} \quad \text{and} \quad T = \{t_1, t_2, \ldots, t_m\}$$

where each element is a positive integer. We can assume the elements of T are sorted in ascending order (in fact, we can assume that T is a set and not a multiset). A solution to the problem is an ordering σ on the set A such that every element of T can be expressed as a prefix sum of A in order σ; formally, σ is a permutation on the set $[n]$, and the solution is valid if for all j there is a positive integer k_j such that $t_j = \sum_{i=1}^{k_j} a_{\sigma(i)}$. We consider the following strategy:

1. Click the input stacks in the order σ;

2. Whenever a gate in a verification gadget closes, complete the activation by clicking the (now isolated) whitespace within the gadget;

3. after all input stacks have been clicked (and all gadgets have been activated—by definition, all gadgets will activate, as all the target sums are expressible as prefix sums of A in the order σ), click on the remaining groups from bottom to top (so each click removes a layer of the group).

This sequence of events will result in the clearing of the board, because L is sufficiently large so that none of the solid-white or solid-red groups ever separate, and the whitespace within each t_j-verification gadget also never separates.

5.2 Not-Sum-Ordering to Not-Clickomania

We now consider the other direction of the proof. In effect, we are trying to show that there is no way to solve the Clickomania instance described except for the 'canonical' solution (which depends on having a solution for the Sum-Ordering instance). Our proof relies heavily on treating the white tiles within the input gadget and within each separation and verification gadget as "bundles" (we avoid the terminology 'group', as this is already used in this paper to refer to clickable groups). We also refer to the red tiles within each verification gadget as "red bundles". The outline of the proof is as follows:

1. Show that (thanks to L being sufficiently large) the "bundles" of white tiles within the verification gadgets and separator gadgets can never become split;
2. Show that all clickable groups are in or are combinations of the following categories:
   (a) input stacks
   (b) bundles (either white or red) or unions of bundles;
   (c) red "bars" (two horizontally-adjacent tiles) created by a closing gate;
   (d) the "leftovers" groups created when the white tiles in the input gadget is clicked (which consists of the red tiles in the balancer sub-gadget, plus all the input stacks which have not been clicked so far);
3. Show that it is impossible to use the white bundles within the verification gadgets to break through the base, without either previously or simultaneously destroying a separation gadget for no reason.

Once we have done this, we will know that (i) it is impossible to create any white groups above the base except for bundles and combinations of bundles; (ii) using a verification bundle to break through a layer of the base implies the waste of at least one separation bundle, and the first use of a verification bundle for this implies the waste of at least *two* separation bundles; (iii) there are precisely enough separation bundles to break through the white layers base. Point (ii) implies that the use of any verification bundle reduces the total number of bundles available, and (iii) therefore implies that using a verification bundle to break through the white layer of the base leaves the player without the necessary number of distinct white clickable groups to successfully break through the base.

We need the following formal definitions (with a slight abuse of terminology, in which we refer to the separator gadgets as "bundles" as well, because they consist *entirely* of white tiles):

Definition 1 (Bundle)**.** *We refer to the following sets of tiles as "bundles":*

1. *the set of white tiles within the input gadget as well as the very top white tile in the base (which is connected to the white tiles in the input); we denote this as* S_0*.*
2. *the set of white tiles within each separation gadget* S_j*; we denote this as* S_j*, because it is all the tiles within the gadget.*
3. *the set of white tiles within each* t_j*-verification gadget* V_j*; we denote this as* C_j*.*
4. *the set of red tiles within each* t_j*-verification gadget* V_j except *the four red tiles involved in the two gates; we denote this as* C_j^**.*

We also define the *bias* of a set of tiles:

Definition 2 (Bias and Balance)**.** *The* bias *of a set* X *of tiles, which we denote as* $b(X)$*, is the number of tiles* X *contains in the right-hand column minus the number of tiles it contains in the left-hand column. A set is* balanced *if its bias is* 0*.*

It is intuitively obvious that if X is a clickable group, $b(X)$ is how much the right-hand column will fall relative to the left-hand column (above X) after it is clicked. We make the following observations:

1. for all $j \in [m]$, $b(S_j) = 4(t_{j+1} - t_j)$, and $b(S_0) = 4t_1 + 1$ (for the additional top white tile of the base, which is on the right-hand column because the base always has even height).
2. for all $j \in [m]$, $b(C_j) = b(C_j^*) = 0$.

We now state the main lemma in the proof:

Lemma 2 (Bundles Stay Together!)**.** *Every bundle has the following property: either all the tiles in the bundle are part of the same clickable group, or all of them are singletons (or all have been removed).*

Proof. We prove this by induction, starting from the lowest bundles. Our base case is the input bundle S_0, which is obvious: nothing below it can be clicked, and within it only the input stacks can be clicked, which obviously cannot disconnect it.

We now suppose that the theorem is true for all bundles below a given bundle X (we do not specify which type because this applies for all types); for this purpose we consider C_j to be below C_j^* because it is impossible to affect C_j by clicking on C_j^* (indeed, C_j^* is not clickable at all until C_j has already been removed). All tiles are contained in bundles except for the following list:

1. base tiles *except* for the white tile in the top layer;
2. input stacks;
3. the red tiles in the 'gate' mechanisms;
4. the red balancer tiles in the input.

Note that the only way a red balancer tile can be clicked is if the bundle S_0 has been removed already. Further note that this will connect not only all the balancer tiles, but connect them to all remaining input stacks as well. Thus, at the moment of elimination, all the red balancer tiles are eliminated, as well as all the input stacks—which has an overall bias of 0. Consequently, removing the balancer tiles can only serve to reduce the overall bias of everything which has been clicked before (because some bias may have been previously incurred by clicking on input stacks).

Note also that except for the top and bottom white base tiles, all base tiles are removed in pairs (one from each column). Thus, they incur no additional bias.

The red tiles in the "gate" mechanisms can only be clicked under three circumstances: (i) if the verification gadget V_j in question has been activated (in which case they are effectively part of the bundle C_j^*); (ii) if the separator gadget below them has been removed, in which case they effectively become part of the red bundle below (or, if $j = 1$, part of the 'input gadget leftovers' group containing the remaining input stacks and red balancer tiles); and (iii) when the gates are closed and the two tiles form a "bar." In each case, because any two red tiles belonging to the same gate are removed at exactly the same time, they cannot contribute any bias no matter how they are clicked.

Thus, we have shown that it is actually impossible to get the left-hand column of X to fall relative to the right-hand column. Furthermore, the right-hand column can only fall by the sum of the input stacks plus the sum of the biases of all bundles below it, which is *at most* (because some bundles may be situated above X)

$$\left(\sum_{i=1}^{n} 4a_i\right) + \left(\sum_{j=2}^{m} 4(t_j - t_{j-1})\right) + 4t_1 + 1 = 4t_m + 4t_m + 1 = 8t_m + 1 \leq L - 1$$

(by the fact that the sum of the elements of A is t_m and the other terms collapse due it being a telescoping sum). But because all bundles have an overlap of L tiles between their right-hand and left-hand column tiles, this amount of relative fall is unable to disconnect them because they will still have at least a remaining overlap of L minus the relative fall (which we just showed is at least 1). □

We know that we need to eliminate all $m+1$ white layers of the base one at a time, using distinct white clickable groups. However, all white tiles not contained in the base belong to bundles. We also note that we have white bundles $S_0, S_1, \ldots, S_m$ and $C_1, C_2, \ldots, C_m$. In the 'canonical' solution, it is $S_0, S_1, \ldots, S_m$ which are used, while $C_1, C_2, \ldots, C_m$ are used to activate the gadgets. Suppose, however, that we want to use some bundle C_j to help

break through the base (this means the tiles in the bundle are directly adjacent to the base, not connected through another bundle). Then, at the time it reaches the base (all bundles below it have been clicked) the gates must be closed. Note that a gate can still divide the whitespace even if it is one away from closing; however, all biases (including, of course, groups with bias zero) are divisible by four, except for the input bundle which has a bias $\equiv 1 \mod 4$. This means that the offset of one cannot make a non-closed gate closed because a non-closed gate is at least four tiles away from being closed. Furthermore, note that *all* the bundles below it must have already been clicked (otherwise it will not reach the base), as well as the balancing red tiles from the input gadget.

We consider the click that removed the white bundle S_{j-1} (directly below our bundle C_j). Because this click did not eliminate C_j, it must have happened when the gates were closed already. However, this cannot have removed a layer of the base, because that would only happen if all groups below S_{j-1} had already been clicked, giving a total incurred bias of $4t_{j-1} + 1$, and therefore the gates of V_j (which had an initial height of $4t_j$) are open at that moment. Thus, the separator gadget S_{j-1} cannot have been used to remove a layer of the base if C_j is used to remove a layer of the base. Furthermore, the separator S_j above also cannot be used to break through the base because either (i) it was removed before C_j, or (ii) it is still there when C_j breaks through the base. Case (i) is easy to handle: if S_j is removed before C_j (which is below it), it cannot have removed any layers of the base. Case (ii) is handled by noting that when C_j is clicked, all the bundles below it have been removed, as well as all the leftover red tiles (and some base tiles, but these are evenly distributed between the columns). The total bias of this set is $4t_{j-1} + 1$, as above, meaning that the gate of V_j is open at this time, so C_j and S_j are directly adjacent. Thus, if any bundle C_j is used to break a layer of the base, then S_{j-1} and S_j cannot be.

Thus, any strategy which uses any C_j to break a layer of the base cannot succeed, because if we let J be the set of indices j for which C_j is used to break the base, then for all $j \in J$, S_j cannot be used to break a layer of the base. Neither can $S_{j'}$ where $j' = (\min_j J) - 1 \neq J$. Thus, strictly fewer than $m+1$ distinct white clickable groups can be used to break the base, so the base cannot be completely removed by such a strategy.

Thus, we have proven that any successful strategy must use only $S_0, S_1, \ldots, S_m$ to break the white layers of the base. However, there are exactly $m + 1$ sets here, so it must use *all* of them to do so—thus, eliminating any of $S_1, S_2, \ldots, S_m$ before eliminating all the tiles in the input group is doomed to fail. Furthermore, if V_j has not been activated, and the gates are open, clicking on S_{j-1} will also destroy S_j, thus preventing a solution. Thus, for each V_j, there must be some point at which the gates are closed. The only way which a non-canonical solution can still happen is if the gates of V_j close because some separator gadgets below were removed. But because all separator gadgets have to be used to eliminate white layers of the base, this could only happen after the entire input gadget was removed. But this means that the total bias incurred is at most the total bias of the separator gadgets below (plus one from the input gadget as a whole), which is $4t_{j-1} + 1$. This is less than $4t_j - 1$ (the smallest amount by which the right-hand column can slide relative to the left and cause the gates of V_j to close).

Thus, every t_j-verification gadget must have its gates close at some point *due only to a set of input stacks being clicked*, which implies that the order the input stacks are removed is a solution to the Sum-Ordering instance.

We have therefore shown that the constructed Clickomania instance has a solution if and only if the original Sum-Ordering instance has a solution. Therefore, this is a valid polynomial-time reduction. Because Sum-Ordering is NP-complete, 2-color 2-column Clickomania is NP-hard; and because 2-color 2-column Clickomania is also in NP, it is NP-complete.

Combining this with Biedl et al.'s proof that 1-column Clickomania is in P, with the trivial fact that 1-color Clickomania is also easy to solve (constant time, in fact, because the solution is "click and win"), and with the fact that having more columns and colors cannot make Clickomania easier, we get the proof to Theorem 1.

5.3 The Score Variant

Having established Theorem 1, we proceed to the score variant. We will use the scoring rule that the number of points awarded at each click is quadratic in the number of removed tiles. However, our constructions and proofs can easily be adjusted for any other mathematically nice scoring rule, provided that it is greater than linear. (More technically, it has to be at least x^c where $c > 1$, as opposed to something like $x \log x$, as otherwise the resulting reduction incurs an exponential blowup in size). We first restate Corollary 1:

Corollary 2. *Score-variant* Clickomania(w, c) *is NP-complete if* $w \geq 2$ *and* $c \geq 2$ *(and* Clickomania$(w, 1)$ *is trivially in P for all values of* w*).*

What we really need to prove is that the score variant is NP-complete, even when restricted to two columns and two colors.

Proof. The construction for the score variant is the same construction described above, except with two "reservoirs" of red tiles, of height R each (where R is the number of tiles in the elimination variant construction, plus 2), one above and one below (see the leftmost diagram in Figure 8), and one additional layer added to the base (hence the definition of R as having an extra +2 term). Thus, the construction has height exactly $5R$. The objective score is then set at $16R^2$.

Suppose that the player can connect the two reservoirs, which contain $2R$ tiles each, and eliminate them with a single click; this immediately awards all $16R^2$ points. If the player cannot connect the two reservoirs, then they can't do better than eliminating one reservoir with as many tiles from the central construction as possible, and eliminating the other, which awards $(3R)^2 + (2R)^2 = 13R^2$ points (this of course is a wild overestimate because there is obviously no way to eliminate most of the central tiles along with a reservoir, because, among other things, most of them are white while the reservoir is red). Therefore, the player can win *only* by connecting the two reservoirs. But, this is only possible by solving the construction in between, which is NP-complete, and so we have shown that the score variant is also NP-complete, even in the 2-color, 2-column case.[2] □

[2]As noted in the introduction, the full table of Clickomania hardness results in Figure 4 does not quite

The following two sections (5 and 6) explore more advanced complexity notions (hardness of approximation and parameterized complexity, respectively), and the material is significantly more difficult. While we will try to provide the reader with all the definitions and explanations necessary to understand the material, these sections can be skipped in favor of continuing directly to Section 7 (conclusion and discussion of open problems).

6 Approximation is Hard, Too

We now consider the hardness of computing approximate solutions to Clickomania problems. We will be using the concept of a *gap-producing* reduction. Let OPT be the highest score (or largest number of tiles which can be eliminated) starting from a given Clickomania configuration. Then computing a c-approximation is hard if the following problem is hard (where an instance of the problem is a Clickomania board and x represents a threshold value):

1. return 'yes' if OPT $\geq cx$;
2. return 'no' if OPT $\leq x$;
3. if $x <$ OPT $< cx$, then either 'yes' or 'no' is accepted.

For this definition, c does *not* have to be a constant—it can be a function of n (where n is the size of the problem). Often, the gold standard for showing that approximation is hard for a given problem is to show that $(n^{1-\varepsilon})$-approximation is hard for all $\varepsilon > 0$.

Intuitively, this problem asks if it is possible to distinguish instances where the optimal value is *large* from where it is *small*. The performance of the algorithm on cases where it is in-between is not important. Note that for approximability, the elimination variant is identical to the score variant if the scoring rule is that the player earns a point for every tile removed.

Using the elimination variant NP-hardness construction, we can get a variety of hardness-of-approximation results.

Theorem 3 (Approximation Hardness)**.** *Where n is the number of tiles in the instance, and for any $\varepsilon > 0$:*

1. *$(2-\varepsilon)$-approximation of 2-color 2-column Clickomania is NP-complete in the score variant.*
2. *$(n^{1-\varepsilon})$-approximation of 3-color 2-column Clickomania is NP-complete (in either variant) if the initial configuration is allowed to* not *be a rectangle.*

apply to the score variant, because the polynomial-time algorithm for the case of one column (the lower-leftmost tile) is only for the elimination variant. We conjecture that one-column Clickomania is not NP-complete for the score variant, but this problem remains open.

3. *$(4-\varepsilon)$-approximation of 3-color 2-column Clickomania is NP-complete in the elimination variant (this result differs from the previous one in that it holds even if the initial configuration* must *be perfectly rectangular).*

4. *for 2-color 6-column Clickomania (where the initial configuration is allowed to* not *be a rectangle):*

   (a) *in the elimination variant, $(n^{1-\varepsilon})$-approximation is NP-complete;*

   (b) *in the score variant (with quadratic scoring), $(n^{2-\varepsilon})$-approximation is NP-complete (if scoring is non-quadratic, this result changes: specifically, if eliminating x tiles with one click yields x^d points, then $(n^{d-\varepsilon})$-approximation is NP-complete).*

The reductions for these results are shown in Figure 8.

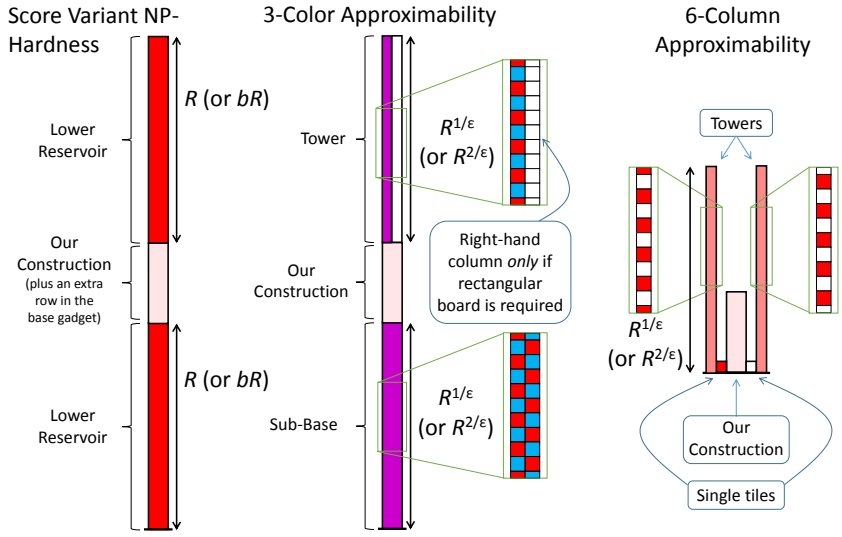

Figure 8: Constructions for approximation reductions. **Left:** The score variant NP-hardness reduction (also proves approximations of order $2+\varepsilon$ are hard). For the NP-hardness reduction of the score variant, the height is R; for the approximation reduction (as described in Theorem 3) the height is bR, where b is a large constant (described in the proof). **Middle:** 3-color 2-column approximability reduction (for both elimination and score variants). If the initial configuration is required to be rectangular, the white column in the tower is included (but makes the hardness result weaker); otherwise it is left out. For the elimination variant, the height is bR, while for the score variant, the height is bR^2. **Right:** 2-color 6-column approximability reduction; only viable if non-rectangular initial boards are allowed. As with the middle reduction, height of the towers is bR for the reduction to the elimination variant, bR^2 for the reduction to the score variant. Single tiles separate 'towers' from the main body; the player must connect the two towers and can then eliminate both of them (they start offset by 1 so they cannot be eliminated, but in the process of connecting them the offset is removed).

All are based on the same idea as the score variant NP-hardness reduction: engineering a board so that the goal (achieving a good approximation of the optimal score or number of

eliminated tiles) can be achieved if and only if a copy of our original construction is solved. This produces a reduction to the approximation task from Sum-Ordering.

Proof. We prove these in order:

1. This follows from the left diagram in Figure 8, where the reservoirs are bR tiles tall rather than R. The highest number of points the player can get *without* connecting the reservoirs is $(2b^2+1)R^2$ (where each reservoir contributes b^2R^2 and the other tiles contribute at most R^2 in total—as in the proof of the score variant's NP-completeness, this is an overestimate). However, if we can connect the reservoirs, we can achieve at least $4b^2R^2$ points in total. Thus, it is NP-complete to get an approximation of

$$\frac{4b^2}{2b^2+1} \geq 2-\varepsilon \text{ for sufficiently large } b$$

   Thus, for any constant $\varepsilon > 0$, it is NP-complete to approximate the optimal score to a factor of $2-\varepsilon$.

2. The reduction used for this is depicted in the middle diagram in the figure. For this reduction, the tower does *not* have the right-hand column of white tiles, as we do not have the requirement that the initial board be perfectly rectangular. In order to remove the sub-base, the player has to connect it to the tower. Neither can be removed without connecting it to the other, which implies a solution to the Sum-Ordering instance. For the elimination variant, the height of the sub-base and tower is $R^{1/\varepsilon}$ and for the score variant their heights are $R^{2/\varepsilon}$. Thus, in the elimination variant, one can eliminate at most R tiles if they are not connected, and can eliminate the $3R^{1/\varepsilon}$ tiles in the tower and sub-base if they are connected. Similarly, in the score variant, eliminating just part of the construction yields at most R^2 points, while eliminating everything yields at least $6R^{2/\varepsilon}$ points (because the least points are awarded for eliminating tiles in pairs, with each pair giving four points. To put it another way, eliminating x tiles yields at least $2x$ points). Because n is the size of the problem, it is proportional to $R^{1/\varepsilon}$ (in the elimination variant) or $R^{2/\varepsilon}$ (in the score variant). Therefore, in order to determine whether one can eliminate n tiles (by connecting the two towers) or less than n^ε tiles (from only the construction in the middle), one has to solve a Sum-Ordering problem (in the score variant, this is whether one can gain n points or n^ε points). The gap between these two is $n^{1-\varepsilon}$, and so $(n^{1-\varepsilon})$-approximation is hard. An important point for this proof is that ε is constant, so the blow-up in size from the original Sum-Ordering problem to the construction remains polynomial.

3. This reduction is also depicted in the middle of Figure 8, though now it needs to be rectangular. Hence, it has the right-hand column of the tower. This makes the reduction unsuitable for the score variant—when the player removes this extra column, it will award more points than the tower and sub-base combined—but the reduction is still valid for the elimination variant. However, the player can now *guarantee* the removal of slightly more than 1/4 of the whole construction, regardless of whether

the corresponding Sum-Ordering instance has a solution. This is because the new column is 1/4 of the "new" tiles (i.e. everything except the original construction), and the player can remove it along with most of the original construction. However, to remove *any* of the sub-base or the tower, the player still has to solve the Sum-Ordering instance. Thus, because sub-base and tower can be made tall enough (by scaling it up by a constant factor) to make them hold arbitrarily close to 3/4 of all the tiles, the construction has the following property:

(a) If the Sum-Ordering instance corresponding to the construction has a solution, then all the tiles can be removed;

(b) If the Sum-Ordering instance does not have a solution, then only 1/4 of the tiles (plus an arbitrarily small fraction) can be removed.

Thus, we get that approximating to a factor of $4 - \varepsilon$, for any $\varepsilon > 0$, is NP-complete.

4. This reduction is depicted on the right of Figure 8. To either side of the original NP-hardness construction, we add towers of alternating red and white tiles, of size $R^{1/\varepsilon}$ (for the elimination variant) or $R^{2/\varepsilon}$ (for the score variant). The proof follows the same logic as for statement (2) of the theorem: the towers can only be removed if the original problem is solved, which connects them to each other.

   For the elimination variant, if the player cannot connect the tower, this limits her to removing at most R tiles. Because the towers are large enough, R is less than n^{ε}, and therefore we have the result that $(n^{1-\varepsilon})$-approximation is NP-complete (because the player can eliminate all n tiles if she connects the towers).

   For the score variant, we have to specify *how* the towers are removed. Although the towers start out at an offset, when they are connected, one of them (specifically, the one on the left) will lose its bottom tile, and the two will come into alignment (i.e. the rows of the resulting 2-column tower will be alternating red and white). The player can then eliminate all the red rows, leaving just the white ones. This will leave $R^{2/\varepsilon}$ white tiles in one large group, yielding $R^{4/\varepsilon}$ points. Specifically, we eliminate almost 1/2 of the tiles in the construction in one click. So, to be conservative, if n is the number of tiles then we get at least $n^2/5$ points. Because of the height of the towers, R is less than $n^{\varepsilon/2}$, leading to a gap of at least $n^{2-\varepsilon/2}/5$, which is greater than $n^{2-\varepsilon}$ for large enough n. Thus, $(n^{2-\varepsilon})$-approximation is NP-complete for the score variant.

□

7 Parameterized Clickomania

We now move on to the parameterized complexity of Clickomania. In Section 7.1 we define our notions of parameterized complexity. We then discuss variants of parameterized Clickomania problems and the associated hardness of these problems in Sections 7.2 and 7.5.

7.1 Fixed-Parameter Tractability and Parameterized Complexity

Recall that we regard NP-complete problems to be ones where we believe no efficient polynomial time algorithm exists. Many such presumably hard problems rely on the use of a parameter k. For example, for problems like k-Vertex Cover, we want to find a set of k vertices in a graph G (with n vertices) such that every edge in G has an endpoint that is a vertex in the set. This problem is NP-complete in the general sense that we do not believe it can be solved in polynomial time with regard to the size of the graph (note that k could potentially be $n/2$, so brute-force methods would need to check $\binom{n}{n/2}$ cases). However, it is fixed-parameter tractable, which means that if we consider k to be small and, thus, can allow exponential blow-up on k only, then it can be solved efficiently. In the fixed-parameter tractability paradigm, we can find efficient algorithms by confining the chaos of exponential blow-up to the parameter k, leaving a largely organized and efficient algorithm with regard to n, the size of the input.

More formally, a problem is considered *fixed-parameter tractable* with regard to a parameter k if there exists a solution algorithm in time $O(f(k) \cdot n^{\alpha})$, where $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is some arbitrary function taking k as the input and α is a constant independent of k and n. As with problems in NP, we believe that certain parameterized problems cannot be solved even in $O(f(k) \cdot n^{\alpha})$ time. The *W-hierarchy* is a collection of parameterized complexity classes used to classify such presumably hard parameterized problems. We consider problems that do not have fixed-parameter algorithms and lie in any of these classes to be *W[1]-hard*.

As in classical complexity, we also perform reductions to show that a parameterized problem is hard. Such a reduction to show that a problem is W[1]-hard is called a *fixed-parameter reduction* (FPT-reduction). A formal definition of an FPT-reduction is given below:

Definition 3 (Fixed-parameter reduction)**.** *Given a finite alphabet Σ, a parameterized problem, (Q, κ), consisting of $Q \in \Sigma^*$, and a polynomial time computable parameterization function, $\kappa : Q \to \mathbb{Z}^+$, an FPT-reduction from one parameterized problem, (Q, κ), to another, (Q', κ'), satisfies the following conditions [DF95, FG06]:*

1. *There exists a mapping $R : \Sigma^* \to \Sigma^*$ such that for every $x \in \Sigma^*$, $x \in Q$ if and only if $R(x) \in Q'$.*
2. *R is computable in time $O(f(\kappa(x))|x|^{O(1)})$ where $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is a computable function.*
3. *There exists a computable function, $g : \mathbb{Z}^+ \to \mathbb{Z}^+$ where $\kappa'(R(x)) \leq g(\kappa(x))$.*
4. *The reduction is linear if $\kappa'(R(x)) = O(\kappa(x))$.*

7.2 k-Click Score-Variant Clickomania

We now define a parameterized variant of Clickomania. In this case, the parameter k is the number of clicks allowed to the player. The *k-Click Score-Variant Clickomania* problem is Clickomania where the player can only make k clicks and wants to achieve a score greater

than some bound S. Here, we are concerned with the game where each cleared tile confers an additive score of one in a game with only two colors. We observe this reduction also holds and can be easily adapted to the scoring variant where given a group of size x tiles, a player obtains $(x-2)^2$ points for clearing that group, a question posed by [BDD+02].

We note that if n is the number of tiles in the initial configuration, then a brute-force algorithm can determine the best line of play in $O(n^{k+1})$ time (n^k possible ways to use k clicks, and $O(n)$ time to update the board after each click). Thus, for fixed k, there is a polynomial-time solution (a feature of virtually all problems parameterized in this way).[3] However, although this algorithm is polynomial for fixed k, we cannot generally assume that k is a fixed constant, and this algorithm is not efficient enough to be considered a fixed-parameter algorithm (as described in Section 7.1). In fact, we show below that k-Click Score-Variant Clickomania is hard in the parameterized sense.

To prove that k-Click Score-Variant Clickomania is W[1]-hard, we provide a linear FPT-reduction from k-Independent Set. The k-Independent Set problem is defined as follows:

k-Independent Set [DF95]. *Given a graph $G(V, E)$, does there exist an independent set (i.e. no two vertices in the set are adjacent) containing k vertices?*

It has been shown that k-Independent Set is W[1]-complete for general graphs [DF95].

The basic reduction scheme from k-Independent Set on a simple graph $G(V, E)$ for the W[1]-hardness of k-Click Score-Variant Clickomania is to create vertex blocks that uniquely identify each vertex $v \in V$. Each vertex block contains identification markers used to distinguish each edge $e \in E$. We will show that the player can only achieve the maximum score by clicking blocks that correspond to a set of k independent vertices in G.

7.3 A Blueprint of the Parameterized Reduction

The idea behind the reduction is to represent the vertices and edges in the k-Independent Set graph instance, G, using Clickomania constructions such that a series of clicks in the Clickomania instance that gives the maximum score will also give an independent set in G and vice versa. We use the following set of gadgets when we construct the Clickomania instance:

1. *Vertex planks* represent every vertex in G. At least k vertex planks must be clicked in the Clickomania instance in order to achieve the maximum score.

2. *Identification markers* are used to represent each edge. An identification marker is a column of tiles of some height that resides on top of a vertex plank. A vertex plank contains an identification marker of an edge if the vertex is one of the edge's two endpoints. The markers are distinguished from each other by its position on the plank.

3. An *infinity block* is a reservoir of points that will result in a large increase in a player's score if clicked.

[3] Knowing this is also interesting because it places k-Click Clickomania in the parameterized complexity class XP. It is known that XP strictly contains all fixed-parameter tractable problems.

4. The *winning block* is a block that is used to connect two infinity blocks so that both infinity blocks can be cleared in one click.

5. *Separator sections* are rows of interchanging colored tiles that are used to separate vertex planks. Their main purpose is to prevent clicking one vertex plank from interfering with clearing other gadgets.

Using these gadgets, the following steps are used to achieve a maximum score in the constructed Clickomania instance given k clicks.

1. Click the k vertex planks that correspond with a set of k independent vertices in G using k clicks.

2. Click the winning block using one click.

See Fig. 10 for a figure relating to the construction. The proof and details of the construction are given in the next few sections.

7.4 Proof of W[1]-hardness of k-Click Score-Variant Clickomania

Given a graph $G(V, E)$, and asked to find an independent set of size k, we create an instance of Clickomania through an FPT-reduction where the maximum score can be achieved with $k + 1$ clicks. We first define in more detail the components involved in the W[1]-hardness proof briefly summarized in Section 7.3.

1. A *vertex plank* is a long horizontal block with some height, $p = 2h|V|$, and width $w+c$ where $w = b|E| + c + 1$, c is some constant where $c \geq 1$, and b is a constant where $b > 1$. Let $h = |V|$ if $|V| \mod 2 = 1$ and $h = |V| + 1$ otherwise. In the figures we use to illustrate our reduction, we choose $c = 11$ and $b = 8$. We create a vertex plank for each $v \in V$.

2. Each edge, $(u, v) \in E$, is uniquely identified in the reduction by *identification markers*. Each edge is represented by a column of size $1 \times 2h$ at a unique distance x_i from the left ends of the vertex planks representing the vertices u and v where $x_i = bi + c + 1$ and $i \in \{0, \ldots, |E| - 1\}$. The height of each identification marker is $2h$. Because each identification marker at distance x_i for $i \in \{0, \ldots, |E| - 1\}$ is used to identify one endpoint of an edge, there are at most 2 identification markers at each distance x_i. Each v_i also has a block of width c and height $4h$ at distance w from the left end of the plank. Fig. 9 shows an example of a vertex plank with identification markers.

3. The vertex planks are stacked vertically one on top of another, separated by regions of interchanging colored horizontal blocks. Each *separator section* is composed of horizontal strips of size $w \times 1$ or $(w+c) \times 1$. Some of these strips intersect identification markers (thus causing them to separate into different segments between two markers). The top of a vertex plank (i.e. the top of the $c \times 4h$ block) and the bottom of a different vertex plank are separated by a separator of height $4h|V|$.

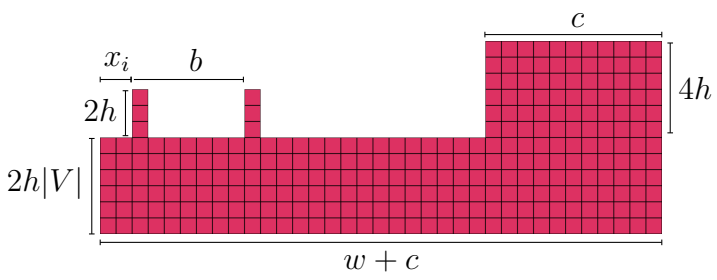

Figure 9: Example vertex plank with identification markers. Here, $c = 11$, $b = 8$, $h = |V|$ and $w = b|E| + c + 1$.

4. On top of the construction is a *winning block* with dimensions $w \times 2h$. Next to the winning block is a column of interchanging colored horizontal strips each with height 1 and width c. There are a total of $2h(k + 1) - 1$ such strips.

5. On top of both the winning block and the column are *infinity blocks* of size $c \times \Psi$ where $\Psi = h[2c(k + 1) + 2w + (6|V| + 4)(w + c)|V|] - c$. Figure 10 shows a diagram of the described reduction. This instance will give a maximum score that is greater than $2c\Psi + hk(2|V|(w + c) + 4c + 2w) + 2hw$ after $k + 1$ clicks if and only if an independent set of size k exists in G. Below, we formally prove that the above reduction is a valid FPT-reduction.

We now proceed to the proof that the stated reduction from k-Independent Set to k-Click Score-Variant Clickomania is valid by Definition 3. Lemma 3 proves that the transformation lies within the stated time bound for a FPT-reduction and Theorem 4 proves that there exists a solution in the given k-Independent Set instance if and only if there exists a solution in the constructed k-Click Clickomania instance.

Lemma 3. *Our transformation, R, of an instance of k-Independent Set into an instance of k-Click Score-Variant Clickomania takes $O(f(\kappa(x))|x|^{O(1)})$ time and there exists a computable function, g, where $\kappa'(R(x)) \leq g(\kappa(x))$.*

Proof. In our transformation, $\kappa'(R(x)) = \kappa(x) + 1$ and so a function g exists to satisfy the required condition. The reduction creates $3\Psi = 3(h[2c(k+1)+2w+(6|V|+4)(w+c)|V|]-c) = O(|V|^5)$ Clickomania tiles where $c = 11$, $h \leq |V| + 1$, $w = b|E| + c + 1$ and $b = 8$. Therefore, the running time is polynomial in the input. □

Lemma 4. *A score greater than $S = 2c\Psi + hk(2|V|(w + c) + 4c + 2w) + 2hw$ using exactly $k + 1$ clicks may be achieved only when k vertex planks are clicked and no two vertex planks share the same identification marker. In addition, players may not cheat by clearing the vertex planks in a specific order.*

Proof. We first argue that clicking less than k vertex planks will not achieve a score greater than $S = 2c\Psi+hk(2|V|(w+c)+4c+2w)+2hw$. We prove this statement in two parts: first, we prove that clicking any set of $k - 1$ vertex planks and the infinity blocks cannot achieve a maximal score; then, we prove that clicking $k - 2$ vertex planks, another component (that is not a vertex plank), and the infinity blocks also cannot achieve the maximum score.

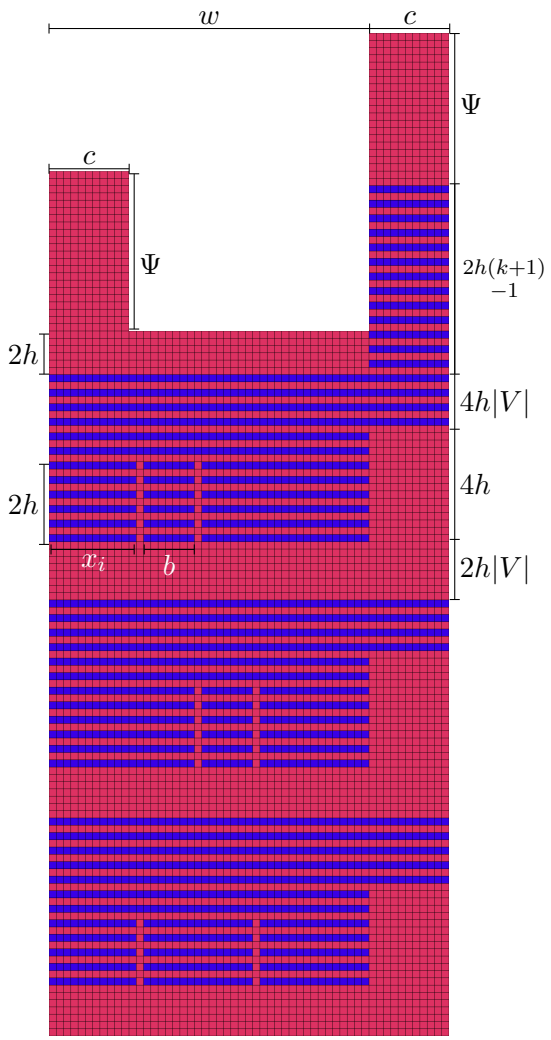

Figure 10: A k-Click Score-Variant Clickomania reduction instance. This figure is not to scale. Here, $c = 11$, $b = 8$, $h = |V|$, $w = b|E| + c + 1$, and $\Psi = h[c(2k-1) + 2w + (6|V| + 4)(w + c)|V|]$. Here, this is an instance corresponding with a cyclic graph, $G(V, E)$ containing vertices $V = \{A, B, C\}$ and $E = \{(A, B), (B, C), (A, C)\}$.

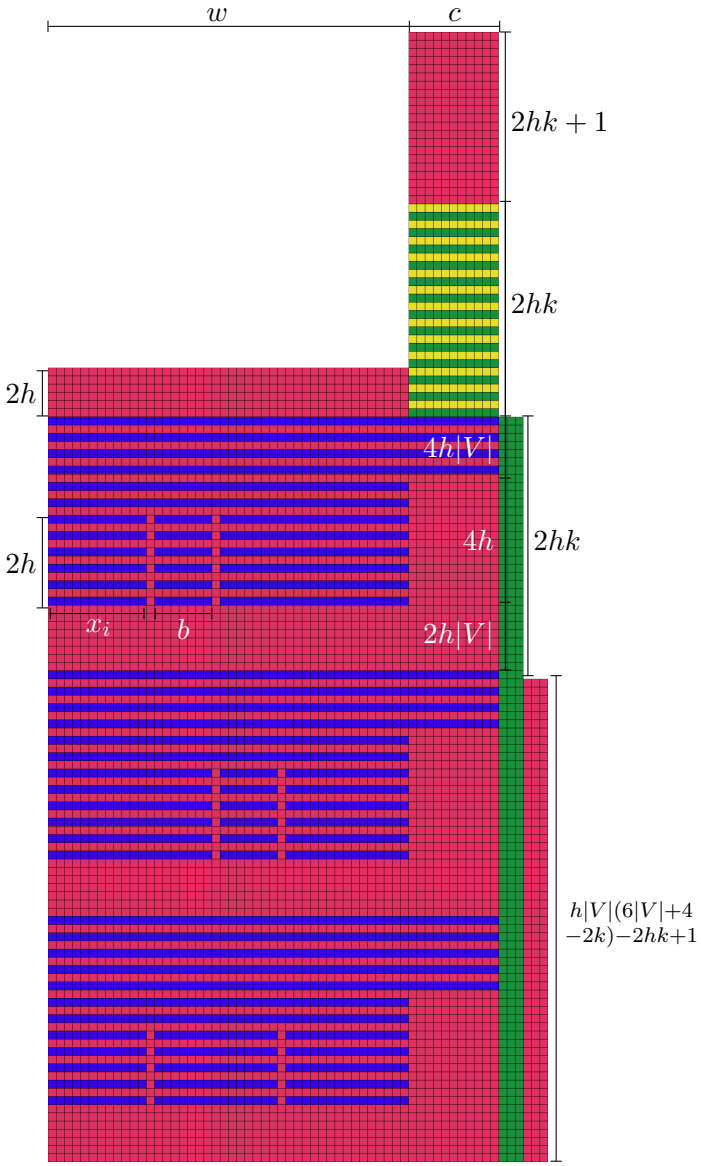

Figure 11: A k-Elimination-Variant Clickomania reduction instance. This figure is not to scale. Here, $c = 11$, $b = 8$, and $h = |V|$, $w = b|E| + c + 1$. Here, this is an instance corresponding with a cyclic graph, $G(V, E)$ containing vertices $V = \{A, B, C\}$ and $E = \{(A, B), (B, C), (A, C)\}$.

1. **Clicking $k-1$ vertex planks and the infinity blocks cannot achieve the maximum score:** To achieve a score of at least $2c\Psi$, one must click both of the infinity blocks because the sum of the scores obtained by clicking all of the other tiles equals Ψ. Therefore, at least one click is required to clear the two infinity blocks. Suppose that only $k-1$ vertex planks were cleared. That means the difference in height between the

infinity block and the winning block in the right column decreases by $2h(k-1)$. There remains $4h-1$ tiles preventing the infinity block from touching the winning block. Therefore, clearing the two infinity blocks takes 2 clicks. Clicking $k-1$ vertex planks results in at most $T=(k-1)h(2|V|(w+c)+4c+2w)+|E|(2h)(k-1)$ points (assuming a completely connected graph). Because $|E|(2h)(k-1) < h(2|V|(w+c)+4c+2w)$ and $T < hk(2|V|(w+c)+4c+2w)$, we conclude that clearing any set of $k-1$ vertex planks and the infinity blocks cannot achieve a score greater than S.

2. **Clicking $k-2$ vertex planks, another component that is not a vertex plank, and the infinity blocks cannot achieve the maximum score:** If one of the $k-1$ clicks was not spent on a vertex plank, then it could be spent clicking something else. It does not benefit to click any of the interspersed horizontal lines because that would result in at most w or c points. The player could alternatively click the merged blue block that results from clicking a vertex plank. This blue block would confer at most $(2h+2)(w+c)+hc < h(2|V|(w+c)+4c+2w)$ points. By performing some clever manipulations, one can see that the maximum number of points that can be conferred by a blue block is $2hk(w+c)$ which is still less than the number of points that is conferred by a vertex plank.

Thus, the only other choice left is for the player to attempt to cheat by clicking the vertex planks in a way that confers an advantage. Below we prove that a player cannot gain an advantage by clearing the vertex planks in a specific order.

Suppose that a player clears a vertex plank that is underneath another vertex plank that they want to clear. The hope is that by doing this, one can potentially clear two different vertex planks with one click or clear the $2hk-1$ horizontal strips separating the infinity block from the winning block with fewer than k clicks. To clear two vertex planks simultaneously, the two planks must be connected by some line of red tiles. However, we show this is impossible to accomplish. First, clearing an identification marker only drops a portion of the vertex plank on top by at most $2h$. Clicking at most $k-1$ vertex planks cannot drop any portion of the plank above by the $4h|V|$ amount necessary to bridge the gap between the two planks.

Furthermore, the gap between any two $4h$ blocks does change from $4h|V|$, so a block of $4h+2h|V|$ height cannot bridge a gap of $4h|V|$ height. The player can choose to click a merged blue block, but clicking this block does not decrease the difference in height between the winning block and the infinity block (in fact it increases it) and also does not help bridge the $4h|V|$ gap.

The last argument is very subtle and requires considering what happens if the vertex planks were cleared in such a way that some of the intervening strips separating the winning block and the infinity block become lined up with some separator section strips that are attached to identification markers. Then, these newly attached strips would be cleared as the vertex plank they are attached to is cleared. In this case, the difference in height between the winning block and the infinity block does not change (through careful checking of cases). The player would only gain $h(w+c)$ more points. If this happens for $k-1$ vertex planks,

then the player would gain a total of $(k-1)h(w+c)$ points which is still less than the amount of points they would gain if they cleared another vertex plank.

Finally, picking two vertex planks with the same identification marker separates the winning block into two pieces because the height of each identification block is $2h$ and the height of the winning block is $2h$. Clicking one vertex block per identification marker does not separate the winning block in two because the entire winning block shifts down by h tiles and the column containing the identification marker shifts down by $2h$, resulting in a change of h height. Clearing two vertex planks with the same identification marker will result in a split in the winning block. □

Theorem 4. *k-Click Score-Variant Clickomania is W[1]-hard.*

Proof. Given a k-Independent Set problem for graph, $G(V, E)$, we can obtain a k-Click Clickomania problem that asks whether the maximum score obtained via $k+1$ clicks is greater than $S = 2c\Psi + hk(2|V|(w+c)+4c+2w)+2hw$. If there exists a k independent set in G, the winning strategy in the Clickomania transformation is to click the vertex planks corresponding to the vertices in the k-sized independent set in G from top to bottom. Then, the remaining click is used to clear the merged infinity blocks and winning block. This is possible because each cleared vertex plank shifts the winning block down by $2h$ and the column containing the infinity block by $4h$ for a net change of $2h$.

If there does not exist a k independent set in G, there does not exist a set of $k+1$ that can solve the Clickomania instance. We proved in Lemma 4 that no other series of clicks can allow the player to cheat in the game. By Lemma 3, we proved that the transformation is polynomial in $|V|+|E|$. Thus, the k-Click Score-Variant Clickomania problem with 2 colors is W[1]-hard by reduction from k-Independent Set. □

7.5 k-Click Elimination-Variant Clickomania is W[1]-hard

The k-Click Elimination-Variant Clickomania game is one where one must clear the Clickomania board using at most k clicks. We provide a reduction from k-Independent Set, similar to the one mentioned in Section 7.2, to prove that k-Click Elimination-Variant Clickomania with 4 colors is W[1]-hard.

Given a graph $G(V, E)$, we construct a Clickomania board shown in Fig. 11 with the same dimensions and components as the max score reduction variant (see Section 7.3) but with two more added columns and the interspersed strips below the infinity block are now interchanging yellow and green strips. We also remove the infinity block on top of the winning block. Although a bit of a misnomer, we still refer to the other infinity block as the *infinity block* despite its purpose is no longer to server as a large points bank. The columns could have any constant width and have heights $h|V|(6|V|+4-2k)$ and $h|V|(6|V|+4-2k)-2hk+1$. In Fig. 11, they have width 3. We now prove this is a valid reduction from k-Independent Set.

Lemma 5. *It is possible to clear the Clickomania board created by the method given above with $k+4$ clicks if and only if there exists a k-independent set in G.*

Proof. First, in the case of $k = |V|$, a polynomial time check will determine whether such an independent set exists. We perform the above detailed reduction in the case when $k < |V|$.

The $k + 4$ clicks are distributed the following way in the Clickomania instance:

1. The player must spend k clicks that clear k vertex planks, no two of which share the same identification marker.

2. The green column of height $h|V|(6|V| + 4 - 2k)$ is clicked to clear the interspersed green rows that have fallen below height $h|V|(6|V| + 4 - 2k)$.

3. Then, the remaining yellow column is clicked which drops the red infinity block by $2hk$ height so that it is barely touching the red column with height $h|V|(6|V| + 4 - 2k) - 2hk + 1$.

4. Clicking any red tile will clear all red tiles from the board.

5. Finally, clicking the remaining merged blue block clears the board.

We first prove that if there does not exist a k-independent set in G, then there does not exist a series of $k + 4$ clicks that will clear the board: First, suppose that $k-1$ vertices are part of an independent set. Then, we can clear $k-1$ vertex planks without disconnecting the winning block. Then, the difference in height between the infinity block and the winning block becomes $2h$. This means that h green strips are not touching the green column. If we click the green column too soon, then there remains at least $2h$ interspersed green and yellow strips which will take $O(|V|)$ clicks to clear (assuming $h = |V|$). Suppose we click another vertex plank. That vertex plank must have the same identification marker as a vertex plank we have clicked previously. Then, the winning block would split into two parts and the red tiles would not be able to be cleared with one click. The other options include clicking the separator section blue strips which would only increase the distance between the infinity block and the winning block. Furthermore, if less than k vertex planks are clicked, then there remains at least $h|V|$ red strips that will not be able to be cleared by using the long column of red tiles. The remaining cases were proven in Lemma 4.

If there exists a k-independent set in G, then there exists a series of $k + 4$ clicks that will clear the board: When a vertex plank is clicked, the horizontal strips in the separator sections are shifted down by an even number of tiles. Therefore, the red and blue horizontal strips maintain their continuity. By the same argument as given in Lemma 4, clicking a set of k vertex planks none of which share identification markers in the same location will eliminate the difference in height between the infinity block and the winning block. Then, clicking the green column only shifts the infinity block by hk down. Clicking the merged yellow block shifts the infinity block down by another hk. At this point, the infinity block is connected to both the winning block and to the column of red tiles. We argued previously that the red strips in the separator sections remain continuous even as we clear vertex planks. Thus, each red strip is either touching a vertex plank, the red column, the infinity block, or the winning block. All vertex planks are connected to the red column.

One click will then clear all the red tiles. Because we eliminated all other colors, the only block that remain is a large blue block, which we can clear with one more click. □

Theorem 5. *k-Click Elimination-Variant Clickomania is W[1]-hard.*

Proof. By Lemma 3, we know that this transformation takes polynomial time because the only difference in the transformation is the addition of two columns of size $h|V|(6|V| + 4 - 2k) - 2hk + 1$ and $h|V|(6|V| + 4 - 2k)$ and the removal of an infinity column of size $\Psi = h[c(2k-1) + 2w + (6|V| + 4)(w + c)|V|]$. By Lemma 5, the transformation guarantees that an answer corresponding with the k-Independent Set instance is the output of the transformation. Therefore, k-Click Elimination-Variant Clickomania using 4 colors is W[1]-hard by reduction from k-Independent Set. □

8 Conclusion

We have shown several hardness results for Clickomania, including NP-completeness for 2-color, 2-column Clickomania, and W[1]-hardness for the parameterized version. In particular, we have closed the question of hardness for Clickomania with the number of columns and colors restricted: as long as there are at least two colors and two columns, the problem is NP-complete, and otherwise it is solvable in polynomial time.

Nonetheless, there are still many open questions regarding the hardness of Clickomania, and we urge anyone who is interested to try to solve these.

1. Recall that we discussed 2-color 2-column Score-Variant Clickomania as a corollary to our NP-completeness result on the elimination variant. However, while this settles the matter for anything with at least 2 colors and 2 columns (and 1 color remains trivial), 1-column Clickomania in the score variant remains open.

   Open Problem 1 (1-Column Score Variant)**.** *What is the complexity of Clickomania in the Score Variant when restricted to one column?*

2. Two natural ways to restrict a Clickomania instance are restricting the number of *copies* (number of tiles of each color), and restricting the number of *rows*. These restrictions are complementary to restricting colors and columns, respectively, because restricting *both* colors and copies (or both rows and columns) leads to only a finite number of valid instances: if there are only c colors and m copies, then the construction as a whole cannot have more than $c\,m$ tiles). Hence, restricting both is not interesting from a complexity perspective, which requires instances that grow to infinity. Thus the interesting combinations are given by the 2×2 Table 1. While this paper has settled the complexity of color-and-column restricted Clickomania, the other three combinations have yet to be solved.

   Open Problem 2 (Alternate Restrictions)**.** *What is the complexity of Clickomania when the other restrictions (number of rows and/or number of copies) are in play?*

	Columns	**Rows**
Colors	Solved	*Open*
Copies	*Open*	*Open*

Table 1: Three out of four combinations of restriction types remain open.

3. We were able to show hardness of approximation in a few instances: hardness of $(2-\varepsilon)$-factor approximation for 2-color 2-column score variant Clickomania, and hardness of achieving *any* constant-factor approximation for 3-color 2-column Clickomania (for both score and elimination variants). This raises the question of approximation for 2-color 2-column Clickomania, both for getting any approximation results for the elimination variant and for possibly improving the result for the score variant.

   Open Problem 3. *For 2 columns and 2 colors, is it hard to approximate (to any constant factor) the number of tiles which can be eliminated? Is it hard to approximate to a factor of 2 (or less) the optimal achievable score in the score variant?*

4. Finally, while we have shown the k-click Clickomania problem to be W[1]-hard and contained within the parameterized complexity class XP, several questions remain open. Most obviously, we have not shown that Clickomania is in W[1] and thus W[1]-complete.

   Open Problem 4. *Is k-Click Clickomania (either elimination-variant or score-variant) in W[1], or does it belong higher in the W hierarchy? What is the parameterized complexity of Clickomania under other restrictions?*

   Our W[1]-hardness reduction also required unrestricted width and height, and in the case of the elimination variant, required 4 colors. Can a similar result be proved under tighter restrictions, or other types of restrictions (as described in Table 1)?

References

[BDD+02] Therese C. Biedl, Erik D. Demaine, Martin L. Demaine, Rudolf Fleischer, Lars Jacobsen, and J. Ian Munro. The complexity of Clickomania. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 389–404. Cambridge University Press, 2002. arXiv:cs.CC/0107031.

[DF95] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1-2):109–131, April 1995.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[HD09] Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. CRC Press, 2009.

[Mor03] Kuniaki "Morisuke" Moribe. Chain-shot! http://www.asahi-net.or.jp/~ky6k-mrb/chainsht.htm, 2003.

[stf15] stfalcon.com. Samegame (swell foop). Google Play app, 2015. https://play.google.com/store/apps/details?id=com.stfalcon.swellfoop.

[Vig12] Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Proceedings of the 6th International Conference on Fun with Algorithms*, pages 357–367, June 2012.