Binary pattern tile set synthesis is NP-hard*

Lila Kari[1], Steffen Kopecki[1**], Pierre-Étienne Meunier[2***], Matthew J. Patitz[3†], and Shinnosuke Seki[2,4‡§]

[1] Department of Computer Science, University of Western Ontario, London ON N6A 1Z8, Canada. {lila,steffen}@csd.uwo.ca

[2] Department of Computer Science, Aalto University, P.O.Box 15400, FI-00076, Aalto, Finland. pierre-etienne.meunier@aalto.fi

[3] Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR, USA. mpatitz@self-assembly.net

[4] Helsinki Institute for Information Technology (HIIT)

Abstract. We solve an open problem, stated in 2008, about the feasibility of designing efficient algorithmic self-assembling systems which produce 2-dimensional colored patterns. More precisely, we show that the problem of finding the smallest tile assembly system which will self-assemble an input pattern with 2 colors (i.e., 2-Pats) is **NP**-hard. One crucial lemma makes use of a computer-assisted proof, which is a relatively novel but increasingly utilized paradigm for deriving proofs for complex mathematical problems. This tool is especially powerful for attacking combinatorial problems, as exemplified by the proof for the four color theorem and the recent important advance on the Erdős discrepancy problem using computer programs. In this paper, these techniques will be brought to a new order of magnitude, computational tasks corresponding to one CPU-year. We massively parallelize our program, and provide a full proof of its correctness. Its source code is freely available online.

1 Introduction

The traditional way for mankind to modify the physical world has been via a top-down process of crafting things with tools, in which matter is directly

* We thank Manuel Bertrand for his infinite patience and helpful assistance with setting up the server and helping debug our network and system problems, and Cécile Barbier, Eric Fede and Kai Poutrain for their assistance with software setup.

** Supported by the NSERC Discovery Grant R2824A01 and UWO Faculty of Science grant to L. K.

*** Supported in part by NSF Grant CCF-1219274

† Supported in part by NSF Grants CCF-1117672 and CCF-1422152

‡ Supported in part by Academy of Finland, Grant 13266670/T30606.

§ Current address: Department of Communication Engineering and Informatics, University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 1828585, Japan. s.seki@uec.ac.jp

manipulated and shaped by those tools. In this work, we are interested in another crafting paradigm called *self-assembly*, a model of building structures from the bottom up. Via self-assembly, it is possible to design molecular systems so that their components autonomously combine to form structures with nanoscale, even atomic, precision. At this scale, tools are no longer the easiest way to build things, and *programming* the assembly of matter becomes at the same time easier, cheaper, and more powerful.

Using this paradigm, researchers have already built a number of things, such as logic circuits [19, 24], DNA tweezers [32], and molecular robots [16], just to name a few. Such examples demonstrate that self-assembly can be used to manufacture specialized geometrical, mechanical, and computational objects at the nanoscale. Potential future applications of nanoscale self-assembly include the production of new materials with specifically tailored properties (electronic, photonic, etc.) and medical technologies which are capable of diagnosing and even treating diseases in vivo, at the cellular level. Furthermore, studying the processes occurring in self-assembling systems yields precious insights about what is physically, even theoretically, possible in these molecular systems. Questions such as "what is the smallest program capable of performing a given task?" arise naturally in these systems, either from experimental applications, or from more fundamental research on the capabilities of natural systems.

The *abstract Tile Assembly Model* (aTAM) was introduced by Winfree [30] to study the possibilities brought by molecular components built by Seeman [25] using DNA. This model is essentially an asynchronous nondeterministic cellular automaton, and can also be seen as a dynamical variant of Wang tiling [29]. In the aTAM, the basic components are translatable but un-rotatable square *tiles* whose sides are labeled with *glues*, each with an integer *strength*. Growth proceeds from a *seed assembly*, one tile at a time, and at each time step a tile can attach to an existing assembly if the sum of the strengths of the glues on its sides, whose types match the existing assembly, is equal to at least a parameter of the model called the *temperature*.

The problem we study in this paper is the optimization of the design of tile assembly systems in the aTAM which self-assemble to form colored input patterns. DNA tiles can be equipped with proteins [31] and nanoparticles such as gold (`Au`) [33]. Assemblies of normal tiles as well as tiles thus modified can be considered a *colored pattern*, as a periodic placement of `Au` nanoparticles on a 2D nanogrid [33] can be considered a 2-colored (i.e., binary) rectangular pattern on which the two colors specify the presence/absence of an `Au` nanoparticle at the position. Various designs of pattern assemblers have been proposed theoretically and experimentally, see, e.g., [4, 6, 22, 33]. The input for this problem is a rectangular pattern consisting of k colors, and the output is a tile set in the aTAM which self-assembles the pattern. Essentially, each type of tile is assigned a "color", and the goal is to design a system consisting of the minimal number of tile types such that they deterministically self-assemble to form a rectangular assembly in which each tile is assigned the same color as the corresponding location in the pattern. This problem was introduced in [17], and has since then

been extensively studied [7,9,11,12,26]. The interest is both theoretical, to determine the computational complexity of designing efficient tile assembly systems, and practical, as the goal of self-assembling patterned substrates onto which a potentially wide variety of molecular components could be attached is a major experimental goal. Known as k-Pats, where k is the number of unique colors in the input pattern, previous work has steadily decreased the value of k for which k-Pats has been shown to be **NP**-hard, from unbounded [7] to 11 [12]. (Additionally, in a variant of k-Pats where the number of tile types of certain colors is restricted, is has been proven to be **NP**-hard for 3 colors [14].) However, the foundational conjecture has been that for $k = 2$, i.e. 2-Pats, the problem is also **NP**-hard. This is our main result, which is thus the terminus of this line of research and a fundamental result in algorithmic self-assembly.

Computer-assisted proofs. In one of its parts (a portion of one direction of the **NP** reduction), our proof of the 2-Pats conjecture requires the solution of a massive combinatorial problem, meaning that one of the lemmas upon which it relies needs a massive exploration of more than $6 \cdot 10^{13}$ cases via a computer program. While this is not a traditional component of mathematical proofs, and may not provide the same level of insight into *why* something is true that a standard proof may, modern hardware and software have now given us the tools to attack combinatorially formidable problems whose proofs, if not augmented by computer programs, would often be impossible or as lacking in their ability to elucidate the reasons for their truth due to explosive case analyses as verification by brute force analysis of a computer program. Indeed, computer science has at the same time introduced combinatorial arguments indicating that most theorems do not have simple proofs, and possible ways to produce *certain facts* anyway, by heavy algorithmic processes. Moreover, the "natural proofs" line of research [1, 5, 20, 23] suggests that understanding "why" complexity classes are separated may be out of reach, and that therefore, the study of these kinds of proofs, and methods to ensure their correctness, are a fundamental direction in computer science today. Asserting the correctness of biological and chemical programs is also an important problem, where "*why*" questions are really not as important as the "*whether*" ones, for instance for therapeutic applications. Computationally intensive proofs are therefore likely to become common in these areas of science.

Historically, Appel and Haken [2,3] were the first to prove a result – the four color theorem – with this kind of method, in 1976. This proof was later simplified in [21]. Since then, important problems in various fields have been solved (fully or partially) with the assistance of computers: the discovery of Mersenne primes [28], the **NP**-hardness of minimum-weight triangulation [18], a special case of Erdős' discrepancy conjecture [15], and the ternary Goldbach conjecture [10], among others. (Over the years, exhaustive exploration and massively parallel programs have also been commonly used in physics, or in combinatorial problems such as solving the Rubik's cube.) However, none of these programs was proven formally, and confidence in the validity of these results thus relies on our trust in the programmers.

Proofs of computer programs. The first rigorous proof of a massive software exploration was for the four colors theorem, recently done in the Coq proof assistant by Gonthier et al. [8]. The order of magnitude of their proof is close to the limits of Coq, and is not comparable with our result, which needs a massively parallel exploration requiring about one CPU-year on very modern, high-end machines (as a sum total over several hundred distributed cores) to complete and verify the correctness of the lemma.

A large parallel cluster was hence employed, which poses a number of new challenges. Indeed, in a sequential program, we often implicitly use the fact that function calls return the output of their computations, which becomes more complicated when using several computers: without using unrealistic hypotheses on the correction of the network and of operating systems, return values could potentially be lost, duplicated or corrupted. Since our program ran for a long time, we cannot make such strong hypotheses, which is why we need to assert the authenticity of messages received by the server by using cryptographic signatures.

Another feature of our proof is the use of a *functional programming language*, OCaml. The conciseness of its code and the proximity of its syntax to mathematical proofs brought us a rigorous proof of the correctness of our program.

The whole framework for carrying out the programmatic part of our proof is reusable for the same kind of tasks in the future.

1.1 Main result

Our result solves an open problem in the field of DNA self-assembly, the so-called *binary pattern tile set synthesis* (2-Pats) problem [17, 26], stated first in 2008. In the general k-Pats for $k \geq 2$, given a placement of k different kinds of nanoparticles, represented in the model as a k-colored rectangular pattern, we are asked to design an optimally small tileset and an L-shaped seed that self-assembles the pattern (see Fig. 1 for an example).

2-Pats has been conjectured to be **NP**-hard since 2008[5]. In [26], Seki proved for the first time the **NP**-hardness of 60-Pats, whose input pattern is allowed to have 60 colors, and the result has since been strengthened to that of 29-Pats [11], and further to 11-Pats [12].

Our main theorem closes this line of research by lowering the number of colors allowed for input patterns to only two. We state the main result of this paper here, although some terms may not formally be defined yet:

Theorem 1. *The* 2-Pats *optimization problem of finding, given a* 2 *colored rectangular pattern P, the minimal colored tileset (together with an L-shaped seed) that produces a single terminal assembly where the color arrangement is exactly the same as in P, is* **NP***-hard.*

The main idea of our proof is similar to the strategies adopted by [11, 12, 26]. We embed the computation of a verifier of solutions for an **NP**-complete problem

[5] This problem was claimed to be **NP**-hard in a subsequent paper by the authors of [17] but what they proved was the **NP**-hardness of a different problem (see [27]).

(in our case, a variant of SAT, which we call M-SAT) in an assembly, which is relatively straightforward in Winfree's aTAM. One can indeed engineer a tile assembly system (TAS) in this model, with colored tiles, implementing a verifier of solutions of the variant of SAT, in which a formula F and a variable assignment $\phi \in \{0,1\}^n$ are encoded in the seed assembly, and a tile of a special color appears after some time if and only if $F(\phi) = 1$. In our actual proof, reported in Sect. 3, we design a set T of 13 tile types and a reduction of a given instance ϕ of M-SAT to a rectangular pattern P_F such that

Property 1. A TAS using tile types in T self-assembles P_F iff F is satisfiable.

Property 2. Any TAS of at most 13 tile types that self-assembles P_F is isomorphic to T.

Therefore, F is solvable if and only if P_F can be self-assembled using at most 13 tile types. In previous works [11,12,26], significant portions of the proofs were dedicated to ensuring their analog of Property 2, and many colors were "wasted" to make the property "manually" checkable (for reference, 33 out of 60 colors just served this purpose for the proof of **NP**-hardness of 60-PATS [26] and 2 out of 11 did that for 11-PATS [12]). Cutting this "waste" causes a combinatorial explosion of cases to test and motivates us to use a computer program to do the verification instead.

Apart from the verification of Property 2 (in Lemma 1), the rest of our proof can be verified as done in traditional mathematical proofs; our proof is in Sect. 3. The verification of Property 2 is done by an algorithm (omitted due to space constraints but described, along with all other proof details, in [13]), which, given a pattern and an integer n, searches for all possible sets of n tile types that self-assemble the pattern. The correctness of the algorithm is proven, and both the (unproven, efficient) C++ code, and the (slower but formally proven) OCaml code implementing the algorithm are freely available online[6]. Both versions were implemented independently and neither is the conversion of the code of the other implementation. The full statistics of the runs are available on demand, and summarized by the Parry user interface: `http://pats.lif.univ-mrs.fr`.

2 Preliminaries

Let $\mathbb{N}$ be the set of nonnegative integers, and for $n \in \mathbb{N}$, let $[n] = \{0, 1, 2, \ldots, n-1\}$. For $k \geq 1$, a *k-colored pattern* is a partial function from $\mathbb{N}^2$ to the set of (color) indices $[k]$, and a *k-colored rectangular pattern* (of width w and height h) is a pattern whose domain is $[w] \times [h]$.

Let Σ be a glue alphabet. A *(colored) tile type* t is a tuple $(g_{\mathtt{N}}, g_{\mathtt{W}}, g_{\mathtt{S}}, g_{\mathtt{E}}, c)$, where $g_{\mathtt{N}}, g_{\mathtt{W}}, g_{\mathtt{S}}, g_{\mathtt{E}} \in \Sigma$ represent the respective north, west, south, and east glue of t, and $c \in \mathbb{N}$ is a color (index) of t. For instance, the right black tile type in Fig. 1 (Left) is (1, 1, 0, 0, black). We refer to $g_{\mathtt{N}}, g_{\mathtt{W}}, g_{\mathtt{S}}, g_{\mathtt{E}}$ as $t(\mathtt{N}), t(\mathtt{W}), t(\mathtt{S}), t(\mathtt{E})$,

[6] `http://self-assembly.net/wiki/index.php?title=2PATS-tileset-search` (C++ version) and `http://self-assembly.net/wiki/index.php?title=2PATS-search-ocaml` (OCaml version)

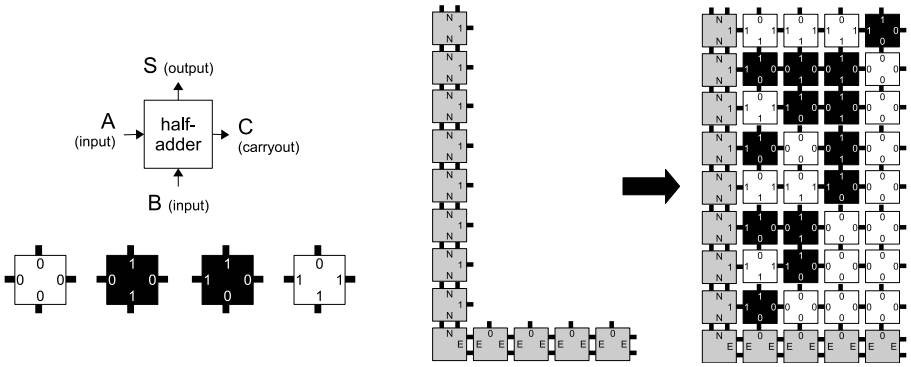

Fig. 1. (Left) Four tile types implement the half-adder with two inputs A, B from the west and south, the output S to the north, and the carryout C to the east. (Right) Copies of the half-adder tiles turn the L-shape seed into the binary counter pattern.

respectively, and by $c(t)$ we denote the color of t. For a set T of tile types, an *assembly* α over T is a partial function from $\mathbb{N}^2$ to T. Its pattern, denoted by $P(\alpha)$, is such that $\mathrm{dom}(P(\alpha)) = \mathrm{dom}(\alpha)$ and $P(\alpha)(x, y) = c(\alpha(x, y))$ for any $(x, y) \in \mathrm{dom}(\alpha)$. Given another assembly β, we say α is a *subassembly* of β if $\mathrm{dom}(\alpha) \subseteq \mathrm{dom}(\beta)$ and, for any $(x, y) \in \mathrm{dom}(\alpha)$, $\beta(x, y) = \alpha(x, y)$.

A *rectilinear tile assembly system* (RTAS) is a pair $\mathcal{T} = (T, \sigma_L)$ of a set T of tile types and an L-shape seed σ_L, which is an assembly over another set of tile types disjoint from T such that $\mathrm{dom}(\sigma)_L = \{(-1, -1)\} \cup ([w] \times \{-1\}) \cup (\{-1\} \times [h])$ for some $w, h \in \mathbb{N}$. The *size* of $\mathcal{T}$ is measured by the number of tile types employed, that is, $|T|$. According to the following general rule that all RTASs obey, it tiles the first quadrant delimited by the seed:

RTAS tiling rule: Tile $t \in T$ can attach to an assembly α at position (x, y) if

1. $\alpha(x, y)$ is undefined,
2. both $\alpha(x-1, y)$ and $\alpha(x, y-1)$ are defined,
3. $t(\mathtt{W}) = \alpha(x-1, y)[\mathtt{E}]$ and $t(\mathtt{S}) = \alpha(x, y-1)[\mathtt{N}]$.

The attachment results in a larger assembly β whose domain is $\mathrm{dom}(\alpha) \cup \{(x, y)\}$ such that for any $(x', y') \in \mathrm{dom}(\alpha)$, $\beta(x', y') = \alpha(x, y)$, and $\beta(x, y) = t$. When this attachment takes place in the RTAS $\mathcal{T}$, we write $\alpha \to_1^{\mathcal{T}} \beta$. Informally speaking, the tile t can attach to the assembly α at (x, y) if on α, both $(x-1, y)$ and $(x, y-1)$ are tiled while (x, y) is not yet, and the west and south glues of t match the east glue of the tile at $(x-1, y)$ and the north glue of the tile at $(x, y-1)$, respectively. This implies that, at the outset, (0, 0) is the sole position where a tile may attach.

Example 1. See Fig. 1 for an RTAS with 4 tile types that self-assembles the binary counter pattern. To its L-shape seed shown there, a black tile of type (1, 1, 0, 0, black) can attach at (0, 0), while no tile of other types can due to glue mismatches. The attachment makes the two positions (0, 1) and (1,

0) attachable. Tiling in RTASs thus proceeds from south-west to north-east *rectilinearly* until no attachable position is left.

The set $\mathcal{A}[\mathcal{T}]$ of *producible* assemblies by $\mathcal{T}$ is defined recursively as follows: (1) $\sigma_L \in \mathcal{A}[\mathcal{T}]$, and (2) for $\alpha \in \mathcal{A}[\mathcal{T}]$, if $\alpha \to_1^{\mathcal{T}} \beta$, then $\beta \in \mathcal{A}[\mathcal{T}]$. A producible assembly $\alpha \in \mathcal{A}[\mathcal{T}]$ is called *terminal* if there is no assembly β such that $\alpha \to_1^{\mathcal{T}} \beta$. The set of terminal assemblies is denoted by $\mathcal{A}_\square[\mathcal{T}]$. Note that the domain of any producible assembly is a subset of $(\{-1\} \cup [w]) \times (\{-1\} \cup [h])$, starting from the seed σ_L whose domain is $\{(-1,-1)\} \cup ([w] \times \{-1\}) \cup (\{-1\} \times [h])$.

A tile set T is *directed* if for any distinct tile types $t_1, t_2 \in T$, $t_1(\mathtt{W}) \neq t_2(\mathtt{W})$ or $t_1(\mathtt{S}) \neq t_2(\mathtt{S})$ holds. An RTAS $\mathcal{T} = (T, \sigma_L)$ is *directed* if its tile set T is directed (the directedness of RTAS was originally defined in a different but equivalent way). It is clear from the RTAS tiling rule that if $\mathcal{T}$ is directed, then it has exactly one terminal assembly, which we call γ. Let γ' be the subassembly of the terminal assembly such that $\mathrm{dom}(\gamma') \subseteq \mathbb{N}^2$, that is, the tiles on γ' did not originate from the seed σ_L but were tiled by the RTAS. Then we say that $\mathcal{T}$ *uniquely self-assembles the pattern* $P(\gamma')$.

The *pattern self-assembly tile set synthesis* (Pats), proposed by Ma and Lombardi [17], aims at computing the minimum size directed RTAS that uniquely self-assembles a given rectangular pattern. The solution to Pats is required to be directed here, but not originally. However, in [9], it was proved that among all the RTASs that uniquely self-assemble the pattern, the minimum one is directed.

To study the algorithmic complexity of this problem on "real size" particle placement problems, a first restriction that can be placed is on the number of colors allowed for the input patterns, thereby defining the k-Pats problem:

k-colored Pats (k-Pats)
Given: a k-colored pattern P
Find: a smallest directed RTAS that uniquely self-assembles P

The **NP**-hardness of this optimization problem follows from that of its decision variant, which decides, given also an integer m, if such an RTAS is implementable using at most m tile types or not. In the rest of this paper, we use the terminology k-Pats to refer to this decision problem, unless otherwise noted.

3 2-Pats is NP-hard

We will prove that Pats is **NP**-hard for binary patterns (2-colored patterns). Our proof is a polynomial-time reduction from *monotone satisfiability with few true variables* (M-Sat) to (the decision variant of) 2-Pats. In M-Sat, we consider a number k and a boolean formula F in conjunctive normal form *without negations* and ask whether or not F can be satisfied by only allowing k variables to be true; the **NP**-hardness of M-Sat is proven in [13]. Given an instance of M-Sat we reduce it to a binary pattern $P_{k,F}$ such that a directed RTAS with 13 or less tile types self-assembles $P_{k,F}$ if and only if the answer to the M-Sat instance is yes, i.e., F can be satisfied with exactly k true variables.

We design the pattern $P_{k,F}$ so as to incorporate, as a subpattern, a gadget pattern G shown in Fig. 3. As formally stated in Lemma 1 below, the gadget pattern G has the property that among all the tilesets of size at most 13, exactly

one (up to isomorphism) can be employed in a directed RTAS to assemble G, and thus any pattern with G as a subpattern has the same property. Let T be this tileset, shown in Fig. 2. Lemma 1 is verified by an exhaustive search by a computer program whose proof of correctness is omitted due to space constraints (all the other parts of our proof of Theorem 1 are manually checkable).

Lemma 1. *If a directed RTAS whose tileset consists of 13 or less tile types self-assembles the gadget pattern G in Fig. 3, then its tileset is isomorphic to T.*

Due to this property of G, in order to decide the reduced 2-Pats instance $(P_{k,F}, 13)$, it suffices to decide whether a directed RTAS with tileset T self-assembles $P_{k,F}$ or not. This is equivalent to finding an L-shape seed σ_L such that the directed RTAS (T, σ_L) self-assembles $P_{k,F}$. A subtlety of our proof comes from the fact that neither F nor k influence the optimal number of tile types that can assemble $P_{k,F}$ if F is satisfiable.

The tileset T works as an M-Sat verifier, when being used by a directed RTAS. It contains 11 white tile types and 2 black ones.

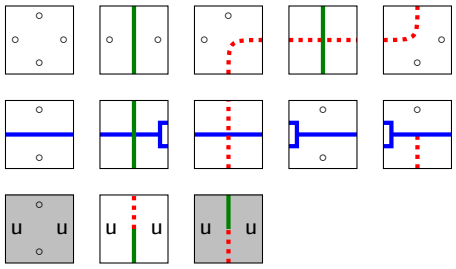

Fig. 2. The tileset T, where the background depicts the color of each tile type and the labels and signals depict the glues (i.e. the glue on a side is equivalent to the label or signal on that side, and the colored signals don't actually appear on the tiles). We refer to the tile types with a gray background as the black tile types. (For better visibility in printouts, the red signals are dotted; blue and green signals can easily be distinguished as blue signals run only horizontally while green signals run only vertically.)

Let us first explain how the RTAS verifies a given M-Sat instance and present its verification visually on its resulting assembly. It does so by "propagating signals" of three kinds (red, green, and blue) via glues from bottom-left to top-right (as the tiles attach in that ordering) and letting them interact with each other. An important fact, that justifies the "signal" vocabulary, is that these signals never fork, i.e. in all the tile types of T, if a signal of type s appears on a west or south glue of a tile $t \in T$, it appears on at most one other side, which is either the east or the north side of t.

We interpret the glues in tile set T as follows. Ten of the white tile types (first and second rows in Fig. 2) simulate three types of signals and their interactions. Recall that in the RTAS, growth begins from an L-shaped seed and proceeds strictly up and to the right. Therefore, as tiles are added by matching the signals

on their bottom and/or left sides, we can think of them as passing the signals to their output (i.e. top and/or right) sides, as indicated by the colored lines showing the signals across each tile. These signals can necessarily, due to the ordering of growth of the assembly and the definitions of the tile types, move only up, right, up and right, or terminate. The signals propagate as follows:

1. blue signals propagate left to right,
2. green signals propagate from bottom to top, and
3. red signals propagate diagonally, bottom left to top right in a wavelike line.

When any two of the signals meet, they simply cross over each other, while the red signal is displaced upwards or rightwards when crossing a blue or green signal, respectively. However when a blue signal crosses a green signal immediately before encountering a red signal, the red signal is destroyed. In order to recognize this configuration, the blue signal is *tagged* when it crosses a green signal; in Fig. 2, the tagging is displayed by the fork in the blue signal. Let us stress that the signals are encoded in the glues of the tiles, and not (at least directly) in their colors.

The other three tile types, all with horizontal glues of type u, are used to start rows called "uncovering rows". A major challenge of the reduction is that we cannot force our signals to appear directly in the pattern, because we have only two colors. Instead, we start these "uncovering rows", and make the signals appear in the pattern by their effects on these rows. More specifically, rows with horizontal u glues are always used in pairs:

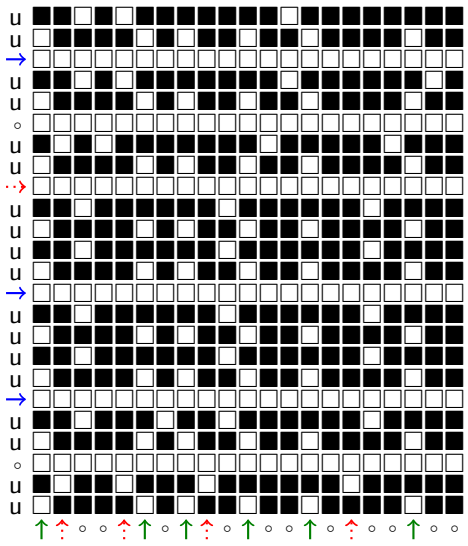

Fig. 3. Binary gadget pattern G, which can only be self-assembled by ≤ 13 tile types by using the tile set T (or one isomorphic to it). To self-assemble G using T one has to use the glues on the L-shaped seed as indicated on the bottom and left. For performance purposes, the bottom row in the pattern was not included in the computerized search; however, because uncovering rows appear in pairs, we add the bottom row here for clarity.

1. one black tile is above another when no signal is received from below,
2. a white tile is below a black when a green signal is received,
3. a black tile is below a white when a red signal is received.

Note that by the definition of the tile set, it's impossible for both signals to be received in the same column. Moreover, blue signals are not "uncovered", since they never reach these rows. Green (resp. red) signals switch to red (resp. green)

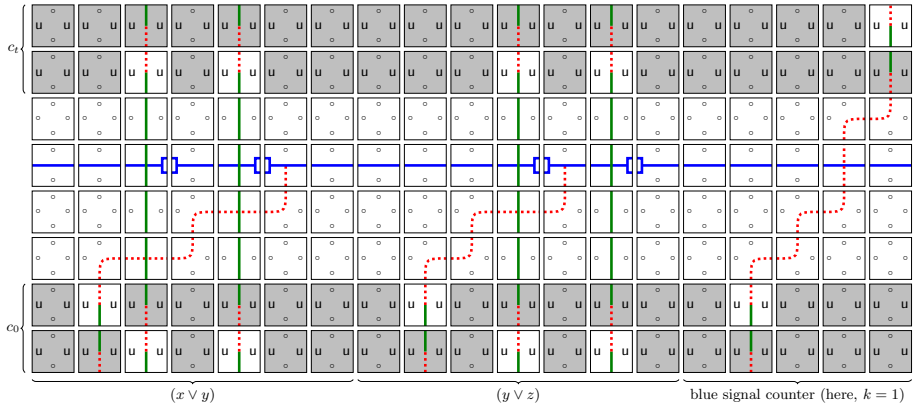

Fig. 4. Subpattern of $P_{k,F}$ for the formula $F = (x \vee y) \wedge (y \vee z)$ with $k = 1$. The position of the blue signal represents the satisfying variable assignment $\phi(y) = 1$. Only the subpattern which encodes F is shown, the gadget pattern and the areas needed to initialize the gadget pattern are omitted here. The different subpatterns shown here are explained in the proof of Theorem 1.

in the first uncover row, but they switch back to their original state in the second uncover row. This allows the enforcement of the encoding of the three possible values of signals (no signal, green signal, or red signal) with exactly two colors. In our construction, uncovering rows always appear in pairs in order to ensure that the original state of each signal is reestablished after passing through a pair of uncovering rows. In our reduction, we'll use this property to "initialize" a gadget area, above the M-SAT verifier in the pattern, forcing use of tileset T.

An example subassembly which represents the formula $F = (x \vee y) \wedge (y \vee z)$ (without the gadget part) is shown in Fig. 4. A more extensive example of a tile assembly with tileset T can be found in Appendix C of [13] which shows the subpattern of $P_{k,F}$ used for initializing and including the gadget pattern G.

The intuition of the construction of the pattern $P_{k,F}$ and its assembly is that on the vertical arm of the L-shaped seed (i.e., the east border of the upward arm of the seed), variables $x_0, x_1, \ldots, x_{n-1}$ are encoded successively, by the presence of a blue signal if the corresponding variable is set to 1, and a tile with no signal else. Each clause of F is, on the other hand, encoded on the horizontal arm of the L-shape seed as a red signal followed by precisely spaced green signals (intervals between these signals specify which variables are in the clause).

For instance, in Fig. 5, the red signal on the left makes it through (i.e., it is not stopped by a tagged blue signal) and appears in the top uncovering rows, while the one on the right does not. The reason for the red signal being stopped on the right, is that the horizontal spacing between the red and the green signal is "compatible" with the vertical location of blue signal. This compatibility of blue, green, and red signals corresponds to a variable in a clause, represented by the red and green signal, which is set true in the variable assignment, represented by the blue signal. More generally, the absence of red signals on the top uncovering rows c_t means that all the clauses have been satisfied, and the presence of a red

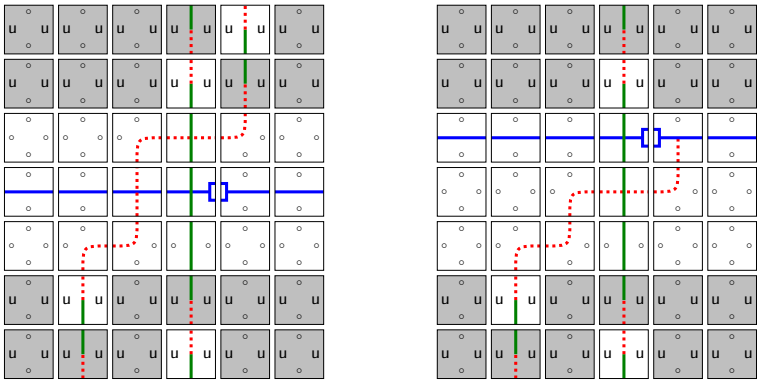

Fig. 5. Example interactions of the signals in the tile set T with uncovering of the configurations: on the left side the red signal can pass through the pattern while the red signal on the right side is destroyed. Note that the position of the blue signal, which is hidden in the horizontal glues, controls whether or not the red signal is destroyed.

signal means that at least one clause could not be satisfied by the assignment. Additionally, note that the positions of blue signals, encoding which variables are set to true in a variable assignment of the M-SAT instance, are not encoded in the pattern, since they travel only through white tiles.

Finally, the part of Fig. 4 which is labeled the "blue signal counter" specifies the number k of true variables in a satisfying variable assignment for F. Note that by the horizontal movement of the red signal from rows c_0 to rows c_t determines the number of blue signals that appear in the white rows in between c_0 and c_t; indeed, the red signal travels one tile to the right in a row without signal, but remains horizontally stationary when passing a row with a blue signal.

References

1. Allender, E., Koucký, M.: Amplifying lower bounds by means of self-reducibility. J. ACM 57(3), 14:1–14:36 (2010)
2. Appel, K., Haken, W.: Every planar map is four colorable. Part I. discharging. Illinois J. Math. 21, 429–490 (1977)
3. Appel, K., Haken, W.: Every planar map is four colorable. Part II. reducibility. Illinois J. Math. 21, 491–567 (1977)
4. Barish, R., Rothemund, P.W.K., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano. Lett. 5(12), 2586–2592 (2005)
5. Chow, T.Y.: Almost-natural proofs. J. Comput. Syst. Sci. 77(4), 728–737 (2011)
6. Cook, M., Rothemund, P.W.K., Winfree, E.: Self-assembled circuit patterns. In: Proc. DNA 9. pp. 91–107. LNCS 2943, Springer (2004)
7. Czeizler, E., Popa, A.: Synthesizing minimal tile sets for complex patterns in the framework of patterned DNA self-assembly. Theor. Comput. Sci. 499, 23–37 (2013)
8. Gonthier, G.: Formal proof – the four-color theorem. Not. Am. Math. Soc. 55(11), 1382–1393 (2008)
9. Göös, M., Lempiäinen, T., Czeizler, E., Orponen, P.: Search methods for tile sets in patterned DNA self-assembly. J. Comput. Syst. Sci. 80, 297–319 (2014)

10. Helfgott, H.A.: The ternary Goldbach conjecture is true (2013)
11. Johnsen, A., Kao, M.Y., Seki, S.: Computing minimum tile sets to self-assemble color patterns. In: Proc. ISAAC 2013. pp. 699–710. LNCS 8283, Springer (2013)
12. Johnsen, A., Kao, M.Y., Seki, S.: A manually-checkable proof for the NP-hardness of 11-colored patterned self-assembly of tile set synthesis. arXiv:1409.1619 (2014)
13. Kari, L., Kopecki, S., Meunier, P.E., Patitz, M.J., Seki, S.: Binary pattern tile set synthesis is NP-hard. arXiv:1404.0967 (2014)
14. Kari, L., Kopecki, S., Seki, S.: 3-color bounded patterned self-assembly. Nat. Comp. (2014), in press
15. Konev, B., Lisitsa, A.: A SAT attack on the Erdös discrepancy conjecture. arXiv: 1402.2184 (2014)
16. Lund, K., Manzo, A.T., Dabby, N., Micholotti, N., Johnson-Buck, A., Nangreave, J., Taylor, S., Pei, R., Stojanovic, M.N., Walter, N.G., Winfree, E., Yan, H.: Molecular robots guided by prescriptive landscapes. Nature 465, 206–210 (2010)
17. Ma, X., Lombardi, F.: Synthesis of tile sets for DNA self-assembly. IEEE T. Comput. Aid. D. 27(5), 963–967 (2008)
18. Mulzer, W., Rote, G.: Minimum-weight triangulation is NP-hard. J. ACM 55(2), Article No. 11 (2008)
19. Qian, L., Winfree, E.: Scaling up digital circuit computation with DNA strand displacement cascades. Science 332(6034), 1196 (2011)
20. Razborov, A.A., Rudich, S.: Natural proofs. In: Proc. STOC '94. pp. 204–213. ACM, New York, NY, USA (1994)
21. Robertson, N., Sanders, D.P., Seymour, P., Thomas, R.: A new proof of the four-colour theorem. Electron. Res. Announc. AMS. 2(1), 17–25 (1996)
22. Rothemund, P.W., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biol. 2(12), 2041–2053 (2004)
23. Rudich, S.: Super-bits, demi-bits, and NP/qpoly-natural proofs. J. Comput. Syst. Sci. 55, 204–213 (1997)
24. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)
25. Seeman, N.C.: Nucleic-acid junctions and lattices. J. Theor. Biol. 99, 237–247 (1982)
26. Seki, S.: Combinatorial optimization in pattern assembly (extended abstract). In: Proc. UCNC 2013. pp. 220–231. LNCS 7956, Springer (2013)
27. Sterling, A.: https://nanoexplanations.wordpress.com/2011/08/13/dna-self-assembly-of-multicolored-rectangles/
28. Tuckerman, B.: The 24th Mersenne prime. Proc. Nat. Acad. Sci. USA 68, 2319–2320 (1971)
29. Wang, H.: Proving theorems by pattern recognition – II. AT&T Tech. J. XL(1), 1–41 (1961)
30. Winfree, E.: Algorithmic Self-Assembly of DNA. Ph.D. thesis, California Institute of Technology (June 1998)
31. Yan, H., Park, S.H., Finkelson, G., Reif, J.H., LaBean, T.H.: DNA-templated self-assembly of protein arrays and highly conductive nanowires. Science 301, 1882–1884 (2003)
32. Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: A DNA-fuelled molecular machine made of DNA. Nature 406(6796), 605–608 (2000)
33. Zhang, J., Liu, Y., Ke, Y., Yan, H.: Periodic square-like gold nanoparticle arrays templated by self-assembled 2D DNA nanogrids on a surface. Nano Letters 6(2), 248–251 (2006)