



Solving satisfiability in the tile assembly model with a constant-size tileset

Yuriy Brun

Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA

ARTICLE INFO

Article history:

Received 7 December 2007

Available online 13 September 2008

Keywords:

Self-assembly

NP-complete

Tile assembly model

Crystal-growth

Molecular computation

Natural computation

Distributed computing

Parallel computing

Satisfiability

3-SAT

ABSTRACT

Biological systems are far more complex and robust than systems we can engineer today. One way to increase the complexity and robustness of our engineered systems is to study how biological systems function. The tile assembly model is a highly distributed parallel model of nature's self-assembly. Previously, I defined deterministic and nondeterministic computation in the tile assembly model and showed how to add, multiply, factor, and solve *SubsetSum*. Here, I present a system that decides satisfiability, a well-known NP-complete problem. The computation is nondeterministic and each parallel assembly executes in time linear in the input. The system requires only a constant number of different tile types: 64, an improvement over previously best known system that uses $\Theta(n^2)$ tile types. I describe mechanisms for finding the successful solutions among the many parallel assemblies and explore bounds on the probability of such a nondeterministic system succeeding and prove that probability can be made arbitrarily close to 1.

© 2008 Elsevier Inc. All rights reserved.

1. Introduction

Self-assembly is a process that is ubiquitous in nature. Systems form on all scales via self-assembly: atoms self-assemble to form molecules, molecules to form complexes, and stars and planets to form galaxies. One manifestation of self-assembly is crystal growth: molecules self-assembling to form crystals. Crystal growth is an interesting area of research for computer scientists because it has been shown that, in theory, under careful control, crystals can compute [1]. The field of DNA computation demonstrated that DNA can be used to compute [2], solving NP-complete problems such as the satisfiability problem [3,4]. This idea of using molecules to compute nondeterministically is the driving motivation behind my work.

NP-complete problems are integral to our everyday lives and yet we know of no efficient algorithms to solve them. Many problems in the realms of resource allocation, scheduling, and protein folding have been shown to be NP-hard, and the ability to solve these problems quickly is highly desirable. Molecular computing models are a possible powerful route toward feasible algorithms for solving NP-complete problems quickly because of the high information density molecules allow.

Winfrey showed that DNA computation is Turing-universal [5]. While DNA computation suffers from relatively high error rates, the study of self-assembly shows how to utilize redundancy to design systems with built-in error correction [6–10]. Researchers have used DNA to assemble crystals with patterns of binary counters [11] and Sierpinski triangles [12], but while those crystals are deterministic, generating nondeterministic crystals may hold the power to solving complex problems quickly.

E-mail address: ybrun@usc.edu.

Two important questions about self-assembling systems that create shapes or compute functions are: “what is a minimal tileset that can accomplish this goal?” and “what is the minimum assembly time for this system?” Here, I study systems that solve NP-complete problems and ask these questions, as well as another that is important to nondeterministic computation: “what is the probability of assembling the crystal that encodes the solution?” Adleman has emphasized studying the number of steps it takes for an assembly to complete (assuming maximum parallelism) and the minimal number of tiles necessary to assemble a shape [13]. He answered these questions for n -long linear polymers [14]. Previously, I have extended these questions to apply to systems that compute functions, rather than assemble shapes, deterministically [15] and nondeterministically [16], and now I extend them to systems that decide sets and solve NP-complete problems.

Adleman proposed studying the complexity of tile systems that can uniquely produce $n \times n$ squares. A series of researchers [17–20] proceeded to answer the questions: “what is a minimal tileset that can assemble such shapes?” and “what is the assembly time for these systems?” They showed that the minimal tileset that assembles $n \times n$ squares is of size $O\left(\frac{\log n}{\log \log n}\right)$ and the optimal assembly time is $\Theta(n)$ [19].

Soloveichik et al. [21] studied assembling all decidable shapes in the tile assembly model and found that the minimal set of tiles necessary to uniquely assemble a shape is directly related to the Kolmogorov complexity of that shape. Interestingly, they found that for the result to hold, scale must not be a factor. That is, the minimal set of tiles they find builds a given shape (e.g., square, a particular approximation of the map of the world, etc.) on some scale, but not on all scales. Thus they showed that smaller versions of the same shape might require larger sets of tiles to assemble.

I proposed and studied systems that compute the sums and products of two numbers using the tile assembly model [15]. I found that in the tile assembly model, adding and multiplying can be done using $\Theta(1)$ tiles (as few as 8 tiles for addition and as few as 28 tiles for multiplication), and that both computations can be carried out in time linear in the input size. I then showed that systems can be combined to create systems with more complex behavior, and designed a system that factors numbers [16]. I have also shown a system that solves the NP-complete problem *SubsetSum* using 49 distinct computational tile types [22]. One interesting aspect of solving satisfiability in the tile assembly model using a constant-size tileset is that writing the input itself is difficult because each Boolean formula has some n distinct variables, and one must encode all variables using a constant number of tiles. Previous attempts have used distinct tiles for each variable. The mechanism I design here can be applied to solving other problems that require an encoding of variables in its input, as well as graph problems that require the encoding of vertices and edges.

Early attempts at nondeterministic computation include a proposal by Lagoudakis et al. [23] to solve the satisfiability problem. They informally define two systems that nondeterministically compute whether or not an n -variable boolean formula is satisfiable using $\Theta(n^4)$ and $\Theta(n^2)$ distinct tiles, respectively. The former system encodes each clause-variable pair as a separate tile, and the latter system encodes each pair of literals as a separate tile. In a DNA implementation of even the smaller system, to solve a 50-variable satisfiability problem, one would need on the order of 2500 different DNA complexes, while current DNA self-assembly systems have on the order of 10 different complexes. In contrast, the system I present in this paper for solving an NP-complete problem uses $\Theta(1)$ distinct tiles and assembles in time linear in the input.

While the constructions in this paper are in some ways analogous to traditional computer programs, and their running times are polynomially related to the running times of Turing machines and nondeterministic Turing machines, Baryshnikov et al. [24] began the study of fundamental limits on the time required for a self-assembly system to compute functions. They consider models of molecular self-assembly and apply Markov models to show lower limits on assembly times.

Researchers have also studied variations of the traditional tile assembly model. Aggarwal et al. [25] and Kao et al. [26] have shown that changing the temperature of assembly from a constant throughout the assembly process to a discrete function reduces the minimal tileset that can build an $n \times n$ square to a size $\Theta(1)$ tileset.

Barish et al. [11] have demonstrated DNA implementations of tile systems, one that copies an input and another that counts in binary. Similarly, Rothmund et al. [12] have demonstrated a DNA implementation of a tile system that computes the *xor* function, resulting in a Sierpinski triangle. These systems grow crystals using double-crossover complexes [27] as tiles. The theoretical underpinnings of these systems are closely related to the work presented here because these systems compute functions.

Rothmund has demonstrated what is currently the state-of-the-art of DNA nanostructure design and implementation with DNA origami, a concept of folding a single long scaffold strand into an arbitrary shape by using small helper strands [28–30]. Similar concepts may be the key to three-dimensional self-assembly, more powerful error-correction techniques, and self-assembly using biological molecules.

Cook et al. [31] have explored using the tile assembly model to implement arbitrary circuits. Their model allows for tiles that contain gates, counters, and even more complex logic components, as opposed to the simple static tiles used in the traditional tile assembly model and in this paper. While they speculate that the tile assembly model logic may be used to assemble logic components attached to DNA, my assemblies require no additional logic components and encode the computation themselves. It is likely that their approach will require fewer tile types and perhaps assemble faster, but at the disadvantage of having to not only assemble crystals but also attach components to those crystals and create connections among those components. Nevertheless, Rothmund’s work with using DNA as a scaffold may be useful in attaching and assembling such components [30].

Some experimental work [2,4] has shown that it is possible to work with an exponential number of components and to solve NP-complete problems. I explore the possibility of nondeterministic computation using the tile assembly model and prove bounds on the probability of successful computation. The probability of successfully solving an instance of the

SAT problem can be made arbitrarily close to 1 by increasing the number of self-assembling components and seeds in the computation.

The rest of this paper is structured as follows: Section 1.1 will describe in detail the tile assembly model, Section 2 will discuss what it means for a tile assembly model system to compute and to decide sets, Section 3 will introduce, define, and prove the correctness of two tile systems that decide satisfiability, a well-known NP-complete problem, and Section 4 will summarize the contributions of this work.

1.1. Tile assembly model

The tile assembly model [1,17,32] is a formal model of crystal growth. It was designed to model self-assembly of molecules such as DNA. It is an extension of a model proposed by Wang [33]. The model was fully defined by Rothmund and Winfree [17], and the definitions here are similar to those, and identical to the ones in [15,16,22], but I restate them here for completeness and to assist the reader. Intuitively, the model has *tiles* or squares that stick or do not stick together based on various binding domains on their four sides. Each tile has a binding domain on its north, east, south, and west side, and may stick to another tile when the binding domains on the abutting sides of those tiles match and the total strength of all the binding domains on that tile exceeds the current temperature. The four binding domains define the type of the tile.

Formally, let Σ be a finite alphabet of binding domains such that $null \in \Sigma$. I will always assume $null \in \Sigma$ even when I do not specify so explicitly. A **tile** over a set of binding domains Σ is a 4-tuple $\langle \sigma_N, \sigma_E, \sigma_S, \sigma_W \rangle \in \Sigma^4$. A **position** is an element of \mathbb{Z}^2 . The set of directions $D = \{N, E, S, W\}$ is a set of 4 functions from positions to positions, i.e., \mathbb{Z}^2 to \mathbb{Z}^2 , such that for all positions (x, y) , $N(x, y) = (x, y + 1)$, $E(x, y) = (x + 1, y)$, $S(x, y) = (x, y - 1)$, $W(x, y) = (x - 1, y)$. The positions (x, y) and (x', y') are neighbors iff $\exists d \in D$ such that $d(x, y) = (x', y')$. For a tile t , for $d \in D$, I will refer to $bd_d(t)$ as the binding domain of tile t on d 's side. A special tile $empty = \langle null, null, null, null \rangle$ represents the absence of all other tiles.

A **strength function** $g: \Sigma \times \Sigma \rightarrow \mathbb{N}$, where g is commutative and $\forall \sigma \in \Sigma$ $g(null, \sigma) = 0$, denotes the strength of the binding domains. It is common to assume that $g(\sigma, \sigma') = 0 \Leftrightarrow \sigma \neq \sigma'$. This simplification of the model implies that the abutting binding domains of two tiles have to match to bind. For the remainder of this paper, I will use $g = 1$ to mean $\forall \sigma \neq null, g(\sigma, \sigma) = 1$ and $\forall \sigma' \neq \sigma, g(\sigma, \sigma') = 0$.

Let T be a set of tiles containing the empty tile. A **configuration** of T is a function $A: \mathbb{Z} \times \mathbb{Z} \rightarrow T$. I write $(x, y) \in A$ iff $A(x, y) \neq empty$. A is finite iff there is only a finite number of distinct positions $(x, y) \in A$.

Finally, a **tile system** \mathbb{S} is a triple $\langle T, g, \tau \rangle$, where T is a finite set of tiles containing $empty$, g is a strength function, and $\tau \in \mathbb{N}$ is the temperature.

If $\mathbb{S} = \langle T, g, \tau \rangle$ is a tile system and A is a configuration of some set of tiles $T' \subseteq \Sigma^4$ then a tile $t \in T$ can **attach** to A at position (x, y) and produce a new configuration A' iff:

- $(x, y) \notin A$, and
- $\sum_{d \in D} g(bd_d(t), bd_{d^{-1}}(A(d(x, y)))) \geq \tau$, and
- $\forall (u, v) \in \mathbb{Z}^2, (u, v) \neq (x, y) \Rightarrow A'(u, v) = A(u, v)$, and
- $A'(x, y) = t$.

That is, a tile can attach to a configuration only in empty positions and only if the total strength of the appropriate binding domains on the tiles in neighboring positions meets or exceeds the temperature τ . For example, if for all σ , $g(\sigma, \sigma) = 1$ and $\tau = 2$ then a tile t can attach only at positions with matching binding domains on the tiles in at least two adjacent positions.

Given a tile system $\mathbb{S} = \langle T, g, \tau \rangle$, a set of tiles Γ , and a **seed configuration** $S_0: \mathbb{Z}^2 \rightarrow \Gamma$, if the above conditions are satisfied, one may attach tiles of T to S_0 . Note that I allow the codomain of S_0 to be Γ , a set of tiles which may be different from T . Let $W_0 \subseteq \mathbb{Z}^2$ be the set of all positions where at least one tile from T can attach to S_0 . For all $w \in W_0$ let U_w be the set of all tiles that can attach to S_0 at w . Let \hat{S}_1 be the set of all configurations S_1 such that for all positions $p \in S_0$, $S_1(p) = S_0(p)$ and for all positions $w \in W_0$, $S_1(w) \in U_w$ and for all positions $p \notin S_0 \cup W_0$, $S_1(p) = empty$. For all $S_1 \in \hat{S}_1$, I say that \mathbb{S} **produces** S_1 on S_0 in one step. If A_0, A_1, \dots, A_n are configurations such that for all $i \in \{1, 2, \dots, n\}$, \mathbb{S} produces A_i on A_{i-1} in one step, then I say that \mathbb{S} produces A_n on A_0 in n steps. When the number of steps taken to produce a configuration is not important, I will simply say \mathbb{S} produces a configuration A on a configuration A' if there exists $k \in \mathbb{N}$ such that \mathbb{S} produces A on A' in k steps. If the only configuration produced by \mathbb{S} on A is A itself, then A is said to be a **final configuration**. If there is only one final configuration A produced by \mathbb{S} on S_0 , then \mathbb{S} is said to produce a **unique final configuration** on S_0 . Finally, if A is a final configuration produced by \mathbb{S} on S_0 and n is the least integer such that A is produced by \mathbb{S} on S_0 in n steps, then n is the **assembly time** of \mathbb{S} on S_0 to produce A .

Note that a system may produce a unique final configuration, even though there exist non-unique sequences of attachments that continue growing at infinitum. Theoretically, such constructions pose no problem, though they may present problems to certain implementations of tile systems. In particular, the infinite configurations might consume all the tiles available for construction. It is possible to limit the definition of a unique final configuration to exclude systems that produce infinite configurations; however, such a restriction seems somewhat arbitrary and would only be helpful for some implementations of the tile assembly model. I choose not to restrict my definitions here, though I note that the systems

presented in this paper do not suffer from this problem and produce no infinite configurations, and thus would satisfy the stricter definitions.

Winfree showed that the tile assembly model with $\tau = 2$ is Turing-universal [1] by showing that a tile system can simulate Wang tiles [33], which Robinson showed to be universal [34]. Adleman et al. [35] showed that the tile assembly model with $\tau = 1$ is Turing-universal.

2. Computation in the tile assembly model

In [15], I define what it means to deterministically compute functions in the tile assembly model. In some implementations of tile assembly, many assemblies happen in parallel. In fact, it is often almost impossible to create only a single assembly, and thus there is a parallelism that my previous definitions did not take advantage of. In [16], I extend the notion of computation in the tile assembly model to nondeterministic assemblies. For deterministic computation, I have defined a tile system to produce a unique final configuration on a seed if for all sequences of tile attachments, all possible final configurations are identical. In nondeterministic computation, different sequences of tile attachments attach different tiles in the same position. Intuitively, a system nondeterministically computes a function iff at least one of the possible sequences of tile attachments produces a final configuration which codes for the solution. Finally, in [22] I defined the notion of a tile system nondeterministically deciding a set.

Since a nondeterministic computation may have unsuccessful sequences of attachments, it is important to distinguish the successful ones. Further, in many implementations of the tile assembly model that would simulate all the nondeterministic executions at once, it is useful to be able to identify which executions succeeded and which failed in a way that allows selecting only the successful ones. For some problems, only an exponentially small fraction of the assemblies would represent a solution, and finding such an assembly would be difficult. For example, a DNA based crystal growing system would create millions of crystals, and only a few of them may represent the correct answer, while all others represent failed computations. Finding a successful computation by sampling the crystals at random would require time exponential in the input. Thus it would be useful to attach a special identifier tile to the crystals that succeed so that the crystals may be filtered to find the solution quickly. It may also be possible to attach the special identifier tile to solid support so that the crystals representing successful computations may be extracted from the solution. I thus specify one of the tiles of a system as an **identifier** tile that only attaches to a configuration that represents a successful sequence of attachments.

Often, computer scientists talk about deciding subsets of the natural numbers instead of computing functions. Deciding a subset of the natural numbers is synonymous with computing a function that has value 1 on arguments that are in the set, and value 0 on arguments that are not in the set. I adapt the definition of nondeterministically computing functions to nondeterministically deciding subsets of natural numbers. (There is also a direct analog of deciding sets deterministically, which I do not bother to formally specify here.) Let $\mathbb{N} = \mathbb{Z}_{\geq 0}$. Since for all constants $n \in \mathbb{N}$, the cardinalities of \mathbb{N}^n and \mathbb{N} are the same, one can encode an element of \mathbb{N}^n as an element of \mathbb{N} . Thus it makes sense to talk about deciding subsets of \mathbb{N}^n . The below defined functions o_{s_m} can depend on the mapping of $\mathbb{N}^n \rightarrow \mathbb{N}$.

Let $v : \Gamma \cup T \rightarrow \{0, 1\}$ code each tile as a 1- or a 0-tile. Let $\hat{m} \in \mathbb{N}$ and let $\Omega \subseteq \mathbb{N}^{\hat{m}}$. For all $0 \leq m < \hat{m}$, let $o_{s_m} : \mathbb{N} \rightarrow \mathbb{Z}^2$ be injections. Let the seed encoding functions $e_{s_m} : \Delta \rightarrow \mathbb{N}$ map a seed S to \hat{m} numbers such that $e_{s_m}(S) = \sum_{i=0}^{\infty} 2^i v(S(o_{s_m}(i)))$ iff for no more than a constant number of (x, y) not in the union of the images of all o_{s_m} , $(x, y) \in S$. Let \mathbb{S} be a tile system with T as its set of tiles, and let $r \in T$. Then I say that \mathbb{S} **nondeterministically decides** a set Ω with identifier tile r iff for all $\vec{a} = \langle a_0, a_1, \dots, a_{\hat{m}-1} \rangle \in \mathbb{N}^{\hat{m}}$ there exists a seed configuration S such that for all final configurations F that \mathbb{S} produces on S , $r \in F(\mathbb{Z}^2)$ iff $\forall 0 \leq m < \hat{m}$, $e_{s_m}(S) = a_m$ and $\vec{a} \in \Omega$.

If for all $\hat{m} \in \mathbb{N}$, for all $0 \leq m < \hat{m}$, the o_{s_m} functions are allowed to be arbitrarily complex, the definition of computation in the tile assembly model is not very interesting because the computational intelligence of the system could simply be encoded in the o_{s_m} functions. For example, suppose h is the halting characteristic function (for all $a \in \mathbb{N}$, $h(a) = 1$ if the a th Turing machine halts on input a , and 0 otherwise) and o_{s_0} is such that the input a is encoded in some straight line if $h(a) = 1$ and in some jagged line otherwise. Then it would be trivial to design a tile system to solve the halting problem. Thus the complexities of the o_{s_m} functions need to be limited.

The problem of limiting the complexities is not a new one. When designing Turing machines, the input must be encoded on the tape and the theoreticians are faced with the exact same problem: an encoding that is too powerful could render the Turing machine capable of computing uncomputable functions. The common solution is to come up with a single straightforward encoding, e.g., for all $m \in \mathbb{N}$, converting the input element of \mathbb{N}^m into an element of \mathbb{N} via a mapping $\mathbb{N}^m \rightarrow \mathbb{N}$ and using the intuitive direct binary encoding of that element of \mathbb{N} on the Turing machine tape for all computations [36]. A similar approach is possible in the tile assembly model, requiring all systems to start with the input encoded the same way. In fact, it has been shown that such a definition conserves Turing universality of the tile systems [1]. However, the assembly time complexity of such systems may be adversely affected. In my definitions, I wish to give the system architect freedom in encoding the inputs for the sake of efficiency of computation; however, I restrict the o_{s_m} functions to be computable in linear time on a Turing machine. Thus these functions cannot add too much complexity-reducing power to the systems (the functions themselves cannot compute anything more complex than what linear-time algorithms can) while allowing the architects the freedom to place the inputs where they wish.

3. Solving satisfiability

The Boolean satisfiability (SAT) problem is a well-known NP-complete problem. Let $n \in \mathbb{N}$, then for all $0 \leq i < n$, let x_i be a Boolean variable that can take on values from the set $\{TRUE, FALSE\}$. Let the set of literals be the set of those variables and their negations $(\bigcup_i \{x_i, \neg x_i\})$, where $\neg TRUE = FALSE$, and $\neg FALSE = TRUE$. A clause is a disjunction of literals, e.g., $(x_0 \vee \neg x_1 \vee x_2)$. A Boolean formula, in conjunctive normal form (CNF), is a conjunction of clauses, e.g., $(x_0 \vee \neg x_1 \vee x_2) \wedge (\neg x_3 \vee x_2 \vee \neg x_2)$. If every clause has exactly k literals, the Boolean formula is said to be in k CNF. A Boolean formula is satisfiable iff there exists some assignment of each variable to an element of $\{TRUE, FALSE\}$ such that the formula evaluates to $TRUE$.

The notions of truth assignment and literal selection are in some sense parallel, and I will use them somewhat interchangeably in this paper. Formally, every literal selection corresponds to a truth assignment. Thus I will sometimes use a literal selection, e.g., $x_0, \neg x_1, x_2$, to specify a truth assignment, in this case $x_0 = x_2 = TRUE$ and $x_1 = FALSE$.

The k -SAT problem is, given a k CNF Boolean formula, to determine whether or not it is satisfiable. It is well known that 1-SAT and 2-SAT can be solved in polynomial time, while each k -SAT for $k \geq 3$ is NP-complete. Formally, k -SAT is the set of all Boolean formula in k CNF that are satisfiable. To solve k -SAT means to decide the set k -SAT.

The rest of this paper discusses solving satisfiability nondeterministically in the tile assembly model. Section 3.1 describes a system that uses $\Theta(n^2)$ distinct tiles to solve k -SAT for an arbitrary $k \in \mathbb{N}$, where n is the number of distinct variables in the Boolean formula. This system is similar to the system presented in [23], but I offer formal proofs of the system's correctness and speed. Section 3.2 describes a system that uses $\Theta(1)$ distinct tiles to solve k -SAT for an arbitrary $k \in \mathbb{N}$.

3.1. Naïve approach

Lagoudakis et al. [23] proposed a tile system for nondeterministically deciding whether a 3CNF Boolean formula is satisfiable. This system uses $\Theta(n^2)$ distinct tiles, for a formula with n distinct variables, and can be adapted to decide the satisfiability of k -SAT for arbitrary $k \in \mathbb{N}$. While Lagoudakis et al. do not formally define what it means for a tile system to solve a problem or compute a function and do not formally argue that their system does in fact solve satisfiability, I believe their system can be made to fit my definitions and proven correct. What I present here is a slight variant of their system, which follows the same basic logic. While it is of some interest to formally prove this system's correctness and speed, as I do below, the main reason for including this system is that it will explain part of the logic of the more complex system presented in Section 3.2.

I now describe a family of tile systems that determine whether a Boolean formula with $n \in \mathbb{N}$ distinct variables is satisfiable (because the size of the system depends on the number of variables, there will be a different system for every n , and thus I refer to the systems for all n as a family). I will refer to these systems as \mathbb{S}_n^2 .

The idea behind encoding the input is to encode the Boolean formula in the 0th row of the seed configuration with a unique tile for each possible literal, prefixing each clause with a special clause tile, and to encode the variables in the 0th column with a unique tile for each variable. Fig. 1 shows the concepts behind the tiles in Γ_n^2 , used by \mathbb{S}_n^2 . The bottom row shows three helper tiles that are the same for all n , and the top row shows the two tiles used to encode the Boolean formula (left) and the variable index (right). Thus, for a given $n \in \mathbb{N}$, the set Γ_n^2 contains the three bottom-row tiles, $2n$ left top-row tiles (where ℓi enumerates over all the $2n$ literals), and n right top-row tiles (where $0 \leq i < n$). Thus $|\Gamma_n^2| = 3n + 3$. For example, for $n = 3$, Fig. 2 shows the 12 tiles of the set Γ_3^2 .

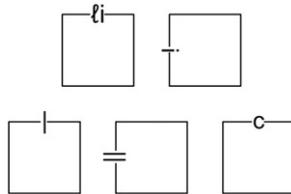


Fig. 1. The concepts behind the tiles in Γ_n^2 . For every $n \in \mathbb{N}$, the set Γ_n^2 contains $3n + 3$ tiles: the three bottom-row tiles, $2n$ left top-row tiles (where ℓi enumerates over all the $2n$ literals), and n right top-row tiles (where $0 \leq i < n$).

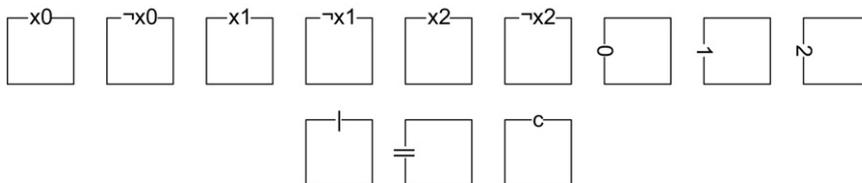


Fig. 2. The 12 tiles in Γ_3^2 .

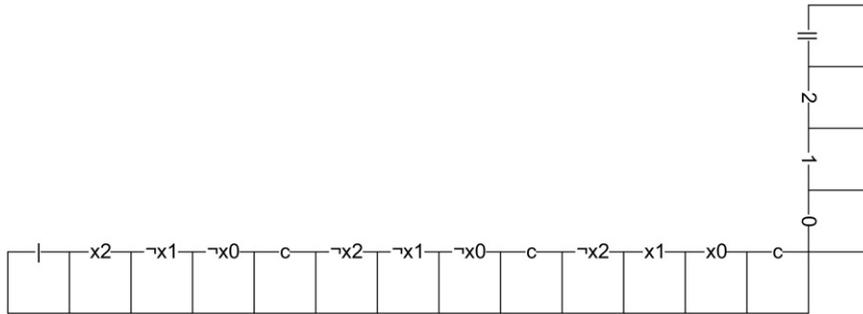


Fig. 3. The seed encoding a 3-variable Boolean formula $(x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ using the tiles from T_3^2 .

I will use the tiles in T_n^2 to encode an n -variable Boolean formula in a specific way. Informally, I will place tiles representing the formula’s literals in the 0th row such that the literals of each clause are together, place the special clause tile to the east of each clause, place the variable tiles in the 0th column, and place special end tiles in the west-most and north-most positions on that row and column. I explain the seed set up more formally below. Fig. 3 shows a sample seed encoding the 3-variable Boolean formula $(x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ using the tiles from T_3^2 . Note that while this example tries to follow some logical order, the order of the variables, the clauses, and the literals within each clause is not important.

The main idea of the computation is to have tiles attach nondeterministically in column -1 to select either *TRUE* or *FALSE* for each variable and then to “sweep” those choices westward, checking if a literal in a clause evaluates to *TRUE*. Whenever a literal evaluates to *TRUE*, that information propagates northward, and along the top row, tiles attach to ensure that at least one literal in every clause evaluates to *TRUE*. Iff that is the case, a special \checkmark tile attaches in the northwest corner.

The system S_n^2 will use the set of computational tiles T_n^2 . Fig. 4 shows the concepts behind the tiles in T_n^2 . The bottom row shows four helper tiles, including the special \checkmark tile, that are the same for all n . For each tile in the middle row there will be $\Theta(n)$ tiles in T_n^2 , with ℓ_i enumerating over all the literals and $0 \leq i < n$. Finally, the top row shows the concept behind the tile which will expand to $\Theta(n^2)$ tiles in T_n^2 , with ℓ_i enumerating over all the literals and ℓ_j enumerating over all the literals such that $\ell_i \neq \ell_j$. Thus, for a given $n \in \mathbb{N}$, the set T_n^2 contains the four bottom-row tiles, $14n$ middle-row tiles, and $4n^2 - 2n$ top-row tiles. Thus $|T_n^2| = 4n^2 + 12n + 4$. For example, for $n = 3$, Fig. 5 shows the 76 tiles of the set T_3^2 . Note that Lagoudakis et al. claim that their systems use $2n^2 + 12n + 10$ distinct tiles, but after careful analysis, I disagree with their calculations and believe their systems actually use significantly more tiles, even more than my system’s $4n^2 + 12n + 4$. Most notably, their analysis assumes that there are $2n^2 - n$ tiles identical to my yellow tiles, whereas in reality there are $4n^2 - 2n$ such tiles.

The tiles of T_3^2 attach to a seed configuration, such as the one in Fig. 3, to nondeterministically select a truth assignment of the variables and check if that assignment satisfies the Boolean formula, as shown in Fig. 6.

Theorem 1. For all $n \in \mathbb{N}$, let $\Sigma_n^2 = \{c, |, ||, i, x_i, \neg x_i, OK\}$, where $0 \leq i < n$. Let T_n^2 be as defined in Fig. 4. Let $g_n^2 = 1$ and $\tau_n^2 = 2$. Let $S_n^2 = \langle T_n^2, g_n^2, \tau_n^2 \rangle$. Then S_n^2 nondeterministically decides k -SAT (for all $k \in \mathbb{N}$) with up to n distinct variables per formula with the black \checkmark tile from T_n^2 as the identifier tile.

Proof. To show that S_n^2 nondeterministically decides k -SAT with up to n distinct variables per formula with the black \checkmark tile from T_n^2 as the identifier tile, I will first describe how to construct the seed from a Boolean formula ϕ with at most n

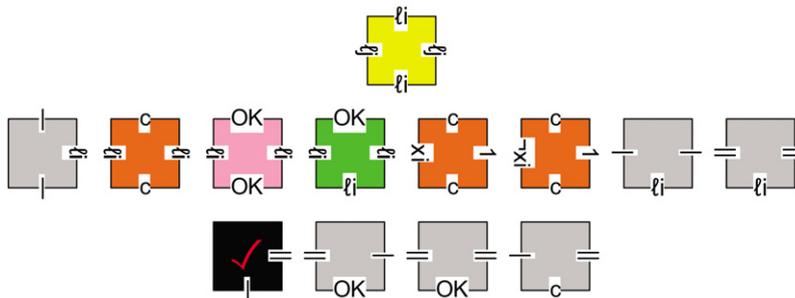


Fig. 4. The concepts behind the tiles in T_n^2 . For every $n \in \mathbb{N}$, the set T_n^2 contains $4n^2 + 12n + 4$ tiles: the four bottom-row tiles, including the special \checkmark tile, $14n$ middle-row tiles, with ℓ_i enumerating over all the literals and $0 \leq i < n$, and $4n^2 - 2n$ top row tiles, with ℓ_i enumerating over all the literals and ℓ_j enumerating over all the literals such that $\ell_i \neq \ell_j$.

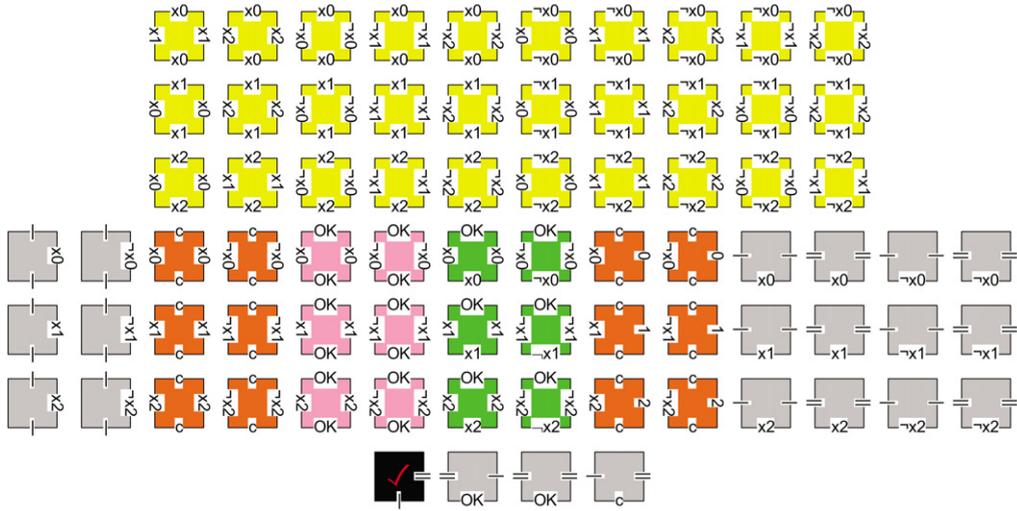


Fig. 5. The 76 tiles in T_3^2 .

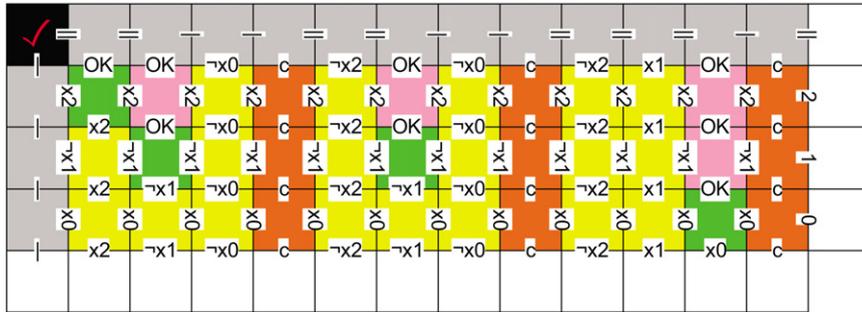


Fig. 6. Tiles from T_n^2 attach to the seed to nondeterministically select a truth assignment. Here $\phi = (x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ and the literal selection is $x_0, \neg x_1, x_2$ meaning that $x_0 = x_2 = \text{TRUE}$ and $x_1 = \text{FALSE}$. Tiles attach to allow the \checkmark tile to attach in the northwest corner iff the assignment satisfies the Boolean formula encoded by the seed.

distinct variables, then argue which tiles will attach to that seed, and finally conclude that the final assembly will contain the \checkmark tile iff there exists a truth assignment that makes ϕ evaluate to TRUE .

Let ϕ be a Boolean formula in $k\text{CNF}$, for some arbitrary $k \in \mathbb{N}$, with at most n distinct variables. Without loss of generality, I assume that the distinct variables are x_0, x_1, \dots, x_{n-1} . Let Γ_n^2 be as defined in Fig. 1. Let m be the number of clauses in ϕ (numbered $0, 1, \dots, m-1$).

To assist the readability of this proof, I will define two helper functions: $x: \mathbb{N} \rightarrow \mathbb{N}$ and $y: \mathbb{N} \rightarrow \mathbb{N}$. These functions will help identify positions on the 2-D grid. For all $\hat{m} \in \mathbb{N}$, let $x(\hat{m}) = -\hat{m}(k+1) - 1$. The intuition is that I will use the $(k+1)$ columns to the west of, and including column $x(\hat{m})$ for clause \hat{m} . For all $\hat{n} \in \mathbb{N}$, let $y(\hat{n}) = \hat{n} + 1$. The intuition is that I will use the row $y(\hat{n})$ for variable \hat{n} .

I define the seed S_n^2 that encodes ϕ as follows:

- $S_n^2(0, y(n)) = \gamma_{||}$, where $\gamma_{||}$ is the tile in Γ_n^2 with the west binding domain $||$.
- For all $0 \leq \hat{n} < n$, $S_n^2(0, y(\hat{n})) = \gamma_{\hat{n}}$, where $\gamma_{\hat{n}}$ is the tile in Γ_n^2 with the west binding domain \hat{n} .
- For all $0 \leq \hat{m} < m$, $S_n^2(x(\hat{m}), 0) = \gamma_c$, where γ_c is the tile in Γ_n^2 with the north binding domain c .
- For all $0 \leq \hat{m} < m$, for all $0 < \hat{k} \leq k$, $S_n^2(x(\hat{m}) - \hat{k}, 0) = \gamma_\ell$, where ℓ is the \hat{k} th literal of the \hat{m} th clause of ϕ , and γ_ℓ is the tile in Γ_n^2 with the north binding domain ℓ .
- $S_n^2(x(m), 0) = \gamma_{|}$, where $\gamma_{|}$ is the tile in Γ_n^2 with the north binding domain $|$.
- And for all other positions (v, w) , $S_n^2(v, w) = \text{empty}$.

Fig. 3 shows a sample seed for a 3-SAT formula ϕ with 3 distinct variables ($\phi = (x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$).

Note that because $\tau_n^2 = 2$, $g_n^2 = 1$, and the seed is in the shape of a horizontally reflected L , a tile may only attach in the S_n^2 system when its south and east neighbors are present, and only if its appropriate binding domains match those

neighbors' appropriate binding domains (if this statement is not clear, see the definition of L-configuration and the unique final configuration lemma (Lemma 2.2) from [15]).

I examine the tiles that can attach in column -1 , in positions $(-1, y(0))$ through $(-1, y(n) - 1)$. Because the west binding domains of the tiles in S_n^2 to the east of those positions are all \hat{n} , where $0 \leq \hat{n} < n$, the only tiles that might attach in this column are the orange tiles with \hat{n} east binding domains. By induction, because the north binding domain of $S_n^2(-1, 0)$ is c , and all the south and north binding domains for those orange tiles are c , those tiles will match their neighbors on the south and east sides, and thus will attach. For each \hat{n} , there are 2 possible tiles that may attach in position $(-1, y(\hat{n}))$. One of these tiles will have the west binding domain $x_{\hat{n}}$ and the other the west binding domain $\neg x_{\hat{n}}$. These tiles will attach nondeterministically, in some sense "selecting" only a single literal for each variable. Every set of tile attachments corresponds to a particular literal selection, and every literal selection has a set of tile attachments associated with it. Given a particular literal selection, the rest of the assembly will be deterministic (note, as I go through the proof, that no position in the rest of the assembly will have an east neighbor with a west binding domain \hat{n} , and that no tile in T_n^2 other than those with \hat{n} as their east binding domain have another tile with the same east and south binding domain pair). For the remainder of this proof, I fix the literal selection made in column -1 and refer to it as the *fixed assignment*.

Examine all the colored (neither black nor gray) tiles in T_n^2 . Other than the tiles with \hat{n} (where $0 \leq \hat{n} < n$) east binding domains that I just showed can only attach in column -1 , these are the only tiles that may attach in the rectangle defined by, and including, columns -2 and $x(m) - 1$ and rows $y(0)$ and $y(n) - 1$. (This fact follows because the gray and black tiles may only attach in columns with a $|$ north binding domain or rows with a $|$ or $||$ west binding domain, and by induction, since gray tiles are the only ones with such north and west binding domains, they can only attach in columns and rows with those binding domains in the seed, and those are column $x(m)$ and row $y(n)$, by definition of the seed.) Let the set T_D contain only those colored tiles with an east binding domain not equal to some \hat{n} . Now observe that for all the tiles t in T_D , $bd_W(t) = bd_E(t)$ and either $bd_N(t) = bd_S(t)$ or $bd_N(t) = \text{OK}$. As these tiles attach to S_n^2 , by induction, as long as every column of the rectangle is complete, the west binding domains of the tiles in row i are the same as the west binding domains of the tiles in row -1 , and therefore encode the fixed literal selection. Similarly, as long as every row of the rectangle is complete, the north binding domain of the tile in column p is either the same as the north binding domain of the seed's tile in position $(p, 0)$ or is OK.

Let ℓ be the \hat{k} th literal of the \hat{m} th clause of ϕ . I now prove that the north binding domain of the tile in position $(x(\hat{m}) - \hat{k}, y(n - 1))$ is OK iff ℓ is in the fixed assignment, and thus the \hat{m} th clause is satisfied. The north binding domain of the tile in position $(x(\hat{m}) - \hat{k}, 0)$ is ℓ , by the definition of the seed. Let \hat{n} be such that ℓ is either $x_{\hat{n}}$ or $\neg x_{\hat{n}}$. Then ℓ cannot match the literals in the assignment in rows $y(0)$ through $y(\hat{n} - 1)$, and thus the north binding domain of the tile in position $(x(\hat{m}) - \hat{k}, y(\hat{n} - 1))$ is ℓ . If ℓ is in the fixed assignment, then the tile that attaches in position $(x(\hat{m}) - \hat{k}, y(\hat{n}))$ must have matching east and south binding domains, and thus must be green, and thus has the north binding domain OK. Otherwise, the east and south binding domains do not match, and a yellow tile must attach, with the north binding domain ℓ . Since ℓ cannot match the literals in the assignment in rows $y(\hat{n} + 1)$ through $y(n) - 1$, iff the green tile had attached, rose tiles attach to the north, propagating the OK binding domain to row $y(n) - 1$, and otherwise yellow tiles attach, propagating the ℓ binding domain. Thus the north binding domain of the tile in position $(x(\hat{m}) - \hat{k}, y(n) - 1)$ is OK iff ℓ is in the fixed assignment.

Consider the tiles that attach in column $x(m)$. Because the seed has a north binding domain $|$ in that column, by induction, the gray tiles with north and south binding domains $|$ may attach. Since there is such a tile for every possible literal on its east binding domain, these tiles will propagate the $|$ binding domain to the north of the tile in position $(x(m), y(n) - 1)$.

Consider the tiles that attach in row $y(n)$. Because the seed has the west binding domain $||$ in that row, by induction the gray tiles with east and west binding domains in the set $\{||, ||\}$ may attach. I now show, by induction, that the west binding domain of the tile in position $(x(m) + 1, y(n))$ is $||$ iff every clause has at least one literal in the fixed assignment, and is thus satisfied. In the base case, by the definition of the seed, the west binding domain of the tile in position $(x(0) + 1, y(n))$ is $||$. Now I assume that the west binding domain of the tile in position $(x(\hat{m}) + 1, y(n))$ is $||$ and show that the west binding domain of the tile in position $(x(\hat{m} + 1) + 1, y(n))$ is $||$ iff the \hat{m} th clause is satisfied. Since the north binding domain of the tile in position $(x(\hat{m}), y(n - 1))$ is c , the gray tile with the south binding domain c must attach in position $(x(\hat{m}), y(n))$, and its west binding domain is $|$. Since there is a gray tile with every possible literal as its south binding domain and $|$ as its east and west binding domains, the $|$ will propagate to the west until either position $(x(\hat{m} + 1) + 1, y(n))$ or some north binding domain is OK. If the binding domain is OK, then some literal in clause \hat{m} is in the fixed assignment and thus this clause is satisfied, the gray tile with the south binding domain OK and west binding domain $||$ attaches, and the $||$ is propagated to the west by the gray tiles with literal and OK south binding domains and $||$ east and west binding domains to the tile in position $(x(\hat{m} + 1) + 1, y(n))$. Observe that there is no tile with a south binding domain c and east binding domain $|$, thus the west binding domain of the tile in position $(x(m) + 1, y(n))$ is $||$ iff every clause has at least one literal in the fixed assignment, and is thus satisfied. If there exists at least one unsatisfied clause, the position $(x(m) + 1, y(n))$ will either be empty, or the tile in that position will have the west binding domain $|$.

The \checkmark tile has an east binding domain $||$ and south binding domain $|$ thus it can only attach in position $(x(m), y(n))$ and only if the west binding domain of the tile in position $(x(m) + 1, y(n))$ is $||$. Thus the \checkmark tile can attach iff every clause is satisfied by the fixed assignment.

Since every possible assignment will be explored nondeterministically, if any one of them satisfies every clause, the \checkmark tile will attach. If no assignment exists that satisfies every clause, then the \checkmark tile will never attach. Thus S_n^2 nondeterministically

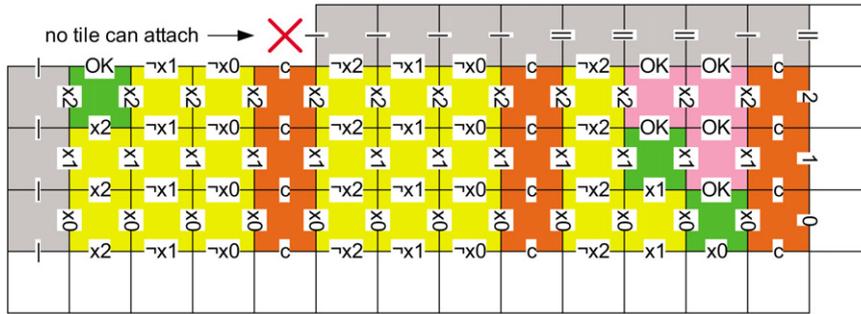


Fig. 7. For some nondeterministic choices of the truth assignment, the Boolean formula encoded by the seed is not satisfied. Here, $\phi = (x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ and the literal selection is x_0, x_1, x_2 meaning that $x_0 = x_1 = x_2 = TRUE$. The second clause is not satisfied and the \checkmark tile never attaches. Note that this is a final configuration and no more tiles may attach.

decides k -SAT (for all $k \in \mathbb{N}$) with up to n distinct variables per formula with the black \checkmark tile from T_n^2 as the identifier tile. \square

Fig. 7 shows a nondeterministically selected truth assignment that does not satisfy the Boolean formula. Because the formula is not satisfied, the \checkmark tile never attaches.

Lemma 1. *The assembly time of S_n^2 is linear in the size of the input ϕ .*

This lemma follows directly from the assembly time lemma (Lemma 2.3) from [15]. The assembly time lemma states that under certain conditions (such as in a tile system with a strength function that is identically 1, operating at temperature 2, and starting from a seed configuration that resembles the horizontally reflected letter L, such as S_n^2), the assembly time is on the order of the sum of the dimensions of the L.

Lemma 2. *For all $n \in \mathbb{N}$, for all Boolean formula ϕ on n distinct variables, assuming each tile that may attach to a configuration at a certain position attaches there with a uniform probability distribution, the probability that a single nondeterministic execution of S_n^2 succeeds in attaching a \checkmark tile if ϕ is satisfiable is at least $(\frac{1}{2})^n$.*

Proof. Only the tiles in column -1 attach nondeterministically, and at each of those positions there are exactly two tiles that may attach. If ϕ is satisfiable, then there exists at least one assignment that satisfies it, and thus a particular choice at each of the n nondeterministic positions in column -1 will select a satisfying assignment. The probability of the correct tile attaching at each location is $\frac{1}{2}$, and thus the probability of the whole column attaching to represent the correct assignment is $(\frac{1}{2})^n$. Since the rest of the assembly is deterministic, $(\frac{1}{2})^n$ is the lower bound on the probability that a single nondeterministic execution of S_n^2 succeeds in attaching a \checkmark tile if ϕ is satisfiable. \square

In summary, S_n^2 decides whether a k CNF Boolean formula ϕ on n variables is in k -SAT, has $4n^2 + 12n + 4 = \Theta(n^2)$ computational tile types and uses $3n + 3 = \Theta(n)$ tile types to encode the input. It computes in time linear in the size of the input, and each assembly has the probability of at least $(\frac{1}{2})^n$ of finding the satisfying assignment, if one exists.

3.2. Constant-size tileset approach

I now describe a nondeterministic tile system S_{SAT} , which will follow a logic similar to that of S_n^2 , but will use only a constant number of tiles. The idea of S_{SAT} is to encode ϕ in the same way S_n^2 did, but instead of using a single tile for each literal, the literals will be encoded by a tile that indicates whether the literal is a negation (I place this tile in the east-most position), and a series of 0 and 1 tiles encoding, in binary, the index of the variable. For example, the literal x_5 would be encoded by a v tile, and by a 1 tile, then a 0 tile, and then a 1 tile (101 v) because $5 = 101_2$. The literal $\neg x_4$ would be encoded by a $\neg v$ tile, and by a 1 tile, then a 0 tile, and then a 0 tile (100 $\neg v$) because $4 = 100_2$. For consistency, I will use the same number of bits to encode all variables, e.g., if my ϕ has 7 distinct variables, I will need three bits to encode x_7 , so I will encode x_1 as 001 v . Similarly, I will encode the variables in the 0th column using this binary encoding method. The assemblies in S_{SAT} will be larger in size than the assemblies in S_n^2 because what used to be encoded by a single tile will now be represented by a $\Theta(\log n) \times \Theta(\log n)$ block of tiles, but the overall logic will remain the same. The key to S_{SAT} is building the logic of the blocks to correctly match literals without affecting the inherited logic of S_n^2 .

There are 12 tiles in Γ_{SAT} , shown in Fig. 8, no matter how large ϕ is or how many variables it contains. I will use the tiles in Γ_{SAT} to encode an n -variable Boolean formula in a specific way. Informally, I will encode the formula's literals in the 0th row, as described above, such that the literals of each clause are together, place the special clause tile to the east

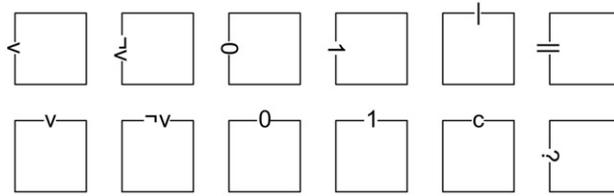


Fig. 8. The 12 tiles in Γ_{SAT} .

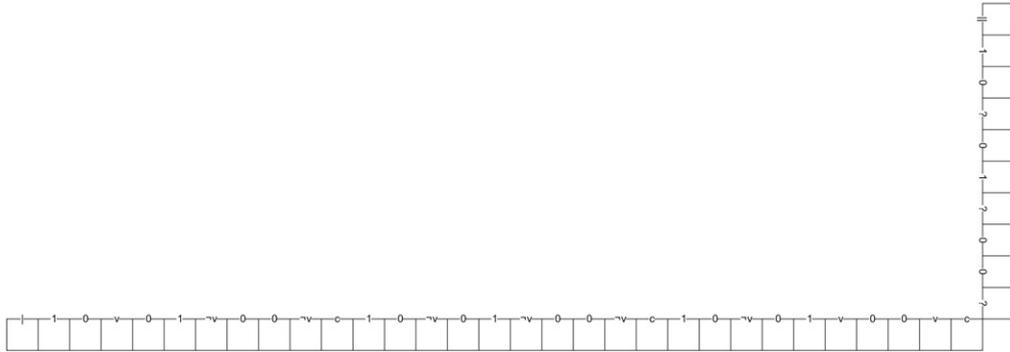


Fig. 9. The seed encoding a 3-variable Boolean formula $(x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ using the tiles from Γ_{SAT} .

of each clause, place the encoded variables in the 0th column, and place special end tiles in the west-most and north-most positions on that row and column. I explain the seed set up more formally below. Fig. 9 shows a sample seed encoding the 3-variable Boolean formula $(x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ using the tiles from Γ_{SAT} . Note that while this example tries to follow some logical order, the order of the variables, the clauses, and the literals within each clause is not important.

Just as before, the main idea of the computation is to have tiles attach nondeterministically in column -1 to select either *TRUE* or *FALSE* for each variable and then to “sweep” those choices westward, checking if a literal in a clause evaluates to *TRUE*. The comparison of literals will take place within a $\Theta(\log n) \times \Theta(\log n)$ block of tiles. Whenever a literal evaluates to *TRUE*, that information propagates northward, and along the top row, tiles attach to ensure that at least one literal in every clause evaluates to *TRUE*. Iff that is the case, a special \checkmark tile attaches in the northwest corner.

The system \mathbb{S}_{SAT} will use the set of computational tiles T_{SAT} . Fig. 10 shows the 64 tiles in T_{SAT} . The colors of the tiles are coordinated with the colors of the T_n^2 system—the tiles of the same colors perform the same functions.

The tiles of T_{SAT} attach to a seed configuration, such as the one in Fig. 9, to nondeterministically select a truth assignment of the variables and check if that assignment satisfies the Boolean formula, as shown in Fig. 11.

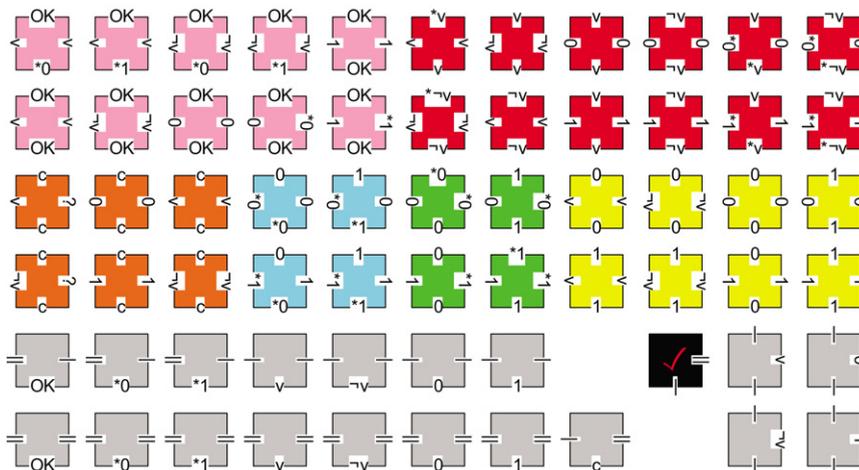


Fig. 10. The 64 tiles in T_{SAT} .

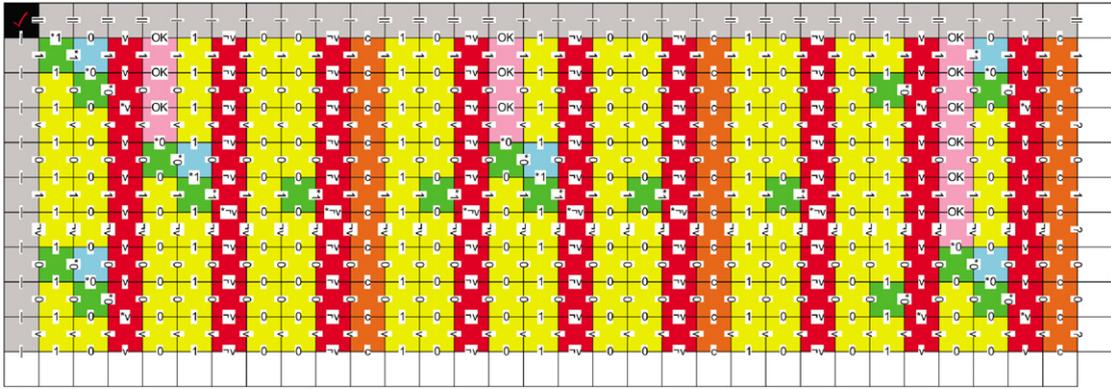


Fig. 11. Tiles from T_{SAT} attach to the seed to nondeterministically select a truth assignment. Here $\phi = (x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ and the literal selection is $x_0, \neg x_1, x_2$ meaning that $x_0 = x_2 = TRUE$ and $x_1 = FALSE$. Tiles attach to allow the \checkmark tile to attach in the northwest corner iff the assignment satisfies the Boolean formula encoded by the seed.

Lemma 3. Let a configuration S (confined to some $v \times v$ square with the southeast tile at position (x_0, y_0)) be such that:

- $bd_N(S(x_0 - 1, y_0)) \in \{v, \neg v\}$,
- $bd_W(S(x_0, y_0 + 1)) \in \{v, \neg v\}$,
- for all $1 < i \leq v$, $bd_N(S(x_0 - i, y_0)) \in \{0, 1\}$,
- for all $1 < i \leq v$, $bd_W(S(x_0, y_0 + i)) \in \{0, 1\}$, and
- for all positions (x, y) within the square, $S(x, y) = empty$.

And let $g_{=} = 1$ and $\tau_{=} = 2$, and $S_{=} = \langle T_{SAT}, g_{=}, \tau_{=} \rangle$. Then $S_{=}$ produces a final unique configuration F on S such that:

- $bd_N(F(x_0 - v, y_0 + v))$ contains a $*$ iff for all $0 < i \leq v$, $bd_N(x_0 - i, y_0) = bd_W(x_0, y_0 + i)$,
- no other tile in row $y_0 + v$ has a north binding domain that contains a $*$, and
- no tile in column $x_0 - v$ has a west binding domain that contains a $*$.

Proof. Let a relationship \approx be defined on binding domains such that given binding domains a and b , $a \approx b$ iff the portion of a that is not $*$ exactly equals the portion of b that is not $*$. For example, $0 \approx 0 \approx *0 \not\approx *1$. Observe that for all tiles $t \in T_{SAT}$, either $bd_N(t) = OK$ or $bd_N(t) \approx bd_S(t)$, and either $bd_E(t) = ?$ or $bd_E(t) \approx bd_W(t)$. That is to say, other than the tiles with a $?$, the east binding domain \approx the west binding domain, and other than the tiles with an OK , the north binding domain \approx the south binding domain. It will become clear in this proof that the tiles with $?$ and OK binding domains will never attach to S , thus the west binding domain of all tiles in row r will $\approx bd_W(S(x_0, r))$ and the north binding domains of all tiles in column c will $\approx bd_N(S(c, y_0))$.

I now show by induction that iff for all $\mu \leq v$, for all $0 < i \leq \mu$, $bd_N(S(x_0 - i, y_0)) = bd_W(S(x_0, y_0 + i))$ then the only tile in row $y_0 + \mu$ whose north binding domain contains a $*$ is in position $(x_0 - \mu, y_0 + \mu)$ and that no tile in column $x_0 - \mu$ has a west binding domain that contains a $*$. Otherwise, none of those binding domains contain a $*$.

First, examine the base case. Let $\mu = 1$. Examine row $(y_0 + \mu)$. By the definition of S , in position $(x_0 - \mu, y_0 + \mu)$, one of the red tiles from T_{SAT} must attach. Iff the east and south binding domains of that red tile are equal (and thus $bd_N(S(x_0 - \mu, y_0)) = bd_W(S(x_0, y_0 + \mu))$) then the north binding domain of the tile contains a $*$. It follows that the yellow tiles will attach in row $(y_0 + \mu)$ to the west of that red tile. The yellow tiles contain no binding domains with $*$.

Now assume that if for all $0 < i \leq \mu$, $bd_N(S(x_0 - i, y_0)) = bd_W(S(x_0, y_0 + i))$ then the only tile in row $y_0 + \mu$ whose north binding domain contains a $*$ is in position $(x_0 - \mu, y_0 + \mu)$ and that no tile in column $x_0 - \mu$ has a west binding domain that contains a $*$, and that otherwise, none of those binding domains contain a $*$. I will now show that if $bd_N(S(x_0 - (\mu + 1), y_0)) = bd_W(S(x_0, y_0 + (\mu + 1)))$, then the only tile in row $y_0 + \mu + 1$ whose north binding domain contains a $*$ is in position $(x_0 - (\mu + 1), y_0 + (\mu + 1))$ and no tile in column $x_0 - (\mu + 1)$ has a west binding domain that contains a $*$, and that otherwise, none of those binding domains contain a $*$. Examine row $(y_0 + (\mu + 1))$. If the north binding domain of the tile in position $(x_0 - \mu, y_0 + \mu)$ contains a $*$ (thus the appropriate north and west binding domains of S have matched up to μ), then a blue tile must attach in position $(x_0 - \mu + 1, y_0 + (\mu + 1))$, with a west binding domain containing a $*$. Then a green tile must attach to the west, in position $(x_0 - (\mu + 1), y_0 + (\mu + 1))$. Iff the east and south binding domains of that green tile are \approx (and thus $bd_N(S(x_0 - (\mu + 1), y_0)) = bd_W(S(x_0, y_0 + (\mu + 1)))$) then the green tile's north binding domain contains a $*$. Yellow tiles attach in column $x_0 - (\mu + 1)$ below that green tile. Thus if $bd_N(S(x_0 - (\mu + 1), y_0)) = bd_W(S(x_0, y_0 + (\mu + 1)))$, then the only tile in row $y_0 + \mu + 1$ whose north binding domain contains a $*$ is in position $(x_0 - (\mu + 1), y_0 + (\mu + 1))$ and no tile in column $x_0 - (\mu + 1)$ has a west binding domain that contains a $*$, and otherwise, none of those binding domains contain a $*$.

The lemma follows when $\mu = v$. \square

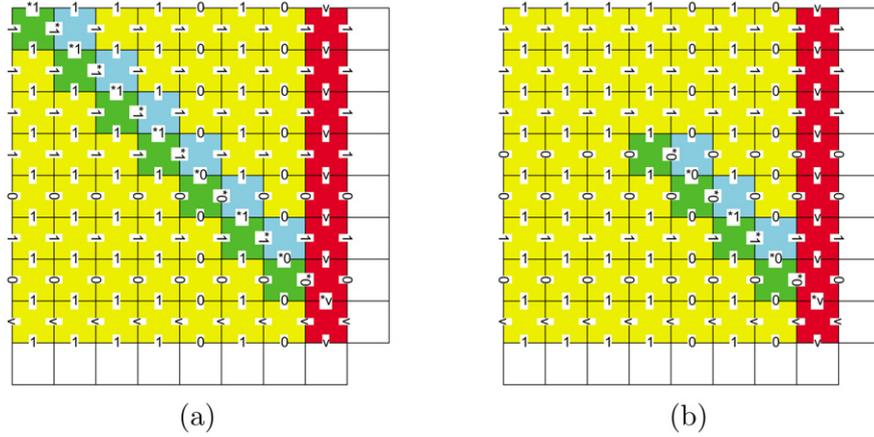


Fig. 12. Tiles comparing two inputs. In (a), the comparison is between 1111010v and 1111010v. Because the two inputs are the same, the northwest tile's north binding domain contains a *, and none of the rest of the exposed binding domains do. In (b), the comparison is between 1111010v and 1110010v. Because the two inputs do not match, no exposed binding domain contains a *.

Lemma 3 describes a subset of T_{SAT} attaching to compare two inputs. Fig. 12(a) shows a comparison of 1111010v and 1111010v. Because the two inputs are the same, the northwest tile's north binding domain contains a *, and none of the rest of the exposed binding domains do. Fig. 12(b) shows a comparison of 1111010v and 1110010v. Because the two inputs do not match, no exposed binding domain contains a *.

Theorem 2. Let $\Sigma_{SAT} = \{c, ?, |, ||, v, *v, -v, *-v, 0, *0, 1, *1, OK\}$. Let T_{SAT} be as defined in Fig. 10. Let $g_{SAT} = 1$ and $\tau_{SAT} = 2$. Let $\mathbb{S}_{SAT} = \langle T_{SAT}, g_{SAT}, \tau_{SAT} \rangle$. Then \mathbb{S}_{SAT} nondeterministically decides k -SAT (for all $k \in \mathbb{N}$) with the black \checkmark tile from T_{SAT} as the identifier tile.

Proof. To show that \mathbb{S}_{SAT} nondeterministically decides k -SAT with the black \checkmark tile from T_{SAT} as the identifier tile, I will first describe how to construct the seed from a Boolean formula ϕ , then argue which tiles will attach to that seed, and finally conclude that the final assembly will contain the \checkmark tile iff there exists a truth assignment that makes ϕ evaluate to *TRUE*.

Let ϕ be a Boolean formula in k CNF, for some arbitrary $k \in \mathbb{N}$. Let n be the number of distinct variables in ϕ , and let $\nu = \lceil \lg n \rceil + 1$. The value ν is the number of tiles used to encode each variable ($\lg n$ tiles to encode the variable's number in binary and 1 tile to encode whether the variable is negated). Note that \mathbb{S}_{SAT} works for all n , and ν depends on n only to properly encode the variables in the seed. Without loss of generality, I assume that the distinct variables of ϕ are x_0, x_1, \dots, x_{n-1} . Let Γ_{SAT} be as defined in Fig. 8. Let m be the number of clauses in ϕ , numbered $0, 1, \dots, m-1$.

To assist the readability of this proof, I will define three helper functions: $x: \mathbb{N} \rightarrow \mathbb{N}$, $v: \mathbb{N} \rightarrow \mathbb{N}$ and $y: \mathbb{N} \rightarrow \mathbb{N}$. These functions will help identify positions on the 2-D grid. For all $\hat{m} \in \mathbb{N}$, let $x(\hat{m}) = -\hat{m}(k\nu + 1) - 1$. The intuition is that I will use the $(k\nu + 1)$ columns to the west of, and including column $x(\hat{m})$ for clause \hat{m} . For all $\hat{k} \in \mathbb{N}$, let $v(\hat{k}) = \hat{k}\nu + 1$. The intuition is that I will use the ν columns to the west of, and including column $x(\hat{m}) - v(\hat{k})$ for the \hat{k} th literal in the \hat{m} clause (where the literals of a clause are $\ell_0, \ell_1, \dots, \ell_{k-1}$). For all $\hat{n} \in \mathbb{N}$, let $y(\hat{n}) = \hat{n}\nu + 1$. The intuition is that I will use ν rows to the north of, and including row $y(\hat{n})$ for variable \hat{n} .

I define the seed S_{SAT} that encodes ϕ as follows:

- $S_{SAT}(0, y(n)) = \gamma_{||}$, where $\gamma_{||}$ is the tile in Γ_{SAT} with the west binding domain $||$.
- For all $0 \leq \hat{n} < n$, $S_{SAT}(0, y(\hat{n})) = \gamma_?$, where $\gamma_?$ is the tile in Γ_{SAT} with the west binding domain $?$.
- For all $0 \leq \hat{n} < n$, for all $0 \leq i < \nu - 1$, $S_{SAT}(0, y(\hat{n}) + i + 1) = \gamma_z$, where z is the i th bit of \hat{n} ($z = \lfloor \frac{\hat{n}}{2^i} \rfloor \bmod 2$) and γ_z is the tile in Γ_{SAT} with the west binding domain z .
- For all $0 \leq \hat{m} < m$, $S_{SAT}(x(\hat{m}), 0) = \gamma_c$, where γ_c is the tile in Γ_{SAT} with the north binding domain c .
- For all $0 \leq \hat{m} < m$, for all $0 \leq \hat{k} < k$, $S_{SAT}(x(\hat{m}) - v(\hat{k}), 0) = \gamma_{-}$, where if the \hat{k} th literal in the \hat{m} th clause is some unnegated variable then γ_{-} is the tile in Γ_{SAT} with the north binding domain v and if the \hat{k} th literal in the \hat{m} th clause is the negation of some variable then γ_{-} is the tile in Γ_{SAT} with the north binding domain $-v$.
- For all $0 \leq \hat{m} < m$, for all $0 \leq \hat{k} < k$, for all $0 \leq i < \nu - 1$, $S_{SAT}(x(\hat{m}) - v(\hat{k}) - i - 1, 0) = \gamma_z$, where if w is such that the \hat{k} th literal in the \hat{m} th clause is either x_w or $\neg x_w$, then z is the i th bit of w ($z = \lfloor \frac{w}{2^i} \rfloor \bmod 2$) and γ_z is the tile in Γ_{SAT} with the north binding domain z .
- $S_{SAT}(x(m), 0) = \gamma_|$, where $\gamma_|$ is the tile in Γ_{SAT} with the north binding domain $|$.
- And for all other positions (v, w) , $S_{SAT}(v, w) = \text{empty}$.

Fig. 9 shows a sample seed for a 3-SAT formula ϕ with 3 distinct variables ($\phi = (x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$).

Note that because $\tau_{SAT} = 2$, $g_{SAT} = 1$, and the seed is in the shape of a horizontally reflected L , a tile may only attach in the S_{SAT} system when its south and east neighbors are present, and only if its appropriate binding domains match those neighbors' appropriate binding domains (if this statement is not clear, see the definition of L -configuration and the unique final configuration lemma (Lemma 2.2) from [15]).

I examine the tiles that can attach in column -1 , in positions $(-1, y(0))$ through $(-1, y(n) - 1)$. First, consider the positions $(-1, y(\hat{n}))$, for all $0 \leq \hat{n} < n$. Because the west binding domains of the tiles in S_{SAT} to the east of those positions are all $?$, the only tiles that might attach in this column are the orange tiles with $?$ east binding domains. In the other positions, the orange tiles with the east and west binding domains 0 and 1 may attach. By induction, because the north binding domain of $S_{SAT}(-1, 0)$ is c , and all the south and north binding domains for those orange tiles are c , those tiles will match their neighbors on the south and east sides, and thus will attach. For each \hat{n} , there are 2 possible tiles that may attach in position $(-1, y(\hat{n}))$. One of these tiles will have the west binding domain v and the other the west binding domain $\neg v$. These tiles will attach nondeterministically, in some sense “selecting” only a single literal for each variable. Every set of tile attachments corresponds to a particular literal selection, and every literal selection has a set of tile attachments associated with it. Given a particular literal selection, the rest of the assembly will be deterministic (note, as I go through the proof, that no position in the rest of the assembly will have an east neighbor with a west binding domain $?$, and that no tile in T_{SAT} other than those with $?$ as their east binding domain have another tile with the same east and south binding domain pair). For the remainder of this proof, I fix the literal selection made in column -1 and refer to it as the *fixed assignment*.

Examine all the colored (neither black nor gray) tiles in T_{SAT} . Other than the tiles with $?$ east binding domains that I just showed can only attach in column -1 , these are the only tiles that may attach in the rectangle defined by, and including, columns -2 and $x(m) - 1$ and rows $y(0)$ and $y(n) - 1$. (This fact follows because the gray and black tiles may only attach in columns with a $|$ north binding domain or rows with a $|$ or $||$ west binding domain, and by induction, since gray tiles are the only ones with such north and west binding domains, they can only attach in columns and rows with those binding domains in the seed, and those are column $x(m)$ and row $y(n)$, by definition of the seed.) Let the set T_D contain only those colored tiles with an east binding domain not equal to $?$. Now observe that for all the tiles t in T_D , $bd_W(t) \approx bd_E(t)$ and either $bd_N(t) \approx bd_S(t)$ or $bd_N(t) = \text{OK}$ (using the definition of \approx from the proof of Lemma 3). Thus I conclude, by induction, that as these tiles attach to S_{SAT} , as long as every column of the rectangle is complete, the west binding domains of the tiles in row i are \approx the west binding domains of the tiles in row -1 , and thus they encode the fixed literal selection, and that as long as every row of the rectangle is complete, the north binding domain of the tile in column p is either \approx the north binding domain or the seed's tile in position $(p, 0)$ or is OK.

Let ℓ be the \hat{k} th literal of the \hat{m} th clause of ϕ . I now prove that the north binding domain of the tile in position $(x(\hat{m}) - (v(\hat{k} + 1) + 1), y(n) - 1)$ is either OK or contains a $*$ iff ℓ is in the fixed assignment, and thus the \hat{m} th clause is satisfied. Examine the v columns to the west of, and including column $x(\hat{m}) - v(\hat{k})$. Those columns' intersections with the rows $y(0)$ through $y(1) - 1$ are a $v \times v$ square matching the description of the $v \times v$ square in Lemma 3. The north binding domains to the south of the square encode ℓ . The west binding domains to the east of the square encode the literal of the fixed assignment that is either x_0 or $\neg x_0$. By Lemma 3, the north binding domain of the tile in position $(x(\hat{m}) - v(\hat{k} + 1) + 1, y(1) - 1)$ contains a $*$ iff ℓ is in the fixed assignment, and thus if the \hat{m} th clause is satisfied. If that binding domain has no $*$, then by induction, ℓ is compared with each other literal in the fixed assignment in rows $y(1)$ through $y(n) - 1$. If ever a north binding domain in column $(x(\hat{m}) - v(\hat{k} + 1) + 1, y(1) - 1)$ contains a $*$ (except in row $y(n) - 1$), then the tile that attaches to it has the east binding domain either v or $\neg v$ and thus must be rose and has the north binding domain OK. That OK binding domain is then propagated to row $y(n) - 1$ by the rose tiles. Thus iff ℓ is in the fixed assignment, the north binding domain of the tile in position $(x(\hat{m}) - v(\hat{k} + 1) + 1, y(n) - 1)$ is either OK or contains a $*$.

Consider the tiles that attach in column $x(m)$. Because the seed has a north binding domain $|$ in that column, by induction, the gray tiles with north and south binding domains $|$ may attach. Since for every element of $\{v, \neg v, 0, 1\}$, there is a tile with that east binding domain, these tiles will propagate the $|$ binding domain to the north of the tile in position $(x(m), y(n) - 1)$.

Consider the tiles that attach in row $y(n)$. Because the seed has the west binding domain $||$ in that row, by induction the gray tiles with east and west binding domains in the set $\{|, ||\}$ may attach. I now show, by induction, that the west binding domain of the tile in position $(x(m) + 1, y(n))$ is $||$ iff every clause has at least one literal in the fixed assignment, and is thus satisfied. In the base case, by the definition of the seed, the west binding domain of the tile in position $(x(0) + 1, y(n))$ is $||$. Now I assume that the west binding domain of the tile in position $(x(\hat{m}) + 1, y(n))$ is $||$ and show that the west binding domain of the tile in position $(x(\hat{m} + 1) + 1, y(n))$ is $||$ iff the \hat{m} th clause is satisfied. Since the north binding domain of the tile in position $(x(\hat{m}), y(n - 1))$ is c , the gray tile with the south binding domain c must attach in position $(x(\hat{m}), y(n))$, and its west binding domain is $|$. Since there is a gray tile with every element of $\{v, \neg v, 0, 1\}$ as its south binding domain and $|$ as its east and west binding domains, the $|$ will propagate to the west until either position $(x(\hat{m} + 1) + 1, y(n))$ or some north binding domain is either OK or contains a $*$. If the binding domain is OK or contains a $*$, then some literal in clause \hat{m} is in the fixed assignment and thus this clause is satisfied, a gray tile with the south binding domain OK or containing a $*$ and west binding domain $||$ attaches, and the $||$ is propagated to the west by the gray tiles with $\{v, \neg v, 0, 1, *0, *1, \text{OK}\}$ south binding domains and $||$ east and west binding domains to the tile in position $(x(\hat{m} + 1) + 1, y(n))$. Observe that there

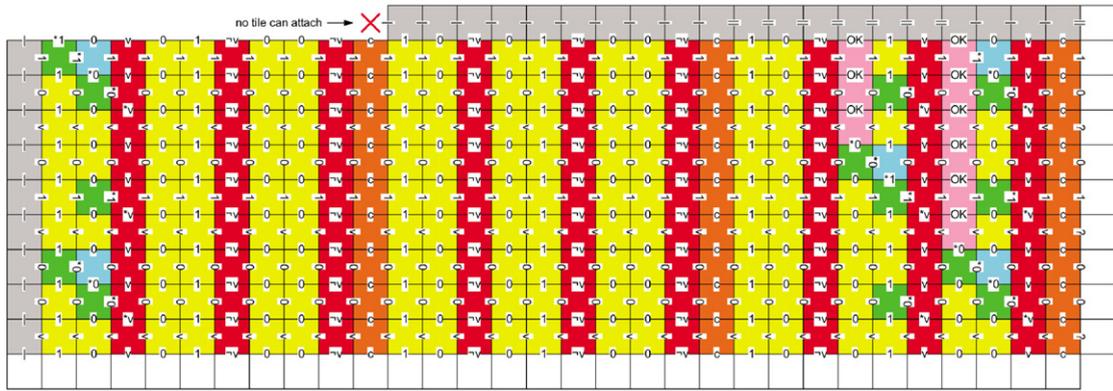


Fig. 13. For some nondeterministic choices of the truth assignment, the Boolean formula encoded by the seed is not satisfied. Here, $\phi = (x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee \neg x_1 \vee \neg x_0) \wedge (\neg x_2 \vee x_1 \vee x_0)$ and the literal selection is x_0, x_1, x_2 meaning that $x_0 = x_1 = x_2 = \text{TRUE}$. The second clause is not satisfied and the \checkmark tile never attaches. Note that this is a final configuration and no more tiles may attach.

is no tile with a south binding domain c and east binding domain $|$, thus the west binding domain of the tile in position $(x(m) + 1, y(n))$ is $||$ iff every clause has at least one literal in the fixed assignment, and is thus satisfied. If there exists at least one unsatisfied clause, the position $(x(m) + 1, y(n))$ will either be empty, or the tile in that position will have the west binding domain $|$.

The \checkmark tile has an east binding domain $||$ and south binding domain $|$ thus it can only attach in position $(x(m), y(n))$ and only if the west binding domain of the tile in position $(x(m) + 1, y(n))$ is $||$. Thus the \checkmark tile can attach iff every clause is satisfied by the fixed assignment.

Since every possible assignment will be explored nondeterministically, if any one of them satisfies every clause, the \checkmark tile will attach. If no assignment exists that satisfies every clause, then the \checkmark tile will never attach. Thus \mathbb{S}_{SAT} nondeterministically decides k -SAT (for all $k \in \mathbb{N}$) with the black \checkmark tile from T_{SAT} as the identifier tile. \square

Fig. 13 shows a nondeterministically selected truth assignment that does not satisfy the Boolean formula. Because the formula is not satisfied, the \checkmark tile never attaches.

Lemma 4. *The assembly time of \mathbb{S}_{SAT} is linear in the number of bits necessary to describe ϕ (i.e., the size of ϕ).*

Proof. For a ϕ in k CNF with m clauses and n distinct variables, each literal can be described using $\Theta(\log n)$ bits. Thus ϕ can be described using $\Theta(mk \log n)$ bits. The dimensions of the rectangle formed by the seed S_{SAT} are $\Theta(mk \log n) \times \Theta(n \log n)$ and it follows from the assembly time lemma (Lemma 2.3) from [15] that the assembly time of \mathbb{S}_{SAT} is $\Theta(mk \log n)$, which is linear in the size of ϕ . \square

Lemma 5. *For all $n \in \mathbb{N}$, for all Boolean formula ϕ on n distinct variables, assuming each tile that may attach to a configuration at a certain position attaches there with a uniform probability distribution, the probability that a single nondeterministic execution of \mathbb{S}_{SAT} succeeds in attaching a \checkmark tile if ϕ is satisfiable is at least $(\frac{1}{2})^n$.*

Proof. The only tiles that attach nondeterministically in \mathbb{S}_{SAT} attach in positions $(-1, y(\hat{n}))$ for all $0 \leq \hat{n} < n$. At each of those positions there are exactly two tiles that may attach. If ϕ is satisfiable, then there exists at least one assignment that satisfies it, and thus a particular choice at each of the n nondeterministic positions in column -1 will select a satisfying assignment. The probability of the correct tile attaching at each location is $\frac{1}{2}$, and thus the probability of the whole column attaching to represent the correct assignment is $(\frac{1}{2})^n$. Since the rest of the assembly is deterministic, $(\frac{1}{2})^n$ is the lower bound on the probability that a single nondeterministic execution of \mathbb{S}_{SAT} succeeds in attaching a \checkmark tile if ϕ is satisfiable. \square

In summary, \mathbb{S}_{SAT} decides whether a k CNF Boolean formula ϕ on n variables is in k -SAT, has $64 = \Theta(1)$ computational tile types and uses $12 = \Theta(1)$ tile types to encode the input. It computes in time linear in the size of the input, and each assembly has the probability of at least $(\frac{1}{2})^n$ of finding the satisfying assignment, if one exists.

4. Contributions

I have designed two systems that solve well-known NP-complete problems k -SAT, for all $k \in \mathbb{N}$, in the tile assembly model. The first system, \mathbb{S}_n^2 , uses $\Theta(n^2)$ distinct tiles to decide a Boolean formula with n distinct variables and is closely related to a previously described though unproven system [23]. The second system, \mathbb{S}_{SAT} , uses $64 = \Theta(1)$ distinct tiles to decide a Boolean formula. I prove the correctness of both systems and analyze their size and time complexities. Both

systems compute in time linear in the input size. Each nondeterministic assembly has a probability of success of at least $(\frac{1}{2})^n$, where n is the number of distinct variables. Thus a parallel implementation of \mathbb{S}_{SAT} , such as a DNA implementation like those in [11,12], with 2^n seeds has at least a $1 - \frac{1}{e} \geq 0.5$ chance of correctly deciding whether a Boolean formula is satisfiable. An implementation with 100 times as many seeds has at least a $1 - (\frac{1}{e})^{100}$ chance. Experiments in DNA self-assembly commonly contain on the order of 10^{17} assemblies [27,37–39]. However, those experiments in no way required a high concentration of assemblies and no specific attempts to achieve a maximum concentration were made. In fact, experiments such as these commonly try to limit the number of parallel assemblies, as all the assemblies are identical and creating many of them is simply a waste of material. Thus it is likely that orders of magnitude larger volumes of solutions orders of magnitude more concentrated than those can be achieved.

Previous related work has solved problems with inputs that did not require unique identifiers on variables, nodes, or edges. For example, the NP-complete problem *SubsetSum* [22]. Tile system solutions to problems that do require such unique identifiers have resorted to using $\Theta(n)$ tiles to encode the input, and no fewer than $\Theta(n^2)$ tiles to compute, for inputs of size n [23]. My proposal is the first constant-size tileset solution that solves such a problem, and the mechanism I design for uniquely addressing variables is completely portable to solving other such problems, such as graph problems, many of which are known to be NP-complete.

While DNA self-assembly suffers from high error-rates, the existence of methods of error-control and error-correction for self-assembly systems present it as a promising direction for molecular computation and early experimental and these theoretical results shine even more promise on self-assembly. Further, early investigations into programming large distributed computer networks by representing each computer as a tile in a tile assembly model system have revealed promising possibilities, thus, generalizing the tile assembly model as a potentially powerful tool in software architecture research [40].

References

- [1] Erik Winfree, Algorithmic self-assembly of DNA, PhD thesis, California Institute of Technology, Pasadena, CA, USA, June 1998.
- [2] Leonard Adleman, Molecular computation of solutions to combinatorial problems, *Science* 266 (1994) 1021–1024.
- [3] Ravinderjit Braich, Cliff R. Johnson, Paul W.K. Rothemund, Darryl Hwang, Nickolas Chelyapov, Leonard Adleman, Solution of a satisfiability problem on a gel-based DNA computer, in: *Proceedings of DNA Computing: 6th International Workshop on DNA-Based Computers (DNA00)*, Leiden, The Netherlands, June 2000, pp. 27–38.
- [4] Ravinderjit Braich, Nickolas Chelyapov, Cliff R. Johnson, Paul W.K. Rothemund, Leonard Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer, *Science* 296 (5567) (2002) 499–502.
- [5] Erik Winfree, On the computational power of DNA annealing and ligation, *DNA Based Computers* (1996) 199–221.
- [6] Erik Winfree, Renat Bekbolatov, Proofreading tile sets: Error correction for algorithmic self-assembly, in: *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS02)*, vol. 2943, Madison, WI, USA, June 2003, pp. 126–144.
- [7] Yuliy Baryshnikov, Ed G. Coffman, Nadrian Seeman, Teddy Yimwadsana, Self correcting self assembly: Growth models and the hammersley process, in: *Proceedings of the 11th International Meeting on DNA Computing (DNA05)*, London, Ontario, June 2005.
- [8] Ho-Lin Chen, Ashish Goel, Error free self-assembly with error prone tiles, in: *Proceedings of the 10th International Meeting on DNA Based Computers (DNA04)*, Milan, Italy, June 2004.
- [9] John H. Reif, Sadheer Sahu, Peng Yin, Compact error-resilient computational DNA tiling assemblies, in: *Proceedings of the 10th International Meeting on DNA Based Computers (DNA04)*, Milan, Italy, June 2004.
- [10] Erik Winfree, Self-healing tile sets, *Nanotechnology Science Comput.* (2006) 55–78.
- [11] Robert Barish, Paul W.K. Rothemund, Erik Winfree, Two computational primitives for algorithmic self-assembly: Copying and counting, *Nano Lett.* 5 (12) (2005) 2586–2592.
- [12] Paul W.K. Rothemund, Nick Papadakis, Erik Winfree, Algorithmic self-assembly of DNA Sierpinski triangles, *PLoS Biology* 2 (12) (2004) e424.
- [13] Leonard Adleman, Towards a mathematical theory of self-assembly, Technical Report 00-722, Department of Computer Science, University of Southern California, Los Angeles, CA, 2000.
- [14] Leonard Adleman, Qi Cheng, Ashish Goel, Ming-Deh Huang, Hal Wasserman, Linear self-assemblies: Equilibria, entropy, and convergence rates, in: *Proceedings of the 6th International Conference on Difference Equations and Applications (ICDEA01)*, Augsburg, Germany, June 2001.
- [15] Yuliy Brun, Arithmetic computation in the tile assembly model: Addition and multiplication, *Theoret. Comput. Sci.* 378 (1) (June 2007) 17–31.
- [16] Yuliy Brun, Nondeterministic polynomial time factoring in the tile assembly model, *Theoret. Comput. Sci.* 395 (1) (2008) 3–23.
- [17] Paul W.K. Rothemund, Erik Winfree, The program-size complexity of self-assembled squares, in: *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC00)*, Portland, OR, USA, May 2000, pp. 459–468.
- [18] Leonard Adleman, Qi Cheng, Ashish Goel, Ming-Deh Huang, David Kempe, Pablo Moisset de Espanés, Paul W.K. Rothemund, Combinatorial optimization problems in self-assembly, in: *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC02)*, Montreal, Quebec, Canada, May 2002, pp. 23–32.
- [19] Leonard Adleman, Ashish Goel, Ming-Deh Huang, Pablo Moisset de Espanés, Running time and program size for self-assembled squares, in: *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC02)*, Montreal, Quebec, Canada, May 2002, pp. 740–748.
- [20] Pablo Moisset de Espanés, Computerized exhaustive search for optimal self-assembly counters, in: *Proceedings of the 2nd Foundations of Nanoscience: Self-Assembled Architectures and Devices (FNANO05)*, Snowbird, UT, USA, April 2005, pp. 24–25.
- [21] David Soloveichik, Erik Winfree, Complexity of self-assembled shapes, *SIAM J. Comput.* 36 (6) (2007) 1544–1569.
- [22] Yuliy Brun, Solving NP-complete problems in the tile assembly model, *Theoret. Comput. Sci.* 395 (1) (2008) 31–46.
- [23] Michail G. Lagoudakis, Thomas H. LaBean, 2D DNA self-assembly for satisfiability, in: *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, vol. 54, 1999, pp. 141–154.
- [24] Yuliy Baryshnikov, Ed G. Coffman, Petar Momcilovic, DNA-based computation times, in: *Springer Lecture Notes in Comput. Sci.*, vol. 3384, 2005, pp. 14–23.
- [25] Gagan Aggarwal, Qi Cheng, Michael H. Goldwasser, Ming-Yang Kao, Pablo Moisset de Espanés, Robert T. Schweller, Complexities for generalized models of self-assembly, *SIAM J. Comput.* 34 (6) (2005) 1493–1515.
- [26] Ming-Yang Kao, Robert Schweller, Reducing tile complexity for self-assembly through temperature programming, in: *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA06)*, Miami, FL, USA, January 2006, pp. 571–580.
- [27] Tsu Ju Fu, Nadrian C. Seeman, DNA double-crossover molecules, *Biochemistry* 32 (13) (1993) 3211–3220.

- [28] Paul W.K. Rothmund, Folding DNA to create nanoscale shapes and patterns, *Nature* 440 (March 2006) 297–302.
- [29] Paul W.K. Rothmund, Design of DNA origami, in: *Proceedings of the International Conference on Computer-Aided Design (ICCAD05)*, San Jose, CA, USA, November 2005.
- [30] Paul W.K. Rothmund, Scaffolded DNA origami: From generalized multicrossovers to polygonal networks, *Nanotechnology Science Comput.* (2006) 3–21.
- [31] Matthew Cook, Paul W.K. Rothmund, Erik Winfree, Self-assembled circuit patterns, in: *Proceedings of the 9th International Meeting on DNA Based Computers (DNA03)*, Madison, WI, USA, June 2003, pp. 91–107.
- [32] Erik Winfree, Simulations of computing by self-assembly of DNA, Technical Report CS-TR:1998:22, California Institute of Technology, Pasadena, CA, USA, 1998.
- [33] Hao Wang, Proving theorems by pattern recognition. II, *Bell System Tech. J.* 40 (1961) 1–42.
- [34] Raphael M. Robinson, Undecidability and nonperiodicity for tilings of the plane, *Invent. Math.* 12 (3) (1971) 177–209.
- [35] Leonard Adleman, Jarkko Kari, Lila Kari, Dustin Dale Reishus, On the decidability of self-assembly of infinite ribbons, in: *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS02)*, Ottawa, Ontario, Canada, November 2002, pp. 530–537.
- [36] Michael Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, 1997.
- [37] Yuriy Brun, Manoj Gopalkrishnan, Dustin Dale Reishus, Bilal Shaw, Nickolas Chelyapov, Leonard Adleman, Building blocks for DNA self-assembly, in: *Proceedings of the 1st Foundations of Nanoscience: Self-Assembled Architectures and Devices (FNANO04)*, Snowbird, UT, USA, April 2004, pp. 2–15.
- [38] Nickolas Chelyapov, Yuriy Brun, Manoj Gopalkrishnan, Dustin Dale Reishus, Bilal Shaw, Leonard Adleman, DNA triangles and self-assembled hexagonal tilings, *J. Amer. Chem. Soc. (JACS)* 126 (43) (October 2004) 13924–13925.
- [39] Dustin Dale Reishus, Bilal Shaw, Yuriy Brun, Nickolas Chelyapov, Leonard Adleman, Self-assembly of DNA double-double crossover complexes into high-density, doubly connected, planar structures, *J. Amer. Chem. Soc. (JACS)* 127 (50) (November 2005) 17590–17591.
- [40] Yuriy Brun, Nenad Medvidovic, An architectural style for solving computationally intensive problems on large networks, in: *Proceedings of Software Engineering for Adaptive and Self-Managing Systems (SEAMS07)*, Minneapolis, MN, USA, May 2007.