Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 395 (2008) 31–46
Theoretical Computer Science
www.elsevier.com/locate/tcs

Solving NP-complete problems in the tile assembly model

Yuriy Brun*

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

Received 22 December 2006; received in revised form 26 July 2007; accepted 28 July 2007

Communicated by A. Condon

Abstract

Formalized study of self-assembly has led to the definition of the tile assembly model, a highly distributed parallel model of computation that may be implemented using molecules or a large computer network such as the Internet. Previously, I defined deterministic and nondeterministic computation in the tile assembly model and showed how to add, multiply and factor. Here, I extend the notion of computation to include deciding subsets of the natural numbers, and present a system that decides *SubsetSum*, 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: 49. 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 one.

Keywords: Self-assembly; NP-complete; Tile assembly model; Crystal-growth; Molecular computation; Natural computation; Distributed computing; Parallel computing; Nondeterministic computation; SubsetSum

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 [40]. The field of DNA computation demonstrated that DNA can be used to compute [1], solving NP-complete problems such as the satisfiability problem [12,11]. This idea of using molecules to compute nondeterministically is the driving motivation behind my work.

Winfree showed that DNA computation is Turing-universal [39]. 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 [43,10,22,28,42]. Researchers have used DNA to assemble crystals with patterns of binary counters [8] and

* Tel.: +1 3023545174.
E-mail address: ybrun@usc.edu.

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.07.052

Sierpinski triangles [34], but while those crystals are deterministic, generating nondeterministic crystals may hold the power to solving complex problems quickly.

Two important questions about self-assembling systems that create shapes or compute functions are: "what is a minimal tile set 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 [2]. He answered these questions for n-long linear polymers [3]. Previously, I have extended these questions to apply to systems that compute functions, rather than assemble shapes, deterministically [13] and nondeterministically [17], 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 [35,4,5,24] proceeded to answer the questions: "what is a minimal tile set that can assemble such shapes?" and "what is the assembly time for these systems?". They showed that, for most n, the minimal tile set that assembles $n \times n$ squares is of size $\Theta(\frac{\log n}{\log \log n})$ and the optimal assembly time is $\Theta(n)$ [5]. A key issue related to assembling squares is the assembly of small binary counters, which theoretically can have as few as 7 tile types [24].

Soloveichik et al. 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 [37].

I proposed and studied systems that compute the sums and products of two numbers using the tile assembly model [13]. 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 [17].

Other early attempts at nondeterministic computation include a proposal by Lagoudakis et al. to solve the satisfiability problem [27]. 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 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. began the study of fundamental limits on the time required for a self-assembly system to compute functions [9]. 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. and Kao et al. have shown that changing the temperature of assembly from a constant throughout the assembly process to a discrete function reduces the minimal tile set that can build an $n \times n$ square to a size $\Theta(1)$ tile set [7,26].

Barish et al. have demonstrated DNA implementations of tile systems, one that copies an input and another that counts in binary [8]. Similarly, Rothemund et al. have demonstrated a DNA implementation of a tile system that computes the xor function, resulting in a Sierpinski triangle [34]. These systems grow crystals using double-crossover complexes [25] as tiles. The theoretical underpinnings of these systems are closely related to the work presented here because these systems compute functions.

Rothemund 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 [32,31,33]. 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. have explored using the tile assembly model to implement arbitrary circuits [23]. 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, Rothemund's work with using DNA as a scaffold may be useful in attaching and assembling such components [33].

Some experimental work [1,11] 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 *SubsetSum* problem can be made arbitrarily close to 1 by increasing the number of self-assembling components and seeds in the computation.

A preliminary version of the constructions presented in this paper has appeared in [14], though without the formal proofs and analysis presented here.

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 a tile system that decides *SubsetSum*, 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 [41,40,35] 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 [38]. The model was fully defined by Rothemund and Winfree [35], and the definitions here are similar to those, and identical to the ones in [13], 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. While this definition does not allow tiles to rotate, it is essentially equivalent to a system with rotating tiles.

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** $g : \Sigma \times \Sigma \to \mathbb{N}$, 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 $g(\sigma, \sigma') = 0 \iff \sigma \neq \sigma'$. 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 $A : \mathbb{Z} \times \mathbb{Z} \to T$. 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 A is a configuration, then within system $\mathbb{S}$, a tile 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** $S : \mathbb{Z}^2 \to \Gamma$, if the above conditions are satisfied, one may attach tiles of T to S. Configurations produced by repeated attachments of tiles from T are said to be **produced** by $\mathbb{S}$ on S. If this process terminates, then the configuration achieved when no more attachments are possible is called a **final configuration**. At certain points in time, it may be possible for more than one tile to attach at a given position, or there may be more than one position where a tile can attach. If for all sequences of tile attachments, all possible final configurations are identical, then $\mathbb{S}$ is said to produce a **unique** final configuration on S.

Note that a system may produce a unique final configuration, even though there exist nonunique 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.

Let $\mathbb{S} = \langle T, g, \tau \rangle$, and let S_0 be a seed configuration such that $\mathbb{S}$ produces a unique final configuration F on S_0. Let $W_0 \subseteq 2^{T \times \mathbb{Z}^2}$ be the set of all tile-position pairs $\langle t, (x, y) \rangle$ such that t can attach to S_0 at (x, y). Let S_1 be the configuration produced by adding all the elements of W_0 to S_0 in one time step. Define $W_1, W_2, \ldots$ and $S_2, S_3, \ldots$ similarly. Let n be the smallest natural number such that $S_n \equiv F$. Then n is the **assembly time** of $\mathbb{S}$ on S_0 to produce F.

I allow the codomain of S to be Γ, a set of tiles which may be different from T. The reason is that I will study systems that compute functions using minimal sets T; but the seed, which has to code for the input of the function, may contain more distinct tiles than there are in T. Therefore, I wish to keep the two sets separate. Note that, at any temperature τ, it takes $\Theta(n)$ distinct tiles to assemble an arbitrary n-bit input such that each tile codes for exactly one of the bits.

Winfree showed that the tile assembly model with $\tau = 2$ is Turing-universal [40] by showing that a tile system can simulate Wang tiles [38], which Robinson showed to be universal [30]. Adleman et al. showed that the tile assembly model with $\tau = 1$ is Turing-universal [6].

2. Computation in the tile assembly model

In [13], 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 [17], 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.

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 $\mathbb{N}^n \to \mathbb{N}$.

Let $v : \Gamma \cup T \to \{0, 1\}$ 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 $o_{s_m} : \mathbb{N} \to \mathbb{Z}^2$ be injections. Let the seed encoding functions $e_{s_m} : \Delta \to \mathbb{N}$ 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 $\vec{a} = \langle a_0, a_1, \ldots, a_{\hat{m}-1} \rangle \in \mathbb{N}^{\hat{m}}$ 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 ath 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 $\mathbb{N}^m \to \mathbb{N}$ and using the intuitive direct binary encoding of that element of $\mathbb{N}$ on the Turning 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 [40]. 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 *SubsetSum* in the tile assembly model

SubsetSum is a well known NP-complete problem. The set *SubsetSum* is a set of pairs: a finite sequence $\vec{B} = \langle B_1, B_2, \ldots, B_n \rangle \in \mathbb{N}^n$, and a target number $v \in \mathbb{N}$, such that $\langle \vec{B}, v \rangle \in \mathit{SubsetSum}$ iff $\exists \vec{c} = \langle c_1, c_2, \ldots, c_n \rangle \in \{0, 1\}^n$ such that $\sum_{i=1}^{n} c_i B_i = v$. In other words, the sum of some subset of numbers of $\vec{B}$ equals exactly v.

In order to explain the system that nondeterministically decides *SubsetSum*, I will first define three smaller systems that perform pieces of the necessary computation. The first system subtracts numbers, and given the right conditions, will subtract a B_i from v. The second system computes the identity function and just copies information (this system will be used when a B_i should not be subtracted from v). The third system nondeterministically guesses whether the next B_i should or should not be subtracted. Finally, I will add a few other tiles that ensure that the computations went as planned and attach an identifier tile if the execution found that $\langle \vec{B}, v \rangle \in \mathit{SubsetSum}$. The system works by nondeterministically choosing a subset of $\vec{B}$ to subtract from v and comparing the result to 0.

In some proofs, I will refer to the unique final configuration corollary (corollary 2.1 from [13]), which states that if all the tiles in a system have unique east–south binding domain pairs, then on some seed configurations, that system always produces a unique final configuration, and to the assembly time corollary (corollary 2.2 from [13]), which states that the final configuration produced by such systems assembles in time linear in the largest dimension of the seed.

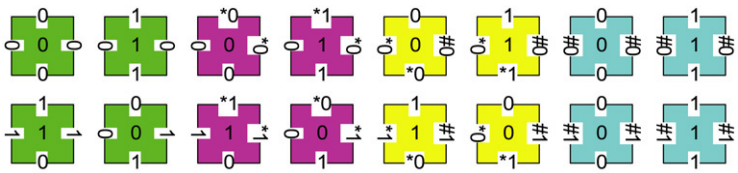
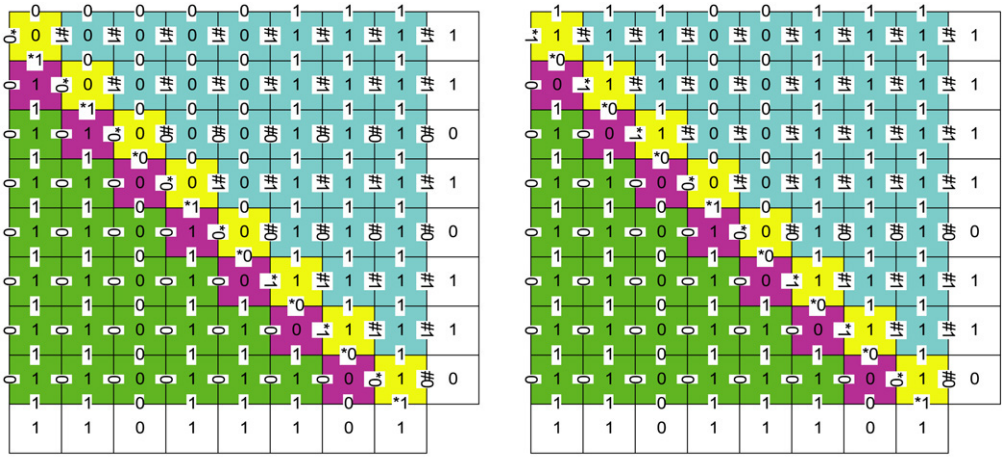


Fig. 1. There are 16 tiles in T_-. The value in the middle of each tile t represents that tile's $v(t)$ value. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) (b)

Fig. 2. An example of $\mathbb{S}_-$ subtracting numbers. In (a), the system subtracts $214 = 11010110_2$ from $221 = 11011101_2$ to get $7 = 111_2$. The inputs are encoded along the bottom row ($221 = 11011101_2$) and rightmost column ($214 = 11010110_2$). The output is on the top row ($7 = 00000111_2$). Note that because $214 \leq 221$, all the west binding domains of the leftmost column contain a 0. In (b), the system attempts to subtract $246 = 11110110$ from $221 = 11011101_2$, but because $246 > 221$, the computation fails, and indicates its failure with the topmost and leftmost west binding domain containing a 1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Whenever considering a number $\alpha \in \mathbb{N}$, I will refer to the size of α, in bits, as n_α. I will further refer to the ith bit of α as α_i; that is, for all $i \in \mathbb{N}$, $\alpha_i \in \{0, 1\}$ such that $\sum_i \alpha_i 2^i = \alpha$. The least significant bit of α is α_0.

3.1. Subtraction

In this section, I will describe a system that subtracts positive integers. It is similar to one of the addition systems from [13], contains 16 tiles, and will subtract one bit per row of computation.

Fig. 1 shows the 16 tiles of T_-. The value in the middle of each tile t represents that tile's $v(t)$ value. Intuitively, the system will subtract the ith bit on the ith row. The tiles to the right of the ith location will be blue; the tile in the ith location will be yellow; the next tile, the one in the $(i+1)$st location, will be magenta and the rest of the tiles will be green. The purpose of the yellow and magenta tiles is to compute the diagonal line, marking the ith position on the ith row. Fig. 2 shows two sample executions of the subtracting system.

Lemma 3.1. *Let $\Sigma_- = \{0, 1, \star 0, \star 1, \#0, \#1\}$, let T_- be the set of tiles defined by Fig. 1, let $g_- = 1$, let $\tau_- = 2$, and let $\mathbb{S}_- = \langle T_-, g_-, \tau_- \rangle$. Let $\alpha, \beta \in \mathbb{N}$ and let $\delta = \alpha - \beta$. Let S_- be a seed configuration such that there exists some $(x_0, y_0) \in \mathbb{Z}^2$ such that:*

- $bd_N(S_-(x_0 - 1, y_0)) = \star\alpha_0$.
- *For all* $i \in \{1, 2, \ldots, n_\alpha - 1\}$, $bd_N(S_-(x_0 - i - 1, y_0)) = \alpha_i$.
- *For all* $j \in \{0, 1, \ldots, n_\beta - 1\}$, $bd_W(S_-(x_0, y_0 + j + 1)) = \#\beta_j$.
- *For all other positions* (x, y), $(x, y) \notin S_-$.

Then $\mathbb{S}_-$ produces a unique final configuration F_- on S_- such that
$\alpha \geq \beta \implies$

- *For all $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(F_-(x_0 - i - 1, y_0 + n_\beta)) \in \{\delta_i, {}^\star\delta_i\}$.*
- *For all $j \in \{0, 1, \ldots, n_\beta - 1\}$, $bd_W(F_-(x_0 - n_\alpha, y_0 + j + 1)) \in \{0, {}^\star 0\}$.*

and $\alpha < \beta \implies$

- *There exists $j \in \{0, 1, \ldots, n_\beta - 1\}$ such that $bd_W(F_-(x_0 - n_\alpha, y_0 + j + 1)) \in \{1, {}^\star 1\}$.*

Proof. By the unique final configuration corollary (corollary 2.1 from [13]), $\mathbb{S}_-$ produces a unique final configuration on S_-. Call that configuration F_-.

To simplify notation, given x_0, y_0, for all $i, j \in \mathbb{Z}$, let $p(i, j) = (x_0 - i - 1, y_0 + j + 1)$. The purpose of this notation is to more easily identify data in the construction. I expect the north binding domain of the tile in position $p(i, j)$ to code for the ith bit of ${}_j\delta$ (see definition of ${}_j\delta$ below), also denoted ${}_j\delta_i$. Further, I will refer to row $y_0 + j + 1$ as row $p(\ldots, j)$, and column $x_0 - i - 1$ as column $p(i, \ldots)$. Thus, it is sufficient to show F_- is such that:
$\alpha \geq \beta \implies$:

- For all $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(F_-(p(i, n_\beta - 1))) \in \{\delta_i, {}^\star\delta_i\}$;
- For all $j \in \{0, 1, \ldots, n_\beta - 1\}$, $bd_W(F_-(p(n_\alpha - 1, j))) \in \{0, {}^\star 0\}$.

and $\alpha < \beta \implies$:

- There exists $j \in \{0, 1, \ldots, n_\beta - 1\}$ such that $bd_W(F_-(p(n_\alpha - 1, j))) \in \{1, {}^\star 1\}$.

For all $j \in \{-1, 0, 1, \ldots, n_\beta - 1\}$, let ${}_j\delta = \alpha - \sum_{k=0}^{j} \beta_k 2^k$. (That is, ${}_j\delta$ is the difference between α and the number formed by the $j+1$ least significant bits of β.) I show, by induction on j, that for all $i \in \{0, 1, \ldots, j, j+2, \ldots, n_\alpha - 1\}$, $bd_N(F_-(p(i, j))) = {}_j\delta_i$ and $bd_N(F_-(p(j + 1, j))) = {}^\star{}_j\delta_{j+1}$.

First, note some properties of ${}_j\delta$: ${}_{-1}\delta = \alpha$, ${}_{n_\beta - 1}\delta = \alpha - \beta = \delta$, and for all $i \in \{0, 1, \ldots, j\}$, ${}_j\delta_i = {}_{j+1}\delta_i$.

Base case: ($j = -1$). Thus ${}_j\delta = {}_{-1}\delta = \alpha$. S_- and F_- must agree everywhere S_- is not *empty*, and by definition of S_-, $bd_N(S_-(p(0, -1))) = {}^\star\alpha_0$ and for all $i \in \{1, 2, \ldots, n_\alpha - 1\}$, $bd_N(S_(- p(i, -1))) = \alpha_i$. Thus, for $j = -1$, for all $i \in \{1, 2, \ldots, n_\alpha - 1\}$, $bd_N(F_-(p(i, j))) = {}_j\delta_i$ and $bd_N(F_-(p(j + 1, j))) = {}^\star{}_j\delta_{j+1}$.

Inductive step: I assume that for all $i \in \{0, 1, \ldots, j, j+2, \ldots, n_\alpha - 1\}$, $bd_N(F_-(p(i, j))) = {}_j\delta_i$ and $bd_N(F_-(j + 1, j)) = {}^\star{}_j\delta_{j+1}$. I will show that for all $i \in \{0, 1, \ldots, j + 1, j + 3, \ldots, n_\alpha - 1\}$, $bd_N(F_-(p(i, j + 1))) = {}_{j+1}\delta_i$ and $bd_N(F_-(p(j + 2, j + 1))) = {}^\star{}_{j+1}\delta_{j+2}$.

Consider row $p(\ldots, j+1)$. Consider the tile t that attaches in position $p(0, j+1)$ (assume for a second that $j \neq 0$). By the definition of S_-, $bd_W(F_-(p(-1, j+1))) = \#\beta_{j+1}$ and by the inductive hypothesis, $bd_N(F_-(p(0, j))) = {}_j\delta_0$. Thus t's east binding domain must be either #0 or #1 and south binding domain must be either 0 or 1, so t must be a blue tile. Therefore, $bd_W(t) = bd_E(t) = \#\beta_{j+1}$ and $bd_N(t) = bd_S(t) = {}_j\delta_0$. Note that this argument holds for the tile to the west of position $p(0, j+1)$, as long as the north binding domain of the tile below does not start with *, and so on. By the inductive hypothesis, for all $i \in \{0, 1, \ldots, j\}$, $bd_N(F_-(p(i, j))) = {}_j\delta_i$ and thus those binding domains do not start with *. Thus, for all $i \in \{0, 1, \ldots, j\}$, $bd_W(F_-(p(i, j+1))) = \#\beta_{j+1}$ and $bd_N(F_-(p(i, j+1))) = {}_j\delta_i = {}_{j+1}\delta_i$. (Note that for $j = 0$, no blue tiles attach so the earlier assumption is justified. For $j = 0$ the argument starts here.)

Observe that for all nonblue tiles (yellow, magenta and green), looking only at the numerical value of the binding domains (ignoring the preceding * and #) the north binding domain is the result of subtracting the east binding domain from the south binding domain, modulo 2, and the west binding domain is 1 iff the east binding domain is greater than the south binding domain.

Consider the tile t that attaches in position $p(j + 1, j + 1)$. I just showed that $bd_E(t)$ must be $\#\beta_{j+1}$. By the inductive hypothesis, $bd_N(F_-(p(j+1, j))) = {}^\star{}_j\delta_{j+1}$, so t must be yellow. From above, a yellow tile's north binding domain is the difference of its south and east binding domains, and its west binding domain is 1 iff the east binding domain is greater than the south binding domain. Thus $bd_N(t)$ is ${}_j\delta_{j+1} - \beta_{j+1} = {}_{j+1}\delta_{j+1}$ and $bd_W(t) = {}^\star 1$ if ${}_j\delta_{j+1} < \beta_{j+1}$, thus "borrowing a 1" and $bd_W(t) = {}^\star 0$ otherwise.

Consider the tile t that attaches in position $p(j + 2, j + 1)$. I just showed that $bd_E(t)$ must be ${}^\star 0$ if there is no need to borrow a 1 and ${}^\star 1$ otherwise. By the inductive hypothesis, $bd_N(F_-(p(j + 2, j))) = {}_j\delta_{j+2}$, so t must be magenta. Thus $bd_N(t)$ is ${}^\star{}_{j+1}\delta_{j+2}$ and $bd_W(t) = 1$ if there is a need to borrow a 1 again, and $bd_W(t) = 0$ otherwise.

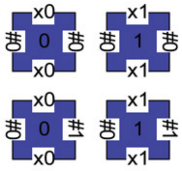


Fig. 3. There are 4 tiles in T_{x}. The value in the middle of each tile t represents that tile's $v(t)$ value.

Consider the tile t that attaches in position $p(j+3, j+1)$. I just showed that $bd_E(t)$ must be 0 if there is no need to borrow a 1 and 1 otherwise. By the inductive hypothesis, $bd_N(F_-(p(j+3, j))) = {}_j\delta_{j+3}$, so t must be green. Thus $bd_N(t)$ is ${}_{j+1}\delta_{j+3}$ and $bd_W(t) = 1$ if there is a need to borrow a 1 again, and $bd_W(t) = 0$ otherwise. The same holds for the tile to west of position $p(j+3, j+1)$, and so on, until position $p(n_\alpha - 1, j+1)$, thus green tiles attach. Also, $bd_W(F_-(p(n_\alpha - 1, j+1))) = 1$ if the number being subtracted exceeds ${}_{j+1}\delta$ and $bd_W(F_-(p(n_\alpha - 1, j+1))) = 0$ otherwise.

Thus, in row $p(\ldots, j+1)$, for all $i \in \{0, 1, \ldots, j+1, j+3, \ldots, n_\alpha - 1\}$, $bd_N(F_-(p(i, j+1))) = {}_{j+1}\delta_i$ and $bd_N(F_-(p(j+2, j+1))) = {}^\star{}_{j+1}\delta_{j+2}$. Also, $bd_W(F_-(p(n_\alpha - 1, j+1))) \in \{1, {}^\star 1\}$ iff the number being subtracted exceeds ${}_{j+1}\delta$.

Therefore, $\alpha \geq \beta \implies$

- Let $j = n_\beta - 1$. Since ${}_{n_\beta - 1}\delta = \delta$, in row $p(\ldots, n_\beta - 1)$, for all $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(F_-(p(i, n_\beta - 1))) \in \{\delta_i, {}^\star\delta_i\}$.
- For all $j \in \{0, 1, \ldots, n_\beta - 1\}$, $bd_W(F_-(p(n_\alpha - 1, j))) \in \{0, {}^\star 0\}$.

and, $\alpha < \beta \implies$

- There exists $j \in \{0, 1, \ldots, n_\beta - 1\}$ such that $bd_W(F_-(p(n_\alpha - 1, j))) \in \{1, {}^\star 1\}$. □

Thus $\mathbb{S}_-$ is a system that is capable of subtracting numbers. Formally, using the definition of a tile assembly model computing a function from [13]:

Theorem 3.1. *Let $f : \mathbb{N}^2 \to \mathbb{N}$ be such that for all $\alpha, \beta \in \mathbb{N}$ such that $\alpha \geq \beta$, $f(\alpha, \beta) = \alpha - \beta$ and for all other α, β, $f(\alpha, \beta)$ is undefined. Then $\mathbb{S}_-$ computes the function f.*

Proof. This theorem follows from Lemma 3.1, and the fact that for all $t \in T_-$, $v(t) = bd_N(t)$. □

Lemma 3.2. *The assembly time of $\mathbb{S}_-$ is $\Theta(n)$ steps to subtract an n-bit number from another n-bit number.*

Proof. This lemma follows directly from the assembly time corollary (corollary 2.2 from [13]). □

Fig. 2 shows sample executions of $\mathbb{S}_-$. In Fig. 2(a), the system subtracts $214 = 11010110_2$ from $221 = 11011101_2$ to get $7 = 111_2$. The inputs are encoded along the bottom row ($221 = 11011101_2$) and rightmost column ($214 = 11010110_2$). The output is on the top row ($7 = 00000111_2$). Note that because $214 \leq 221$, all the west binding domains of the leftmost column contain a 0. In Fig. 2(b), the system attempts to subtract $246 = 11110110$ from $221 = 11011101_2$, but because $246 > 221$, the computation fails, and indicates its failure because the topmost leftmost west binding domain contains a 1.

This system is very similar to an adding system from [13], but not the smallest adding system from [13]. While this system has 16 tiles, it is possible to design a subtracting system with 8 tiles, that is similar to the 8-tile adding system from [13].

3.2. Identity

I now describe a system that ignores the input on the rightmost column, and simply copies upwards the input from the bottom row. This is a fairly straightforward system that will not need much explanation.

Lemma 3.3. *Let $\Sigma_{\mathrm{x}} = \{\mathrm{x0}, \mathrm{x1}, \#0, \#1\}$, let T_{x} be the set of tiles defined by Fig. 3, let $g_{\mathrm{x}} = 1$, let $\tau_{\mathrm{x}} = 2$, and let $\mathbb{S}_{\mathrm{x}} = \langle T_{\mathrm{x}}, g_{\mathrm{x}}, \tau_{\mathrm{x}}\rangle$. Let $\alpha, \beta \in \mathbb{N}$. Let S_{x} be a seed configuration such that there exists some $(x_0, y_0) \in \mathbb{Z}^2$ such that:*

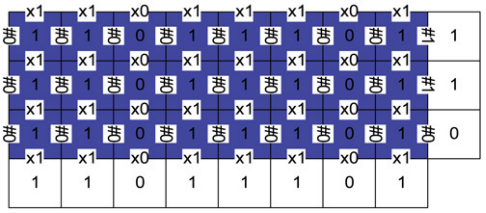


Fig. 4. An example of an $\mathbb{S}_{\mathrm{x}}$ execution. The system simply copies the input on the bottom row upwards, to the top column.

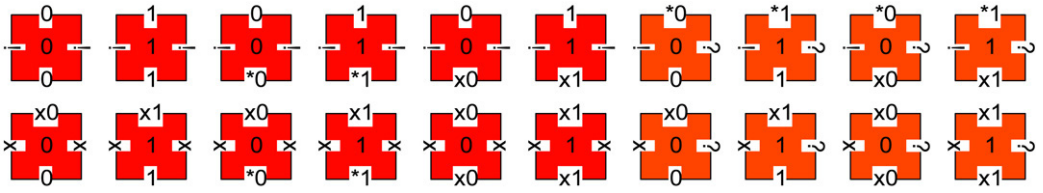


Fig. 5. The are 20 tiles in $T_?$. The value in the middle of each tile t represents that tile's $v(t)$ value. Unlike the red tiles, the orange tiles do not have unique east–south binding domain pairs, and thus will attach nondeterministically. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- *For all* $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(S_{\mathrm{x}}(x_0 - i - 1, y_0)) = \mathrm{x}\alpha_i$.
- *For all* $j \in \{0, 1, \ldots, n_\beta - 1\}$, $bd_W(S_{\mathrm{x}}(x_0, y_0 + j + 1)) = \#\beta_j$.
- *For all other positions* (x, y), $(x, y) \notin S_{\mathrm{x}}$.

Then $\mathbb{S}_{\mathrm{x}}$ *produces a unique final configuration* F_{x} *on* S_{x}, *and for all* $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(F_{\mathrm{x}}(x_0 - i - 1, y_0 + n_\beta)) = \mathrm{x}\alpha_i$ *and for all* $j \in \{0, 1, \ldots, n_\beta - 1\}$, $bd_W(F_{-}(x_0 - n_\alpha, y_0 + j + 1)) = \mathrm{x0}$.

Proof. By the unique final configuration corollary (corollary 2.1 from [13]), $\mathbb{S}_{\mathrm{x}}$ produces a unique final configuration on S_{x}. Call that configuration F_{x}. It is clear that the configuration will fill the rectangle outlined by the seed, with the exception of the bottom right corner, because for every possible pair of west-north binding domains of the tiles in T_{x} and in S_{x}, there is a tile with a matching east–south binding domain.

Every tile $t \in T_{\mathrm{x}}$ has $bd_S(t) = bd_N(t)$ so for all $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(F_{\mathrm{x}}(x_0 - i - 1, y_0 + n_\beta)) = \mathrm{x}\alpha_i$. Every tile $t \in T_{\mathrm{x}}$ has $bd_W(t) = \mathrm{x0}$ so for all $j \in \{0, 1, \ldots, n_\beta - 1\}$, $bd_W(F_{-}(x_0 - n_\alpha, y_0 + j + 1)) = \mathrm{x0}$. □

Lemma 3.4. *The assembly time of* $\mathbb{S}_{\mathrm{x}}$ *is* $\Theta(n_\alpha + n_\beta)$.

Proof. This lemma follows directly from the assembly time corollary (corollary 2.2 from [13]). □

Fig. 4 shows a sample execution of the $\mathbb{S}_{\mathrm{x}}$ system. The system simply copies the input on the bottom row upwards, to the top column.

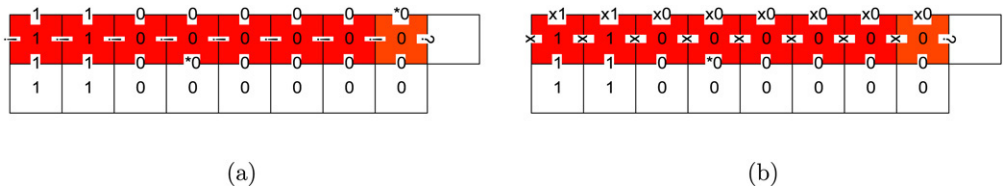
3.3. Nondeterministic guess

In this section, I describe a system that nondeterministically decides whether or not the next B_i should be subtracted from v. It does so by encoding the input for either the $\mathbb{S}_{-}$ system or the $\mathbb{S}_{\mathrm{x}}$ system.

Lemma 3.5. *Let* $\Sigma_? = \{?, !, 0, 1, \mathrm{x0}, \mathrm{x1}, {}^\star 0, {}^\star 1\}$, *let* $T_?$ *be the set of tiles defined by Fig. 5, let* $g_? = 1$, *let* $\tau_? = 2$, *and let* $\mathbb{S}_? = \langle T_?, g_?, \tau_? \rangle$. *Let* $\alpha \in \mathbb{N}$. *Let* $S_?$ *be a seed configuration such that there exists some* $(x_0, y_0) \in \mathbb{Z}^2$ *such that:*

- $bd_N(S_?(x_0 - 1, y_0)) \in \{\alpha_i, \mathrm{x}\alpha_i\}$;
- *For all* $i \in \{1, 2, \ldots, n_\alpha - 1\}$, $bd_N(S_?(x_0 - i - 1, y_0)) \in \{\alpha_i, {}^\star\alpha_i, \mathrm{x}\alpha_i\}$;
- $bd_W(S_?(x_0, y_0 + 1)) = ?$;
- *For all other positions* (x, y), $(x, y) \notin S_?$.

Then $\mathbb{S}_?$ *produces one of two final configurations* $F_?$ *on* $S_?$. *Either,*



(a) (b)

Fig. 6. Two examples of $\mathbb{S}_?$ executions. In (a), the system attaches tiles with ! east-west binding domains, preparing a valid seed for $\mathbb{S}_-$, and in (b), the system attaches tiles with x east-west binding domains, preparing a valid seed for $\mathbb{S}_\mathrm{x}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

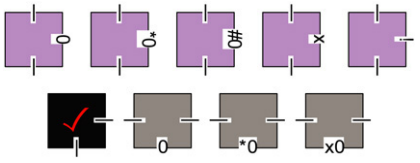


Fig. 7. There are 9 tiles in $T_\checkmark$. The black tile with a ✓ in the middle will serve as the identifier tile. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- *for all* $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(F_?(x_0 - i - 1, y_0 + 1)) = \mathrm{x}\alpha_i$ *and* $bd_W(F_?(x_0 - n_\alpha, y_0 + 1)) = \mathrm{x}$, *or*
- $bd_N(F_?(x_0 - 1, y_0 + 1)) = {}^\star\alpha_i$, *and for all* $i \in \{1, 2, \ldots, n_\alpha - 1\}$, $bd_N(F_?(x_0 - i - 1, y_0 + 1)) = \alpha_i$, *and* $bd_W(F_?(x_0 - n_\alpha, y_0 + 1)) = !$.

Proof. Because $bd_W(S_\mathrm{x}(x_0, y_0 + 1)) = ?$, only an orange tile may attach in position $(x_0 - 1, y_0 + 1)$. Suppose an orange tile t attaches such that $bd_W(t) = \mathrm{x}$. Then $bd_S(t) \in \{\alpha_0, \mathrm{x}\alpha_0\} \implies bd_N(t) = \mathrm{x}\alpha_0$. Further, to the west of that position, only red tiles with west binding domain x can attach, and for all of those, since their south binding domains $\in \{\alpha_i, {}^\star\alpha_i, \mathrm{x}\alpha_i\}$, their north binding domains must be $\mathrm{x}\alpha_i$. Thus, for all $i \in \{0, 1, \ldots, n_\alpha - 1\}$, $bd_N(F_?(x_0 - i - 1, y_0 + 1)) = \mathrm{x}\alpha_i$ and $bd_W(F_?(x_0 - n_\alpha, y_0 + 1)) = \mathrm{x}$.

Now suppose an orange tile t attaches at position $(x_0 - 1, y_0 + 1)$ such that $bd_W(t) = !$. Then $bd_S(t) \in \{\alpha_0, \mathrm{x}\alpha_0\} \implies bd_N(t) = {}^\star\alpha_0$. Further, to the west of that position, only red tiles with west binding domain ! can attach, and for all of those, since their south binding domains $\in \{\alpha_i, {}^\star\alpha_i, \mathrm{x}\alpha_i\}$, their north binding domains must be α_i. Thus, $bd_N(F_?(x_0 - 1, y_0 + 1)) = {}^\star\alpha_i$, and for all $i \in \{1, 2, \ldots, n_\alpha - 1\}$, $bd_N(F_?(x_0 - i - 1, y_0 + 1)) = \alpha_i$, and $bd_W(F_?(x_0 - n_\alpha, y_0 + 1)) = !$. □

Lemma 3.6. *The assembly time of* $\mathbb{S}_?$ *is* $\Theta(n_\alpha)$.

Proof. This lemma follows directly from the assembly time corollary (corollary 2.2 from [13]). □

Fig. 6 shows two possible executions of $\mathbb{S}_?$. In Fig. 6(a), the system attaches tiles with ! east-west binding domains, preparing a valid seed for $\mathbb{S}_-$, and in Fig. 6(b), the system attaches tiles with x east-west binding domains, preparing a valid seed for $\mathbb{S}_\mathrm{x}$. Only one tile, the orange tile, attaches nondeterministically, determining which tiles attach to its west.

3.4. Deciding SubsetSum

I have described three systems that I will now use to design a system to decide *SubsetSum*. Intuitively, I plan to write out the elements of $\vec{B}$ on a column and v on a row, and the system will nondeterministically choose some of the elements from $\vec{B}$ to subtract from v. The system will then check to make sure that no subtracted element was larger than the number it was being subtracted from, and whether the result is 0. If the result is 0, then a special identifier tile will attach to signify that $\langle \vec{B}, v \rangle \in \mathit{SubsetSum}$.

Theorem 3.2. *Let* $\Sigma_{SS} = \Sigma_- \cup \Sigma_\mathrm{x} \cup \Sigma_? \cup \{|\}$. *Let* $T_{SS} = T_- \cup T_\mathrm{x} \cup T_? \cup T_\checkmark$, *where* $T_\checkmark$ *is defined by Fig. 7. Let* $g_{SS} = 1$ *and* $\tau_{SS} = 2$. *Let* $\mathbb{S}_{SS} = \langle T_{SS}, g_{SS}, \tau_{SS} \rangle$. *Then* $\mathbb{S}_{SS}$ *nondeterministically decides SubsetSum with the black* ✓ *tile from* $T_\checkmark$ *as the identifier tile.*

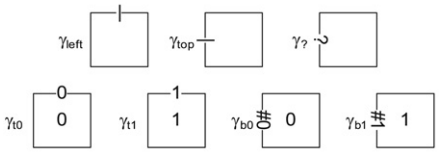


Fig. 8. There are 7 tiles in Γ_{SS}. The value in the middle of each tile t represents that tile's $v(t)$ value and each tile's name is written on its left.

Proof. Let $n \in \mathbb{N}$, let $\vec{B} = \langle B_1, B_2, \ldots, B_n \rangle \in \mathbb{N}^n$, and let $v \in \mathbb{N}$. Let Γ_{SS} be as defined by Fig. 8. Let the seed S_{SS} be as follows (see Fig. 9(a) for an example of a valid seed):

- For all $i \in \{0, 1, \ldots, n_v - 1\}$, $S_{SS}(-i, 0) = \gamma_{tv_i}$.
- $S_{SS}(-n_v, 0) = \gamma_{\text{left}}$.
- For all $k \in \{1, 2, \ldots n\}$, $S_{SS}(1, 1 + \sum_{j=1}^{k-1}(n_{B_j} + 1)) = \gamma_?$.
- For all $k \in \{1, 2, \ldots n\}$, for all $i \in \{0, 1, \ldots, n_{B_k} - 1\}$, $S_{SS}\left(1, 2 + i + \sum_{j=1}^{k-1}\left(n_{B_j} + 1\right)\right) = \gamma_{b(B_k)_i}$.
- $S_{SS}(1, 1 + \sum_{j=1}^{n}(n_{B_j} + 1)) = \gamma_{\text{top}}$.
- And for all other positions (x, y), $(x, y) \notin S_{SS}$.

Because T_-, T_{x}, $T_?$, and $T_\checkmark$ have disjoint sets of south-east binding domain pairs, and because a tile attaches to the assembly only when its south and east binding domains match (as described in the step configuration lemma (lemma 2.1 from [13])), only one of those sets contains tiles that can attach at each position.

$S_{SS}(1, 1) = \gamma_?$, so tiles from $T_?$ may attach in position $(0, 1)$. By Lemma 3.5, one of two things will happen on row 1: either tiles with east-west binding domains ! will attach, or tiles with east-west binding domains x will attach. Thus one nondeterministic sequence of attachments (case 1) will result in the north binding domains of the tiles in row 1 encoding the bits of v, and the other nondeterministic sequence of attachments (case 2) will result in the north binding domains of the tiles in row 1 encoding bits of v with x preceding every bit. In case 1, tiles from T_- will attach, and by Lemma 3.1, these tiles will attach deterministically to encode the bits of $v - B_1$ in the north binding domains of row $n_{B_1} + 1$. In case 2, tiles from T_{x} will attach, and by Lemma 3.3, these tiles will attach deterministically to encode the bits of v in the north binding domains of row $n_{B_1} + 1$.

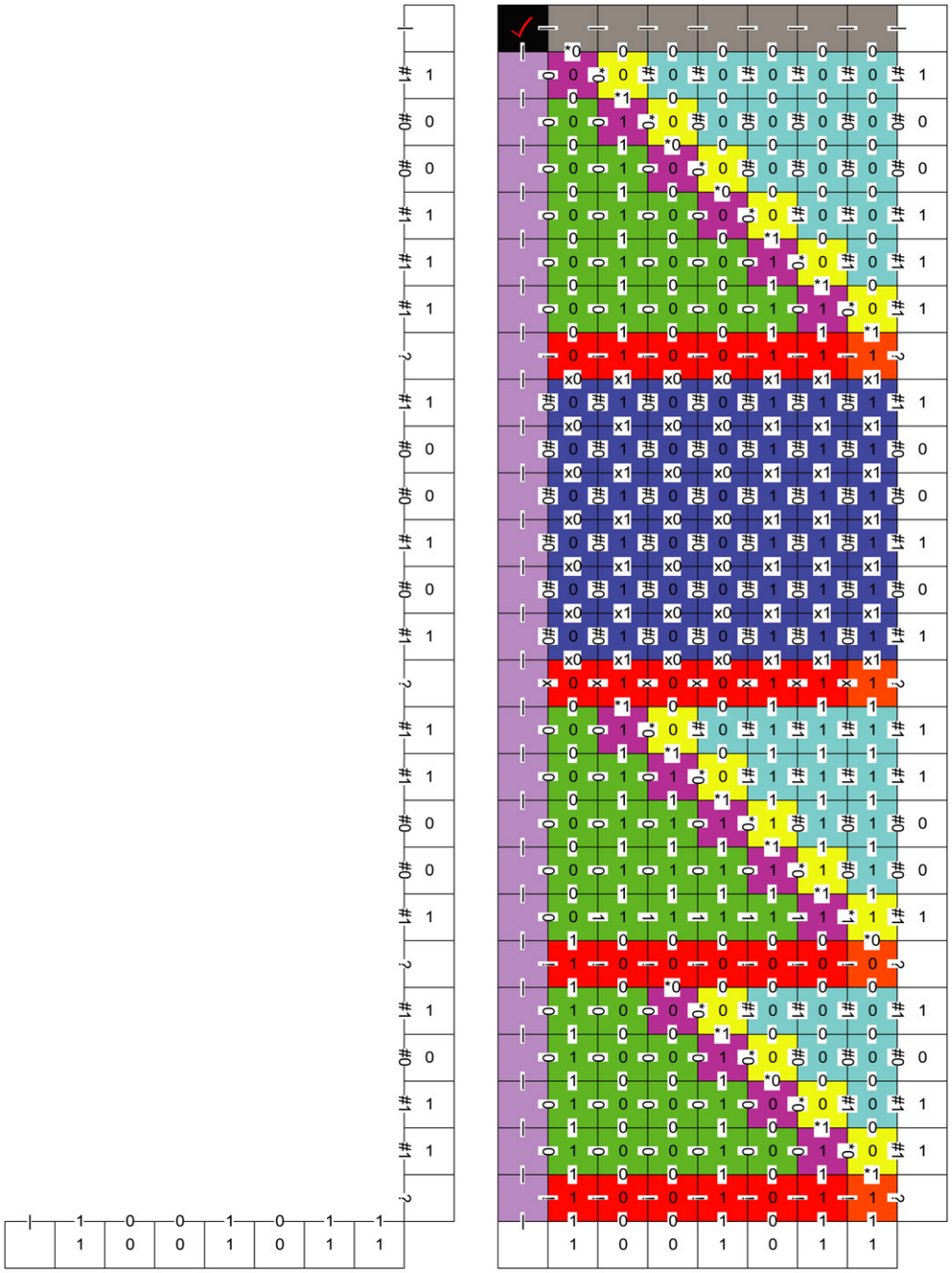
Note that $S_{SS}(1, n_{B_1} + 2) = \gamma_?$ so the process can repeat with B_2, and so on. Thus for each nondeterministic sequence of attachments, in the final configuration, the north binding domains of the tiles in row $\sum_{j=1}^{n}(n_{B_j} + 1)$ encode $v - \sum_{j=1}^{n} c_j B_j$, where $\vec{c} = \langle c_1, c_2, \ldots, c_n \rangle \in \{0, 1\}^n$ and there exists a nondeterministic execution for each of the 2^n possible assignments of $\vec{c}$.

Only lavender tiles can attach in column $-n_v$ because their north-south binding domains are | and thus they can only attach to each other or north of γ_{left}, which only occurs in position $(-n_v, 0)$. By Lemma 3.1, one of the B_j being subtracted is greater than the number it is being subtracted from iff there exists a q such that the tile t that attaches at position $(-n_v - 1, q)$ has $bd_W(t) \in \{1, \star 1\}$. In column $-n_v$, lavender tiles from $T_\checkmark$ attach only to tiles with west binding domains x, !, or that contain a 0. Thus a tile attaches in position $(-n_v, \sum_{j=1}^{n}(n_{B_j} + 1))$ iff no subtracted numbers have exceeded the number they were subtracted from. Since that tile must be lavender, its north binding domain is |.

The black ✓ tile may only attach at position $(-n_v, 1 + \sum_{j=1}^{n}(n_{B_j} + 1))$ because that is the only position with a tile to its south that may have the | north binding domain and the tile to its east that may have the | west binding domain. The gray tiles in $T_\checkmark$ attach in row $1 + \sum_{j=1}^{n}\left(n_{B_j} + 1\right)$, starting from column 0, then 1, etc. iff all the north binding domains of the row $\sum_{j=1}^{n}(n_{B_j} + 1)$ contain 0. Thus a tile can attach in position $(-n_v + 1, 1 + \sum_{j=1}^{n}(n_{B_j}) + 1)$ only if there exists a choice of $\vec{c} \in \{0, 1\}^n$ such that $v - \sum_{j=1}^{n} c_j B_j = 0$.

Therefore, the black ✓ tile attaches iff there exists a choice of $\vec{c} \in \{0, 1\}^n$ such that $v = \sum_{j=1}^{n} c_j B_j$, or in other words, $\langle \vec{B}, v \rangle \in \textit{SubsetSum}$. Therefore, $\mathbb{S}_{SS}$ nondeterministically decides *SubsetSum*. □

Fig. 9 shows an example execution of $\mathbb{S}_{SS}$. Fig. 9(a) encodes a seed configuration with $v = 75 = 1001011_2$ along the bottom row and $\vec{B} = \langle 11 = 1011_2, 25 = 11001_2, 37 = 100101 + 2, 39 = 100111_2 \rangle$ along the rightmost column. Tiles from T_{SS} attach to the seed configuration, nondeterministically testing all possible values of $\vec{c} \in \{0, 1\}^4$. Fig. 9(b)



(a) (b)

Fig. 9. An example of $\mathbb{S}_{SS}$ solving a *SubsetSum* problem. Here, $v = 75 = 1001011_2$, and $\vec{B} = \langle 11 = 1011_2, 25 = 11001_2, 37 = 100101_2, 39 = 100111_2\rangle$. The seed configuration encodes v on the bottom row and $\vec{B}$ on the rightmost column (a). The fact that $75 = 11 + 25 + 39$ implies that $\langle \vec{B}, t\rangle \in$ *SubsetSum*, thus at least one final configuration (b) contains the ✓ tile.

shows one such possible execution, the one that corresponds to $\vec{c} = \langle 1, 1, 0, 1\rangle$. Because $11 + 25 + 39 = 75$, the ✓ tile attaches in the top left corner.

I have described configurations that code for the correct $\vec{c}$ to allow the ✓ tile to attach. It is also interesting to see what happens if improper nondeterministic choices of $\vec{c}$ are made. Fig. 10(a) shows a final configuration in which

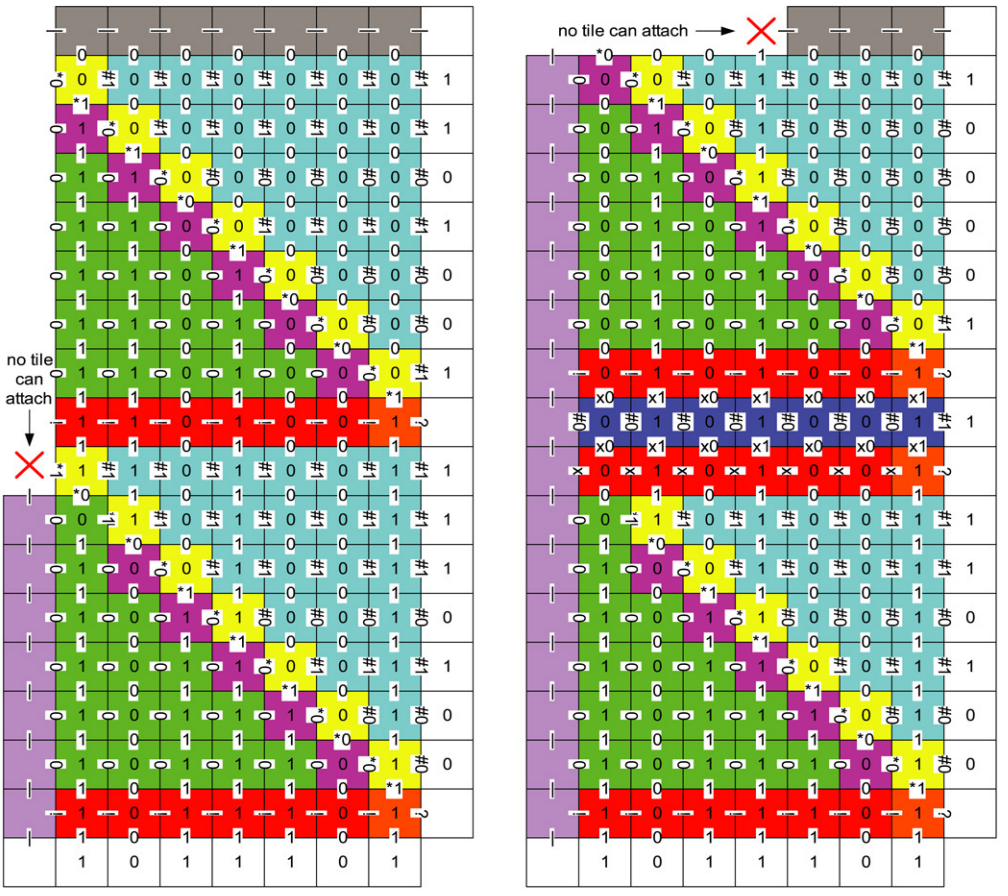


Fig. 10. If the execution of $\mathbb{S}_{SS}$ does not prove membership in *SubsetSum*, the ✓ tile does not attach. If one or more of the B_i values is bigger than the number it is subtracted from, the leftmost column of tiles does not complete and the ✓ tile cannot attach (a). Similarly, if final result of subtracting some B_i values does not equal 0, the top row does not complete and the ✓ tile does not attach (b). Both these configurations are final configurations and no more tiles can attach.

one of the B_i values being subtracted is bigger than v. The leftmost column of tiles does not complete and the ✓ tile cannot attach. Fig. 10(b) shows a final configuration of an execution that never tries to subtract a number that is too big, but the result does not equal 0. Thus the top row does not complete and the ✓ tile does not attach. Both these configurations are final, and no more tiles can attach.

Lemma 3.7. *The assembly time of $\mathbb{S}_{SS}$ is linear in the size of the input (number of bits in $\langle \vec{B}, v \rangle$).*

Proof. This lemma follows from Lemmas 3.2, 3.4 and 3.6, and the facts that $\Theta(\sum_{k=1}^{n} n_{B_k})$ tiles attach in column $-n_v$ and that $\Theta(n_v)$ tiles attach on row $1 + \sum_{j=1}^{n}(n_{B_j} + 1)$. □

Lemma 3.8. *For all $n \in \mathbb{N}$, for all $\langle \vec{B}, v \rangle \in \mathbb{N}^n \times \mathbb{N}$, 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 $\mathbb{S}_{SS}$ succeeds in attaching a ✓ tile if $\langle \vec{B}, v \rangle \in$ SubsetSum is at least $\left(\frac{1}{2}\right)^n$.*

Proof. If $\langle \vec{B}, v \rangle \in$ *SubsetSum*, then there exists at least one choice of $\vec{c} \in \{0, 1\}^n$ such that $v = \sum_{j=1}^{n} c_j B_j$. When tiles of $\mathbb{S}_{SS}$ attach to S_{SS}, only the tiles with the ? east binding domains may attach nondeterministically, there are two

choices of tiles of the ones that do have the ? east binding domain, and there are exactly n places where such tiles may attach. Thus at least $\left(\frac{1}{2}\right)^n$ of the assemblies attach a ✓ tile. □

Lemma 3.8 implies that a parallel implementation of $\mathbb{S}_{SS}$, such as a DNA implementation like those in [8,34], with 2^n seeds has at least a $1 - \frac{1}{e} \geq 0.5$ chance of correctly deciding whether a $\langle \vec{B}, v \rangle \in \mathit{SubsetSum}$. An implementation with 100 times as many seeds has at least a $1 - \left(\frac{1}{e}\right)^{100}$ chance.

Note that T_{SS} has 49 computational tile types and uses 7 tile types to encode the input.

In [17], I showed a 50-tile system that factors numbers at temperature three. Because *SubsetSum* is NP-complete, the decision version of factoring numbers ($\langle n, a \rangle \in \mathit{Factor}$ iff there exists a prime factor of n less than a) can be reduced to *SubsetSum* and solved using $\mathbb{S}_{SS}$, thus using fewer tiles and executing the model at a lower temperature, although at the cost of having to execute several instances of $\mathbb{S}_{SS}$ (one has to solve polynomially many factoring decision problems in order to find the actual factors of a number).

4. Contributions

The tile assembly model is a formal model of self-assembly and crystal growth. In [13], I explored what it means for a system to compute a function via deterministic assembly and identified two important quantities: the number of tile types and the assembly speed of the computation. In [17], I extended the definition of a tile system computing a function to include nondeterministic computation, and adapted the two measures to nondeterministic computations, as well as identified a third important measure: the probability of a nondeterministic assembly identifying the solution. Here, I defined what it means for a tile system to decide a set and explored these measures for a system that decides an NP-complete problem *SubsetSum*. The system I designed computes at temperature two, uses 49 computational tile types, and 7 tile types to encode the input. The system computes in time linear in the input size and each nondeterministic assembly has a probability of success of at least $\left(\frac{1}{2}\right)^n$. Thus a parallel implementation of $\mathbb{S}_{SS}$, such as a DNA implementation like those in [8,34], with 2^n seeds has at least a $1 - \frac{1}{e} \geq 0.5$ chance of correctly deciding whether a $\langle \vec{B}, v \rangle \in \mathit{SubsetSum}$. An implementation with 100 times as many seeds has at least a $1 - \left(\frac{1}{e}\right)^{100}$ chance. Experiments in DNA self-assembly commonly contain on the order of 10^{17} assemblies [18,21,29,25]. 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.

On the way to defining a system that decides *SubsetSum*, I also defined a system that deterministically subtracts numbers. This system uses 16 computational tile types and executes in time linear in the input size. I speculate that there exists an 8-tile subtracting system, based on the 8-tile adding system from [13].

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 alternative to 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 [15,19,20,16].

Acknowledgments

I wish to immensely thank Dustin Reishus and Leonard Adleman for their help in formalizing and stabilizing the ideas presented in this paper, for helpful discussions, and for all their other support. I also wish to thank the anonymous reviewers for their helpful comments.

References

[1] Leonard Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 1021–1024.

[2] Leonard Adleman, Towards a mathematical theory of self-assembly, Technical Report 00-722, Department of Computer Science, University of Southern California, Los Angleles, CA, 2000.
[3] Leonard Adleman, Qi Cheng, Ahish 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, ICDEA 2001, Augsburg, Germany, June 2001.
[4] Leonard Adleman, Qi Cheng, Ashish Goel, Ming-Deh Huang, David Kempe, Pablo Moisset de Espanes, Paul Rothermund, Combinatorial optimization problems in self-assembly, in: ACM Symposium on Theory of Computing, STOC02, Montreal, Quebec, Canada, 2002, pp. 23–32.
[5] Leonard Adleman, Ashish Goel, Ming-Deh Huang, Pablo Moisset de Espanes, Running time and program size for selfassembled squares, in: ACM Symposium on Theory of Computing, STOC02, Montreal, Quebec, Canada, 2001, pp. 740–748.
[6] Leonard Adleman, Jarkko Kari, Lila Kari, Dustin Reishus, On the decidability of self-assembly of infinite ribbons, in: The 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS'02, Ottawa, Ontario, Canada, November 2002, pp. 530–537.
[7] Gagan Aggarwal, Qi Cheng, Michael H. Goldwasser, Ming-Yang Kao, Pablo Moisset de Espanes, Robert T. Schweller, Complexities for generalized models of self-assembly, SIAM Journal on Computing 34 (6) (2005) 1493–1515.
[8] Robert Barish, Paul Rothemund, Erik Winfree, Two computational primitives for algorithmic self-assembly: Copying and counting, Nano Letters 5 (12) (2005) 2586–2592.
[9] Yuliy Baryshnikov, Ed G. Coffman, Petar Momcilovic, DNA-based computation times, in: Springer Lecture Notes in Computer Science, vol. 3384, 2005, pp. 14–23.
[10] 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, DNA 2005, London, Ontario, June 2005.
[11] Ravinderjit Braich, Nickolas Chelyapov, Cliff Johnson, Paul Rothemund, Leonard Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer, Science 296 (5567) (2002) 499–502.
[12] Ravinderjit Braich, Cliff Johnson, Paul Rothemund, Darryl Hwang, Nickolas Chelyapov, Leonard Adleman, Solution of a satisfiability problem on a gel-based DNA computer, in: DNA Computing: 6th International Workshop on DNA-Based Computers, DNA2000, Leiden, The Netherlands, June 2000, pp. 27–38.
[13] Yuriy Brun, Arithmetic computation in the tile assembly model: Addition and multiplication, Theoretical Computer Science 378 (June) (2007) 17–31.
[14] Yuriy Brun, Asymptotically optimal program size complexity for solving np-complete problems in the tile assembly model, in: Proceedings of the 13th International Meeting on DNA Computing, DNA07, Memphis, TN, USA, June 2007, 231–240.
[15] Yuriy Brun, A discreet, fault-tolerant, and scalable software architectural style for internet-sized networks, in: Proceedings of the Doctoral Symposium at the 29th International Conference on Software Engineering, ICSE07, Minneapolis, MN, USA, May 2007, pp. 83–84.
[16] Yuriy Brun, Discreetly distributing computation via self-assembly, Technical Report USC-CSSE-2007-714, Center for Software Engineering, University of Southern California, 2007.
[17] Yuriy Brun, Nondeterministic polynomial time factoring in the tile assembly model, Theoretical Computer Science 395 (1) (2008) 3–23.
[18] Yuriy Brun, Manoj Gopalkrishnan, Dustin 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, FNANO'04, Snowbird, UT, April 2004.
[19] 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.
[20] Yuriy Brun, Nenad Medvidovic, Fault and adversary tolerance as an emergent property of distributed systems' software architectures, in: Proceedings of the 2nd International Workshop on Engineering Fault Tolerant Systems, EFTS07, Dubrovnik, Croatia, September 2007.
[21] Nickolas Chelyapov, Yuriy Brun, Manoj Gopalkrishnan, Dustin Reishus, Bilal Shaw, Leonard Adleman, DNA triangles and self-assembled hexagonal tilings, Journal of American Chemical Society (JACS) 126 (43) (2004) 13924–13925.
[22] Ho-Lin Chen, Ashish Goel, Error free self-assembly with error prone tiles, in: Proceedings of the 10th International Meeting on DNA Based Computers, DNA 2004, Milan, Italy, June 2004.
[23] Matthew Cook, Paul Rothemund, Erik Winfree, Self-assembled circuit patterns, in: Proceedings of the 9th International Meeting on DNA Based Computers, DNA 2004, Madison, WI, June 2003, pp. 91–107.
[24] Pablo Moisset de Espanes, Computerized exhaustive search for optimal self-assembly counters, in: The 2nd Annual Foundations of Nanoscience Conference, FNANO'05, Snowbird, UT, April 2005, pp. 24–25.
[25] Tsu Ju Fu, Nadrian C. Seeman, DNA double-crossover molecules, Biochemistry 32 (13) (1993) 3211–3220.
[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, SODA 2006, Miami, FL, January 2006, pp. 571–580.
[27] Michail G. Lagoudakis, Thomas H. LaBean, 2D DNA self-assembly for satisfiability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 54 (1999) 141–154.
[28] 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, DNA 2004, Milan, Italy, June 2004.
[29] Dustin Reishus, Bilal Shaw, Yuriy Brun, Nickolas Chelyapov, Leonard Adleman, Self-assembly of DNA double-double crossover complexes into high-density, doubly connected, planar structures, Journal of American Chemical Society (JACS) 127 (50) (2005) 17590–17591.
[30] Raphael M. Robinson, Undecidability and nonperiodicity for tilings of the plane, Inventiones Mathematicae 12 (3) (1971) 177–209.
[31] Paul Rothemund, Design of DNA origami, in: Proceedings of the International Conference on Computer-Aided Design, ICCAD 2005, San Jose, CA, November 2005.
[32] Paul Rothemund, Folding DNA to create nanoscale shapes and patterns, Nature 440 (2006) 297–302.

[33] Paul Rothemund, Scaffolded DNA origami: From generalized multicrossovers to polygonal networks, Nanotechnology: Science and Computation (2006) 3–21.
[34] Paul Rothemund, Nick Papadakis, Erik Winfree, Algorithmic self-assembly of DNA Sierpinski triangles, PLoS Biology 2 (12) (2004) e424.
[35] Paul W.K. Rothemund, Erik Winfree, The program-size complexity of self-assembled squares, in: Proceedings of the ACM Symposium on Theory of Computing, STOC00, Portland, OR, USA, May 2000, pp. 459–468.
[36] Michael Sipser, Introduction to the Theory of Computation, PWS Publishing Company, 1997.
[37] David Soloveichik, Erik Winfree, Complexity of self-assembled shapes, in: Proceedings of the 10th International Meeting on DNA Based Computers, DNA 2004, Milan, Italy, June 2004.
[38] Hao Wang, Proving theorems by pattern recognition, I. Bell System Technical Journal 40 (1961) 1–42.
[39] Erik Winfree, On the computational power of DNA annealing and ligation, DNA Based Computers (1996) 199–221.
[40] Erik Winfree, Algorithmic self-assembly of DNA, Ph.D. Thesis, California Insitute of Technology, Pasadena, CA, June 1998.
[41] Erik Winfree, Simulations of computing by self-assembly of DNA, Technical Report CS-TR:1998:22, California Insitute of Technology, Pasadena, CA, 1998.
[42] Erik Winfree, Self-healing tile sets, Nanotechnology: Science and Computation (2006) 55–78.
[43] Erik Winfree, Renat Bekbolatov, Proofreading tile sets: Error correction for algorithmic self-assembly, in: The 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS'02, vol. 2943, Madison, WI, June 2003, pp. 126–144.