Combinatorial Optimization Problems in Self-Assembly*

Leonard Adleman
University of Southern California

Qi Cheng
University of Oklahoma

Ashish Goel
University of Southern California

Ming-Deh Huang
University of Southern California

David Kempe
Cornell University

Pablo Moisset de Espanés
University of Southern California

Paul Wilhelm Karl Rothemund
California Institute of Technology

ABSTRACT

Self-assembly is the ubiquitous process by which simple objects autonomously assemble into intricate complexes. It has been suggested that intricate self-assembly processes will ultimately be used in circuit fabrication, nano-robotics, DNA computation, and amorphous computing. In this paper, we study two combinatorial optimization problems related to efficient self-assembly of shapes in the Tile Assembly Model of self-assembly proposed by Rothemund and Winfree [18]. The first is the Minimum Tile Set Problem, where the goal is to find the smallest tile system that uniquely produces a given shape. The second is the Tile Concentrations Problem, where the goal is to decide on the relative concentrations of different types of tiles so that a tile system assembles as quickly as possible. The first problem is akin to finding optimum program size, and the second to finding optimum running time for a "program" to assemble the shape.

We prove that the first problem is NP-complete in general, and polynomial time solvable on trees and squares. In order to prove that the problem is in NP, we present a polynomial time algorithm to verify whether a given tile system uniquely produces a given shape. This algorithm is analogous to a program verifier for traditional computational systems, and may well be of independent interest. For the second problem, we present a polynomial time $O(\log n)$-approximation algorithm that works for a large class of tile systems that we call *partial order systems*.

*Author emails: adleman@cs.usc.edu, qcheng@cs.ou.edu, agoel@cs.usc.edu, huang@cs.usc.edu, kempe@cs.cornell.edu, pmoisset@cs.usc.edu, pwkr@dna.caltech.edu . Leonard Adleman's research was supported in part by grants from NASA/JPL, NSF, ONR, and DARPA. Qi Cheng's research was supported in part by NSF Grant CCR-9820778. Ashish Goel's research was supported in part by a grant from NASA. Ming-deh Huang's research was supported in part by a grant from NASA and by NSF Grant CCR-9820778. David Kempe's research was supported by an Intel Fellowship. Pablo Moisset de Espanés's research was supported in part by a grant from NASA. Paul Rothemund's research was supported by a Beckman Senior Research Fellowship.

STOC'02, May 19-21, 2002, Montreal, Quebec, Canada.

1. INTRODUCTION

In this paper we study two optimization problems related to efficient self-assembly of geometric shapes. Self-assembly is the ubiquitous process by which simple objects autonomously assemble into intricate complexes. Nature provides many examples: Atoms react to form molecules. Molecules react to form crystals and supramolecules. And self-assembly of cells plays a part in the development of organisms. Simple self-assembly schemes are already widely used in chemical syntheses – it has been suggested that more complicated schemes will ultimately be useful for circuit fabrication, nano-robotics, DNA computation [21, 15, 14, 7, 24], and amorphous computing [1]. Several promising experiments along these lines have been recently reported; a summary of some recent experimental activity is presented later in this introduction.

In accordance with its practical importance, self-assembly has received increased theoretical attention over the last few years. Adleman [2] has proposed a mathematical model of linear self-assembly, and Adleman *et al.* [3] studied the equilibrium states of these systems. For 2-dimensional self-assembly, Winfree has proposed the Tile Assembly Model [23]. This model considers the assembly of rigid square objects or *tiles*. Elementary steps under this model are simple – assembly starts from a single seed tile and proceeds by the addition of single tiles – yet the structures that may be formed are surprisingly complex. In theory, the Tile Assembly Model is universal and relatively small sets of tiles may be used to simulate arbitrary Turing Machines or cellular automata [23]. Perhaps the most interesting facet of this result is not that self-assembly may be able to compute, *per se*, but rather that a set of tiles and binding interactions may be viewed as a program to build a desired pattern or shape. In the language of this paper such a program is known as a *tile system*.

Working under the assumption that an arbitrary tile system can be implemented, researchers have begun to explore complexity-theoretic questions about the Tile Assembly Model. In particular, some of the recent work has focused on the assembly of $n \times n$ squares. Rothemund and Winfree [18] studied the program size complexity (the number of different tile-types required) of assembling such squares. Adleman *et al.* [4] added the notion of time complexity to the Tile Assembly Model and applied it to such squares. To analyze the assembly time for a shape, they describe the assembly process as a continuous time Markov chain. The assembly time depends not only on the tile system, but also on the relative concentration of each tile-type. They also described a tile system for assembling $n \times n$ squares that simultaneously achieved the asymptotically optimum program size of $\Theta(\log n / \log \log n)$ and the asymptotically optimum expected time complexity of $\Theta(n)$.

The study of self-assembled squares has been fruitful and yielded insights into "programming techniques" for tile systems. Thus it may be worthwhile to study the program size and time complexity for assembling other important specific shapes, such as spirals, fractals, or rings. In this paper, however, we consider two combinatorial optimization problems for generalized shapes:

1. **The Minimum Tile Set Problem:** Given a shape, find the tile system with the minimum number of tile-types that can uniquely self-assemble into this shape.

2. **The Tile Concentrations Problem:** Given a shape and a tile system that uniquely produces the given shape, assign concentrations to each tile-type so that the expected assembly time for the shape is minimized.

We believe that good solutions to these problems are important for developing a mature, algorithmic theory of self-assembly and for applying this theory to real-life scenarios. In this paper we report partial progress towards resolving these problems.

We first prove that the decision version of the Minimum Tile Set problem is in NP. For proving membership in NP, we need an algorithm to verify in polynomial time whether a given tile system uniquely assembles into a given shape. We note that since tile systems are analogous to programs for assembling a shape, such an algorithm is analogous to a program verifier and may well be of independent interest in practical settings. In this paper, we present such an algorithm for the case where the tile system has a unique seed, and the shape must have a single occurrence of the seed. A recent result by Cook *et al.* [5, 17] allows an extension of our basic verification algorithm to the general case where the tile system can have multiple seeds and there can be multiple occurrences of the seed within the shape. We then show that the Minimum Tile Set Problem is polynomial time tractable on tree-shapes and squares [1]. To prove NP-hardness, we reduce 3-CNF SAT to the Minimum Tile Set Problem. We need to use the polynomial time algorithm for the minimum tile set problem on tree-shapes as a subroutine in the reduction. We also use some ideas found in a paper by Lagoudakis and LaBean [14]. An obvious and important open problem is to obtain upper and lower bounds on the approximation ratio for the Minimum Tile Set Problem.

For the Tile Concentrations Problem, we conjecture that finding an optimum solution is #P-hard. We then define *partial order systems*; such systems include tile systems that count [18], assemble into squares [18, 4] and trees, perform base-conversion of numbers [4], simulate Turing machines [23, 18], and simulate 1-dimensional cellular automata. We present a polynomial time, $O(\log n)$-approximation algorithm for the Tile Concentrations Problem restricted to partial order systems. Our algorithm approximates the stochastic self-assembly process by a carefully defined deterministic process. The optimization problem for the deterministic process is a convex program and may have exponentially many constraints. However, a polynomial time membership oracle exists for this program, allowing us to optimize the deterministic process in polynomial time. It is interesting to note that any tile system that uniquely produces a tree-shape must be a partial order system; further, we prove that for tree-shapes, the minimum tile set also leads to the fastest assembly. Finding the exact computational status of the Tile Concentrations Problem remains an important open question.

We define the Tile Assembly Model and the two optimization problems formally in section 2. The proof that the Minimum Tile Set Problem is in NP is given in section 3; in particular this section contains a description of the algorithm which verifies whether a given tile system uniquely assembles into a given shape. We outline our polynomial time algorithm for the Minimum Tile Set Problem on tree-shapes and squares in section 4. The proof of NP-hardness of the Minimum Tile Set Problem is sketched in section 6. Section 5 details our $O(\log n)$-approximation algorithm for the Tile Concentrations Problem on partial order systems.

We now summarize some recent experimental activity in the field of self-assembly. This experimental activity gives us hope that sophisticated self-assembly systems may be implemented in the near future and also offers some insight into the kind of theoretical questions that may be important for self-assembling systems.

1.1 Summary of recent experimental activity

While tile systems that generate interesting computations, patterns and shapes are not very complex, the use of natural systems to implement them seems very difficult. Instead, researchers have attempted to engineer them using either macroscopic plastic tiles that assemble at an oil/water interface or nanoscale multi-stranded DNA motifs designed to act as closely to the abstract tiles of the tile assembly model as possible. A number of challenges must be overcome before either system can be used to implement interesting tile systems, in particular (i) nucleation of structures must be controlled so that their growth begins with a specified seed tile (ii) the rate of errors (events in which a tile binds where it should not bind) must be minimized. Using centimeter-sized plastic tiles Rothemund [16] has explored the simulation of simple 1-dimensional cellular automata on a particular input. Controlled nucleation could not be achieved, so the desired input could not be specified. Instead, random simulations of the cellular automata were formed with an error rates of 2.8 %. DNA systems, more technologically interesting than the plastic tile systems because of their size, are being actively pursued by at least two research groups and the problems of errors and controlled nucleation are being emphasized. Winfree *et al.* [22] demonstrated that (i) DNA strands can self-assemble into 13 nanometers $\times$ 4 nm $\times$ 2 nm DNA tiles and (ii) that these tiles can further self-assemble into periodic crystals of approximately a quarter million tiles and several microns in size. Mao *et al.*[13] have reported the creation of small 2-dimensional assemblies of DNA that compute the cumulative XOR of pairs of 4-bit strings. For this computation, the first reported using 2-

[1] The algorithm for squares assumes a constant *temperature*; see section 4 for details.

dimensional self assembly, error rates on the order of 3-5 % were observed. While controlled nucleation is part of the design of this experiment, it was not measured. In theory, with proper control of experimental temperature, DNA systems should be able to achieve reasonably large structures (say 10,000 tiles) with error rates of less than 10^{-4} in reasonable amounts of time (say a week) and nucleation should be well-controlled [23]. New experimental DNA systems, designed explicitly so that error and nucleation rates may be easily measured, are now being explored [17]. Already tiles have been held in a supersaturated state (one prerequisite for controlled nucleation) for approximately 6 hours [17]. The outlook for experimental self-assembly using these systems is promising.

Gracias *et al.*[8] have used the self-assembly of small plastic and metal objects to fabricate simple 3-d circuits of light emitting diodes. Lopinski *et al.*[11] have demonstrated the self-assembly of molecular silicon structures. And, most recently, Bell Labs has announced the construction, by self-assembly, of the world's first molecular-thickness transistor [19].

2. DEFINITIONS

The Tile Assembly Model was originally proposed by Rothemund and Winfree [18]. It extends the theoretical model of tiling by Wang [20] to include a mechanism for growth based on the physics of molecular self-assembly. Informally each unit of an assembly is a square with glues of various types on each edge. The tile "floats" on a two dimensional plane and when two tiles collide they stick if their abutting sides have compatible glues.

Formally, a tile is an oriented unit square with the north, east, south and west edges labeled from some alphabet Σ of glues. We begin with a triple $\langle T, g, \tau \rangle$ where T is a finite set of tile-types, $\tau \in \mathbf{Z}^+$ is the *temperature*, and g is the *glue strength* function from $\Sigma \times \Sigma$ to $\mathbf{Z}^+ \cup \{0\}$, where Σ is the set of edge labels and $\mathbf{N}$ is the set of natural numbers. It is assumed that $null \in \Sigma$, $g(x,y) = g(y,x)$ for $x, y \in \Sigma$, and $g(null, x) = 0$ for all $x \in \Sigma$. For each tile type $i \in T$, the labels of its four edges are denoted $\sigma_N(i)$, $\sigma_E(i)$, $\sigma_S(i)$, and $\sigma_W(i)$.

A *configuration* is a map from $\mathbf{Z}^2$ to $T \cup \{\mathbf{empty}\}$. For $t \in T$, $\Gamma_t^{(x,y)}$ is the configuration such that $\Gamma_t^{(x,y)}(i,j) = t$ iff $(i,j) = (x,y)$ and **empty** otherwise. Let C and D be two configurations. Suppose there exist some $i \in T$ and $(x,y) \in \mathbf{Z}^2$ such that $C(x,y) = \mathbf{empty}$, $D = C$ except at (x,y), $D(x,y) = i$, and

$$g(\sigma_E(i), \sigma_W(D(x+1,y))) + g(\sigma_W(i), \sigma_W(D(x-1,y))) +$$
$$g(\sigma_N(i), \sigma_W(D(x,y+1))) + g(\sigma_S(i), \sigma_W(D(x,y-1))) \geq \tau.$$

Then we say that the position (x,y) in C is *attachable*, and we write $C \rightarrow_{\mathbf{T}} D$ to denote the transition from C to D in attaching tile i to C at position (x,y). Informally, $C \rightarrow_{\mathbf{T}} D$ iff D can be obtained from C by adding a tile to it such that the total strength of interaction in adding the tile to C is at least τ.

A *tile system* is a quadruple $\mathbf{T} = \langle T, \mathcal{S}, g, \tau \rangle$, where T, g, τ are as above and $\mathcal{S}$ is a set of supertiles called *seed supertiles*. Intuitively a supertile is a connected and finite assembly of tiles. Supertiles are the initial configuration of the assembly process or the objects obtained by adding individual tiles to an existing supertile.

Let $\rightarrow_{\mathbf{T}}^*$ denote the reflexive transitive closure of $\rightarrow_{\mathbf{T}}$, and let $\rightarrow_{\mathbf{T}}^+$ be the transitive closure of $\rightarrow_{\mathbf{T}}$. A *derived supertile* of the tile system $\mathbf{T}$ is a supertile such that $s \rightarrow_{\mathbf{T}}^* A$ for some $s \in \mathcal{S}$. A *terminal supertile* of the tile system $\mathbf{T}$ is a derived supertile A such that there is no supertile B, different from A, such that $A \rightarrow_{\mathbf{T}}^* B$. If there is a terminal supertile A such that for all derived supertile B, $B \rightarrow_{\mathbf{T}}^* A$, we say that the tile system *uniquely produces* A. We use the term $Prod(\mathbf{T})$ to refer to the set of derived supertiles of $\mathbf{T}$, and the term $Term(\mathbf{T})$ to refer to the set of terminal supertiles of $\mathbf{T}$.

Given a tile system $\mathbf{T}$ which uniquely produces A, we say that the program size complexity of the system is $|T|$ i.e. the number of tile-types.

In this paper, we adopt the restriction, suggested by Rothemund and Winfree [18], that $\mathcal{S}$ contains a single seed s consisting of a single tile, and that $g(\alpha, \beta) = 0$ for $\alpha, \beta \in \Sigma$ with $\alpha \neq \beta$.

Adleman *et al.* [4] proposed a definition of the time complexity of self-assembly, which we will now explain. We associate with each tile type $i \in T$ a nonnegative probability $P(i)$, such that $\sum_{i \in T} P(i) = 1$. We assume that the tile system has an infinite supply of each tile type, and $P(i)$ models the concentration of tile i in the system – the probability that a tile of type i is chosen when a tile is drawn at random. Now self-assembly of the tile system $\mathbf{T}$ corresponds to a continuous time Markov process where the states are in a one-one correspondence with derived supertiles, and the initial state corresponds to the seed s. There is a transition of state B to C iff $B \rightarrow_{\mathbf{T}} C$, and the rate of the transition is $P(i)$ if C is obtained from B by adding a tile of type i. Suppose the tile system uniquely produces a supertile A_T. It would follow that A_T is the unique sink state. Given the Markov process, the time for reaching A_T from s is a random variable. The time complexity for producing A_T from s is defined as the expected value of this random variable.

Informally our definition of time models a system wherein a seed "floats" in solution encountering tiles at random. The higher the concentration of a particular tile the higher the rate at which it is encountered. When a tile is encountered which has sufficiently strong interaction with the seed, the tile is incorporated. By this process of accretion the seed grows larger and larger.

A tile position is a pair (i,j) of integers. A pair $(i_1, j_1), (i_1, j_2)$ of tile positions is said to be adjacent if $|i_1 - i_2| + |j_1 - j_2| = 1$. For a finite set A of tile positions, let $G(A)$ denote the graph on A defined by the adjacency relation; we will call this the adjacency graph. A set A of tile positions is said to be connected if the adjacency graph is connected; we will often use the term "shape" to refer to a finite connected set of tile positions. Two sets of tile positions are said to be isomorphic if they can be made identical by translation[2]. Given two shapes of size n each, we can check whether they are isomorphic in time $\Theta(n)$.

A tile system is said to uniquely produce a shape if it uniquely produces a supertile that has this shape.

DEFINITION 2.1. **Minimum tile set problem:** *Given a shape A and a temperature τ, find $\mathbf{T} = \langle T, \{s\}, g, \tau \rangle$ such that $|T|$ is minimal and $\mathbf{T}$ uniquely produces A.*

[2]If the adjacency graphs of two shapes are isomorphic, it doesn't necessarily imply that the shapes themselves are isomorphic.

DEFINITION 2.2. **Minimum tile set decision problem:** *Let* $L = \{(A, c, \tau)/\exists \mathbf{T} = \langle T, \{s\}, g, \tau\rangle$ *with* $|T| \leq c$ *that uniquely produces the shape* $A\}$ *Given a triple* (A, c, τ)*, determine whether* $(A, c, \tau) \in L$.

Let $\mathbf{T} = \langle T, \{s\}, g, \tau\rangle$. P is said to be a *concentrations function* iff $P : T \to [0, 1]$ *and* $\sum_{\forall i \in T} P(i) = 1$.

DEFINITION 2.3. **Tile Concentrations Problem:** *Let* A *be a shape, and let* $\mathbf{T} = \langle T, \{s\}, g, \tau\rangle$ *be a tile system that uniquely produces* A*. Let* t *be the time to assemble* A*. Given* $\mathbf{T}$*, find a concentrations function* P *such that* $E[t]$ *is minimum.*

3. VERIFYING UNIQUE ASSEMBLY OF A SHAPE BY A TILE SYSTEM

We will first present an algorithm for verifying whether a given tile system uniquely produces a given *supertile* and then later extend it to shapes. Our algorithm establishes that the Minimum Tile Set problem is in NP, and may also be of independent interest in practical settings to verify a proposed mechanism for self-assembling a desired shape. As explained in the introduction, we will assume that the tile system has a single seed tile-type, and there can only be one occurrence of the seed in the shape; we will further assume that this seed tile must be at position $(0, 0)$. Recent generalizations of our work by Cook *et al.*[6] remove these restrictions.

Throughout this section, we assume that we have some fixed tile system $\mathbf{T} = \langle T, \mathcal{S}, g, \tau\rangle$. Whenever we speak of a supertile being produced, uniquely produced etc., it is with respect to this tile system.

At the core of deciding unique production is a simple greedy algorithm that grows the maximum produced subtile of a given supertile A. A' is called a subtile of A if $A'(x, y) = A(x, y)$ for each site (x, y) with $A'(x, y) \neq$ **empty**. Further, A' is said to be a maximum produced subtile of A if A' is a subtile of A, A' is in $Prod(\mathbf{T})$, and in addition, each produced subtile A'' of A is also a subtile of A'. The existence of a unique maximum produced subtile follows implicitly from the correctness of the following algorithm:

`Algorithm Greedy-Grow` (A)
1. `Start with` $A' = \Gamma^{(0,0)}_{A(0,0)}$.
2. `While there is a site` (x, y) `with` $A(x, y) = t$ `and` $A'(x, y) =$ **empty** `such that` t `can be added to` A' `at` (x, y)`, add it.`

LEMMA 3.1. *The algorithm Greedy-Grow computes the maximum produced subtile of* A*. It can be implemented to run in time* $O(|A|)$.

PROOF. It is immediately clear that the output A' is a produced subtile, so we only need to prove that it is maximum.

Assume that it is not, and let B be another produced subtile of A with a site (x, y) such that $A'(x, y) =$ **empty** and $B(x, y) = A(x, y) \neq$ **empty**. Fix some order of tile additions for B, and let (x, y) be the first site at which a tile with the above property was added in this order. Let B' be the subtile immediately before the addition — by definition, B' is also a produced subtile of A'. Because the bond strengths are non-negative, and the tile $A(x, y)$ could be added to B' at site (x, y), it can still be added to A' at site (x, y), contradicting the termination condition in the loop of the greedy algorithm.

To implement the algorithm to run in linear time, one can simply maintain a list of all sites at which tiles can be added immediately. Whenever a tile is added, all its neighbors are checked, and added to the list if they are now candidates for addition. ■

The algorithm for deciding unique production tests all three necessary properties separately, i.e. whether a supertile is produced, terminal, and unique.

`Algorithm Unique-Supertile` (A)
1. `Let` $A' =$ `Greedy-Grow`(A).
   `If` $A' \neq A$`, then` A `is not produced.`
2. `For all non-empty sites` (x, y)`, test whether any tile` t `can be added at an adjacent site.`
   `If yes, then` A `is not terminal.`
3. `For all non-empty sites` (x, y)`, let` $A_{(x,y)}$ `be the supertile` A `with the tile at` (x, y) `removed.`
   `Let` $A'_{(x,y)} =$ `Greedy-Grow`$(A_{(x,y)})$.
   `If a tile` $t \neq A(x, y)$ `can be added to` $A'_{(x,y)}$ `at` (x, y)`, then` A `is not uniquely produced.`

`If` A `does not fail any of the above three tests, then` A `is uniquely produced and terminal.`

THEOREM 3.2. *The above algorithm decides unique terminal production in time* $O(|A|^2 + |A||T|)$.

PROOF. The first and second steps obviously filter out exactly non-produced and non-terminal assemblies. It is also clear that when the algorithm claims that a supertile is not uniquely produced, it possesses a witness for the non-uniqueness (in the form of the subtile $A'_{(x,y)}$), so it only remains to show that if a terminal supertile A is not produced uniquely, then the algorithm will discover a witness for the non-uniqueness.

Let A be not uniquely produced, and B be another terminal supertile produced by $\mathbf{T}$. Fix some order in which the tiles of B were added, and let (x, y) be the first site in this order at which a tile $t = B(x, y) \neq A(x, y)$ was added. Let B' denote the subtile right before t was added at site (x, y).

By definition of (x, y), B' is a subtile of A, and more importantly, of $A_{(x,y)}$. Because B' is produced, and $A'_{(x,y)}$ is the maximum produced subtile of $A_{(x,y)}$, B' is also a subtile of $A'_{(x,y)}$. The bond strength function is non-negative, so the tile t can also be added to $A'_{(x,y)}$ at site (x, y), and the algorithm will discover this fact in step 3, thus finding a witness for non-uniqueness.

To bound the running time, notice that step 1 takes time $O(|A|)$, step 2 time $O(|A||T|)$, because we check each of possibly $|T|$ kinds of tiles at each of at most $4|A|$ locations, and step 3 takes time $O(|A|^2)$, because we are running $|A|$ times Greedy-Grow. ■

Building on this algorithm, we can test for the unique production of shapes.

`Algorithm Unique-Shape` (S)
1. `Let` A `be any supertile grown by` $\mathbf{T}$`, where growth is terminated after` $|S| + 1$ `steps if necessary.`

```
2.  If A does not have shape S, then the
    tile system does not uniquely produce S.
3.  Otherwise, use Unique-Supertile (A) to
    determine whether A is uniquely produced.
    S is uniquely produced iff A is.
```

THEOREM 3.3. *The algorithm Unique-Shape (S) decides in time $O(|A|^2 + |A||T|)$ whether a tile system uniquely produces the shape S.*

PROOF. Obvious. ■

4. MINIMUM TILE SET PROBLEM FOR TREES AND SQUARES

4.1 Tree-shapes

We will assume in this section that the temperature τ is 1; it is not hard to see that any tile system to uniquely produce a tree can be transformed into one that has $\tau = 1$ without increasing the size of the tile system. We will assume that we are given a tree-shape A, and also that we know the position of the seed. We will call this the root. For any tile position $(i, j) \in A$, define $A(i, j)$ to be the set of tile positions corresponding the sub-tree rooted at (i, j) in $G(A)$; we will refer to this set of tile positions as the sub-shape at (i, j). Also, let $P(i, j)$ denote the tile position which is the parent of (i, j) in the tree $G(A)$ i.e. the tile position which occurs just before (i, j) in the path from the root to (i, j) in $G(A)$. Observe that each tile position except the root has a unique parent; the parent of the root is defined to be the root itself.

DEFINITION 4.1. *Two tile positions in a tree-shape A are said to be* isomorphic *if their corresponding sub-shapes are isomorphic.*

DEFINITION 4.2. *Two tile positions (i_1, j_i) and (i_2, j_2) in a tree-shape A are said to be* identically entered *if the vectors $P(i_1, j_1) - (i_1, j_1)$ and $P(i_2, j_2) - (i_2, j_2)$ are identical.*

DEFINITION 4.3. *Two tile positions in a tree-shape are said to be* compatible *if they are identically entered as well as isomorphic.*

The "compatible" relation defined above is an equivalence relation and hence partitions the set of tile positions in A into equivalence classes. Let Γ represent a set of tile-types that uniquely assembles into a tree-shape A with the seed at the root. Let $\Gamma(i, j)$ represent the tile-type that goes to position $(i, j) \in A$. We now state the following two lemmas; the proof of these lemmas is straightforward and is omitted from this version.

LEMMA 4.1. *If $\Gamma(i_1, j_1) = \Gamma(i_2, j_2)$, then (i_1, j_1) and (i_2, j_2) must be compatible.*

For positions (i_1, j_1). (i_2, j_2) in A such that $\Gamma(i_1, j_1) \neq \Gamma(i_2, j_2)$, let MERGE$(\Gamma, i_1, j_1, i_2, j_2)$ represent the set of tile-types obtained from Γ by performing the following transformations:

1. For each glue-label g_2 on an edge of tile-type $\Gamma(i_2, j_2)$, replace all occurrences of g_2 in all tile-types by g_1, where g_1 is the glue-label on the corresponding edge of $\Gamma(i_1, j_1)$.
2. Remove the tile-type $\Gamma(i_2, j_2)$ from Γ.

Intuitively, MERGE$(\Gamma, i_1, j_1, i_2, j_2)$ is the set of tile-types obtained by allowing the tile-type $\Gamma(i_1, j_1)$ to attach wherever the tile-type $\Gamma(i_2, j_2)$ could attach in the original set of tile-types Γ; the tile-type $\Gamma(i_2, j_2)$ then becomes redundant and is removed.

LEMMA 4.2. *If (i_1, j_1) and (i_2, j_2) are compatible but $\Gamma(i_1, j_1) \neq \Gamma(i_2, j_2)$, then the set* MERGE$(\Gamma, i_1, j_1, i_2, j_2)$ *also uniquely assembles A.*

The following theorem is immediate from lemmas 4.1 and 4.2; we omit the proof. This theorem directly leads to a polynomial time algorithm for the Minimum Tile Set Problem on trees, and is also useful for the proof of NP-hardness in section 6.

THEOREM 4.3. *A set of tile-types Γ that uniquely assembles into a tree-shape A (with the seed at the root) is the smallest such set iff the following is true:*

> *$\Gamma(i_1, j_1) = \Gamma(i_2, j_2)$ iff (i_1, j_1) and (i_2, j_2) are compatible.*

As observed before, isomorphism of two shapes can be checked in time linear in the size of the shapes. Finding whether two tiles are identically oriented takes $\Theta(1)$ time. Thus, we can find whether a pair of tile positions is compatible in time $\Theta(n)$, and consequently, divide the entire set of tile positions (i, j) into equivalence classes in time $\Theta(n^3)$. By theorem 4.3, the number of equivalence classes is exactly the minimum number of tile-types required to assemble the given tree-shape; a different tile-type is associated with each equivalence class. The above discussion pertained to tree-shapes with a given seed. If we are not given the seed then we can guess each of the tile positions to be the seed in turn and pick the best solution.

In order to complete the construction, we need to describe how to assign glues to each tile-type in the optimum solution. For each tile-type T, consider a tile position (i, j) which belongs to the equivalence class corresponding to T. We will now explain how to assign a glue to the "West" edge of T. Consider the tile position $(i-1, j)$. If this tile position does not belong to the tree-shape, then assign a *null* glue to the "West" edge of T. If the tile position $(i-1, j)$ does belong to the tree-shape, then let T' be the tile-type corresponding to the equivalence class of tile position $(i-1, j)$. Label the "West" edge of T by the set $\{T, T'\}$. Assign glues to the other edges of T in a similar fashion. It is easy to see that in the case considered above, the "West" side of T and the "East" side of T' will receive the same glue.

4.2 Squares

Let us assume that the temperature τ is bounded by some constant. Let $N(K)$ denote the number of tile systems with at most K tile-types. There are at most $4K$ sides, and each must choose one of at most $4K$ glues and at most τ different glue strengths. Therefore, $N(K) \leq (4K)^{f(K)}$ for some function $f(K) = O(K)$. Adleman *et al.* [4] proved that the optimum program size for assembling an $n \times n$ square is $O(\log n / \log \log n)$. Consequently, the number of tile systems which use at most $O(\log n / \log \log n)$ tile-types is $n^{O(1)}$. For each of these (polynomially many) tile systems, we can use the algorithm outlined in section 3 to determine whether they uniquely assemble into an $n \times n$ square. Note that we assume that the input size is the number of tiles in the square, and *not* the number of bits needed to represent n. This is consistent with our definition of the size of the problem as the size of the shape. The algorithm outlined above will also work for "thick" rectangles ie. rectangles where the width is at least logarithmic in the height; details are omitted from this version.

5. TILE CONCENTRATIONS PROBLEM

In section 4 we have addressed the problem of finding the smallest tile system that produces a particular shape. In this section we will discuss the problem of minimizing the time it takes to finish an assembly process given a tile system. In this section, we will concentrate on *partial order systems*, which we define below. This is a large class of tile systems that includes tile systems that count [18], assemble into squares [18, 4] and trees, perform base-conversion of numbers [4], simulate Turing machines [23, 18], and simulate 1-dimensional cellular automata.

Assume that we are given a tile system $\mathbf{T}$ and a shape S that is uniquely produced by $\mathbf{T}$. Let A denote the supertile that is uniquely produced by $\mathbf{T}$ and has shape S. Further assume that the tile system has a unique seed tile-type that occurs only once in A; without loss of generality, assume that the position in the shape S corresponding to the seed is $(0, 0)$. Let $\Gamma(i, j)$ represent the tile-type at position (i, j). Let $t_{i,j}$ represent the time when the tile at position (i, j) attaches to the growing assembly. Define a partial order $\prec$ on the tile positions in S such that $(i, j) \prec (p, q)$ iff $t_{i,j} \leq t_{p,q}$ for all possible ways of assembling shape S using $\mathbf{T}$.

DEFINITION 5.1. *A tile system* $\mathbf{T}$ *is said to be a* partial order system *if it is singly seeded, uniquely produces a shape* S *which has only one occurrence of the seed, and if for all adjacent positions* (i, j),(p, q) *in* S*, either* $(i, j) \prec (p, q)$*, or* $(p, q) \prec (i, j)$*, or the strength of the glues connecting tiles at positions* (i, j) *and* (p, q) *is zero.*

Given a partial order system $\mathbf{T}$ and its uniquely produced shape S, consider the directed acyclic graph G defined on S by the relation $\prec$. This graph will have only one source node (ie. the seed, $(0, 0)$). Let $\mathcal{P}$ be the set of all source-to-sink paths in G. Let $X_{i,j}$ represent the time it takes for the tile at position (i, j) to attach after it becomes attachable. $X_{i,j}$ is an exponential random variable with mean $1/C(\Gamma(i, j))$ where $C(\Gamma(i, j))$ is the concentration of the tile-type at position (i, j). For any path P in $\mathcal{P}$, let $X_P = \sum_{(i,j)\in P} X_{i,j}$. Now, the assembly time is $X(C) = \max_{P\in\mathcal{P}} X_P$; we use the notation $X(C)$ to make explicit the dependence of X on the vector of concentrations C. Now, the tile concentration problem is to find non-negative concentrations $C(\Gamma)$ for each tile-type Γ such that $\sum_{\Gamma\in T} C(\Gamma) = 1$ and $\mathbf{E}[X(C)]$ is minimized.

This is an involved problem, since max and $\mathbf{E}$ do not commute. We have not been able to resolve whether this problem is NP-hard or in PTIME. However, based on the similarity of this problem to one studied by Louth, Mitzenmacher and Kelly [12] we make the following conjecture:

CONJECTURE 5.1. *Solving the Tile Concentrations Problem for partial order systems is #P-hard.*

In order to obtain an $O(\log n)$-approximation algorithm for this problem, we approximate the assembly process by a deterministic process where the time $Y(i, j)$ for tile (i, j) to attach is a *deterministic* variable with value $1/C(\Gamma(i, j))$ ie. $Y(i, j) = \mathbf{E}[X_{i,j}]$. We define Y_P and $Y(C)$ analogous to X_P and $X(C)$. Observe that $Y(C) = \max_P \mathbf{E}[X_P]$ whereas $X(C) = \max_P X_P$. The following theorem establishes that optimizing Y immediately leads to an $O(\log n)$-approximation for X.

THEOREM 5.2. *Let* C^*_Y *represent a vector of concentrations that minimizes* Y *and* C^*_X *represent a vector of concentrations that minimizes* X*. Then,* $\mathbf{E}[X(C^*_Y)] = O(\log n) \cdot \mathbf{E}[X(C^*_X)]$.

PROOF SKETCH: For any vector of concentrations C, we have $Y(C) = \max_P \mathbf{E}[X_P]$; let P' be the path where this maximum is achieved. Now, $Y(C) = \mathbf{E}[X_{P'}]$. Clearly, $X_{P'} \leq \max_P X_P$, and hence, by taking expectations on both sides, we obtain $\mathbf{E}[X_{P'}] \leq \mathbf{E}[\max_P X_P]$. This immediately implies that $Y(C) \leq \mathbf{E}[X(C)]$. Let $\gamma = \max_{(i,j)\in S}(X_{i,j}/Y_{i,j})$. Since there are n tiles in the shape, and $X_{i,j}$ are exponential random variables with mean $Y_{i,j}$, it follows that $\mathbf{E}[\gamma] = O(\log n)$. Also, $X_P \leq \gamma Y_P$ which implies that $\mathbf{E}[X(C)] = O(\log n) \cdot Y(C)$.

Now, $\mathbf{E}[X(C^*_Y)] = O(\log n)Y(C^*_Y)$. But $Y(C^*_Y) \leq Y(C^*_X)$ by the optimality of C^*_Y for Y, and hence, $Y(C^*_Y) \leq \mathbf{E}[X(C^*_X)]$. This completes the proof of the theorem. ∎

The problem now reduces to determining C^*_Y. Define $x_\Gamma = 1/(C(\Gamma)$. We will need the following lemma before proceeding:

LEMMA 5.3. *For all tile-types* Γ*,* $C^*_Y(\Gamma) \geq 1/n^2$.

PROOF. The length of the longest path in G is at most n, and there are at most n tile-types. Assigning a concentration of $1/n$ to each tile-type results in an upper bound of n^2 for Y. Hence, $Y(C^*_Y) \leq n^2$. But $Y(C^*_Y) \geq 1/C^*_Y(\Gamma)$ for all tile-types Γ, which proves the result. ∎

The optimization problem is now the following:

$$
\begin{aligned}
\text{Find } Y^* \quad &= \quad \min Y, \quad \text{such that} \\
&\quad \sum_{(i,j)\in P} x_{\Gamma(i,j)}) \leq Y \quad \text{for all source-sink paths} P \in \mathcal{P} \\
&\quad \sum_{\Gamma\in T} 1/x_\Gamma \leq 1 \\
&\quad 0 \leq x_\Gamma \leq n^2
\end{aligned}
$$

The above program is not linear because of the constraint $\sum_{\Gamma\in T} 1/x_\Gamma \leq 1$. Also, since the number of source-sink paths may be exponential, the number of constraints in exponential. However, it is easy to find a membership oracle as well as a separation oracle for the problem, using the longest path problem on directed acyclic graphs. Further, even though the above problem is not an LP, it is still a convex program, and has a good initial bounding box because of the constraints $0 \leq x_\Gamma \leq n^2$. Hence, the optimum solution to this program can be found in polynomial time using the ellipsoid algorithm [10], completing the $O(\log n)$-approximation algorithm. A thorough discussion on how to solve convex programs from membership, separation and violation oracles can be found in [9].

5.1 Optimal Tile Concentrations and Smallest Tile Set problems for trees:

In computational problems it is common to find time-space tradeoffs. The fastest program to perform a computation is not necessarily the smallest. In self assembling trees under our model, however, there is no such tradeoff. In section 4 we presented an algorithm for solving the Minimum Tile Set Problem on trees. Given a tile system $\mathbf{T}$ that uniquely produces a shape S, let $X^*(\mathbf{T})$ refer to the optimum solution to the Tile Concentrations Problem; note that we do not know how to compute $X^*(\mathbf{T})$ in polynomial time. We omit the proof of the following theorem:

THEOREM 5.4. *For all tree-shapes S, the tile system $\mathbf{T}$ that minimizes $X^*(\mathbf{T})$ is also a minimum size tile system that uniquely assembles S.*

6. NP-HARDNESS OF MINIMUM TILE SET PROBLEM

Theorem 3.3 in section 3 states that the decision version of the Minimum Tile Set Problem is in NP. In this section, we sketch the proof that this problem is also NP-hard, and hence NP-Complete. In order to prove NP-Hardness, we reduce the 3CNF-SAT problem to the Minimum Tile Set Problem.

The language we wish to decide is:

$\mathbf{L} = \{(A, c, \tau)/$ There exists a tile system $\mathbf{T} = \langle T, \mathcal{S}, g, \tau \rangle$ with $|T| \leq c$ that uniquely assembles $A\}$

where A is a shape, c is a positive integer and $p \in A$.

Given a 3CNF-formula φ, we design two shapes. One is a tree $\Upsilon(\varphi)$ for which we can compute the minimum tile set efficiently; let c be the size of this minimum tile set. We show that φ is satisfiable iff the second shape $M(\varphi)$ can also be assembled using c distinct tile-types. An interesting part of the proof is that we rely on the polynomial time solution of the Minimum Tile Set Problem for trees.

Assume φ has n clauses $C_1, C_2, \cdots, C_n$ and m variables $v_1, v_2, \cdots, v_m$. Assume also the temperature $\tau = 2$.

6.1 Description of the tree

Informally, the tree is an encoding of the 3CNF-formula into a shape. There is a horizontal straight line L in the tree. Its length will be determined later. There are $4 \times n \times m + n$ gadgets (subtrees). Each gadget is on top of a vertical pillar attached to the horizontal line. The height of the pillar will be determined later. For ease of exposition, we assume that we can specify where the seed is placed. In particular, we assume the seed is placed in L immediately to the west of the westmost pillar. We further assume that the seed is placed at the same position in $\Upsilon(\varphi)$ and in $M(\varphi)$. It is easy to remove these assumption; details are omitted from this version.

Each gadget may be of one out of four possible classes.

1. $ST_{i,j}$, which represents that C_i is satisfied by at least one variable v_k with $k < j$, and v_j is true;

2. $UT_{i,j}$, which represents that C_i is unsatisfied by all variables v_k with $k < j$ and v_j is true;

3. $SF_{i,j}$, which represents that C_i is satisfied by at least one variable v_k with $k < j$, and v_j is false;

4. and $UF_{i,j}$, which represents that C_i is unsatisfied by all variables v_k with $k < j$, and v_j is false.

Each gadget is roughly a rectangle, with part of the south-east corner missing. The gadget has teeth on its edges. The teeth on the external side of the west and east edges represent whether C_i has been satisfied.

The teeth on the external side of the north and south edges represent the value of v_j. If v_j appears in C_i, C_i is unsatisfied by all variables v_k for $k < j$ and v_j makes C_i satisfied, then the gadget changes the tooth position on its east side accordingly. Otherwise a gadget will simply pass the information from west to east, and from south to north.

If $j = 1$, the west side has no teeth and we have only two gadgets for each $1 \leq i \leq n$: $UT_{i,j}$ and $UF_{i,j}$. If $i = 1$, the south side has no teeth. If $i = n$, the north side has no teeth.

All gadgets have teeth on the internal side of west edges, they encode a position as an $(i,\ j)$ ordered pair. The gadget can be fit into a $13 \times 2\lceil \log nm \rceil$ rectangle. There are n extra "side gadgets." The teeth on the external size of their west edges represent whether C_i has been satisfied. A typical regular gadget and a typical side gadget are shown in Figure 1, along with how the gadgets will be fit together to transfer information.

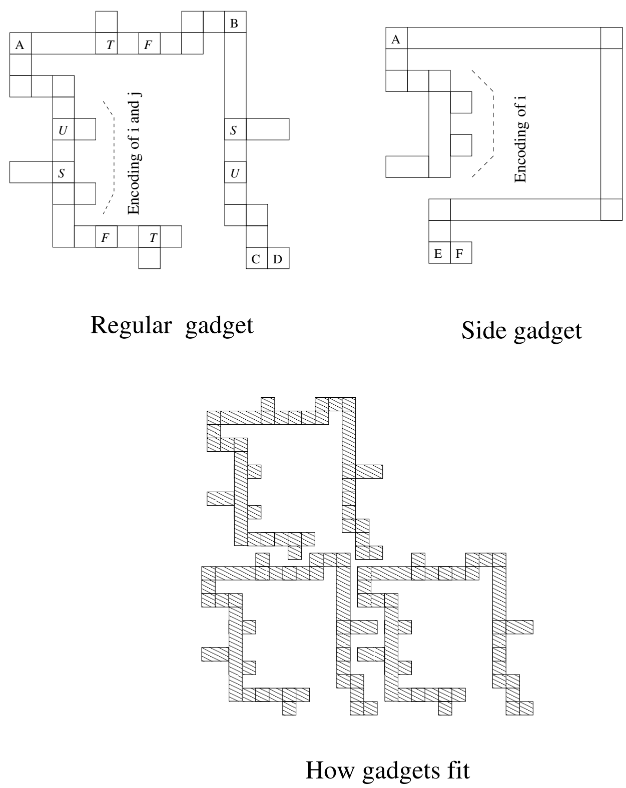
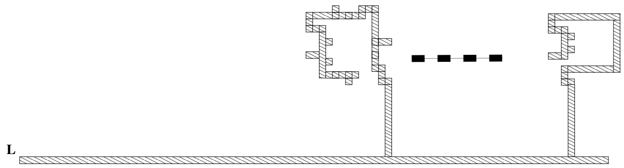


Figure 1: A typical regular ST gadget and a side gadget for the NP-hardness reduction

Each gadget attaches to a vertical line (pillar) of height $26m$, the line then sits on L.

The whole tree is sketched in Figure 2.



The tree

Figure 2: The sketch of a tree for the proof of NP-hardness

6.2 Description of the second shape

Given a 3CNF-formula, $M(\varphi)$ has $\Upsilon(\varphi)$ as a sub-shape. Besides $\Upsilon(\varphi)$, we will have a substructure, which can be thought as an $m \times n$ matrix of gadgets. The bottom level of gadgets in the matrix attach to pillars of height $26m$. The pillars supporting the matrix are attached to line L. The leftmost pillar supporting the matrix sits on a tile at position P, while the rightmost pillar supporting the matrix sits on tile at position Q. See Figure 3 for the sketch of the second structure.

The position encoding parts of gadgets will be arranged in such a way that i starts from 1 and increases by 1 in each gadget northward until it reaches n, j starts from 1 and increases by 1 in each gadget eastward until it reaches m. The side gadgets are placed on the east side of the matrix. Note that in the final assembly all teeth must be present.

The intuition behind the construction is: The tree substructure $\Upsilon(\varphi)$ encodes the clauses of the formula φ. The assembly of the matrix substructure of $M(\varphi)$ mimics the evaluation of φ. Choosing a particular set of gadgets is equivalent to choosing a truth assignment. Building the entire matrix using only gadgets is equivalent to satisfying φ.

6.3 The proof

The B tiles in the following lemma refer to Figure 1

LEMMA 6.1. *For all 3CNF-formulas φ, for all tile systems $\mathbf{T} = \langle T, \mathcal{S}, g, \tau \rangle$ that uniquely assemble $\Upsilon(\varphi)$, let w be the supertile assembled by $\mathbf{T}$ with shape $\Upsilon(\varphi)$. For all gadgets g let w_1 be the sub-supertile corresponding to g. The tile at position B in w_1 is of different type from any other tile in w.*

LEMMA 6.2. *For all 3CNF-formulas φ, for all tile systems $\mathbf{T} = \langle T, \mathcal{S}, g, \tau \rangle$ that uniquely assemble $\Upsilon(\varphi)$, let w be the supertile assembled by $\mathbf{T}$ with shape $\Upsilon(\varphi)$. For all pairs of positions (p_1, p_2) such that p_1 and p_2 are in the PQ segment and $p_1 \neq p_2$, the tiles at positions p_1 and p_2 in w are of different types.*

Both lemmas are proved by showing the tiles involved are part of different equivalent classes, as defined before Lemma 4.1. The details of the proofs are omitted.

Lemmas 6.3 through 6.6 describe some relations between the minimum number of tile-types to build $\Upsilon(\varphi)$ and $M(\varphi)$. We will use $A_{i,j}, B_{i,j}$ etc. to refer to tile-positions A, B etc. in the (i,j)-th gadgets.

LEMMA 6.3. *For all satisfiable 3CNF-formulas φ, if c is the minimum possible number of tiles types to uniquely assemble the tree $\Upsilon(\varphi)$, then there exists a tile system $\mathbf{T} = \langle T, \mathcal{S}, g, \tau \rangle$ such that $|T| = c$ and $\mathbf{T}$ uniquely assembles $M(\varphi)$.*

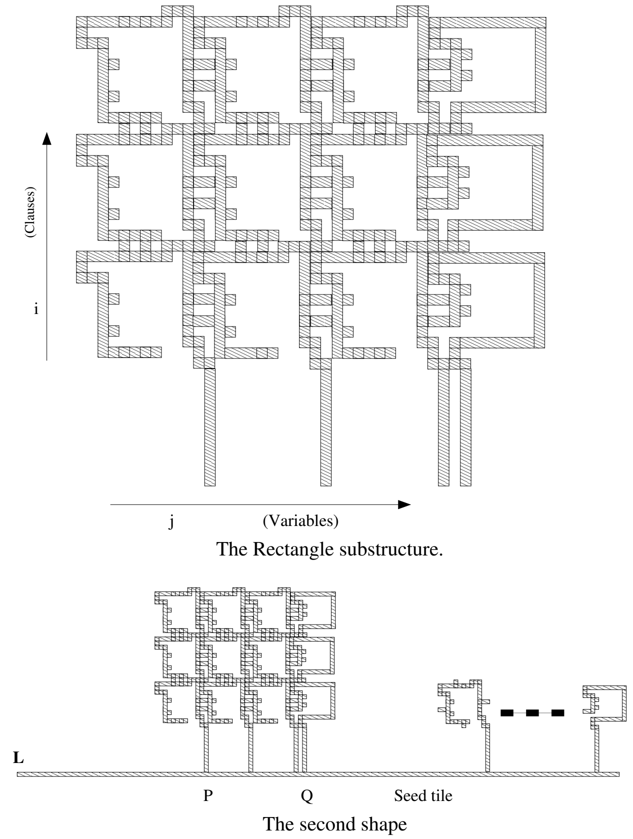


Figure 3: A sketch of the second structure used in the NP-hardness proof

PROOF SKETCH: Assume we know a satisfying assignment and we know the tile system $\mathbf{T} = \langle T, \mathcal{S}, g, \tau \rangle$ to build $\Upsilon(\varphi)$. It is possible to modify some glues in the tile-types in $\mathbf{T}$ to define a tile system $\mathbf{T}'$ that uniquely assembles $M(\varphi)$.

From lemma 6.1, B tiles are unique. We could derive an analogous lemma for C tiles, so we also know C tiles are unique. We can arrange the glues on east side of $B_{i,j}$ and north side of $A_{i+1,j}$, correspondingly the west and south side of $C_{i,j+1}$, to provide enough strength for the proper gadget of index $i, j+1$ to grow. Similarly, by Lemma 6.2 the tiles in segment PQ are unique, we can change the glue on the north side of the tiles without increasing the size of the tile set. ■

LEMMA 6.4. *For all 3CNF-formulas φ, for all $\mathbf{T} = \langle T, \mathcal{S}, g, \tau \rangle$ with $|T| \leq c$ that uniquely assembles $M(\varphi)$, if c is the minimum possible number of tiles types to uniquely build the tree $\Upsilon(\varphi)$, all pairs of adjacent tiles in the PQ segment in the supertile produced by $\mathbf{T}$, are connected by strength-2 bonds.*

PROOF SKETCH: Assume there is at least one strength-1 bond between adjacent tiles in PQ or two adjacent tiles in PQ are not bonded. L can be assembled only if at least one of the pillars supporting the matrix grows southward. The tiles used to construct a pillar growing southward cannot be of the same type of tiles using pillars growing upwards. If we tried to reuse tile-types as suggested in the previous sentence it would be possible to construct supertiles that do not have the shape $M(\varphi)$. The pillars are taller than $2|PQ|$, and we would have to introduce that many new tile-types to uniquely build $M(\varphi)$. The savings obtained by using tiles of the same type cannot outweigh the cost of building a pillar southward. The number of tile-types needed to build the substructure to the west of P and to the east of Q is independent of how we build PQ. The tile system would have more than c tiles. ■

LEMMA 6.5. *For all 3CNF-formulas φ, if c is the minimum possible number of tiles types to uniquely build the tree $\Upsilon(\varphi)$, then there is no tile system with less than c tile-types that uniquely assembles $M(\varphi)$. Further, given a tile system that uniquely assembles $M(\varphi)$ with c tile-types, it is possible to create a tile system of minimum size that uniquely assembles $\Upsilon(\varphi)$ by modifying only the north glues of tile-types used to build the PQ segment.*

PROOF SKETCH: Assume there is a tile system $\mathbf{T} = \langle T, \mathcal{S}, g, \tau\rangle$ with $|T| \leq c$ that uniquely assembles $M(\varphi)$ and $|T| < c$. Note that no tile of the same type as one used to build the PQ segment may appear in the terminal supertile except in the PQ segment. If this were possible, $\mathbf{T}$ would assemble something other than $M(\varphi)$. Consider the tile system $\mathbf{T}' = \langle T', \mathcal{S}, g, \tau\rangle$ created by modifying the tile-types used to build the PQ segment. The modification is removing the glue on the north side of these tile-types. $\mathbf{T}'$ assembles $\Upsilon(\varphi)$, the removal of north glues prevents assembling the matrix and its supporting pillars. By Lemma 6.4, all consecutive tiles in the PQ segment are connected by strength-2 bonds so the removal of north glues cannot prevent the growth of the PQ segment. The construction of the substructures to the to the west of P and to the east of Q is not affected.

$\mathbf{T}'$ then uniquely assembles $\Upsilon(\varphi)$, but $|T'| < c$. ■

Lemma 6.3 tell us that if φ is satisfiable then there exist a tile system of size c that uniquely assembles $M(\varphi)$, while Lemma 6.5 says that there is no tile system smaller than c to uniquely assembles $M(\varphi)$. Now we have to consider the case when φ is unsatisfiable:

LEMMA 6.6. *For all 3CNF-formulas φ, for all tile systems $\mathbf{T} = \langle T, \mathcal{S}, g, \tau\rangle$, if c is the minimum possible number of tiles types to uniquely build the tree $\Upsilon(\varphi)$ and $\mathbf{T}$ uniquely assembles $M(\varphi)$ and $|T| = c$ then φ is satisfiable.*

PROOF SKETCH: First we observe that all pillars supporting the matrix of $M(\varphi)$ must grow northward. The choice of glues to start the pillar growth represents a truth assignment to variables. We also observe that the matrix is built out gadgets. This means position (i, j) in the matrix is guaranteed to be built by a replica of a gadget with an (i, j) identifier. Uniqueness of assembly implies that the order used to add tiles is irrelevant wrt. the terminal supertile produced. We note that the matrix can be built from row 1 to row n. Further each row can be assembled from west to east. If the matrix is built in that order it is easy to see that if row k can be built then clause C_k is satisfied by the truth assignment. If the entire matrix can be built then all clauses can be satisfied by a single truth assignment. ■

As a corollary, if φ is unsatisfiable, we need more than c tile-types to uniquely assemble $M(\varphi)$. From Lemmas 6.3 and 6.6 we trivially prove a theorem that implies the Minimum Tile Set Problem is NP-hard.

THEOREM 6.7. *For all 3CNF-formulas φ, if c is the minimum possible number of tiles types to uniquely build the tree $\Upsilon(\varphi)$ then φ is satisfiable iff there exists a tile system $\mathbf{T} = \langle T, \mathcal{S}, g, \tau\rangle$ such that $|T| = c$ and $\mathbf{T}$ uniquely assembles $M(\varphi)$.*

7. OPEN PROBLEMS

1. Algorithm Unique-Shape in Section 3 assumes the strength of every bond is nonnegative. If the assumption does not hold, we do not know if it is possible to do the verification in polynomial time.

2. The algorithm to solve the Minimum Tile Set Problem for squares presented in Section 4, assumes the temperature is a constant. If we let the temperature be function of the size of the square, it is not clear if we can solve the problem by enumeration of tile systems in polynomial time.

3. As we proved, the Minimum Tile Set Problem is NP-hard in general, but for some families of shapes can be solved in polynomial time. In this paper we presented algorithms for trees and squares and "thick rectangles." We would like to know if we can solve the problem for other families of shapes such as thin rectangles or shapes without holes. Another interesting family could be the set of all shapes that are convex wrt. horizontal and vertical lines.

4. Our conjecture about the #P-hardness of the Optimal Concentrations problem has to be proved.

Acknowledgements

We would like to thank Matt Cook, Lila Kari, and Erik Winfree for useful discussions about self-assembly in general, and the topics studied in this paper in particular.

8. REFERENCES

[1] H. Abelson, D, Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman and R. Weiss. Amorphous Computing. Communications of the ACM vol 43, p 74-82, 2000.

[2] L. Adleman. Towards a mathematical theory of self-assembly. Technical Report 00-722, Department of Computer Science, University of Southern California, (2000)

[3] L. Adleman, Q. Cheng, A. Goel, M. Huang and Hal Wasserman. Linear Self-Assemblies: Equilibria, Entropy, and Convergence Rates. Unpublished.

[4] L. Adleman, Q. Cheng, A. Goel and M. Huang, Running time and program size for self-assembled squares, ACM Symposium on Theory of Computing (STOC) 2001. pages 740-748.

[5] M. Cook, D. Kempe, P. Rothemund, E. Winfree.

[6] M. Cook, D. Kempe, P. Rothemund, E. Winfree. Personal communication

[7] M. Gomez-Lopez, J. Preece, and J. Stoddart, The art and science of self-assembling molecular machines, Nanotechnology, Vol. 7, No. 3, pp. 183-192, September 1996

[8] D. Gracias, J. Tien, T. Breen, C. Hsu and G. Whitesides, Forming Electrical Networks in Three Dimensions by Self-Assembly, Science 289, 5482, p 1170-1173 (2000)

[9] M. Grotschel, L. Lovasz, and A. Schrijver. Geometric Algorithms and Combinatorial Optimization. Springer-Verlag, 1993 (2nd corrected edition).

[10] L. Khachian, A Polynomial Algorithm in Linear Programming. Soviet Mathematics Doklady 20, 191-194 (1979).

[11] G. Lopinski, D. Wayner and R. Wolkow. Self-Directed Growth of Molecular Nano-Structures on Silicon. Nature 406, 48 (2000).

[12] G. Louth, M. Mitzenmacher and F. Kelly. Computational complexity of loss networks, Theoretical Computer Science journal, (1994) vol 125, p 45-59.

[13] C. Mao, T. LaBean, J. Reif and N. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. Nature.407, 493-496. (2000)

[14] Lagoudakis and T. LaBean. 2D DNA Self-Assembly for Satisfiability. in DIMACS Series in Discrete Mathematics and Theoretical Computer Science 1999, Volume 54, Editors: E. Winfree and D.K. Gifford, Proceedings of the 5th DIMACS Workshop on DNA Based Computers; MIT: Cambridge. ISBN 0-8218-2053-2

[15] J. Reif. Local Parallel Biomolecular Computation. Third Annual DIMACS Workshop on DNA Based Computers, University of Pennsylvania, June 23-26, 1997. Published in DNA Based Computers, III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol 48 (ed. H. Rubin), American Mathematical Society, p 217-254, (1999)

[16] P. Rothemund. Using lateral capillary forces to compute by self-assembly. Proceedings of the National Academy of Sciences, vol 9. p 984-989. (2000)

[17] P. Rothemund. Theory and Experiments in Algorithmic Self-Assembly University of Southern California Ph.D. Thesis Copyright 2001

[18] P. Rothemund and E. Winfree. The program-size complexity of self-assembled squares. ACM Symposium on Theory of Computing (STOC) 2001. pages 459-468.

[19] J.H. Schön, H. Meng and Z. Bao. Self-assembled monolayer organic field-effect transistors. Nature. 413, 713-715. (2001)

[20] H. Wang. Proving theorems by pattern recognition. II. *Bell Systems Technical Journal*, 40:1-42, (1961)

[21] E. Winfree, X. Yang and N. Seeman, Universal Computation via Self-assembly of DNA: Some Theory and Experiments, Proceedings of the Second Annual Meeting on DNA Based Computers, Princeton University, June 10-12, (1996)

[22] E. Winfree, F. Liu, L. Wenzler, N. Seeman. Design and self-assembly of two-dimensional DNA crystals, 6 pages. (Nature 394, 539-544 (Aug. 6, 1998) Article)

[23] E. Winfree. Algorithmic Self-Assembly of DNA, Ph.D. thesis. California Institute of Technology, Pasadena, (1998)

[24] G. Whitesides, J. Mathias and Christopher T. Seto. Molecular self-assembly and nanochemistry: a chemical strategy for the synthesis of nanostructures, Science, vol 254, p 1312–1319. Nov 1991.