A Limit to the Power of Multiple Nucleation in Self-Assembly

Aaron D. Sterling*

Department of Computer Science, Iowa State University, Ames, IA 50014, USA.
sterling@cs.iastate.edu

Abstract. Majumder, Reif and Sahu presented in [7] a model of reversible, error-permitting tile self-assembly, and showed that restricted classes of tile assembly systems achieved equilibrium in (expected) polynomial time. One open question they asked was how the model would change if it permitted multiple nucleation, *i.e.,* independent groups of tiles growing before attaching to the original seed assembly. This paper provides a partial answer, by proving that no tile assembly model can use multiple nucleation to achieve speedup from polynomial time to constant time without sacrificing computational power: if a tile assembly system $\mathcal{T}$ uses multiple nucleation to tile a surface in constant time (independent of the size of the surface), then $\mathcal{T}$ is unable to solve computational problems that have low complexity in the (single-seeded) Winfree-Rothemund Tile Assembly Model. The proof technique defines a new model of distributed computing that simulates tile assembly, so a tile assembly model can be described as a distributed computing model.
Keywords: self-assembly, multiple nucleation, locally checkable labeling.

1 Introduction

1.1 Overview

Nature is replete with examples of the self-assembly of individual parts into a more complex whole, such as the development from zygote to fetus, or, more simply, the replication of DNA itself. In his Ph.D. thesis in 1998, Winfree proposed a formal mathematical model to reason algorithmically about processes of self-assembly [15]. Winfree connected the experimental work of Seeman [12] (who had built "DNA tiles," molecules with unmatched DNA base pairs protruding in four directions, so they could be approximated by squares with different "glues" on each side) to a notion of tiling the integer plane developed by Wang in the 1960s [14]. Rothemund, in his own Ph.D. thesis, extended Winfree's original Tile Assembly Model [10].

Informally speaking, Winfree effectivized Wang tiling, by requiring a tiling of the plane to start with an individual *seed tile* or a connected, finite *seed assembly.* Tiles would then accrete one at a time to the seed assembly, growing a *seed*

* This research was supported in part by National Science Foundation Grants 0652569 and 0728806.

supertile. A *tile assembly system* is a finite set of differently defined *tile types*. Tile types are characterized by the names of the "glues" they carry on each of their four sides, and the binding strength each glue can exert. We assume that when the tiles interact "in solution," there are infinitely many tiles of each tile type. Tile assembly proceeds in discrete stages. At each stage s, from all possibilities of tile attachment at all possible locations (as determined by the glues of the tile types and the binding requirements of the system overall), one tile will bind. If more than one tile type can bind at stage s, a tile type and location will be chosen uniformly at random. Winfree proved that his Tile Assembly Model is Turing universal, so it is a robust model of computation.

The standard Winfree-Rothemund tile assembly model is *error-free* and *irreversible*—tiles always bind correctly, and, once a tile binds, it can never unbind. Adleman *et al.* were the first to define a notion of time complexity for tile assembly, using a one-dimensional error-permitting, reversible model, where tiles would assemble in a line with some error probability, then be scrambled, and fall back to the line [1]. Adleman *et al.* proved bounds on how long it would take such models to achieve equilibrium. Majumder, Reif and Sahu have recently presented a two-dimensional stochastic model for self-assembly [7], and have shown that some tiling problems in their model correspond to *rapidly mixing Markov chains*—Markov chains that reach stationary distribution in time polynomial in the state space. The tile assembly model in [7], like the standard model, allows only for a single seed assembly, and one of the open problems in [7] was how the model might change if it allowed multiple nucleation, *i.e.*, if multiple supertiles could build independently before attaching to a growing seed supertile.

The main result of this paper provides a time complexity lower bound for tile assembly models that permit multiple nucleation: there is no way to use multiple nucleation to achieve a speedup to tiling a surface in constant time (time independent of the size of the surface) without sacrificing computational power. This result holds for tile assembly models that are reversible, irreversible, error-permitting or error-free. In fact, a speedup to constant time is impossible, even if we relax the model to allow that, at each step s, there is a positive probability for every available location that a tile will bind there (instead of requiring that exactly one tile bind per stage).

Our method of proof appears novel: given a tile assembly model and a tile assembly system $\mathcal{T}$ in that model, we construct a distributed network of processors that can simulate the behavior of $\mathcal{T}$ as it assembles on a surface. Our result then follows from the theorem by Naor and Stockmeyer that locally checkable labeling (LCL) problems have no local solution in constant time [8]. This is true for both deterministic and randomized algorithms, so no constant-time tile assembly system exists that solves an LCL problem with a positive probability of success. We consider one LCL problem in specific, the weak c-coloring problem, and demonstrate a tile set of only seven tile types that solves the weak c-coloring problem in the Winfree-Rothemund Tile Assembly Model, even though weak c-coloring is impossible to achieve in constant time by multiple nucleation, regardless of the rate of convergence to equilibrium.

1.2 Background

In the standard Tile Assembly Model, one tile is added per stage, so the primary complexity measure is not one of time, but of how much information a tile set needs in order to solve a particular problem. Several researchers [1] [3] [4] [11] [13] have investigated the *tile complexity* (the minimum number of distinct tile types required for assembly) of finite shapes, and sets of "scale-equivalent" shapes (essentially a $\mathbb{Z} \times \mathbb{Z}$ analogue of the Euclidean notion of similar figures). For example, it is now known that the number of tile types required to assemble a square of size $n \times n$ (for n any natural number) is $\Omega(\log n / \log \log n)$ [11]. Or, if T is the set of all discrete equilateral triangles, the asymptotically optimal relationship between triangle size and number of tiles required to assemble that triangle, is closely related to the Kolmogorov Complexity of a program that outputs the triangle as a list of coordinates [13].

Despite these advances in understanding of the complexity of assembling finite, bounded shapes, the self-assembly of infinite structures is not as well understood. In particular, there are few lower bounds or impossibility results on what infinite structures can be self-assembled in the Tile Assembly Model. The first such impossibility result appeared in [6], when Lathrop, Lutz and Summers showed that no finite tile set can assemble the discrete Sierpinski Triangle by placing a tile only on the coordinates of the shape itself. (By contrast, Winfree had shown that just seven tile types are required to tile the first quadrant of the integer plane with tiles of one color on the coordinates of the discrete Sierpinski Triangle, and tiles of another color on the coordinates of the complement [15].) Recently, Patitz and Summers have extended this initial impossibility result to other discrete fractals [9], and Lathrop *et al.* [5] have demonstrated sets in $\mathbb{Z} \times \mathbb{Z}$ that are Turing decidable but cannot be self-assembled in Winfree's sense.

To date, there has been no work comparing the strengths of different tile assembly models with respect to infinite (nor to finite but arbitrarily large) structures. Since self-assembly is an asynchronous process in which each point has only local knowledge, it is natural to consider whether the techniques of distributed computing might be useful for comparing models and proving impossibility results in nanoscale self-assembly. This paper is an initial attempt in that direction.

Aggarwal *et al.* in [3] proposed a generalization of the standard Tile Assembly Model, which they called the q-Tile Assembly Model. This model permitted multiple nucleation: tiles did not need to bind immediately to the seed supertile. Instead, they could form independent supertiles of size up to some constant q before then attaching to the seed supertile. While the main question considered in [3] was *tile* complexity, we can also ask whether multiple nucleation would allow an improvement in *time* complexity. Intuitively, Does starting from multiple points allow us to build things strictly faster than starting from a single point?

As mentioned above, Majumder, Reif and Sahu recently presented a stochastic, error-permitting tile assembly model, and calculated the rate of convergence to equilibrium for several tile assembly systems [7]. The model in [7] permitted only a single seed assembly, and addition of one tile to the seed supertile at each

stage. Majumder, Reif and Sahu left as an open question how the model might be extended to permit the presence and binding of multiple supertiles.

Therefore, we can rephrase the "intuitive" question above as follows: Can we tile a surface of size $n \times n$ in a constant number of stages, by randomly selecting nucleation points on the surface, building supertiles of size q or smaller from those points in $\leq q$ stages, and then allowing $\leq r$ additional stages for tiles to fall off and be replaced if the edges of the supertiles contain tiles that bind incorrectly? (The assembly achieves equilibrium in constant time because q and r do not depend on n.)

The main result of this paper is that the answer is: Not without losing significant computational power.

Section 2 of this paper describes the "standard" Winfree-Rothemund Tile Assembly Model, and then considers generalizations of the standard model that permit multiple nucleation. Section 3 reviews the distributed computing results of Naor and Stockmeyer needed to prove the impossibility result. In Section 4 we present our main result. Section 5 concludes the paper and suggests directions for future research.

2 Description of Tile Assembly Models

2.1 The Winfree-Rothemund Tile Assembly Model

Winfree's objective in defining the Tile Assembly Model was to provide a useful mathematical abstraction of DNA tiles combining in solution in a random, nondeterministic, asynchronous manner [15]. Rothemund [10], and Rothemund and Winfree [11], extended the original definition of the model. For a comprehensive introduction to tile assembly, we refer the reader to [10]. In our presentation here, we follow [6], which gives equal status to finite and infinite tile assemblies. Throughout this paper, we will consider only two-dimensional tile assemblies.

Intuitively, a tile of type t is a unit square that can be placed with its center on a point in the integer lattice. A tile has a unique orientation; it can be translated, but not rotated. We identify the side of a tile with the direction (or unit vector) one must travel from the center to cross that side. The literature often refers to west, north, east and south sides, starting at the leftmost side and proceeding clockwise. Each side $\overrightarrow{u} \in U_2$ (where U_2 is the set of unit vectors in two coordinates) of a tile is covered with a "glue" that has *color* $\text{col}_t(\overrightarrow{u})$ and *strength* $\text{str}_t(\overrightarrow{u})$. Figure 1 shows how a tile is represented graphically.

If tiles of types t and t' are placed adjacent to each other (*i.e.*, with their centers at $\overrightarrow{m}$ and $\overrightarrow{m} + \overrightarrow{u}$, where $\overrightarrow{m} \in \mathbb{Z}^2$ and $\overrightarrow{u} \in U_2$) then they will *bind* with strength $\text{str}_t(\overrightarrow{u}) \cdot [\![t(\overrightarrow{u}) = t'(-\overrightarrow{u})]\!]$, where $[\![\phi]\!]$ is the Boolean value of the statement ϕ. Note that this definition of binding implies that if the glues of the adjacent sides do not have the same color or strength, then their binding strength is 0. Later, we will permit pairs of glues to have negative binding strength, to model error occurrence and correction.

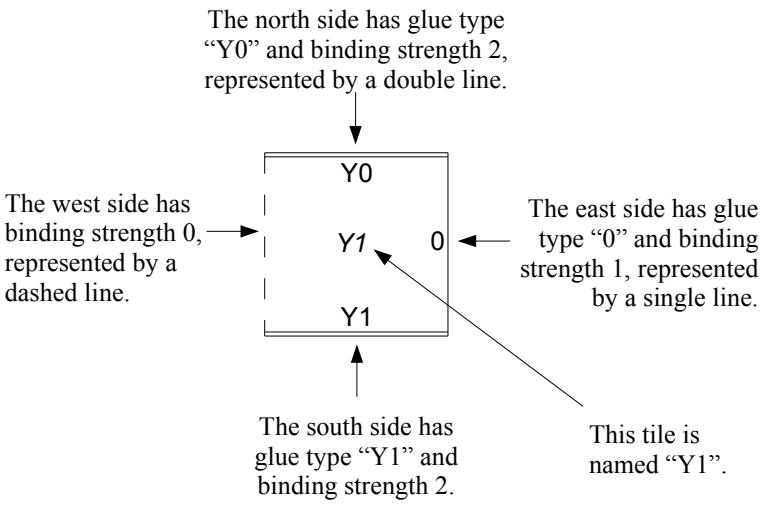

Fig. 1. An example tile with explanation.

One parameter in a tile assembly model is the minimum binding strength required for tiles to bind "stably." This parameter is usually termed *temperature* and denoted by τ, where $\tau \in \mathbb{N}$.

As we consider only two-dimensional tile assemblies, we limit ourselves to working in $\mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z}$. U_2 is the set of all unit vectors in $\mathbb{Z}^2$.

A *binding function* on an (undirected) graph $G = (V, E)$ is a function $\beta : E \longrightarrow \mathbb{N}$. If β is a binding function on a graph $G = (V, E)$ and $C = (C_0, C_1)$ is a cut of G, then the *binding strength* of β on C is

$$\beta_C = \{\beta(e) \mid e \in E, \{e\} \cap C_0 \neq \emptyset, \text{ and } \{e\} \cap C_1 \neq \emptyset\} \ .$$

The *binding strength* of β on G is then $\beta(G) = \min\{\beta_C \mid C \text{ is a cut of } G\}$. Intuitively, the binding function captures the strength with which any two neighbors are bound together, and the binding strength of the graph is the minimum strength of bonds that would have to be severed in order to separate the graph into two pieces.

A *binding graph* is an ordered triple $G = (V, E, \beta)$ where (V, E) is a graph and β is a binding function on (V, E). If $\tau \in \mathbb{N}$, a binding graph $G = (V, E, \beta)$ is *τ-stable* if $\beta(V, E) \geq \tau$.

Recall that a *grid graph* is a graph $G = (V, E)$ where $V \subseteq \mathbb{Z} \times \mathbb{Z}$ and every edge $\{\vec{m}, \vec{n}\} \in E$ has the property that $\vec{m} - \vec{n} \in U_2$.

Definition 1. *A* tile type *over a (finite) alphabet Σ is a function $t : U_2 \longrightarrow \Sigma^* \times \mathbb{N}$. We write $t = (\mathrm{col}_t, \mathrm{str}_t)$, where $\mathrm{col}_t : U_2 \longrightarrow \Sigma^*$, and $\mathrm{str}_t : U_2 \longrightarrow \mathbb{N}$ are defined by $t(\vec{u}) = (\mathrm{col}_t(\vec{u}), \mathrm{str}_t(\vec{u}))$ for all $\vec{u} \in U_2$.*

Definition 2. *If T is a set of tile types, a* T-configuration *is a partial function* $\alpha : \mathbb{Z}^2 \dashrightarrow T$.

Definition 3. *The* binding graph *of a T-configuration $\alpha : \mathbb{Z}^2 \dashrightarrow T$ is the binding graph $G_\alpha = (V, E, \beta)$, where (V, E) is the grid graph given by*

$$V = \mathrm{dom}(\alpha),$$
$$E = \big\{\{\vec{m}, \vec{n}\} \in [V]^2 \mid \vec{m} - \vec{n} \in U_2, \mathrm{col}_{\alpha(\vec{m})}(\vec{n} - \vec{m}) = \mathrm{col}_{\alpha(\vec{n})}(\vec{m} - \vec{n}), \text{ and } \mathrm{str}_{\alpha(\vec{m})}(\vec{n} - \vec{m}) > 0\big\},$$

and the binding function $\beta : E \longrightarrow \mathbb{Z}^+$ is given by $\beta(\{\vec{m}, \vec{n}\}) = \mathrm{str}_{\alpha(\vec{m})}(\vec{n} - \vec{m})$ for all $\{\vec{m}, \vec{n}\} \in E$.

Definition 4. *For T a set of tile types, a T-configuration α is* stable *if its binding graph G_α is τ-stable. A τ-T-*assembly *is a T-configuration that is τ-stable. We write $\mathcal{A}_T^\tau$ for the set of all τ-T-assemblies.*

Definition 5. *Let α and α' be T-configurations.*

1. *α is a* subconfiguration *of α', and we write $\alpha \sqsubseteq \alpha'$, if $\mathrm{dom}(\alpha) \subseteq \mathrm{dom}(\alpha')$ and, for all $\vec{m} \in \mathrm{dom}(\alpha)$, $\alpha(\vec{m}) = \alpha'(\vec{m})$.*
2. *α' is a* single-tile extension *of α if $\alpha \sqsubseteq \alpha'$ and $\mathrm{dom}(\alpha') \smallsetminus \mathrm{dom}(\alpha)$ is a singleton set. In this case, we write $\alpha' = \alpha + (\vec{m} \mapsto t)$, where $\{\vec{m}\} = \mathrm{dom}(\alpha') \smallsetminus \mathrm{dom}(\alpha)$ and $t = \alpha'(\vec{m})$.*
3. *The notation $\alpha \xrightarrow[\tau,T]{1} \alpha'$ means that $\alpha, \alpha' \in \mathcal{A}_T^\tau$ and α' is a single-tile extension of α.*

Definition 6. *Let $\alpha \in \mathcal{A}_T^\tau$.*

1. *For each $t \in T$, the τ-t-*frontier *of α is the set*

$$\partial_T^\tau \alpha = \left\{ \vec{m} \in \mathbb{Z}^2 \smallsetminus \mathrm{dom}(\alpha) \middle| \sum_{\vec{u} \in U_2} \mathrm{str}_t(\vec{u}) \cdot [\![\alpha(\vec{m} + \vec{u})(-\vec{u}) = t(\vec{u})]\!] \geq \tau \right\} .$$

2. *The τ-*frontier *of α is the set*

$$\partial^\tau \alpha = \bigcup_{t \in T} \partial_t^\tau \alpha \ .$$

Definition 7. *A τ-T-*assembly sequence *is a sequence $\vec{\alpha} = (\alpha_i \mid 0 \leq i < k)$ in $\mathcal{A}_T^\tau$, where $k \in \mathbb{Z}^+ \cup \{\infty\}$ and, for each i with $1 \leq i + 1 < k$, $\alpha_i \xrightarrow[\tau,T]{1} \alpha_{i+1}$.*

Definition 8. *The* result *of a τ-T-assembly sequence $\vec{\alpha} = (\alpha_i \mid 0 \leq i < k)$ is the unique T-configuration $\alpha = \mathrm{res}(\vec{\alpha})$ satisfying: $\mathrm{dom}(\alpha) = \cup_{0 \leq i < k} \mathrm{dom}(\alpha_i)$ and $\alpha_i \sqsubseteq \alpha$ for each $0 \leq i < k$.*

Definition 9. *Let $\alpha, \alpha' \in \mathcal{A}_T^\tau$. A τ-T-*assembly sequence from α to α' *is a τ-T-assembly sequence $\vec{\alpha} = (\alpha_i \mid 0 \leq i < k)$ such that $\alpha_0 = \alpha$ and $\mathrm{res}(\vec{\alpha}) = \alpha'$. We write $\alpha \xrightarrow[\tau,T]{} \alpha'$ to indicate that there exists a τ-T-assembly from α to α'.*

Definition 10. *An assembly $\alpha \in \mathcal{A}_T^\tau$ is* terminal *if $\partial^\tau \alpha = \emptyset$.*

Intuitively, a configuration is a set of tiles that have been placed in the plane, and the configuration is stable if the binding strength at every possible cut is at least as high as the temperature of the system. Informally, an assembly sequence is a sequence of single-tile additions to the frontier of the assembly constructed at the previous stage. Assembly sequences can be finite or infinite in length. We are now ready to present a definition of a tile assembly system.

Definition 11. *Write $\mathcal{A}_T^\tau$ for the set of configurations, stable at temperature τ, of tiles whose tile types are in T. A* tile assembly system *is an ordered triple $\mathcal{T} = (T, \sigma, \tau)$ where T is a finite set of tile types, $\sigma \in \mathcal{A}_T^\tau$ is the* seed assembly*, and $\tau \in \mathbb{N}$ is the* temperature*. We require* $\mathrm{dom}(\sigma)$ *to be finite.*

Definition 12. *Let $\mathcal{T} = (T, \sigma, \tau)$ be a tile assembly system.*

1. *Then the* set of assemblies produced by $\mathcal{T}$ *is*

$$\mathcal{A}[\mathcal{T}] = \left\{ \alpha \in \mathcal{A}_T^\tau \middle| \sigma \underset{\tau,T}{\longrightarrow} \alpha \right\} \ ,$$

   where "$\sigma \underset{\tau,T}{\longrightarrow} \alpha$" means that tile configuration α can be obtained from seed assembly σ by a legal addition of tiles (as formalized in Appendix A).
2. *The* set of terminal assemblies produced by $\mathcal{T}$ *is*

$$\mathcal{A}_\square[\mathcal{T}] = \{\alpha \in \mathcal{A}[\mathcal{T}] \mid \alpha \text{ is terminal}\} \ ,$$

   where "terminal" describes a configuration to which no tiles can be legally added.

If we view tile assembly as the programming of matter, the following analogy is useful: the seed assembly is the input to the computation; the tile types are the legal (nondeterministic) steps the computation can take; the temperature is the primary inference rule of the system; and the terminal assemblies are the possible outputs.

We are, of course, interested in being able to *prove* that a certain tile assembly system always achieves a certain output. In [13], Soloveichik and Winfree presented a strong technique for this: local determinism.

Informally, an assembly sequence $\overrightarrow{\alpha}$ is locally deterministic if (1) each tile added in $\overrightarrow{\alpha}$ binds with the minimum strength required for binding; (2) if there is a tile of type t_0 at location $\overrightarrow{m}$ in the result of α, and t_0 and the immediate "OUT-neighbors" of t_0 are deleted from the result of α, then no other tile type in $\mathcal{T}$ can legally bind at $\overrightarrow{m}$; the result of α is terminal.

Definition 13 (Soloveichik and Winfree [13]). *A τ-$\mathcal{T}$-assembly sequence $\overrightarrow{\alpha} = (\alpha_i \mid 0 \leq i \leq k)$ with result α is* locally deterministic *if it has the following three properties.*

1. *For all* $\overrightarrow{m} \in dom(\alpha) - dom(\alpha_0)$*,*

$$\sum_{\overrightarrow{u} \in \mathrm{IN}^{\alpha}(\overrightarrow{m})} \mathrm{str}_{\alpha_{i_\alpha(\overrightarrow{m})}}(\overrightarrow{m}, \overrightarrow{u}) = \tau \ .$$

2. *For all* $\overrightarrow{m} \in dom(\alpha) - dom(\alpha_0)$ *and all* $t \in \mathcal{T} - \{\alpha(\overrightarrow{m})\}$, $\overrightarrow{m} \notin \partial_t^{\tau}(\overrightarrow{\alpha} \backslash \overrightarrow{m})$.
3. $\partial^{\tau} \alpha = \emptyset$.

Definition 14 (Soloveichik and Winfree [13]). *A tile assembly system* $\mathcal{T}$ *is* locally deterministic *if there exists a locally deterministic* τ*-*$\mathcal{T}$*-assembly sequence* $\alpha = (\alpha_i \mid 0 \leq i < k)$ *with* $\alpha_0 = \sigma$.

Local determinism is important because of the following result.

Theorem 1 (Soloveichik and Winfree [13]). *If* $\mathcal{T}$ *is locally deterministic, then* $\mathcal{T}$ *has a unique terminal assembly.*

2.2 Generalizations of the Winfree-Rothemund Tile Assembly Model

We will consider three generalizations of the standard tile assembly model: (1) multiple nucleation; (2) assembly in which glues bind incorrectly according to some error probability; and (3) negative glue strengths, allowing incorrectly bound tiles to be released from the assembly so it is possible for a correctly-binding tile to attach in that space. We move from an *irreversible* tiling model, in which tiles are placed in an error-free manner and can never be removed, to a *reversible* tiling model, in which a terminal assembly is defined by equilibrium, not by the disappearance of a frontier to which tiles can be legally added.

Aggarwal *et al.* in [3] formulated and studied a model that permitted multiple nucleation, which they called the *q-tile* or *multiple tile* model. Essentially, they allowed supertiles to form, independent of the seed, up to size bounded by a constant q. Then the independent supertile would have to bind to the growing seeed supertile. Legal supertiles were defined recursively: each tile type was a legal supertile, and any two supertiles whose combined size was $\leq q$ could form a legal supertile if the binding strength at their adjacent frontiers was at least the temperature of the system.

Models of reversible tiling have been considered in [15] and [1], and more recently in [7], which contains a summary of previous work in the area. Majumder, Reif and Sahu in [7] introduced the concept of *bond pair equilibrium*, as follows.

Definition 15 (Majumder, Reif and Sahu [7]). *Suppose* α *is a finite configuration that contains* m *different tile types* $t_1, \ldots, t_m$*, with* γ_i *the relative fraction of tiles of type* t_i *(so* $\sum \gamma_i = 1$*).*

1. *Define* a_{ij} *to be the fraction of* t_i *tiles bonded to the east to a* t_j *tile.*
2. *Define* b_{ik} *to be the fraction of* t_i *tiles bonded to the north to a* t_k *tile.*
3. *Define* p_{ij} *to be the fraction of* t_i *tiles bonded to the west to a* t_j *tile.*
4. *Define* q_{ik} *to be the fraction of* t_i *tiles bonded to the south to a* t_k *tile.*

5. $A_{ij} = \gamma_i a_{ij}$. $B_{ik} = \gamma_i b_{ik}$.

Definition 16 (Majumder, Reif and Sahu [7]). *A configuration α in an error-permitting, reversible tile assembly system has achieved* bond pair equilibrium *when, for every tile type t_i in α, the (expected value of the) number of pairs (A_{ij}, B_{kj}) is invariant over time steps.*

Informally, bond pair equilibrium is achieved when, if the configuration is considered as a whole, the quantity of each distinct bond interaction does not change over time. If we assume the system has a property of *bond independence*—the bond on one side of a tile does not affect the binding on the other three sides—then bond pair equilibrium is a sufficient condition for thermodynamic equilibrium.

Theorem 2 (Majumder, Reif and Sahu [7]). *Bond pair equilibrium and bond independence implies strong (thermodynamic) equilibrium.*

This theorem provides justification for us to replace the notion of terminal assembly with the notion of assembly that has achieved bond pair equilibrium, if we relax the Winfree-Rothemund Tile Assembly Model to include the possiblity of error in binding, and the reversibility of tile assembly.

Majumder, Reif and Sahu studied the rate of convergence of several tile assembly systems in a model that only permitted addition of one tile at a given time step. They defined the notion of a Markov Chain that corresponds to an assembly system, and demonstrated several tile assembly systems whose Markov chains were rapidly mixing, *i.e.,* they reached stationary distribution in time polynomial in the state space.

In what follows, we will see that a speedup to constant time is impossible without losing computational power, even if we add multiple nucleation to a model of reversible tile assembly. First, though, we review the distributed computing impossibility results that imply this.

3 Distributed Computing Results of Naor and Stockmeyer

In a well known distributed computing paper, Naor and Stockmeyer investigated whether "locally checkable labeling" problems could be solved over a network of processors in an entirely local manner, where a local solution means a solution arrived at "within time (or distance) independent of the size of the network" [8]. One locally checkable labeling problem Naor and Stockmeyer considered was the *weak c-coloring problem.*

Definition 17 (Naor and Stockmeyer [8]). *For $c \in \mathbb{N}$, a* weak c-coloring *of a graph is an assignment of numbers from $\{1, \ldots, c\}$ (the possible "colors") to the vertices of the graph such that for every non-isolated vertex v there is at least one neighbor w such that v and w receive different colors. Given a graph G, the* weak c-coloring problem for G *is to weak c-color the nodes of G.*

In the context of tiling, to solve the weak c-coloring problem for an $n \times n$ surface means tiling the surface so each tile has at least one neighbor (to the north, south, east or west) of a different color. In the next section, we will present a simple solution to the weak c-coloring problem in the Winfree-Rothemund Tile Assembly Model. By contrast, Naor and Stockmeyer showed that no local, constant-time algorithm can solve the weak c-coloring problem for grid graphs.

Theorem 3 (Naor and Stockmeyer [8]). *For any c and t, there is no local algorithm with time bound t that solves the weak c-coloring problem for the class of finite square grid graphs over the integer lattice.*

This theorem is a consequence of Theorem 6.3 in [8]. The original result is a stronger statement.

A second theorem from the same paper says that randomization does not help. As before, the original result is stronger than the formulation I provide here.

Theorem 4 (Naor and Stockmeyer [8]). *Fix a class $\mathcal{G}$ of graphs closed under disjoint union. If there is a randomized local algorithm P with time bound t that solves the weak c-coloring problem for $\mathcal{G}$ with error probability ϵ for some $\epsilon < 1$, then there is a deterministic local algorithm A with time bound t that solves the weak c-coloring problem for $\mathcal{G}$.*

4 Proof of Main Result

In order to apply the theorems of Naor and Stockmeyer to the realm of tile assembly, we build a distributed network of processors that simulates assembly of tile assembly system $\mathcal{T}$ in tile assembly model $\mathcal{M}$. We accomplish this by defining a class of tile assembly models that generalize the standard model and permit multiple nucleation; and we show that for any tileset defined in that class of models, there is a system of distributed processors that simulates the assembly behavior of that tileset.

Theorem 5. *For any (reversible or irreversible) tile assembly model $\mathcal{M}$ that permits multiple nucleation, and any tile set $\mathcal{T}$ in $\mathcal{M}$, there is a model of distributed computing $\mathcal{N}$ that simulates the assembly of $\mathcal{T}$ on a surface of size n^2, using n^2 processors laid out in a grid graph, and constant-size message complexity.*

Proof. Fix a tile assembly model $\mathcal{M}$ with the following properties:

1. The binding function β of $\mathcal{M}$ assigns a real number to each pair of glue types. This assignment can be positive, zero or negative.
2. The definition of the binding function β and the definition of each tile type t_i induces a function

$$\hat{\beta} : T \times (\{\text{glue colors of } T, \text{glue strengths of } T\} \cup \{\emptyset\})^4 \longrightarrow [0, 1] \ ,$$

such that for any T-configuration α and any location $\overrightarrow{m}$ at stage s,

$$\hat{\beta}\big[\alpha(\overrightarrow{m}), \alpha(\overrightarrow{m}+(1,0)), \alpha(\overrightarrow{m}+(-1,0)), \alpha(\overrightarrow{m}+(0,1)), \alpha(\overrightarrow{m}+(0,-1))\big]$$

is the probability that the tile at location $\overrightarrow{m}$ will remain in that location at the end of stage s. (In words, $\hat{\beta}$ is a function from a tile type and each possible set of glues—including no glue—adjacent to that tile type, to a probability that the tile will remain in that location at the end of the stage.) Note that in a model of irreversible tiling, if there is a tile in location $\overrightarrow{m}$ that is part of configuration α, then we can drop the part of $\hat{\beta}$ that depends on the tile's neighbors, and $\hat{\beta}[\alpha(\overrightarrow{m})]$ always takes the value 1.

3. $\mathcal{M}$ can allow multiple nucleation. In addition to the placement of the seed assembly at the first stage of assembly, there is some probability π such that (at the first stage of assembly only) a tile is placed on each location of the surface in question with probability π, determined uniformly at random. (Note that if $\pi = 0$, then $\mathcal{M}$ does not allow multiple nucleation.)
4. At each stage s of assembly, there is a probability $\pi_{s,\overrightarrow{m}}$ for each location $\overrightarrow{m}$ in the frontier of each supertile that a tile will be placed there. In particular, it is possible to place more than one tile per stage. Tiles that are placed in stage s do not interact with one another (with either positive or negative binding strength) until stage $s+1$.

For example, if we want $\mathcal{M}$ to be the standard Winfree-Rothemund Tile Assembly Model, we set all values of β to 0 or a positive integer, all values of $\hat{\beta}$ to 1, $\pi = 0$, and the values of $\pi_{s,\overrightarrow{m}}$ sufficiently small for all stages s and locations $\overrightarrow{m}$ that, with high probability, at most one tile appears per stage. Then we count time steps only when a tile is added to the existing configuration.

We simulate assembly sequences of $\mathcal{T}$ on an $n \times n$ surface by a network of processors $\mathcal{N}$ whose network graph is an $n \times n$ grid graph. Each processor will simulate the presence or absence of a tile in the same location on the $n \times n$ tiling surface. Processors do not have unique ID's, and do not know their own coordinates. Each processor $p_i \in \mathcal{N}$ is of the following form.

Processor p_i

Four input message buffers: $\text{inbuf}_{i,n}$, $\text{inbuf}_{i,s}$, $\text{inbuf}_{i,e}$ and $\text{inbuf}_{i,w}$.

Four output message buffers: $\text{outbuf}_{i,n}$, $\text{outbuf}_{i,s}$, $\text{outbuf}_{i,e}$ and $\text{outbuf}_{i,w}$.

A color variable: COLOR_i, a variable that can take a value from $\{1, \ldots, c\}$, where c is a global constant.

A local state: Each processor is in one of $|T|+1$ different local states q during a given execution stage s. There is one stage q_k to simulate each tile type $t_k \in T$, and an additional stage QUIET, to simulate the absence of a tile from the surface location that p_i is simulating.

A state transition function: This function takes the current processor state and the messages received in the current round, and (deterministically or probabilistically, depending on $\mathcal{M}$) directs what state the processor will adopt in the next round.

The messages processors send on the network are of form ⟨glue type, glue strength⟩. The input message buffers of processor p_i simulate the glue types of the edges the tile at p_i's location is adjacent to. The output message buffers of p_i simulate the glues on the edges of the tile p_i is simulating. The purpose of COLOR_i is to simulate the color of the tile placed at the location simulated by p_i.

All processors in $\mathcal{N}$ are hardcoded with the same state transition function, which is determined from the definition of $\hat{\beta}$ in $\mathcal{M}$, in the natural way: if, in round r of the algorithm execution, p_i is in state q_k, a simulation of $t_k \in T$, and hears messages that simulate glue types $g_1, \ldots, g_4$, then at the end of round r, if $\hat{\beta}(t_k, g_1, g_2, g_3, g_4) = \gamma$, then with probability γ the transition function directs p_i to remain in state q_k, and with probability $1 - \gamma$ to enter state QUIET.

To simulate the process of tile assembly, we run the following distributed algorithm on $\mathcal{N}$.

Algorithm execution proceeds in synchronized rounds. Before execution begins, all processors start in state QUIET. In round $r = 0$, (through the intervention of an omniscient operator) each processor in the locations corresponding to the seed assembly enters the stage to simulate the tile type at that location in the seed assembly.

Also in round $r = 0$, each processor not simulating part of the seed assembly "wakes up" (enters a state other than QUIET) with probability π. If a processor wakes up, it enters state $q \neq$ QUIET, chosen uniformly at random. For any round $r > 0$, each processor runs either Algorithm 1 or Algorithm 2, depending on whether it is in state QUIET.

Algorithm 1 For p_i in state QUIET at round r

```
if r = 0 then
    wake up with probability π, and cease execution for this round.
end if
if r > 0 then
    Read the four input buffers.
    if no messages were received then
        cease execution for the round
    else
        let q_0 be the state change (probabilistically) indicated by the value of β̂ for
        a location that has adjacent glue types that are simulated by the messages
        received this round.
        Send the messages indicated by state q_0.
        Set the value of COLOR_i according to q_0.
        Enter state q_0 and cease execution for this round.
    end if
end if
```

The interaction between tiles in $\mathcal{M}$ is completely defined by the glues of a tile's immediate neighbors, as specified in the function $\hat{\beta}$, and the processors of

Algorithm 2 For p_i in state $q \neq$ QUIET (at any round)

Read the four input buffers.
if no messages were received **then**
  Send the messages indicated by state q and cease execution for this round.
else
  Let q_0 be the state change directed by the function $\hat{\beta}$ applied to the glue types simulated by the messages received this round. {Note that q_0 will either equal q or QUIET, and q_0 might be chosen probabilistically.}
  Send the messages indicated by state q_0.
  Set the value of COLOR$_i$ according to q_0.
  Enter state q_0 and cease execution for this round.
end if

$\mathcal{N}$ simulate that behavior with Algorithm 2. Since the processors of $\mathcal{N}$ simulate empty spaces with Algorithm 1, by a straightforward induction argument, $\mathcal{N}$ can simulate all possible $\mathcal{T}$-assembly sequences, and the theorem is proved.

Combining Theorem 5 and the impossibility results of Naor and Stockmeyer, we obtain our main result, as follows.

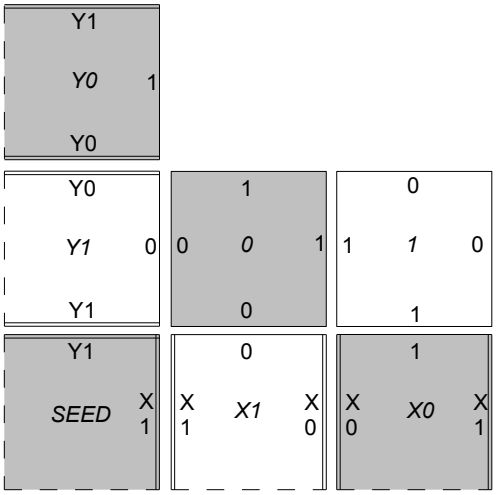

Fig. 2. The tileset $\mathcal{T}^*$ used in the proof of Lemma 2.

Theorem 6 (Main Result). *Any (multiply nucleating) tileset that tiles a surface in constant time is unable to solve the weak c-coloring problem, even though*

the weak c-coloring problem has a low-complexity solution in the Winfree-Rothemund Tile Assembly Model.

We break down the proof of this theorem into the following two lemmas.

Lemma 1. *Let $\mathcal{T}$ and $\mathcal{M}$ be such that, for all n sufficiently large, the expected time $\mathcal{T}$ takes to assemble on an $n \times n$ is some constant k, independent of n. Then $\mathcal{T}$ does not weak c-color the surface.*

Proof. Suppose $\mathcal{M}$ is an irreversible tiling model. If $\mathcal{T}$ can weak c-color surfaces in constant time, then there is a deterministic algorithm for the distributed network $\mathcal{N}$ that weak c-colors $\mathcal{N}$ locally, and in constant time. By Theorem 3 that is impossible.

So assume $\mathcal{M}$ is a reversible tiling model, and when $\mathcal{T}$ assembles, it weak c-colors the tiling surface, and achieves bond pair equilibrium in constant time. Then there is a local probabilistic algorithm for $\mathcal{N}$ that weak c-colors $\mathcal{N}$ in constant time, with positive probability of success. By Theorem 4 that is impossible as well. Therefore, no $\mathcal{T}$ exists that weak c-colors surfaces in constant time.

Lemma 2. *There is a tileset in the Winfree-Rothemund model that weak c-colors the first quadrant.*

Proof. Figure 2 exhibits a tileset $\mathcal{T}^*$ that assembles into a weak c-coloring of the first quadrant, starting from an individual seed tile placed at the origin. One can verify by inspection that $\mathcal{T}^*$ is locally deterministic, so it will always produce the same terminal assembly. All assembly sequences generated by $\mathcal{T}^*$ produce a checkerboard pattern in which a monochromatic "+" configuration never appears. Hence, it solves the weak c-coloring problem for the entire first quadrant, and also for all $n \times n$ squares, for any n.

The main result of the paper follows immediately from Lemmas 1 and 2.

5 Conclusion

In this paper, we showed that no tile assembly model can use multiple nucleation to solve locally checkable labeling problems in constant time, even though the Winfree-Rothemund Tile Assembly Model can solve a locally checkable labeling problem using just seven tile types. This was the first application of a distributed computing impossibility result to the field of nanoscale self-assembly.

There are still many open questions regarding multiple nucleation. Aggarwal *et al.* asked in [3] whether multiple nucleation might reduce the tile complexity of finite shapes. The answer is not known. Furthermore, we can ask for what class of computational problems does there exist some function f such that we could tile an $n \times n$ square in time $\mathcal{O}(1) < \mathcal{O}(f) < \mathcal{O}(n^2)$, and "solve" the problem with "acceptable" probability of error, in a tile assembly model that permits multiple nucleation. Finally, we hope that this is just the start of a conversation between researchers in distributed computing and biomolecular computation.

Acknowledgements

I am grateful to Soma Chaudhuri, Dave Doty, Jim Lathrop and Jack Lutz for helpful discussions on earlier versions of this paper.

References

1. Adleman, L., Cheng, Q., Goel, A., Huang, M.D.: Running time and program-size for self-assembled squares. In: Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing, pp. 740-748 (2001)
2. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations, and Advanced Topics, second edition. Wiley Series on Parallel and Distributed Computing (2004)
3. Aggarwal, G., Goldwasser, M., Kao, M.-Y., Schweller, R.: Complexities for Generalized Models of Self-Assembly. In: Proceedings of the fifteenth annual ACM-SIAM Symposium on Discrete Algorithms, pp. 880-889 (2004)
4. Cheng, Q., de Espanes, P.M.: Resolving two open problems in the self-assembly of squares. Technical Report 793, University of Southern California (2003)
5. Lathrop, J., Lutz, J., Patitz, M., Summers, S.: Computability and complexity in self-assembly. In: Logic and Theory of Algorithms: Proceedings of the Fourth Conference on Computability in Europe, to appear (2008)
6. Lathrop, J., Lutz, J., Summers, S.: Strict self-assembly of discrete Sierpinski triangles. In: Computation and Logic in the Real World: Proceedings of the Third Conference on Computability in Europe, pp. 455-464. Springer, Heidelberg (2007)
7. Majumder, U., Reif, J., Sahu, S.: Stochastic Analysis of Reversible Self-Assembly. In: Computational and Theoretical Nanoscience, to appear (2008)
8. Naor, M., Stockmeyer, L.: What can be computed locally? SIAM Journal of Computing, 24(6), pp. 1259-1277 (1995)
9. Patitz, M., Summers, S.: Self-assembly of discrete self-similar fractals. To appear in: Proceedings of the Seventh International Conference on Unconventional Computation. Springer-Verlag, Heidelberg (2008)
10. Rothemund, P. W. K.: Theory and Experiments in Algorithmic Self-Assembly. Ph.D. thesis, University of Southern California, Los Angeles (2001)
11. Rothemund, P., Winfree, E.: The program-size complexity of self-assembled squares. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, pp. 459-468 (2000)
12. Seeman, N.: Denovo design of sequences for nucleic-acid structural-engineering. Journal of Biomolecular Structure and Dynamics, 8(3), pp. 573-581 (1990)
13. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM Journal of Computing, 36(6), pp. 1544-1569 (2007)
14. Wang, H.: Proving theorems by pattern recognition II. Bell Systems Technical Journal, 40, pp. 1-41 (1961)
15. Winfree, E.: Algorithmic Self-Assembly of DNA. Ph.D. thesis, California Institute of Technology, Pasadena (1998)