Capabilities and Limits of Compact Error Resilience Methods for Algorithmic Self-assembly in Two and Three Dimensions*

Sudheer Sahu and John H. Reif

Department of Computer Science, Duke University
Box 90129, Durham, NC 27708-0129, USA
{sudheer, reif}@cs.duke.edu

Abstract. Winfree's pioneering work led the foundations in the area of error-reduction in algorithmic self-assembly [26], but the construction resulted in increase of the size of assembly. Reif et. al. contributed further in this area with compact error-resilient schemes [15] that maintained the original size of the assemblies, but required certain restrictions on the Boolean functions to be used in the algorithmic self-assembly. It is a critical challenge to improve these compact error resilient schemes to incorporate arbitrary Boolean functions, and to determine how far these prior results can be extended under different degrees of restrictions on the Boolean functions. In this work we present a considerably more complete theory of compact error-resilient schemes for algorithmic self-assembly in two and three dimensions. First we consider two-dimensional algorithmic self-assembly. We present an error correction scheme for reduction of errors from ϵ to ϵ^2 for arbitrary Boolean functions in two dimensional algorithmic self-assembly. Then we characterize the class of Boolean functions for which the error reduction can be done from ϵ to ϵ^3, and present an error correction scheme that achieves this reduction. Then we prove ultimate limits on certain classes of compact error resilient schemes: in particular we show that they can not provide reduction of errors from ϵ to ϵ^4 is for any Boolean functions. Further, we develop the first provable compact error resilience schemes for three dimensional tiling self-assemblies. We also extend the work of Winfree on self-healing in two-dimensional self-assembly [25] to obtain a self-healing tile-set for three-dimensional self-assembly.

1 Introduction

Self-assembly is the ubiquitous process in which smaller objects combine together to form larger complex objects. Recently, it has been demonstrated as an efficient mechanism for bottom-up construction of nanostructures in nanotechnology [19, 27, 11, 8, 32, 31, 3, 10]. The potential of self-assembly is not limited to nanofabrication. The ability of two-dimensional and three-dimensional assemblies to perform parallel universal computations has been explored in development of self-assembly of DNA tiles as a tool for nanocomputation [9, 14, 23, 28, 30]. Self-assembly has been demonstrated at

* The work is supported by NSF EMT Grants CCF-0523555 and CCF-0432038.

C. Mao and T. Yokomori (Eds.): DNA12, LNCS 4287, pp. 223–238, 2006.

larger scales (meso-scale) using capillary forces for interactions between meso-scale tiles [2, 16]. However, major hurdle in harnessing the capabilities of algorithmic self-assembly are the errors that occur during the assembly. Incorrect tiles are incorporated in the growing structure with error rate ranging from 1% to 5% [26]. There are two approaches to combat the errors. The first is to reduce the inherent error rate by optimizing the physical conditions [24] or using newer molecular mechanisms [4], while the other approach is to improve the tile design so that the total number of errors in the final structure is reduced in spite of the intrinsic error-rate remaining the same [26, 15, 5].

Winfree's pioneering work in error-correction [26] laid the foundations towards improving the tile-design to reduce the errors in assembly. Though it resulted in the total size of assembly to be 2×2 times for error reduction to ϵ^2 and 3×3 times for error reduction to ϵ^3, it paved the way for further work in error-reduction using the concept of redundancy. The basic idea was that an error in the assembly of a tile forced more errors in the immediate neighborhood of that tile, making it extremely prone to detachment, and hence reducing the error. Later, the snaked proof-reading scheme that could correct both growth and nucleation errors in the self-assembly was built upon this construction [5]. However, it required replacing a tile by a $k \times k$ block of tiles. Later a method was proposed to control nucleation errors programmably [18]. However, each of these schemes significantly scaled up the overall size of assembly. In applications like molecular fabrication tasks where the scale of final pattern is of critical importance, this scaling up is undesirable. Reif et al. [15] proposed a *compact error-resilient tiling schemes* in which errors could be reduced to ϵ^2 (2-way overlay redundancy) and ϵ^3 (3-way overlay redundancy) without increasing the size of the assembly. The analysis of error was done in the equilibrium state of the assembly. Another distinction of this scheme was that it considered the error resilience in the whole pattern and not only in the output row. It means that this scheme had a tendency to remove any incorrectly placed tile from the assembly even if the ongoing computation was not affected by that tile. This is important in the assembly of a nanostructure of desired pattern, where any incorrect placement of any tile is a defect (even though it might not have interfered with the subsequent growth of assembly). But it had its limitations on the Boolean functions that could be used for the error-resilient algorithmic assembly. In particular, it required one of the function to be *XOR*, and for reduction to ϵ^3 the additional requirement was that the other function should be input-sensitive to one of the inputs. A Boolean function $f(x)$ is called *input-sensitive* to a Boolean variable x if whenever x changes $f(x)$ also changes. It is thus a critical challenge to improve these compact error-correction schemes to incorporate any arbitrary Boolean functions. In case that is not possible, it is important to characterize the class of Boolean functions to which these error-correction schemes can be extended. Recently Winfree [20] presented a *compact error resilient scheme* based on Chen et al [5]. They also overlooked the errors that did not affect the ongoing computation.

Self-assembly in three dimensions is extremely promising in the field of microelectronics assembly, where independent manipulation of each component is required. It is already being seen as promising candidate for heterogeneous three-dimensional integration of next-generation microsystems [29, 12, 6, 22]. In light of the inherent parallelism, three-dimensional nature and larger range (nanoscale to mesoscale) of application of

self-assembly, it has a great potential as tool for building complex systems from microscaled templates. Apart from this, the utility of three-dimensional structures for computing has been known for a long time [7]. Simple examples of algorithmic computation in three dimensions includes the generalization of Pascal triangle to 3D [1] and three dimensional multiplexers (the latter would provide a mechanism for 3D memory addressing with the appropriate affixed molecular electronic components). Analogous to the simulation of a finite state automata through two-dimensional self-assembly, three dimensional self-assembly can be used to simulate a two-dimensional cellular automata, where the third spatial dimension of the 3D tiling is the time step of the cellular automata. The tiles in a horizontal plane will represent the current state of all the cells of a two-dimensional cellular automata, then the tiles assembled in horizontal plane on top of it will be states at next time instance. This allows one to derive 3D tiling assemblies from a wide variety of known two-dimensional cellular automata designs, including matrix multiplication, integer multipliers, context free language recognition, etc. Recently crystal structure of three-dimensional DNA lattices formed by self-assembly was demonstrated [13]. The question of fault-tolerance naturally arises with the increasing popularity of self-assembly for construction of three dimensional self-assembled structures. It will be critical to determine how successfully can the error-correction techniques used for two-dimensional assemblies be extended to three-dimensions.

Self-healing is a very important process in nature. The damage to the living cells can be caused by an external intruder or some mechanical impulse or unfavorable physical conditions. The one property of biological systems that make them robust is their ability to self-heal in case of damages. It would be really interesting to design the DNA tiles that forms the lattices having the ability to self-heal, thereby imparting them the much desired robustness. Winfree [25] gave a construction in which he replaced a single tile with 3×3 (for simple assemblies like Sierpinski triangles) , 5×5 (for general assemblies) and 7×7 (for additional robustness to nucleation errors) block of tiles for self-healing in a two-dimensional assembly. It would be interesting to know if compact self-healing tilesets can be formed and whether the techniques given by Winfree can be extended to three dimensions.

In this paper, we follow the notion of *compactness* as presented in [15], which requires the new error-resilient tiling assembly to be of no larger size than the original assembly. Like [15] we consider any incorrect placement of a tile anywhere in the assembly as an error and aim at reducing them as well, even though these errors might not affect the ongoing computation. As mentioned earlier, this is important for construction of nanostructures of desired pattern. In this paper, the analysis of the error in the assembly is done in the equilibrium state of the assembly. Throughout this paper *redundancy based compact error resilient scheme* refers to any error resilient scheme that does not scale up the assembly and in which the encodings on the pads of the tiles are used to create redundancy. In the event of an error this redundancy forces more errors, which makes the incorrectly placed tiles and their neighborhoods more unstable and prone to removal from assembly, thereby reducing the error. Also we refer to k*-expansive error resilient schemes* as the error correction schemes that work by replacement of a tile by a block of multiple tiles. In case of three dimensional tiling, we carry forward this notion of *redundancy based compact error resilient schemes*.

In this paper, we present a comprehensive theory of redundancy based compact error resilient tiling schemes and examine the prospects of constructing compact self-healing tile sets in two and three-dimensions. The error analysis throughout this paper is in the equilibrium state of the assembly. In Section 2, first we present a compact error correction schemes in two dimensional self-assembly that reduces the error from ϵ to ϵ^2 for arbitrary Boolean functions. Then we characterize the class of Boolean functions for which error reduction from ϵ to ϵ^3 is possible using redundancy based compact error resilient schemes. Also we prove that error reduction from ϵ to ϵ^4 is impossible using redundancy based compact error resilient schemes. Next in Section 3 we examine three-dimensional self-assembly. First we present a compact error resilient scheme that reduces error to ϵ^2 for arbitrary Boolean functions and ϵ^3 for a restricted class of input-sensitive Boolean functions. We also prove that error reduction to ϵ^4 can not be obtained for arbitrary Boolean functions using redundancy based compact error resilient schemes. In Section 4 we extend the idea of Winfree's construction for self-healing in two-dimensions [25] to three-dimensional assembly. In the conclusion, we review our results and state various open problems and conjectures. We conjecture stronger results that error reduction to ϵ^3 in three dimensions can not be achieved outside the previously characterized class, and error reduction to ϵ^4 is impossible to achieve for any Boolean functions using these error resilient techniques.

2 Error Correction in Self-assembly in Two Dimensions

2.1 Assembly in Two Dimensions

We will consider a general assembly problem in two dimensions consisting of the assembly of a two-dimensional Boolean array of size $N \times M$, where the elements of each column are indexed from 0 to $N-1$ from right to left and rows are indexed from 0 to $M-1$ from bottom to top. The bottom row and the rightmost column provide the inputs to the assembly. Let $V(i,j)$ be the value of the ith column (from the right) in the jth row(from the bottom). Let $V(i,j+1)$ be the value communicated to the position $(i,j+1)$ and $U(i+1,j)$ be the value communicated to the position $(i+1,j)$. We define $U(i+1,j) = U(i,j)OP_1V(i,j)$ and $V(i,j+1) = U(i,j)OP_2V(i,j)$ for two Boolean functions OP_1 and OP_2.

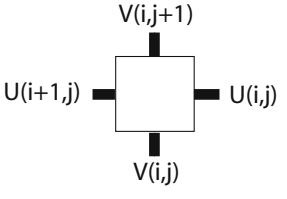

Fig. 1. Two dimensional algorithmic self-assembly

Figure 1 shows a computational tile that can be used for constructing two dimensional self-assembly. Bottom and right pads are the input pads, while the pads on top and left are output pads. A pad *matches* with the neighbor's contiguous pad if the values communicated by these pads are the same. $U(i,j)$ and $V(i,j)$ are the right and bottom input pads, respectively, to the ith column from right and jth row from bottom. Then $U(i+1,j)$ the left output pad is given by $U(i+1,j) = U(i,j)OP_1V(i,j)$, while $V(i,j+1)$ the top output pad is given by $V(i,j+1) = U(i,j)OP_2V(i,j)$. Examples of simple two dimensional assemblies: sierpinski triangle and binary counter, are given

in [15]. Highly complex two-dimensional assemblies are possible due to the universal computability of two-dimensional self-assembly [21, 28].

2.2 The Error Model

We assume that error probability ϵ is defined as the probability that there is mismatch between two tiles and they still stay together in the equilibrium. This probability is independent of any other match or mismatch and hence we term this probabilistic model the *independent error model*. We also want to put emphasis on the correct assembly of all the tiles in the assembly (and hence on the correctness of complete pattern), and not just on the correctness of final output only. There might be wrong placement(s) of tile(s), that do not affect the ongoing computation. But in our error model, we count them as errors and need the error correction schemes to reduce such errors as well. In this way we differ from [20], who overlooked the errors that did not affect the ongoing computation.

Consider a tile $T(i,j)$ in a $N \times M$ tiling assembly where $0 < i < N-1, 0 < j < M-1$. We define the *immediate neighborhood* of a tile $T(i,j)$ as 8 tiles surrounding it, whose coordinates differ from (i,j) by at most 1. Formally speaking, $\{T(i',j') : |i'-i| \leq 1, |j'-j| \leq 1\} \setminus \{T(i,j)\}$. Tile $T(i',j')$ is said to be *a-dependent* (for assembly dependent) on tile $T(i,j)$ if $i' \geq i$ and $j' \geq j$ and *a-independent* otherwise. Next we examine the schemes to reduce the errors in self-assembly. To reiterate, throughout this paper, we refer to redundancy based compact error resilient scheme as error reduction scheme, where redundancy is created by encodings in the pads with absolutely no scale up of the assembly.

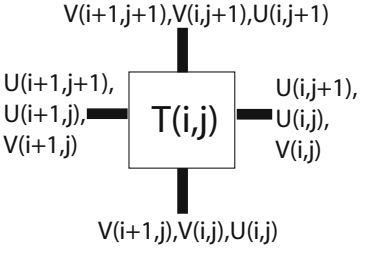

Fig. 2. Construction for error reduction to ϵ^2

Proposition 1. *Under our independent error model, if an error in a pad in a tile enforces k further mismatches in the assembly in the immediate neighborhood of that tile, then error probability is reduced to ϵ^{k+1}.*

Proof. If one error guarantees k more errors, then the probability that the tile and its neighborhood in the assembly will stay together in the equilibrium in spite of these $k+1$ errors is ϵ^{k+1}. And hence the error reduction.

2.3 Error Reduction to ϵ^2

It is known that if an error in a tile can guarantee another error in immediate neighborhood, then it reduces the rate of errors from ϵ to ϵ^2 [26, 15]. Next we describe our construction to achieve this goal in the form of Theorem 1.

Theorem 1. *There exists a compact error correction scheme that will reduce the error from ϵ to ϵ^2 for two-dimensional algorithmic self-assembly for any arbitrary Boolean functions OP_1 and OP_2.*

Proof. **Construction.** Before we begin the proof we would like to emphasize the wholeness of the pad. Each side of the tile has one pad in Figure 2, and it encodes the triplet shown in the Figure. Disagreement between corresponding elements of two such triplets in any two pads results in the total mismatch between those two pads. Consider the tile with input $U(i,j)$ and $V(i,j)$ at the right and bottom pads respectively. Our goal is to guarantee one more error in the immediate vicinity of this tile if there is one error. For that, we construct an error checking portion ($V(i,j)$) in the right side pad and one error checking portion ($U(i,j)$) in the bottom pad. We will need corresponding parts in the pads on the top ($U(i,j+1)$) and the left side ($V(i+1,j)$) also, which will match with the error checking parts in the bottom pad of the top neighbor $T(i,j+1)$ and right pad of the left neighbor $T(i+1,j)$ respectively. Now since top output pad

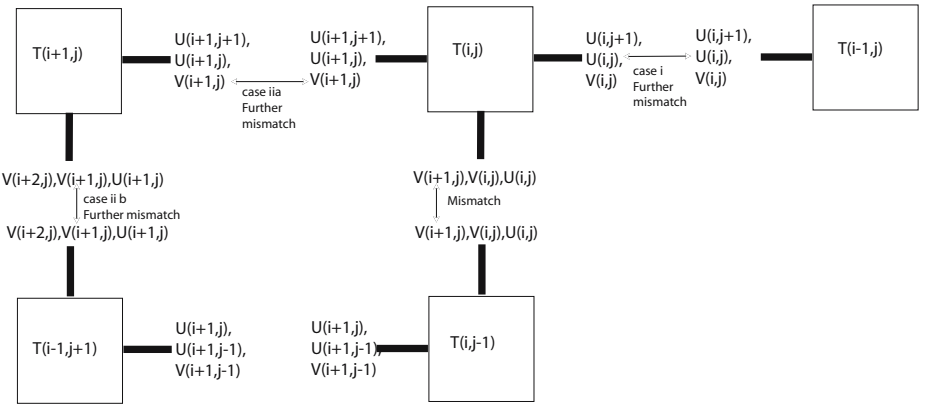

Fig. 3. Case 1 b) A further mismatch is caused by an error in the input pads

depends on the value of $U(i,j+1)$ (which is the right input of the top neighbor) we need to incorporate it in our input pads. It is necessary otherwise there will be multiple type of tiles for any given set of input pads. But for successful functioning of algorithmic self-assembly it is required that there should be only one possible tile-type for every set of input pads. So, we need one more portion in the right input pad ($U(i,j+1)$) and hence a corresponding part in the left output pad ($U(i+1,j+1)$). Similarly, the need for another portion in bottom input pad ($V(i+1,j)$) and subsequently, in top output pad ($V(i+1,j+1)$) can be explained.

This completes our description of a tile in the required tile-set. It should be noted that the number of different tile types in this tile-set will be 4 times as compared to number of tiles in a tileset without any error-correction. It can be attributed to the two possible values for each of $U(i,j+1)$ and $V(i+1,j)$, for every value of the inputs $U(i,j)$ and $V(i,j)$.

Error-Analysis: We show that if the neighborhood tiles a-independent of $T(i,j)$ are assembled correctly then a pad binding error in any of the input pads in $T(i,j)$ causes an additional mismatch error in its neighborhood in equilibrium. We need to consider

only the cases where the pad binding error occurs in either the bottom or the right pad of tile $T(i,j)$. Otherwise, if the error occurs in left (or top) pad of $T(i,j)$ then we can consider the right pad of $T(i+1,j)$ (or bottom pad of $T(i,j+1)$ for the analysis. The following case analysis provides the required proof.

1. If the bottom pad of $T(i,j)$ has a mismatch:
   (a) If $V(i,j)$ on the bottom pad has a mismatch, then $V(i,j)$ on right pad is incorrect, which causes an additional mismatch.
   (b) If $V(i,j)$ on the bottom pad is correct and $V(i+1,j)$ on right pad has a mismatch, $V(i+1,j)$ on left pad is incorrect. Now we will prove that it causes a further mismatch by exactly same technique as used by Reif et al [15]. We have assumed that all the rows and columns that are a-independent of tile $T(i,j)$ are correctly assembled so $T(i+1,j-1)$ is correctly assembled and has correct values of its top output pad. Hence $T(i,j)$'s left neighbor $T(i+1,j)$ is dependent upon the incorrect value communicated by the left pad of $T(i,j)$ and correct values communicated by top pad of $T(i+1,j-1)$. Now consider the pads of $T(i+1,j)$. The right pad includes $U(i+1,j+1), U(i+1,j), V(i+1,j)$ and bottom pads include $V(i+2,j), V(i+1,j), U(i+1,j)$. Since the value $V(i+1,j)$ communicated by $T(i+1,j-1)$ is correct and the value $V(i+1,j)$ communicated by $T(i,j)$ is wrong, this implies there will be a mismatch at the right or bottom pad of Tile $T(i+1,j)$.
2. If there is no error in bottom pad, but the right pad of $T(i,j)$ has mismatch:
   (a) If $U(i,j)$ on the right pad has a mismatch, then $U(i,j)$ on bottom pad is incorrect, which causes an additional mismatch.
   (b) If $U(i,j)$ on right pad is correct but $U(i,j+1)$ on right pad is incorrect, then $U(i,j+1)$ on top output pad is incorrect. Now we will show that it causes a further mismatch as argued above. Since we assume that all the rows and columns that are a-independent of tile $T(i,j)$ are correctly assembled $T(i-1,j+1)$ is correctly assembled and has correct values of its left output pad. Hence $T(i,j)$'s top neighbor is dependent upon the incorrect value communicated by the top pad of $T(i,j)$ and correct values communicated by left pad of $T(i-1,j+1)$. Now consider the pads of $T(i,j+1)$. The right pad includes $U(i,j+2), U(i,j+1), V(i,j+1)$ and bottom pads include $V(i+1,j+1), V(i,j+1), U(i,j+1)$. Since $V(i+1,j)$ communicated by $T(i-1,j+1)$ is correct and the value $V(i+1,j)$ communicated by $T(i,j)$ is wrong, this implies there will be a mismatch at the right or bottom pad of Tile $T(i,j+1)$.

Hence any mismatch on the right or bottom pad of tile $T(i,j)$ causes one more mismatch in the vicinity of the tile. Together with the Proposition 1 this implies that this scheme can reduce the pad mismatch errors from ϵ to ϵ^2.

2.4 Error Reduction to ϵ^3

At this point we would like to reiterate that *redundancy based compact error resilient scheme* refers to any error resilient scheme that does not scale up the assembly and in which only the encodings on the pads of the tiles are used to create redundancy. Also, a

Boolean function $f(x)$ is said to be *input-sensitive* to Boolean input x if it changes for every change in the value of x.

Theorem 2. *For arbitrary OP_1 and OP_2, there does not exist any redundancy based compact error resilient scheme for two-dimensional self-assembly that can reduce the error from ϵ to ϵ^3.*

Proof.

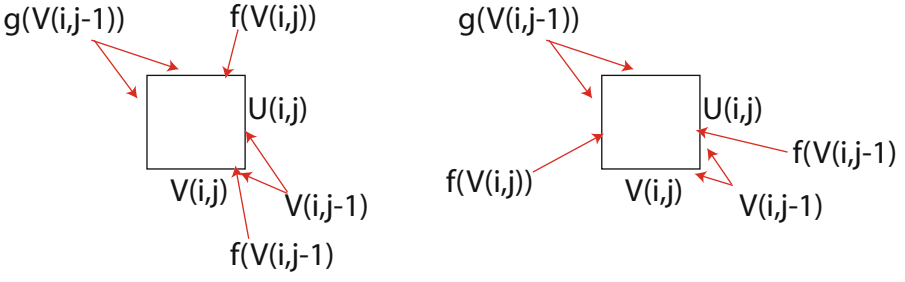

Fig. 4.

For errors to reduce from ϵ to ϵ^3, an error in any input pad, say $V(i,j)$ should cause two further mismatches in the immediate neighborhood. At least one of those mismatches should be caused because of an error on one of the output pads. It should be noted that if OP_1 and OP_2 are arbitrary Boolean functions then the output $U(i+1,j)$ or $V(i,j+1)$ cannot be guaranteed to be wrong for incorrect value of $V(i,j)$. Hence, in at least one of the output pads an additional error checking portion $f(V(i,j))$ (that is input-sensitive to $V(i,j)$ and hence can reflect the error in $V(i,j)$) is required. It can be located on the top or left output pad.

- Assume that $f(V(i,j))$ is located on top side, which implies $f(V(i,j-1))$ is located on the bottom side.
  1. If $V(i,j-1)$ does not exist within the input pads, then we need to consider the case when $f(V(i,j-1))$ has a mismatch. Since we require two further errors in the neighborhood of $T(i,j)$, as argued above it requires an additional error checking function $g(f(V(i,j-1)))$ (that is input-sensitive to $f(V(i,j-1))$) on at least one of the top or left output pad.
  2. If $V(i,j-1)$ exists in the input pads, then in case when $V(i,j-1)$ is mismatched, and two further errors in the neighborhood of $T(i,j)$ are required, it needs an additional error checking function $g'(V(i,j-1))$ (that is input-sensitive to $V(i,j-1)$) on at least one of the top or left output pad.
- Assume that $f(V(i,j))$ is located on left side, which implies $f(V(i-1,j))$ is located on the right side.
  1. If $V(i-1,j)$ does not exist within the input pads, we need to consider the case when $f(V(i-1,j))$ is mismatched. Since two further errors are required, as argued above it requires an additional error checking function $h(f(V(i-1,j)))$ (that is input-sensitive to $f(V(i-1,j))$) to be located on at least one of the top or left output pad.

2. If $V(i-1,j)$ exists in the input pads, then in case when $V(i-1,j)$ is mismatched, and two further errors are required, it requires an additional error checking function $h'(V(i-1,j))$ (that is input-sensitive to $V(i-1,j)$) to be present on at least one of the top or left output pads.

Hence, an additional error checking pad ($g(f(V(i,j-1)))$, $g'(V(i,j-1))$ or $h(f(V(i-1,j)))$or $h(V(i-1,j))$) is required on at least one of the output pads. Arguing in the same manner as above it can concluded that this cycle will keep on repeating. Hence, it is not possible to construct tile with a bounded number of parameters in the pads and we conclude that redundancy based compact error resilient schemes can not reduce error from ϵ to ϵ^3.

However, it will be proved that for a rather restricted class of Boolean functions OP_1 and OP_2, error can be reduced to ϵ^3 by using the construction of Figure 2, which is stated as Theorem 3.

Before we proceed with the error-analysis, it will be useful to understand the function class characterized in the Theorem 3. OP_1 and OP_2 are such that:**(1)** $U(i,j)$ OP_1 $V(i,j)$ is input-sensitive to $U(i,j)$, if $V(i,j)$ is kept constant and $U(i,j)$ OP_2 $V(i,j)$ is input-sensitive to $V(i,j)$ if $U(i,j)$ is kept constant. **(2)** When both of them change at least one of the $U(i,j)$ OP_1 $V(i,j)$ or $U(i,j)$ OP_2 $V(i,j)$ should also change. For $U(i,j) = 0$, there are 2 possible assignments to $U(i,j)OP_1V(i,j)$ maintaining its input-sensitivity to $V(i,j)$. Similarly, for $U = 1$ there are 2 possible assignments to $U(i,j)OP_1V(i,j)$ conditioned to its input-sensitivity to $V(i,j)$. Similarly for $V(i,j)$=0 and $V(i,j)$=1 there are 2 independent assignments each. But among these half of the assignments do not satisfy the second condition. Hence the total number of Boolean functions in this class are 8. An example of such a function is given in the Table 1.

Table 1. An example of the OP_1 and OP_2

U	V	UOP_1V	UOP_2V
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	1

Theorem 3. *For restricted class of Boolean functions OP_1 and OP_2 such that at least one of the $U(i+1,j)$ or $V(i,j+1)$ changes for any change in $U(i,j)$ or $V(i,j)$, there exists a redundancy based compact error resilience scheme that can reduce the error to ϵ^3, and one such scheme is as shown in Figure 2.*

Proof. If OP_1 and OP_2 are restricted to be as described, and if the neighborhood tiles that are a-independent of $T(i,j)$ are assembled correctly, then a pad binding error in any of the input pads in $T(i,j)$ causes two additional mismatch errors in its neighborhood. As explained earlier, we need to consider only the cases where the pad binding error occurs in either the bottom or the right pad of tile T(i,j). The following case analysis provides the required proof.

1. If the bottom pad of $T(i,j)$ has a mismatch:
   (a) If $V(i,j)$ in bottom pad of $T(i,j)$ has a mismatch, then the $V(i,j)$ in the right pad of $T(i,j)$ is incorrect. This causes a mismatch because according to our assumption, all the tiles a-independent of $T(i,j)$ are assembled correctly. Also:

i. If $U(i,j)$ on right pad is correct, $V(i,j+1)$ on top pad is incorrectly computed because of restrictions on OP_1 and OP_2. This will cause further mismatch at the right or bottom pad of the top neighbor $T(i,j+1)$, as argued in the proof of Theorem 1.

ii. If $U(i,j)$ on right pad has a pad-mismatch, then at least one of the $V(i,j+1)$ on top pad or $U(i+1,j)$ on left pad is incorrectly computed, because of the restrictions on OP_1 and OP_2. This will cause a further mismatch at right or bottom pad of the left neighbor ($T(i+1,j)$) or top neighbor($T(i,j+1)$) in the same way as argued earlier.

(b) If $V(i,j)$ on bottom pad is correct and $V(i+1,j)$ on bottom pad has mismatch, then $V(i+1,j)$ on the left pad is incorrect, which causes a further mismatch in the right or bottom pad of the left neighbor $T(i+1,j)$. Also:

i. If $U(i,j)$ on right pad is incorrect, then this causes a mismatch on the right pad of $T(i,j)$, because according to our assumption, all the tiles a-independent of $T(i,j)$ are assembled correctly.

ii. If $U(i,j)$ on right pad is correct, then $U(i+1,j)$ on left output pad is correct. But since $V(i+1,j)$ has a mismatch, $V(i+1,j+1)$ on the top pad is incorrectly computed, because of the restriction on OP_1 and OP_2. This causes a further mismatch on the bottom or the right pad of the top neighbor tile $T(i,j+1)$.

2. If there is no error in the bottom pad and there is mismatch in right pad:

(a) If $U(i,j)$ on right pad has a pad-mismatch, then at bottom $U(i,j)$ is incorrect, and causes a mismatch. However since $V(i,j)$ on the bottom pad is correct so $U(i+1,j)$ on left pad is incorrectly computed because of the restriction on OP_1 and OP_2. This causes a further mismatch on right or bottom pad of left neighbor as explained earlier.

(b) If $U(i,j)$ on right pad is correct and $U(i,j+1)$ has a mismatch, then $U(i,j+1)$ on top pad is incorrect., which causes a further mismatch in right or bottom pad of the top neighbor tile $T(i,j+1)$. Also since $V(i,j)$ is correct, $V(i,j+1)$ is also correct, and hence $U(i+1,j+1)$ on left pad is incorrectly computed because of restriction on OP_1 and OP_2. This causes a further mismatch in the right or bottom pad of the left neighboring tile $T(i+1,j)$.

Hence any mismatch on the right or bottom side of the tile $T(i,j)$ causes two further mismatches in the vicinity of tile $T(i,j)$ and this results in error reduction from ϵ to ϵ^3.

For any other combination of Boolean functions OP_1 and OP_2, which do not satisfy the conditions of Theorem 3, the redundancy based compact error resilient schemes fail to achieve the reduction from ϵ to ϵ^3. It can be proven along the similar lines of reasoning as the proof of Theorem 2. Therefore we state it without proof.

Theorem 4. *For any combination of Boolean functions OP_1 and OP_2 outside the restricted class of Theorem 3, there exists no redundancy based compact error correction schemes that can reduce the error from ϵ to ϵ^3 in two-dimensional self-assembly.*

2.5 Error Reduction to ϵ^4

Theorem 5. *For any Boolean functions OP_1 and OP_2, there exists no redundancy based compact error correction scheme that can reduce error from ϵ to ϵ^4 in two-dimensional self-assembly.*

Proof. For the reduction of error from ϵ to ϵ^4, a mismatch in any input pad should cause 3 more mismatches. It means that for any error in one of the input pads both the output pads should have errors. In case an output pad requires any additional error checking portion to detect an error in an input, then by arguments similar to the proof of Theorem 2, it can be shown that such a tile cannot be constructed.

Hence, the only possibility is when, the left and top outputs $U(i+1, j)$ and $V(i, j+1)$ both change for any change in the input $U(i, j)$ or $V(i, j)$. This means that we have different values for each of $U(i + 1, j)$ and $V(i, j + 1)$ for 4 different values of input pair, which is not possible as $U(i + 1, j)$ and $V(i, j + 1)$ are Booleans.

3 Error Correction in Self-assemblies in Three Dimensions

Three dimensional self-assembly is being described as the most promising tool for heterogeneous integration of next generation microsystems. Its potential to build complex systems from microscale templates can not be overlooked [29, 12, 6, 22]. Besides the assembled three-dimensional structures can be extremely useful in computations [7]. It is possible to simulate a two-dimensional cellular automata, using three-dimensional self-assembly, which then paves way to perform a rich class of computations including matrix multiplication, integer multiplications, context-free language recognition etc.

3.1 Assembly in Three Dimensions

The assembly problem in three-dimensions can be generalized from the two-dimensional assembly as the assembly of a three-dimensional Boolean array of size $N \times M \times P$, where the elements are indexed from 0 to $N - 1$ from right to left, 0 to $M - 1$ from bottom to top, and 0 to $P - 1$ from front to back. The rightmost plane, bottommost plane and frontmost plane provide the inputs to the assembly.

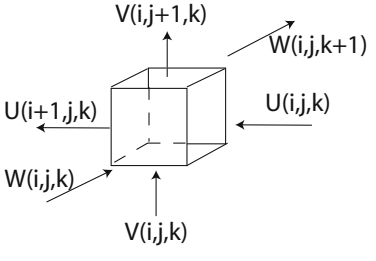

Fig. 5. Three dimensional algorithmic self-assembly

Let $V(i, j, k)$ be the i-th value from right, j-th from bottom, and k-th from front. Let $U(i, j, k)$ be the value communicated to the position $(i+1, j, k)$, $V(i, j, k)$ be communicated to the position $(i, j+1, k)$, and $W(i, j, k)$ be communicated to the position $(i, j, k + 1)$. Figure 5 shows a computational tile that can be used for construction of three-dimensional assembly. $U(i, j, k)$, $V(i, j, k)$ and $W(i, j, k)$ are inputs at right pad, bottom pad and front pad respectively, to the tile located at position (i, j, k). Then $U(i + 1, j, k)$, $V(i, j + 1, k)$ and $W(i, j, k + 1)$ are the output values at left, top and back pads, respectively. Also,$U(i, j, k) = f_1(U(i-1, j, k)$,

$V(i,j-1,k), W(i,j,k-1)), V(i,j,k) = f_2(U(i-1,j,k), V(i,j-1,k), W(i,j,k-1)), W(i,j,k) = f_3(U(i-1,j,k), V(i,j-1,k), W(i,j,k-1))$ where f_1, f_2 and f_3 are the ternary Boolean functions that take as input three Boolean values and give a Boolean output. It is assumed that initially a frame is assembled, with $M \times P$ tiles in rightmost plane, $N \times P$ tiles in bottommost plane and $N \times P$ tiles in frontmost plane. Next we examine the error resilience in three-dimensional self-assembly.

3.2 The Error Model

We extend the error model in two-dimensions to three-dimensional assembly in an obvious way. We follow the *independent error model* for three dimensional assembly. We also want to emphasize on the correct assembly of all the tiles in the assembly (and hence on the correctness of complete pattern), and not just on the correctness of final output only. We want to emphasize that the error analysis is done in the equilibrium state of the assembly. Consider a tile $T(i,j,k)$ in a $N \times M \times P$ tiling assembly where $0 < i < N-1, 0 < j < M-1, 0 < k < P-1$. We define the *immediate neighborhood* of a tile $T(i,j,k)$ as 26 tiles surrounding it, whose coordinates differ from (i,j,k) by at most 1. Formally speaking, $\{T(i',j',k') : |i'-i| \leq 1, |j'-j| \leq 1, |k'-k| \leq 1\} \setminus \{T(i,j,k)\}$. Tile $T(i',j',k')$ is said to be *a-dependent* on tile $T(i,j,k)$ if $i' \geq i$, $j' \geq j$, and $k' \geq k$ and *a-independent* otherwise. Next we examine the schemes to reduce the errors in self-assembly. As mentioned earlier *redundancy based compact error resilient scheme* refers to an error resilient scheme that does not scale up the assembly and in which the encodings on the pads of the tiles are used to create redundancy.

3.3 Error Reduction to ϵ^2

Theorem 6. *There exists a redundancy based compact error resilient tiling scheme in three dimensional assembly which can reduce the error from ϵ to ϵ^2 for any arbitrary Boolean functions f_1, f_2, and f_3, and it is shown in Figure 6.*

Construction. Before we describe the construction, we would like to emphasize on the wholeness of pad. Each side of the tile has one pad in Figure 6, that encodes a 5-tuple as shown in the Figure. Disagreement between corresponding elements of two such 5-tuples in any two pads results in the total mismatch between those two pads. Consider the tile $T(i,j,k)$ with inputs $U(i,j,k)$, $V(i,j,k)$ and $W(i,j,k)$ on the right, bottom and front pads respectively. Our goal is to guarantee one more error in the vicinity of this tile if there is one error in any of the input pads.

We add error checking parts to the right, bottom and front pads as shown in the Figure 6: $V(i,j,k)$ and $W(i,j,k)$ on right pad, $W(i,j,k)$ and $U(i,j,k)$ on bottom pad and $U(i,j,k)$ and $V(i,j,k)$ on front pad. Corresponding to these, we need to add $V(i+1,j,k)$ and $W(i+1,j,k)$ on left pad, $W(i,j+1,k)$ and $U(i,j+1,k)$ on top pad and $U(i,j,k+1)$ and $V(i,j,k+1)$ on back pad, as explained in the case of two-dimensional tile.

As described in two-dimensional assembly, every value in the output pads should be uniquely derivable from the values on the input pads. For $V(i+1,j,k)$ and $W(i+1,j,k)$ on the left pad we add $V(i+1,j,k)$ on the bottom pad, and $W(i+1,j,k)$ on the front pad. For $U(i,j+1,k)$ and $W(i,j+1,k)$ on the top pad, we add $U(i,j+1,k)$ to

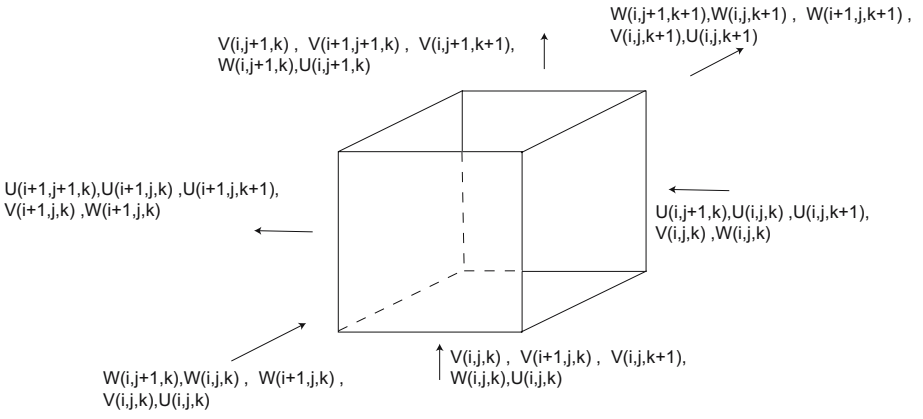

Fig. 6. Construction for error reduction to ϵ^2

the right pad and $W(i, j+1, k)$ to the front pad. For $U(i, j, k+1)$ and $V(i, j, k+1)$ on the back pad, we add $U(i, j, k+1)$ to the right pad and $V(i, j, k+1)$ to the bottom pad. The construction is complete with addition of $U(i+1, j+1, k)$ and $U(i+1, j, k+1)$ to left pad, $V(i+1, j+1, k)$ and $V(i, j+1, k+1)$ to top pad, and $W(i+1, j, k+1)$ and $W(i, j+1, k+1)$ to back pad.

This completes our description of a tile in the required tile-set. It should be noted that the number of different tile types in this tile set will be 64 times as compared to number of tiles in a tileset without any error-correction. It can be attributed to the two values for each of the $U(i, j+1, k)$, $U(i, j, k+1)$, $V(i+1, j, k)$, $V(i, j, k+1)$, $W(i+1, j, k)$ and $W(i, j+1, k)$, for every value of the inputs $U(i, j, k)$, $V(i, j, k)$ and $W(i, j, k)$.

Refer to [17] for detailed error analysis.

3.4 Error Reduction to ϵ^3

Theorem 7. *If Boolean functions f_1, f_2, and f_3 satisfy the following conditions:*

- *for fixed $V(i, j, k)$ and $W(i, j, k)$, $f_1(U, V, W)$ is input-sensitive to $U(i, j, k)$.*
- *for fixed $U(i, j, k)$ and $W(i, j, k)$, $f_2(U, V, W)$ is input-sensitive to $V(i, j, k)$.*
- *for fixed $U(i, j, k)$ and $V(i, j, k)$, $f_3(U, V, W)$ is input-sensitive to $W(i, j, k)$.*

Then there exists a compact error resilient scheme to reduce error from ϵ to ϵ^3 for three-dimensional self-assembly, and it is shown in Figure 6.

Refer to [17] for detailed proof.

3.5 Error Reduction to ϵ^4

Theorem 8. *For arbitrary Boolean functions f_1, f_2, and f_3, there exists no redundancy based compact error resilient scheme that can reduce error from ϵ to ϵ^4 in three-dimensional self-assembly.*

Refer [17] for proof.

4 Self-healing Tile Set for Three Dimensional Assembly

Winfree [25] provided the basis for studying self-healing in the self-assembly in a rigorous manner. We need to consider the repairability of a self-assembled structure in the face of a damage. A tile-set is called self-healing, if at any point during error-free growth, when n tiles are removed, subsequent error free growth will repair the damage rapidly [25]. Winfree's scheme of correctly repairing the damage (hole) is by ensuring that the holes are filled in the original forward direction of the algorithmic assembly and there is no backward growth in the holes.

Winfree proposed constructions of self-healing tile-sets for two dimensional algorithmic self-assembly by replacing a single tile by a 3×3 (for simple assemblies like sierpinsky triangles) , 5×5 (for general assemblies) and 7×7 (for additional robustness to nucleation errors) block. We can extend his construction to three-dimensions. We have discussed the construction of self-healing tile set by replacing a tile by $3 \times 3 \times 3$ block of tiles in [17].

5 Discussion

In this paper, we presented a theoretical analysis of redundancy based compact error resilient tiling in two and three dimensions. We conjecture the following stronger results for three-dimensional assemblies. Currently these conjectures are open questions. We state them without proofs as follows:

Conjecture 1. For arbitrary Boolean functions f_1, f_2, and f_3, there exists no redundancy based compact error correction scheme that will reduce error from ϵ to ϵ^3 in three-dimensional self-assembly.

Conjecture 2. For any functions f_1, f_2, and f_3 that are outside the restricted class of the functions defined in Theorem 7 there exists no redundancy based compact error correction scheme that will reduce error from ϵ to ϵ^3 in three-dimensional self-assembly.

Conjecture 3. For any Boolean functions f_1, f_2, and f_3, , there exists no redundancy based compact error resilient scheme that can reduce error from ϵ to ϵ^4 in three-dimensional self-assembly.

The immediate future work will be to prove or disprove these conjectures. We have presented a three-dimensional extension to Winfree's self-healing tile set in two-dimensions. It remains an open question if it is possible to design a compact self-healing tile set for two and three-dimensional self-assembly.

References

1. B.A. Bondarenko. *Generalized Pascal Triangles and Pyramids, Their Fractals, Graphs and Applications*. The Fibonacci Association, 1993. Translated from Russion and edited by R.C. Bollinger.
2. N. Bowden, A. Terfort, J. Carbeck, and G.M. Whitesides. Self-assembly of mesoscale objects into ordered two-dimensional arrays. *Science*, 276(11):233–235, 1997.

3. N. Chelyapov, Y. Brun, M. Gopalkrishnan, D. Reishus, B. Shaw, and L. Adleman. DNA triangles and self-assembled hexagonal tilings. *J. Am. Chem. Soc.*, 126:13924–13925, 2004.
4. H.L. Chen, Q. Cheng, A. Goel, M.D. Huang, and P.M. de Espanes. Invadable self-assembly: Combining robustness with efficiency. In *Proceedings of the 15th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 890–899, 2004.
5. H.L. Chen and A. Goel. Error free self-assembly using error prone tiles. In *DNA Based Computers 10*, pages 274–283, 2004.
6. T. D Clark, R Ferrigno, J Tien, K E Paul, and G M Whitesides. Template-directed self-assembly of 10-microm-sized hexagonal plates. *J Am Chem Soc.*, 124(19):5419–26, 2002.
7. N. Jonoska, S.A. Karl, and M. Saito. Three dimensional DNA structures in computing. *BioSystems*, 52:143–153, 1999.
8. T.H. LaBean, H. Yan, J. Kopatsch, F. Liu, E. Winfree, J.H. Reif, and N.C. Seeman. The construction, analysis, ligation and self-assembly of DNA triple crossover complexes. *J. Am. Chem. Soc.*, 122:1848–1860, 2000.
9. M.G. Lagoudakis and T.H. LaBean. 2-D DNA self-assembly for satisfiability. In *DNA Based Computers V*, volume 54 of *DIMACS*, pages 141–154. American Mathematical Society, 2000.
10. D. Liu, M. Wang, Z. Deng, R. Walulu, and C. Mao. Tensegrity: Construction of rigid DNA triangles with flexible four-arm dna junctions. *J. Am. Chem. Soc.*, 126:2324–2325, 2004.
11. C. Mao, W. Sun, and N.C. Seeman. Designed two-dimensional DNA holliday junction arrays visualized by atomic force microscopy. *J. Am. Chem. Soc.*, 121:5437–5443, 1999.
12. B. R. Martin, D. C. Furnange, T. N. Jackson, T. E. Mallouk, and T. S. Mayer. Self-alignment of patterned wafers using capillary forces at a water-air interface. *Advanced Functional Materials*, 11:381–386, 2001.
13. P.J. Paukstelis, J. Nowakowski, J.J. Birktoft, and N.C. Seeman. Crystal structure of a continuous three-dimensional DNA lattice. *Chemistry and Biology*, 11:1119–1126, 2004.
14. J.H. Reif. Local parallel biomolecular computation. In H. Rubin and D.H. Wood, editors, *DNA-Based Computers 3*, volume 48 of *DIMACS*, pages 217–254. American Mathematical Society, 1999.
15. J.H. Reif, S. Sahu, and P. Yin. Compact error-resilient computational DNA tiling assemblies. In *Proc. 10th International Meeting on DNA Computing*, pages 248–260, 2004.
16. P.W.K. Rothemund. Using lateral capillary forces to compute by self-assembly. *Proc. Natl. Acad. Sci. USA*, 97(3):984–989, 2000.
17. S. Sahu and J.H.Reif. Capabilities and limits of compact error resilience methods for algorithmic self-assembly in two and three dimensions. Technical Report CS-2006-04, Duke University, 2006.
18. R. Schulman and E. Winfree. Programmable control of nucleation for algorithmic self-assembly. In *DNA Based Computers 10*, LNCS, 2005.
19. N.C. Seeman. DNA in a material world. *Nature*, 421:427–431, 2003.
20. D. Soloveichik and E. Winfree. Complexity of self-assembled shapes. In *DNA Based Computers 10*, LNCS, 2005.
21. H. Wang. Proving theorems by pattern recognition ii. *Bell Systems Technical Journal*, 40:1–41, 1961.
22. George M. Whitesides and Bartosz Grzybowski. Self-assembly at all scales. *Science*, 295:2418 – 242, 2002.
23. E. Winfree. Complexity of restricted and unrestricted models of molecular computation. In R. J. Lipton and E.B. Baum, editors, *DNA Based Computers 1*, volume 27 of *DIMACS*, pages 187–198. American Mathematical Society, 1996.
24. E. Winfree. Simulation of computing by self-assembly. Technical Report 1998.22, Caltech, 1998.

25. E. Winfree. Self-healing tile sets. *Nanotechnology: Science and Computation*, 2006. Preprint. One-page abstract in proceedings of FNANO 2005.
26. E. Winfree and R. Bekbolatov. Proofreading tile sets: Error correction for algorithmic self-assembly. In *DNA Based Computers 9*, volume 2943 of *LNCS*, pages 126–144, 2004.
27. E. Winfree, F. Liu, L.A. Wenzler, and N.C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–544, 1998.
28. E. Winfree, X. Yang, and N.C. Seeman. Universal computation via self-assembly of DNA: Some theory and experiments. In L.F. Landweber and E.B. Baum, editors, *DNA Based Computers II*, volume 44 of *DIMACS*, pages 191–213. American Mathematical Society, 1999.
29. X. Xiong, Y. Hanein, J. Fang, Y. Wang, W. Wang, D. Schwartz, and K. Bohringer. Controlled multibatch self-assembly of microdevices. *Journal Of Microelectromechanical Systems*, 12:117–127, 2003.
30. H. Yan, L. Feng, T.H. LaBean, and J.H. Reif. Parallel molecular computation of pair-wise xor using DNA string tile. *J. Am. Chem. Soc.*, 125(47), 2003.
31. H. Yan, T.H. LaBean, L. Feng, and J.H. Reif. Directed nucleation assembly of DNA tile complexes for barcode patterned DNA lattices. *Proc. Natl. Acad. Sci. USA*, 100(14):8103–8108, 2003.
32. H. Yan, S.H. Park, G. Finkelstein, J.H. Reif, and T.H. LaBean. DNA-templated self-assembly of protein arrays and highly conductive nanowires. *Science*, 301(5641):1882–1884, 2003.