Self-Correcting Self-Assembly: Growth Models and the Hammersley Process

Yuliy Baryshnikov[1], Ed Coffman[2], Nadrian Seeman[3], and Teddy Yimwadsana[2]

[1] Bell Labs, Lucent Technologies, Murray Hill, NJ 07974
[2] Electrical Engineering Dept., Columbia University, New York, NY 10027
[3] Chemistry Dept., New York University, New York 10003

1 Introduction

In many respects, the current state of DNA-based computing resembles the state of standard, electronic computing a half century ago: a fascinating prospect is slow to develop owing to inflexible interfaces and unacceptably low reliability of the computational processes. We concentrate in this paper on the latter aspect, specifically addressing the interplay between the reliability and speed of DNA computing.

While DNA-based computational devices are known to be extremely energy efficient, their reliability is seen as the greatest obstacle to becoming a viable computing environment. As DNA based computing becomes more fully developed, the speed of self assembly will become a crucial factor; but as of now, little is known concerning the fundamental question of computation times. We emphasize the intrinsic connection between the two problems of reliability and speed, because of the unavoidable trade-off that exists between them. A clear understanding of the limitations of self-assembly reliability and speed, specifically that of DNA-based computing, and the interplay between these properties, will be paramount in determining the full potential of the paradigm.

In our past work, which is briefly reviewed below, we analyzed, for a given function, the time required to determine its value on given inputs, and therefore, established theoretical limits on the performance of DNA-based computers. In the simplest instance, the analysis of computation times has surprising connections with interacting particle systems and variational problems, as shown in [1], and as further developed here. The critical new dimension of this paper lies in (a) a novel approach to dramatic improvements in the reliability of computations and (b) in the analysis of the inevitable performance losses of reliable computations.

Computation speed The early theoretical work on DNA-based computation focused chiefly on various measures of complexity, in particular, program-size and time complexity [2–4]. However, Adleman et al [2, 5] also investigated interesting combinatorial questions such as the minimum number of tile types needed for universality, and stochastic optimization questions such as the choice of concentrations that leads to minimum expected assembly times. Apart from these works, the mathematical foundations of computational speed in a stochastic context appear to be restricted to the ground breaking work of Adleman et al [6] and to the more extensive work of Baryshnikov et al [7, 8, 1, 9]. The former work studies random self assembly in one dimension. In a problem called $n-$*linear polymerization*, elementary particles or monomers combine to form progressively larger polymers. The research of Baryshnikov et al [8] on linear self assembly has resulted in exact results for dimer self assembly, which reduces to an interesting maximal matching problem.

Error correction. Any implementation of DNA computing is constrained fundamentally by the fact that all basic interactions have energy thresholds that are much lower than those in electronic devices. This means that any realistic computational device based on organic structures like DNA is forced to operate at signal-to-noise ratios several orders of magnitude lower than those in electronic computing. Therefore, error correction at the computation stage becomes a necessity. Recent research has focused on methods of error correction based on approaches that are analogous, at some level, to the repetition coding of information theory, or, in the computer science context, the concurrent execution of the same computational algorithm with subsequent comparison of results. Note also that biological computations achieve redundancy at little extra cost, as by the inherent virtues of the process, many copies of it are run independently.

However, in our view, an essential component has to be introduced into this paradigm. It involves the notion of *pulsing* and is analogous to checkpointing[4], in which computations are periodically tested to decide their validity. If a test certifies the outputs of the running processes, the inner state of the computing device is saved, and the computation resumes. Otherwise, the computation restarts from the last checkpoint. In our DNA-computing context, saving a state need not entail a specific operation as in classical computing applications. Yet the process of returning from an invalid state to a most recent valid state (i.e., *rescuing* the latest stage of a self assembled structure that was still error-free) also requires an operation that usually has a non-negligible (delay) cost, a cost that must be balanced by a higher speed of checkpointed computations.

Tile model. Formally, the now standard *tiling system* introduced and validated as a universal computational framework in [11, 12] will be our abstract model, and follows the adaptation to elementary logic units of DNA computing described by Winfree and Rothemund [13, 3]. The tile is modeled as shown in Figure 1 as a marked or labeled square. Briefly, in the simplest version, the label values are 0 or 1, and they break down into two input labels on one edge and two output labels on the opposite edge. As illustrated in Figure 1, a computational step consists of one tile bonding to others according to given rules that match input labels of one tile to the output labels of one or two adjacent tiles. Successive bonding of tiles in a self assembly process performs a computation.

---

[4] Checkpointing techniques have been in use since the early days of computing; see, e.g., [10]

Fig. 1. Since a tile glues to other tiles only if their corner labels match, operation $0 \oplus 1 = 1$ is performed. The correct output tile simply attaches itself to the preassembled input tiles, effectively yielding the output bit.

Currently, in a typical implementation of this scheme, the tiles are DNA-based molecular structures moving randomly, in solution, and capable of functioning independently and in parallel in a self assembly process. This process results in a crystal-like construct modeled as a two dimensional array whose allowable structural variations correspond to the possible results of a given computation. We emphasize the contrast with classical computing paradigms: the random phenomena of self assembly create a randomness in the time required to perform a given computation.

2 Growth Models

The tiling self assemblies of the last section are growth processes. Through the abstraction described next, the times to grow constructs or patterns can be related to classical theories of particle processes; growth in the latter processes is again subject to rules analogous to those governing the self assembly process of the previous section.

An initial set of tiles (the input) is placed along the coordinate axes, and growth proceeds outward in the positive quadrant; the placement of a new tile is allowed only if there are already tiles to the left and below the new tile's position which match labels as before. The left-and-below constraint is equivalent to requiring a newly added tile to bond at both of its input labels. The completion of the computation occurs at the attachment of the upper-right corner tile which can happen only after all other tiles are in place.

The fundamental quantity of interest is the computation time, or equivalently, the time until the final square (represented by the upper right corner square in position (M,N)) is in place. Let $T_{i,j}$ be the time it takes for a tile to land at position (i,j) once the conditions are favorable; that is, once both positions $(i,j-1)$ and $(i-1,j)$ are tiled. In a reference theory for self assembly, it is natural to take the $T_{i,j}$'s as independent exponential random variables with unit means. Let $C_{i,j}$ be the time until the square (i,j) becomes occupied, so that the random completion time of interest is given by $C_{M,N}$.

On discovering the isomorphic relationship between the self assembly process and the totally asymmetric simple exclusion process (TASEP), Baryshnikov et al [1] exploited the results on TASEP behavior in the hydrodynamic limit to show that, as N, M grow to infinity such that M/N tends to a positive constant, one has [14, p. 412] $C_{M,N}/(\sqrt{M} + \sqrt{N})^2 \sim 1$, a formula quantifying the degree of parallelism in the computation. One can generalize this formula to more general shapes and to schemes where tiles can depart as well (like the schemes described in [15]). Growth is no longer monotonic, but still can be mapped to a generalization of TASEP for which similar results are known. One obtains [1]

Theorem 1 *The time $E_{\lambda\mathcal{D},\rho}$ required to complete computation on a DNA computer of shape $\lambda\mathcal{D}$ with tiles arriving at rate 1 and departing at rate ρ is given by*

$$\lim_{\lambda \to \infty} \lambda^{-1} E_{\lambda\mathcal{D}} = \frac{1}{1-\rho} \sup_{\gamma} \int \left(\sqrt{\frac{d\xi}{dz}} + \sqrt{\frac{d\eta}{dz}} \right)^2 dz. \tag{1}$$

3 Error Correction with Checkpointing

Checkpointing is a popular error-correction method implemented in operating systems. Typically a control mechanism periodically creates checkpoints, at different times (milestones) in the process. State information is stored at a checkpoint so that the process can restart from that location without having to perform work done before the checkpoint. Thus, when the controlled process fails, it is rolled back to the most recent checkpoint and resumes from that point.

We expect the checkpointing method, being a simple but elegant error-correction technique to become a viable tool in the area, at least until a dramatic change in the underlying chemical technology takes place. The narrow question we address below is how to apply it to DNA tile self-assembly. First, however, we briefly review other approaches.

Alternative approaches The two most frequent errors in DNA self-assembly are growth errors and nucleation errors. Growth errors occur when a wrong type of tile, an *error* tile, attaches to the lattice; a sublattice that forms with the error tile at its origin will then be corrupt. A nucleation error occurs when only one side of a tile attaches to the lattice, and hence at a wrong position. Thermodynamic controls that slow down growth can be introduced to help ensure the relatively early separation of error tiles.

A tile can also be designed to have its own error-correction capability, or a new type of tile that assists the self-assembly process in lowering error rate can be introduced. Several methods for this have been proposed. For example, Winfree and Bekbolatov's Proofreading Tile Set [15] shows that the error rate can be reduced significantly by creating an original Wang Tile using four or nine smaller tiles (2×2 or 3×3) in order to ensure that the small incorrect tiles will fall off before they are assembled to form an incorrect Wang tile. Chen and Goel's Snake Tile Set [16] improves the Proofreading Tile Set by ensuring that the smaller tiles can be assembled only in certain directions.

Reif et al [17] use pads to perform error checking when a new tile is attached to the lattice. Each pad acts as a kind of adhesive, connecting two Wang tiles together, whereas in the original approach the Wang tiles attach to each other. This method allows for redundancy: a single pad mismatch between a tile and its immediate neighbor forces at least one further pad mismatch between a pair of adjacent tiles. This padding method can be extended further to increase the level of redundancy.

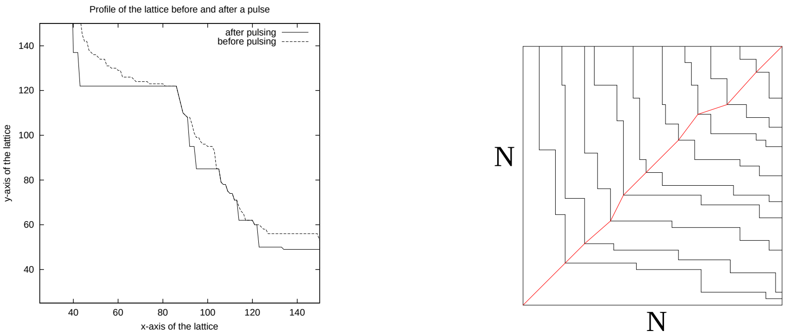
Chen et al's Invadable Tile Set [18] applies the invading capability of the DNA strand to emulate the invasion of a tile. In this model, the tiles are designed so that the correct tile can invade any incorrect tile during the lattice growth process. Fujibayashi and Murata's Layered Tile Model [19] significantly reduces the error rate by using two layers of tiles: the Wang tile layer and the protective tile layer. The protective layer does not allow tiles to attach to the lattice incorrectly. When the attachment is correct, the protective tile releases the rule tile, to which the next tile attaches itself. As one must expect, all methods have one or more shortcomings or costs associated with them, such as prolonged self-assembly times, enlarged lattices, potential instabilities, and failure to deal effectively with both error types.

Modeling checkpointing in DNA self-assembly Temperature pulsing, a widely used method to grow crystals, is analogous to checkpointing. Periodic pulses remove the defective parts of a crystal; in particular, the hydrogen bonds between improperly attached DNA tiles are broken so that defective substructures can separate from the lattice, thus restarting growth at an earlier fault-free structure. Parameters other than temperature can also be considered in the pulsing approach. Pulsing applied to the DNA tile self-assembly model removes the incorrectly attached tiles from the assembly at a higher rate than the correct ones. More targeted pulsing systems can employ enzymatic or conformational ways to shift the binding energy.

In our model of self-assembly with checkpointing, we consider a lattice of size $N \times N$. (While our results are valid for general shapes, the square lattice helps focus discussion.) We study the standard growth process described earlier, with the modification that there are two competing populations of tiles to be attached, correct tiles and erroneous tiles. With an appropriate rescaling, the waiting time until a tile attaches at a vacant position is taken to be exponential with mean 1 (all attachment times are independent). Attached tiles are erroneous with probability q.

We call a tile that attaches to a valid structure a *seed error tile.* Any tile attached to the north east of the seed error tile is automatically an error tile as it participates in an incorrect computation. (See Figure 2(a)). In our initial analysis we assume that a pulse succeeds in washing out all defective regions of the structure. (See Figure 2(a).)

A growth *stage* consists of a growth period of duration P between consecutive pulses. At the end of one such stage, the locations of the seed error tiles define the boundary of the lattice for the next growth period, on which more tiles will be attached. A growth *layer* is the region of tiles that attach to the lattice during



(a) T_A (b) L_p

Fig. 2. (a) - the profile of DNA tile self-assembly process before and after a pulse. (b) the number of layers and the longest increasing subsequence

one growth step. The number of stages required to complete the $N \times N$ lattice is the number of pulses or layers required to complete the lattice.

The shapes at the beginning of each growth stage form a Markov process which is clearly significantly more complicated than the growth process without pulsing. Moreover, it is easily seen that these processes cannot be mapped onto 1-dimensional particle processes with local interaction. Hence to evaluate performance, we are forced to resort to asymptotic analysis and simulation studies.

Small q asymptotics The number of pulses required to complete a crystal can be approximated using Hammersley's Process. In our version of this process, we have an underlying Poisson process in two dimensions with samples ω taken from the square $S = [0, a] \times [0, a]$. For each $z = (x, y) \in S$ let $n(z)$ be the length of the longest *monotone subsequence* in ω between $(0, 0)$ and z, that is the maximal length ℓ of a chain of points $(x_1, y_1) < (x_2, y_2) < \ldots < (x_\ell, y_\ell) < (x, y)$, where $(u, v) < (u', v')$ iff $u < u'$ and $v < v'$. See Figure 2(b).

The problem of finding this length is closely related to the famous problem of finding the longest increasing subsequence in a random permutation (Ulam's problem). It turns out that the expected value $\mathbb{E}\ell \sim 2a$, as $a \to \infty$, see, e.g., [20].

This implies immediately some information on the asymptotic scaling of the number of pulses L_p in the limit of small q and large N. Indeed, we have

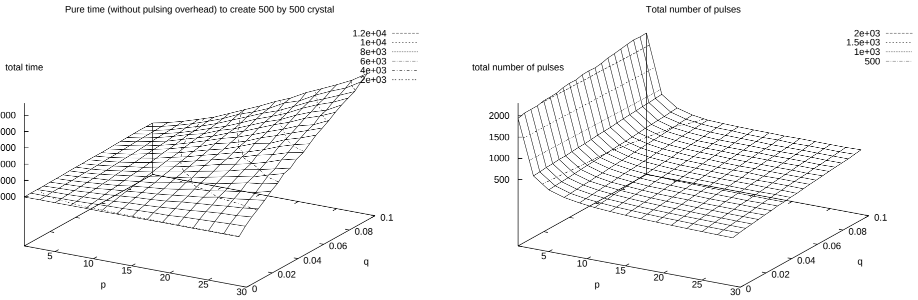
Theorem 2 *As $P, N \to \infty$, and $q \to 0$ in such way that $q = o(P/N)$, the number L_p of pulses required for an error-free assembly grows as*

$$L_p \sim 2N\sqrt{q}.$$

Using this result, we can adjust the value of P so as to obtain an estimate of the minimal time required to complete the lattice for any given N, q, and average time p_s taken by a pulse to remove error tiles. More details can be found in [21].

Remark. The Layered Tile Set [19] can be seen as a variant of our method where P is 1 time unit. When the value of P is very small, the total number of pulses becomes very large because the process pulses once per time unit. As a result, when the value of P is small, the completion time for the formation of the crystal is inversely large. Furthermore, in the case of a high p_s, a very low value of P will not be suitable for the process because of the length of time required during the checkpointing process. Therefore, if P is adjusted appropriately, our checkpointing method will be better than the Layered Tile Set technique.

Simulation analysis The total crystal completion time, T, consists of the total time required by tile attachment, T_A, pulsing setup time and the pulsing overhead, T_p. Our simulations determine the effect of P and q on T_A and T_p. The simulation of a 500×500 lattice yielded T_A and T_p for different values of P and q. The total pulsing overhead time, T_p, is given by $T_p = p_s L_p$ (recall that p_s is the average time taken by a pulse to remove all erroneous tiles).



(a) T_A

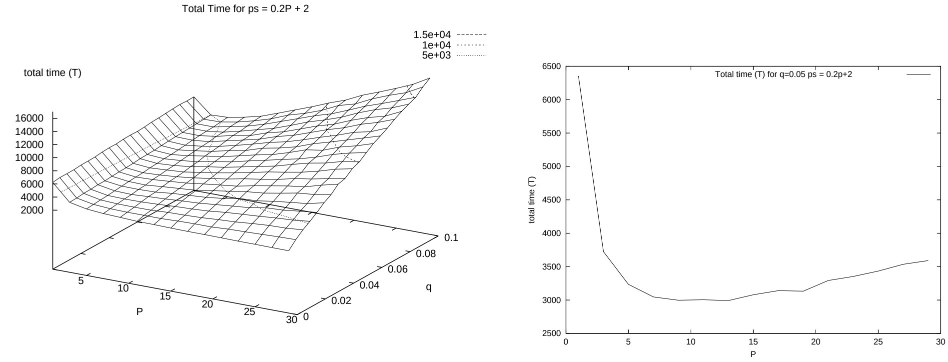
(b) L_p

Fig. 3. The performance of pulsing for various P and q. Crystal size: 500×500

Our self assembly simulations created more than a million tiles. Developing the simulator was a challenge in itself, given current limits on computer memory. We designed our simulator so that it contains only the information of the crystal, which for our purposes will suffice without having to assign memory space for each tile. Implementation details can be found in [21].

Figure 3 shows the effects of P and q on the performance of self-assembly with pulsing. Since the total time T required to complete the crystal is $T_p + T_A$, we see that T in general has an optimal point for given p_s.

For example, Figure 4(a) shows the total-time surface plot as a function of (P, q). For simplicity, we assume that the time required for each pulse, p_s, is



(a) Total completion time

(b) Total time when $q = 0.05$

Fig. 4. (a) total-time pulsing performance, various (P,q). (b) cross section of (a), q=0.05

linearly proportional to the growth time, $p_s = 0.2P + 2$, to show how p_s can affect the total time, T. For a given value of q, there is an optimal P that minimizes the total time to complete the self-assembly. Figure 4(b) shows the total time for different values of P with the error probability $q = 0.05$. The figure shows that one obtains the highest over-all lattice growth rate when P is approximately 9 time units.

4 Future Directions

We have introduced and analyzed the performance of an error-correcting self assembly pulsing/checkpointing technique. This comprises the modeling and analysis component of an over-all project that will include the essential experimental component. To fully verify the validity of the method, we propose an experimental setup that produces a simple periodic system capable of errors, as a testbed for examining our proposed error-correcting techniques. It is possible to 'tag' particular elements in 2D crystals; the addition of a DNA hairpin that sticks up from the plane of the array is the simplest tag. Thus, rather than using a single motif to act as a 'tile' in forming the crystal, we can use two or more different tiles, say, an A-tile and a B-tile. An example of this approach is shown for DX molecules in Figure 5.

The experiments we propose here are based on this notion of two differently marked tiles. The idea is to make a 2-D array with two tiles whose sticky ends are distinct, as they are in the DX example above. It is unlikely that individual 4 nm x 16 nm DX tiles can be recognized, but many motifs with large dimensions in both directions exist (see e.g., [22]), and they would be appropriate for this experiment. The uniqueness of the sticky ends is central to the robust formation

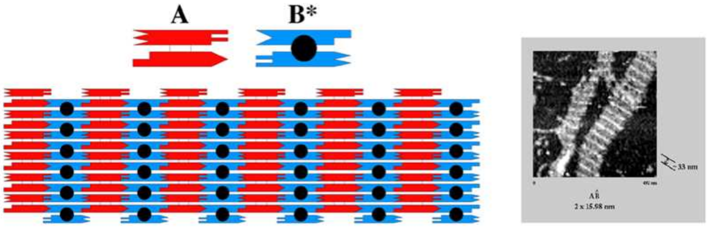


Fig. 5. A Schematic DX Array with 2 Components, A and B*; B* contains a hairpin (the black circle) that protrudes from the plane. Sticky ends are shown geometrically. Note that the A and B* components tile the plane. Their dimensions in this projection are 4 × 16 nm, so a 32 nm stripe is seen on the AFM image to the right.

of the pattern shown above. Were the sticky ends of the two molecules identical, a random array of protruding features would result, rather than the well-formed stripes shown. It is clear that there must exist some middle ground between unique sticky ends and identical sticky ends. Each tile contains four sticky ends, each of six or so nucleotides. We propose to explore steps on the way from uniqueness to identity (starting from unique) so that we can get a set of tiles that produce an array with a well-defined error rate, low but detectable.

Once we have such a system, it will then be possible to do prototype rescue operations, the results of which are expected to complement and validate the modeling approach. The basis of these operations will be thermodynamic pulsing, but the study of techniques based on recognition of structural differences in the 2D array, or some combination of the two basic approaches, will be pursued as well.

References

1. Baryshnikov, Y., Coffman, E., Momčilović, P.: DNA-based computation times. In: Proc. Tenth International Meeting on DNA Computing, Milan, Italy (2004)
2. Adleman, L., Cheng, Q., Goel, A., Huang, M.D.: Running time and program size for self-assembled squares. In: Proc. ACM Symp. Th. Comput. (2001) 740–748
3. Rothemund, P., Winfree, E.: The program-size complexity of self-assembled squares. In: Proc. ACM Symp. Th. Comput. (2001) 459–468
4. Winfree, E.: Complexity of restricted and unrestricted models of molecular computation. In Lipton, R., Baum, E., eds.: DNA Based Computing. Am. Math. Soc., Providence, RI (1996) 187–198
5. Adleman, L., Cheng, Q., Goel, A., Huang, M.D., Kempe, D., de Espanés, P.M., Rothemund, P.: Combinatorial optimization problems in self-assembly. In: Proc. ACM Symp. Th. Comput., Montreal, Canada (2002) 23–32
6. Adleman, L., Cheng, Q., Goel, A., Huang, M.D., Wasserman, H.: Linear self-assemblies: Equilibria, entropy, and convergence rates. In Elaydi, Ladas, Aulbach, eds.: New progress in difference equations, (Taylor and Francis, London (2004))
7. Baryshnikov, Y., Coffman, E., Momčilović, P.: Incremental self-assembly in the fluid limit. In: Proc. 38th Ann. Conf. Inf. Sys. Sci., Princeton, NJ (2004)

8. Baryshnikov, Y., Coffman, E., Winkler, P.: Linear self-assembly and random disjoint edge selection. Technical Report 03-1000, Electrical Engineering Dept., Columbia University (2004)
9. Baryshnikov, Y., Coffman, E., Momčilović, P.: Phase transitions and control in self assembly. In: Proc. Foundations of Nanoscience: Self-Assembled Architectures and Devices, Snowbird, UT (2004)
10. E. G. Coffman, J., Flatto, L., Wright, P.E.: A stochastic checkpoint optimization problem. SIAM J. Comput. **22** (1993) 650–659
11. Wang, H.: Dominoes and AEA case of the decision problem. In: Proc. of the Symposium in the Mathematical Theory of Automata, Polytechnic Press, Brooklyn, NY (1963)
12. Berger, R.: The undecidability of the domino problem. In: Memoirs, Amer. Math. Soc. (66). (1966)
13. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology, Pasadena, CA (1998)
14. Liggett, T.M.: Interacting Particle Systems. Springer-Verlag, New York (1985)
15. Winfree, E., Bekbolatov, R.: Proofreading tile sets: Error correction for algorithmic self-assembly. In: Proceedings of the Ninth International Meeting on DNA Based Computers, LNCS volume 2943. (2004) 126–144
16. Chen, H.L., Goel, A.: Error Free Self-Assembly with Error Prone Tiles. In: the Proceedings of the 10th International Meeting on DNA Based Computers, Milan, Italy, month = (2004)
17. Reif, J.H., Sahu, S., Yin, P.: Compact error-resilient computational dna tiling assemblies. In: Proceedings, Tenth International Meeting on DNA Based Computers, Lecture Notes in Computer Science, Springer-Verlag, New York (2004)
18. Chen, H.L., Cheng, Q., Goel, A., Huang, M.D., de Espanes, P.M.: Invadable self-assembly: Combining robustness with efficiency. In: ACM-SIAM Symposium on Discrete Algorithms. (2004)
19. Fujibayashi, K., Murata, S.: A method of error suppression for self-assembling DNA tiles. In: Proceedings of the Tenth International Meeting on DNA Based Computers: Lecture Notes in Computer Science, Springer Verlag, New York (2004) 284–293
20. Aldous, D., Diaconis, P.: Hammersley's interacting particle process and longest increasing subsequences. Probab. Th. Rel. Fields **103** (1995) 199–213
21. Baryshnikov, Y., Coffman, E., Yimwadsana, T.: Analysis of self-correcting self-assembly growth models. Technical Report 03-1001, Electrical Engineering Dept., Columbia University (2005)
22. Mao, C., Sun, W., Seeman, N.: Designed two-dimensional dna holliday junction arrays visualized by atomic force microscopy. J. Am. Chem. Soc. **121** (1999) 5437–5443