Connecting the Dots: Molecular Machinery for Distributed Robotics

Yuriy Brun and Dustin Reishus

University of Southern California, Los Angeles, CA, USA
{ybrun,reishus}@usc.edu

Abstract. Nature is considered one promising area to search for inspiration in designing robotic systems. Some work in swarm robotics has tried to build systems that resemble distributed biological systems and inherit biology's fault tolerance, scalability, dependability, and robustness. Such systems, as well as ones in the areas of active self-assembly and amorphous computing, typically use relatively simple components with limited computation, memory, and computational power to accomplish complex tasks, such as forming paths in the presence of obstacles. We demonstrate that such tasks can be accomplished in the well-studied tile assembly model, a model of molecular self-assembly that is strictly simpler than other biologically-inspired models. Our systems use a small number of distinct components to find minimal-length paths in time linear in the length of the path while inheriting scalability and fault tolerance of the underlying natural process of self-assembly.

1 Introduction

Swarm robotics, active self-assembly, and amorphous computing are fields that focus on designing systems of small, simple components that are capable of cooperating to complete complex tasks. Many of these systems have been inspired by biological systems seen in nature, so we will refer to them as biologically-inspired systems. Work on biologically-inspired systems started in theoretical explorations [1,2,3,4,5,6] and fueled the creation of distributed robotic systems in hardware, in which individual robots with limited capabilities come together in swarms to exhibit complex emergent behaviors [7,8,9,10,11]. Because systems are built out of simple, and therefore cheap, components, creating a large number of components is typically not a concern, but the number of distinct components is. To further reduce the cost of the component-manufacturing process, many of the systems strive to allow for unreliable components. In large, much of the work in these fields is inspired by biological systems that not only build complex systems out of simple and cheap components, but that also often deal with faulty and malicious agents.

Biologically-inspired systems are typically made up of a large number of identical components that are resource- and computational power-limited agents. Various researchers have defined distinct models for studying such systems; the models differ in the components, in the types of interactions between components, and in the environmental resources available to the components. The primary goal behind the creation of many of these models is to use the simplest components to achieve complex behavior, such as the assembly of shapes or formation of paths between points. While these may not seem like complex tasks on first inspection, these behaviors can be used as primitives to accomplish more practical results. For example, the path-forming primitive can be used to form wires between two electrodes on a surface.

In our approach, we show that an extremely simple model of components with practically no memory, no communication, and no control requirements are capable of accomplishing many of the tasks commonly presented in related literature. To that end, we leverage the tile assembly model [12,13,14], a formal model of crystal growth. It was designed to model self-assembly of molecules such as DNA, and thus its components are no more complex than oversimplified biological molecules. It is an extension of a model proposed by Wang [15] in 1961. In essence, a component is a square with a label on each of its four sides. Components cannot change their labels, nor input or output any information. They can, however, attach to other components if the labels on their abutting sides match. The tile assembly model is a formal mathematical model, which allows for the study of assembly time and tileset complexities. Many other biologically-inspired systems lack the formalism to allow this type of study. We will present tile systems that find paths between points, and show that these systems exhibit the same robustness demonstrated by other biologically-inspired systems.

2 Related Work

The work presented in this paper builds on the tile assembly model to solve problems commonly found in biologically-inspired systems literature, such as path finding. Thus, we will first discuss the work in biologically-inspired systems in Section 2.1 and then review work related to the tile assembly model in Section 2.2. We will define the tile assembly model in Section 3 and explain our path-finding system in Section 4. Finally, we will conclude in Section 5.

2.1 Biologically-Inspired Systems

Perhaps the first instance of using simple components to solve the path-finding problem was in the paintable computing model. Paintable computing was inspired by the idea of placing cheap, unreliable, and tiny (invisible to the naked eye) components into paint, and covering a wall or other surface with that paint. The components remain stationary on the wall and are able to communicate wirelessly with their neighbors to accomplish certain tasks, for example forming a wire between a light switch and a light fixture on the wall (an instance of the path-finding problem) or displaying a photograph (an instance of shape construction). Pushpin computing was the first physical implementation of a paintable-computing-like system. Butera created cubic-inch-sized immobile robots that could be pinned to a special wall made of foil [7]. The wall provided the robots with power, and they, in turn, could communicate with a small radius of neighbors and turn on and off LED lights.

Abelson et al. [1] formally defined an amorphous computer to be a 2-D sheet with randomly placed immobile robots. The robots have wireless communication capability with radius far smaller than the sheet, and their computational abilities are restricted to be less powerful than Turing machines, but are otherwise left open. In general, these robots are expected to have some memory and a finite control. This definition formed a medium for researchers to test the power of simple components and to experiment with programming those components to complete complex tasks. The path-finding problem in this model was solved in [7] with the use of messages similar to chemical gradients used in biological systems. Several extensions of this model exist, and the path-finding problem has been solved in almost all of them. Nagpal's extension to the amorphous computer model allows the 2-D sheet to fold along a line. She solved the path-finding problem and related mathematical work on origami to show that it is possible to compile an origami-folding procedure for a given structure into a program, such that when an identical copy of the program is loaded onto each of the robots, the robots self-organize to create that shape [5].

Clement et al. looked at ways of making the amorphous computing algorithm that solves the path-finding problem more robust to failing robots [3]. While most algorithms are resilient to holes and broken robots at the time of self-organization, this work looked at situations in which robots may fail during or after the algorithm's execution. They came up with modified algorithms to generate lines between points that, essentially, continually check for a line's validity, and if a line is no longer valid, regenerate a new line to fix the problem.

Later, Nagpal et al. showed that it is possible for the robots of an amorphous computer to self-organize into coordinate systems, with each robot knowing its coordinate, and thus display preprogrammed images (given some ability to shine light) [6]. Their robots send out messages similar to the gradient discussed in the line-formation procedure, and robots can, in essence, triangulate their positions on the sheet.

Arbuckle et al. developed their own model, similar to the amorphous computer. They concentrated on limiting the robots to only a few bits of memory, and allow the robots to move on a 2-D surface. They demonstrated the ability to build paths, assemble shapes, and repair formed paths and shapes [2].

A number of researchers have worked on implementing systems of robots in hardware. These implementations commonly have dozens of robots, rather than millions as is often assumed in the theoretical work, and each robot is actually far more complex and expensive than the theoreticians would like them to be. Werfel et al. showed the ability for distributed robots to move blocks to form shapes [11]. McLurkin et al. used mobile robots to assemble into groups based on the sounds they were making, self-organizing into robotic orchestras [9]. Klavins worked with triangular robots with programmable side interfaces that can attract or repel each other to assemble shapes and study assembly dynamics [8]. Shen et al. demonstrated reconfigurable robots, made up of identical basic units, self-organizing to crawl, walk, climb, as well as perform other complex tasks [10].

While a wealth of literature exists on biologically-inspired systems, this literature lacks the organization and common definitions necessary to effectively compare the complexity of the basic components or the complexity of the tasks performed by the systems. Our systems presented in this paper use components far simpler than the ones described in this section (they have no finite control, minimal read-only memory, and incredibly limited communication abilities) and perform some of the same tasks we have described thus far.

2.2 Self-assembly

Research in self-assembly attempts to explain how simple objects come together on their own to form more complex objects capable of more complex behaviors. Self-assembly is a process that is ubiquitous in nature. Systems form on all scales via self-assembly, e.g., atoms self-assemble to form molecules, molecules to form complexes, and stars to form galaxies. One manifestation of self-assembly is crystal growth: molecules self-assemble to form crystals. The tile assembly model [12,13,14] is a formal model of such crystal growth.

One of the potential applications of the tile assembly model is self-assembling electronic circuits [16,17]. Researchers have shown that it is possible to attach simple, electronically-active components to DNA tiles and use the self-assembling interactions of the tiles to arrange these components [18]. One of the most basic tasks one might want to use self-assembly to complete is constructing a wire between two points. This task involves finding a path between them. One might further specify that the path should be short, use no extraneous components, or perhaps avoid certain regions (for example, other circuit elements). We will present a system that accomplishes these tasks.

One similarity between the study of self-assembly and other biologically-inspired systems is that researchers have identified the problem of forming shapes as important in both fields. It is possible to build shapes using tiles, simpler components than the ones used in other biologically-inspired systems, to create arbitrary computable shapes. Adleman proposed studying the complexity of tile systems that can uniquely produce $n \times n$ squares. A series of researchers [14,19,20,21] proceeded to answer the questions “what is a minimal tile set that can assemble such shapes?” and “what is the assembly time for these systems?” They showed that the minimal tile set that assembles $n \times n$ squares is of size $O\left(\frac{\log n}{\log \log n}\right)$ and the optimal assembly time is $\Theta(n)$ [20]. A key issue related to assembling squares is the assembly of small binary counters, which theoretically can have as few as 6 or 7 tile types [22,21].

The path-finding problem is related to the “domino snake problem” that asks whether a given tileset can form a path between two points. On the whole plane, the domino snake problem turns out to be decidable; however, if there are obstacles or regions that the path must avoid, the problem may become undecidable [23].

Researchers have also studied variations on the traditional tile assembly model. Aggarwal et al. and Kao et al. have shown that changing the temperature of assembly from a constant throughout the assembly process to a discrete function reduces the minimal tile set that can build an $n \times n$ square to a size $\Theta(1)$ tile set [24,25]. In our work with path-finding systems, we allow the temperature to change once.

Soloveichik et al. studied assembling all decidable shapes in the tile assembly model and found that the size of the minimal set of tiles necessary to uniquely assemble a shape is directly related to the Kolmogorov complexity of that shape [26]. One of the tasks commonly used to demonstrate power in biologically-inspired systems is the construction of simple shapes. What Soloveichik et al. showed is that systems in the tile assembly model are capable of assembling all decidable shapes, on some scale. While we do not go into great depth on shape construction in this paper, tile assembly model's ability to construct shapes is one indicator of this model's ability to perform the same tasks other biologically-inspired systems perform.

3 Tile Assembly Model

The tile assembly model [12,14], a formal mathematical model of self-assembly, can compute functions and is Turing universal. It is an extension of a model proposed by Wang [15]. It was designed to model crystal growth via self-assembly of molecules such as DNA. The model was fully defined by Rothemund and Winfree [14], and the definitions here are similar to those, though we make a slight extension to allow for growth and decay of crystals. Full formal definitions can be found in [27].

Intuitively, the model has *tiles*, or squares, that stick or do not stick together based on various *binding domains* on their four sides. Each tile has a binding domain on its north, east, south, and west side. The four binding domains, elements of a finite alphabet Σ, define the type of the tile. The strength of the binding domains are defined by the *strength function* g. The placement of some tiles on a 2-D grid is called a *configuration*, and a tile may *attach* in empty positions on the grid if the total strength of all the binding domains on that tile that match its neighbors exceeds the current *temperature* and *detach* if the total strength of all the binding domains on a tile in a configuration that match its neighbors is below the current temperature. Finally, a *melting tile system* $\mathbb{S}$ is a quadruple $\langle T, g, \tau_g, \tau_m \rangle$, where T is a finite set of tiles, g is a strength function, and $\tau_g, \tau_m \in \mathbb{N}$ are two temperatures , where $\mathbb{N} = \mathbb{Z}_{\geq 0}$.

Starting from a *seed configuration* S, tiles may attach or detach at temperature τ_g to form new configurations. At some *switching time*, the temperature changes to τ_m and tiles continue to attach and detach. If that process terminates, the resulting configuration is said to be *final.* At some times, there may be a position where more than one tile can attach, there may be more than one position where a tile can attach, or there may be more than one position where a tile can detach. If, for all sequences of tile attachments, all possible final configurations are identical, then $\mathbb{S}$ is said to produce a *unique* final configuration on S. The *assembly time* of the system is the minimal number of steps it takes to build a final configuration, assuming maximum parallelism.

4 Path-Finding

Path finding is the problem of forming a path between two points on a 2-D plane. Intuitively, given a seed configuration with a single start tile, a single goal tile, and some number of special *obstacle* tiles, a path-finding system should attach tiles to connect the start to the goal, avoiding all the *obstacle* tiles. It is straightforward to design such a system that leaves extraneous tiles: simply fill fill the plane with tiles and claim that the path is there. Thus we wish to restrict systems to leave no extraneous tiles in the final configuration.

Informally, for a tile system $\mathbb{S}$ to solve the path-finding problem, starting from a configuration with a single S tile, a single G tile, and some *obstacle* tiles, $\mathbb{S}$ must produce a final configuration F that contains a path of connected tiles from S to G of minimal length, and every tile in F must be on such a path. Due to space limitations, we refer the reader to [27] for the formal definition of the path-finding problem.

The path-finding problem is analogous to the path-finding relatives that researchers have solved previously [1,2,3,5]. Demonstrating that there exists a tile system that solves the path-finding problem indicates that it can be solved with simpler basic components than those used in the related work.

We now describe the melting tile system $\mathbb{S}_{cpf}$ that solves the path-finding problem. Figure 1(a) shows the start (S) and goal (G) tiles. The start tile has an $\mathsf{n}, \mathsf{e}, \mathsf{s}$, and w binding domain on its north, east, south, and west sides, respectively, the goal tile is covered with γ binding domains, and the *obstacle* tile is covered with x binding domains.

Figure 1(b) shows the four tiles of T_{cpf}. Each tile is labeled with an arrow (we explain the meaning of the arrow later). The glue strength function g_{cpf} is defined as follows:

- The x binding domain binds with strength 0 to every other binding domain,
- The γ binding domain binds with strength 1 to every other binding domain, except x, and
- All other binding domains bind with strength 2 to themselves and 0 to others.

Let us examine the intuition behind $\mathbb{S}_{cpf} = \langle T_{cpf}, g_{cpf}, 2, 3\rangle$. Figure 2(a) shows a sample seed configuration with a four-tile obstacle. At temperature 2, tiles will

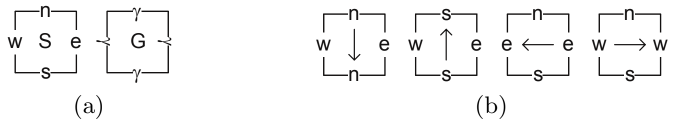

Fig. 1. $\mathbb{S}_{cpf}$ uses a special start (S) and goal (G) tile (a) and four "working" tiles (b)

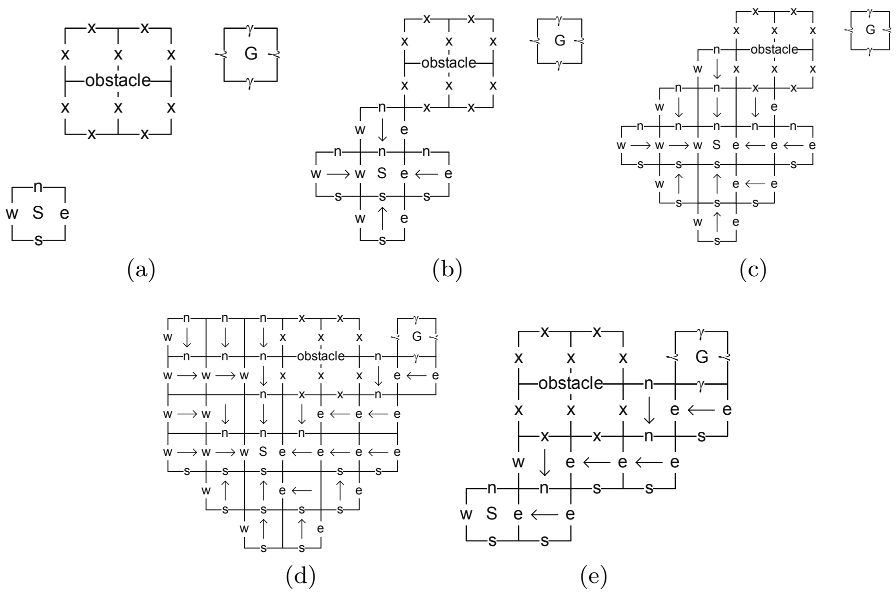

Fig. 2. An example execution of $\mathbb{S}_{cpf}$. $\mathbb{S}_{cpf}$ works at temperature 2 (a-d) to build possible paths and then at temperature 3 (e) to prune unsuccessful paths. Only binding domains that are exposed or attached have been labeled.

attach to S to create possible paths toward G. Each of the four tiles in T_{cpf} is designed to attach in a specific way to an existing assembly: each of the tiles has exactly one of its four domains be "irregular" (where "regular" means n for north, e for east, s for south, and w for west). The growing assembly will always have regular domains on all its exposed sides, thus tiles may only attach via their single irregular domain. Figures 2(b) and 2(c) show tile attachments after 1 and 2 steps, respectively. Paths may turn and fork in their attempts to reach G. Once G is reached, the last tile attaches with a total strength of 3 (2 via its irregular binding domain and 1 to G). Figure 2(d) shows some possible attachments after the system has been running for some time and has reached G. We can now increase the temperature to 3. Partial paths detach, one tile at a time, because the tiles on one end of each of those paths are only connected by a single strength 2 attachment. All partial paths detach, while the successful paths remain. Figure 2(e) shows the final configuration encoding a single path from S to G that avoids the obstacles.

To show that $\mathbb{S}_{cpf}$ solves the path-finding problem, we need a notion of distance in systems with obstacles. The notion we choose is the obstructed Manhattan distance. The obstructed Manhattan distance between two points on a 2-D grid is the fewest number of unit-sized steps one has to take from one point to get to the other, without stepping on an obstacle. Note that the obstructed Manhattan distance can be quite a bit larger than the Manhattan distance, and even infinite between two points that are unreachable from each other via a walk.

The melting tile system $\mathbb{S}_{cpf}$ solves the path-finding problem with the switch time on the order of the obstructed Manhattan distance between the start and the goal, if we assume maximum parallelism: whenever a tile can attach, it does, and whenever a tile can detach, it does. In actual physical implementations of the tile assembly model, it is far more likely that attachments and detachments happen stochastically, with rates that are related to the bond strength. $\mathbb{S}_{cpf}$ may not always produce minimal-length paths without the maximum parallelism assumption, but with high probability, the length of the solution path is on the order of the obstructed Manhattan distance. Due to space constraints, we cannot provide the proofs of these statements here, and we refer the reader to [27] for these proofs, as well as the explanation of a slightly simpler system that solves a variant of the path-finding problem without obstacles.

While the theoretical definitions do not require knowing the proper switch time, in practice it may be helpful to know when to increase the temperature. The switch time is $\Theta(d)$, where d is the obstructed Manhattan distance between S and G. More specifically, the proper minimum switch time is exactly $d - 1$. In practice, if the distance d is known, one can wait that long to increase the temperature. If d is unknown, one can devise an algorithm of increasing and decreasing the temperature repeatedly, perhaps increasing the length of switch time exponentially, such that in expectation a path can be found quickly.

5 Contributions

A number of nature-inspired systems [1,2,4,5,6,10,11] attempt to use simple components with limited computational power to come together to accomplish complex tasks. The tasks commonly accomplished by these systems include finding paths between two points in 2-D space and assembling shapes. The reason for the desire to use simple components is that they are cheap to produce in bulk. We have presented a tile assembly system that finds paths between two points with obstacles present. This system uses components far simpler than those used in the related work. The components' interfaces are static and each component performs no computation, has no controlled movement, and has no writable memory (the binding domains are read-only memory). Components such as these have been built out of DNA [28,29,30,31,32] at incredibly low costs.

While we have not discussed fault-tolerance within tile systems, an entire field of related research exists on making tile systems tolerant to individual tile failures [33]. One could apply the ideas in that related work directly to the tile systems we describe in this paper to make these systems able to perform successfully despite high probabilities of tiles failing, usually at the cost of slowing down the system assembly. We have also discussed related work demonstrating that tiles can be used to assemble arbitrary computable shapes and to solve Turing-complete problems, showing strong reason to believe that even though tiles are simpler than the components described in the related work, together they can accomplish the same tasks as those components.

References

1. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight Jr., T.F., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R.: Amorphous computing. Communications of the ACM 43(5), 74–82 (2000)
2. Arbuckle, D.J., Requicha, A.A.G.: Active self-assembly. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004), New Orleans, LA, USA, pp. 896–901 (April 2004)
3. Clement, L., Nagpal, R.: Self-assembly and self-repairing topologies. In: Proceedings of the Workshop on Adaptability in Multi-Agent Systems, RoboCup Australian Open (January 2003)
4. Kondacs, A.: Biologically-inspired self-assembly of two-dimensional shapes using global-to-local compilation. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico (August 2003)
5. Nagpal, R.: Programmable Self-Assembly: Constructing Global Shape Using Biologically-Inspired Local Interactions and Origami Mathematics. PhD thesis, Massachussetts Institute of Technology, Cambridge, MA, USA (June 2001)
6. Nagpal, R., Shrobe, H.E., Bachrach, J.: Organizing a global coordinate system from local information on an ad hoc sensor network. In: Zhao, F., Guibas, L.J. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 333–348. Springer, Heidelberg (2003)
7. Butera, W.J.: Programming a Paintable Computer. PhD thesis, Massachussetts Institute of Technology, Cambridge, MA, USA (February 2002)
8. Klavins, E.: Programmable self-assembly. Control Systems Magazine 24(4), 43–56 (2007)
9. McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., Schmidt, B.: Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In: Proceedings of the AAAI Spring Symposium, Stanford, CA, USA (March 2006)
10. Shen, W.M., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., Venkatesh, J.: Multimode locomotion for reconfigurable robots. Autonomous Robots 20(2), 165–177 (2006)
11. Werfel, J., Bar-Yam, Y., Rus, D., Nagpal, R.: Distributed construction by mobile robots with enhanced building blocks. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2006), Orlando, FL, USA (May 2006)
12. Winfree, E.: Simulations of computing by self-assembly of DNA. Technical Report CS-TR:1998:22, California Institute of Technology, Pasadena, CA, USA (1998)
13. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology, Pasadena, CA, USA (June 1998)
14. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares. In: Proceedings of STOC 2000, Portland, OR, USA, pp. 459–468 (May 2000)
15. Wang, H.: Proving theorems by pattern recognition. II. Bell System Technical J. 40, 1–42 (1961)
16. Cook, M., Rothemund, P.W.K., Winfree, E.: Self-assembled circuit patterns. In: Chen, J., Reif, J.H. (eds.) DNA 2003. LNCS, vol. 2943, pp. 91–107. Springer, Heidelberg (2004)
17. Reishus, D.: Design of a self-assembled memory circuit. In: Proceedings of the 5th Foundations of Nanoscience: Self-Assembled Architectures and Devices (FNANO 2008), Snowbird, UT, USA, pp. 239–246 (April 2008)

18. Yan, H., Park, S.H., Finkelstein, G., Reif, J.H., LaBean, T.H.: DNA-templated self-assembly of protein arrays and highly conductive nanowires. Science 301, 1882–1884 (2003)
19. Adleman, L., Cheng, Q., Goel, A., Huang, M.D., Kempe, D., Moisset de Espanés, P., Rothemund, P.W.K.: Combinatorial optimization problems in self-assembly. In: Proceedings of STOC 2002, Montreal, Quebec, Canada, pp. 23–32 (May 2002)
20. Adleman, L., Goel, A., Huang, M.D., Moisset de Espanés, P.: Running time and program size for self-assembled squares. In: Proceedings of STOC 2002, Montreal, Quebec, Canada, pp. 740–748 (May 2002)
21. Moisset de Espanés, P., Goel, A.: Toward minimum size self-assembled counters. Natural Computing 7(3), 317–334 (2008)
22. Chen, H.L.: Towards minimum tile self-assembled counters. In: Proceedings of the 5th Foundations of Nanoscience: Self-Assembled Architectures and Devices (FNANO 2008), Snowbird, UT, USA, pp. 218–223 (April 2008)
23. Etzion-Petruschka, Y., Harel, D., Myers, D.: On the solvability of domino snake problems. Theoretical Computer Science 131(2), 243–269 (1994)
24. Aggarwal, G., Cheng, Q., Goldwasser, M.H., Kao, M.Y., Moisset de Espanés, P., Schweller, R.T.: Complexities for generalized models of self-assembly. SIAM J. on Computing 34(6), 1493–1515 (2005)
25. Kao, M.Y., Schweller, R.: Reducing tile complexity for self-assembly through temperature programming. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006), Miami, FL, USA, pp. 571–580 (January 2006)
26. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM J. on Computing 36(6), 1544–1569 (2007)
27. Brun, Y., Reishus, D.: Path finding in the tile assembly model. Theoretical Computer Science (in press, 2008)
28. Barish, R., Rothemund, P.W.K., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano Letters 5(12), 2586–2592 (2005)
29. Chelyapov, N., Brun, Y., Gopalkrishnan, M., Reishus, D., Shaw, B., Adleman, L.: DNA triangles and self-assembled hexagonal tilings. JACS 126(43), 13924–13925 (2004)
30. Fu, T.J., Seeman, N.C.: DNA double-crossover molecules. Biochemistry 32(13), 3211–3220 (1993)
31. Reishus, D., Shaw, B., Brun, Y., Chelyapov, N., Adleman, L.: Self-assembly of DNA double-double crossover complexes into high-density, doubly connected, planar structures. JACS 127(50), 17590–17591 (2005)
32. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biology 2(12), 424 (2004)
33. Winfree, E., Bekbolatov, R.: Proofreading tile sets: Error correction for algorithmic self-assembly. In: Proceedings of FOCS 2002, Madison, WI, USA, vol. 2943, pp. 126–144 (June 2003)