Research online Publishing www.ronpub.com

Open Access

Open Journal of Internet of Things (OJIOT)
Volume 4, Issue 1, 2018

http://www.ronpub.com/ojiot
ISSN 2364-7108

Techniques for the Generation of Arbitrary Three-Dimensional Shapes in Tile-Based Self-Assembly Systems

Florian-Lennert Lau, Kristof Stahl, Stefan Fischer

Institute of Telematics, University of Lübeck, Ratzeburger Allee 160, 23562, Lübeck, Germany,
{lau,stahl,fischer}@itm.uni-luebeck.de

Abstract

A big challenge in nanorobotics is the construction of nanoscale objects. DNA is a bio-compatible tool to reliably and constructively create objects at the nanoscale. A possible tool to build nano-sized structures are tile-based self-assembly systems on the basis of DNA. It is challenging and time-consuming to efficiently design blueprints for the desired objects. This paper presents basic algorithms for the creation of tilesets for $n \times n \times n$-cubes in the aTAM model. Only few publications focus on three-dimensional DNA crystals. Three-dimensional shapes are likely to be of more use in nanorobotics. We present three variations: hollow cubes, cube-grids and filled cubes. The paper also presents a basic algorithm to create arbitrary, finite, connected, three-dimensional and predefined shapes at temperature 1, as well as ideas for more efficient algorithms. Among those are algorithms for spheres, ellipsoids, red blood cells and other promising designs. The algorithms and tilesets are tested/verified using a software that has been developed for the purpose of verifying three-dimensional sets of tiletypes and was influenced by the tool ISU TAS. Others can use the simulator and the algorithms to quickly create sets of tiletypes for their desired nanostructures. A long learning process may thus be omitted.

Type of Paper and Keywords

Regular research paper: *tile-based self-assembly, computational complexity, aTAM, nanodevices*

1 Introduction to Nanorobotics

Sicknesses and injuries are among the oldest human ailments. Many generations of alchemists and scientists tried to find solutions to these problems. Nanotechnologies and especially nanomedicine is the next logical step in the long lasting search for a general cure.

In nanomedicine, nanorobots are often proposed for the treatment of diseases in countless scenarios. Some concepts for nanorobots are based on DNA as a building block [22]. In 2005 and 2006 Rothemund [20] and Seeman [23] presented several nanoscale two-dimensional objects that were assembled out of DNA. Since then, Akyldiz [1] proposed nanonetworks as a design principle for nanoscale computing/robotics. However, the actual implementation of nanorobots still remains unclear to this day.

In [14], a 3D DNA-nanomachine has been realized and tested in vivo. One of the basic components for the self-assembly of DNA-crystals are DX-molecules (see Figure 1). These molecules can be used for DNA computing because their open ends can interact with those of other molecules. Other, three-dimensional building blocks have also been

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2018)* in conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil. The proceedings of VLIoT@VLDB 2018 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

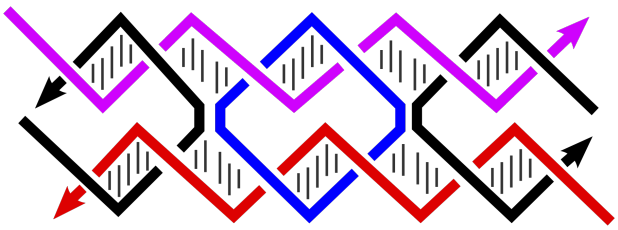


Figure 1: The basic two-dimensional molecule that enables the implementation of tile-based self-assembly systems [6]

realized. Few publications focus on three-dimensional DNA crystals [28, 10, 9]. Since three-dimensional structures have already been realized and due to a variety of applications, we assume that the discipline will progress in the near future.

We focus on an alternative, DNA tile-based system, while [28, 10] use a non-hierarchical approach. Additionally, the growth of their structures in the third dimension is limited. 3D tile-based self-assembly systems have no such limitations once the appropriate building blocks are available.

To tackle the problem of creating nanorobots, we present an approach for the algorithmic construction of simple, three-dimensional objects based on tiles. Since there are no available tools to manipulate matter at the atomic scale, it appears advantageous to use self-organization for the assembly process. Our solution is based on the abstract Tile Assembly Model (aTAM) from Eric Winfree [27].

The resulting DNA-based structures may be used to shelter medical payload [14] (with DNA "keys" to open the lid of a nanobox, e.g.) or serve as a “casing” to connect components like batteries and circuits. They might also be used to transport medical payload to a target location. Many other practical scenarios are possible.

The rest of the paper is structured as follows. Section 2 explains the mathematical preliminaries for Tile Assembly Systems. Section 3 shows an algorithm for different types and sizes of cubes. Section 4 generalizes the basic algorithm to arbitrary three-dimensional shapes. Section 5 analyzes the algorithmic complexity of the algorithms and simulation tools and Section 6 summarizes the paper and explains open problems.

2 Modelling Preliminaries

Since there are numerous definitions for different nanodevices, we introduce a formal definition for the constructs the presented algorithms yield. In new areas like nanonetworking, words like “nanobot”, “nanorobot”,‘ ‘nanomachine” and “nanodevice” are often used synonymously. Every person has a different concept for the respective words. We also briefly define the required

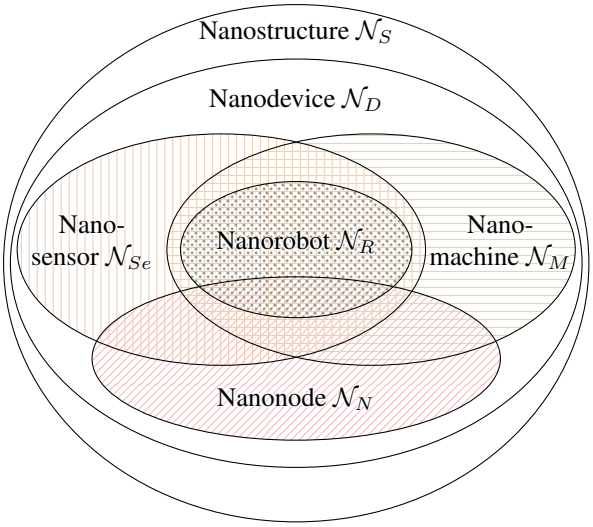


Figure 2: An overview of various nanostructures and their relationships

terminology to understand Tile Assembly Systems.

In [5] Büther et al. introduced a hierarchy for various nanodevices. Figure 2 depicts all formally defined nanostructures.

The newly introduced term is *nanostructures*, denoting the most primitive type of nanoscale constructs. Nanostructures require no functional component other than a framework that constitutes its physical appearance. They may be described as a simple three-dimensional shape. Formally:

Definition 2.1. A *nanostructure* $\mathcal{N}_S = K_{\text{opt}}$ is a nanoscale, artificial construct, designed to fulfill a predefined function in an environment Γ. A nanostructure consists of 0 or more optional components $K_{\text{opt}} \subseteq \{A, C, I, L, M, P, S, T\}$.

For the sake of this paper, we focus on nanostructures in their most basic form, i.e., without any optional components. The possible components are *Actuation* (A), *Computation* (C), *Information processing* (I), *Locomotion* (L), *Memory* (M), *Power Supply* (P), *Sensor* (S) and *Time* (T). The Environment Γ may be the human body. The function of nanostructures might be the connection of the aforementioned components.

In contrast, *nanodevices* require at least a power supply P. *Nanomachines* additionally have an actuatory A component to manipulate the environment. *Nanosensors* require a sensory component S instead of an actuator. *Nanorobots* are the most specific kind of nanomachine. They require all previously mentioned components and locomotion L, as well as information processing I and memory M. *Nanonodes* require at least a power supply P and the ability to communicate C.

The basic components for self-assembly systems are *tiles*

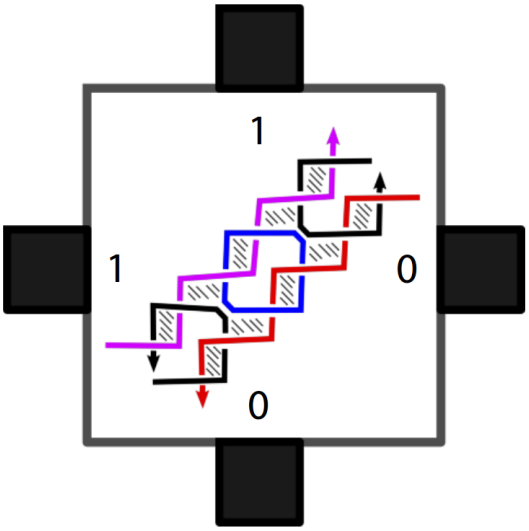
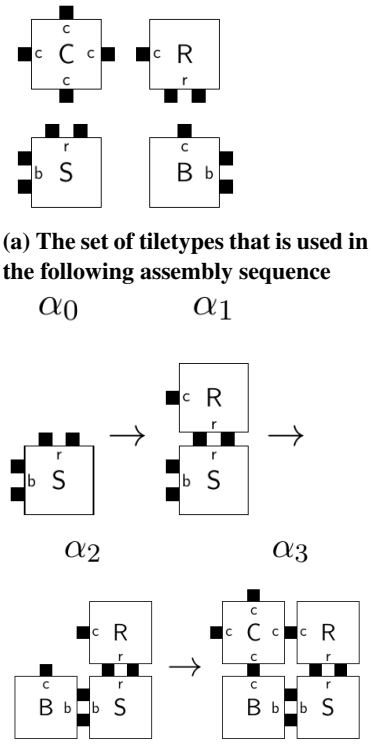


Figure 3: An overlay of the DX-molecule and a mathematical tile

[19]. The idea was originally introduced by Eric Winfree and motivated by Wang [27, 25]. Since the available space in this paper is very limited, we just give a very brief introduction and refer to [19, 13] for a more detailed description of DNA-based tile assembly. We use the notation of [11]. Tiles are mathematical objects that can be implemented with DNA-molecules. A tile may "fill" a cube in the three-dimensional space. Tiles may have one or more *glues* of a certain color attached to each side. Figure 3 explains how the DX-molecule relates to abstract tiles. The sticky ends of the DNA constitute the glues and their colors.

Only glues with the same color may form a stable binding of two tiles. For the sake of modeling, we assume that there is an unlimited supply of every *tiletype* (this is not true for the constituting DNA molecules, but due to the limited size of our desired constructs, it will not be a restriction). Tiles start to randomly form bindings once a *seed tile* is introduced to the medium. The process is illustrated in Figure 4. Figure 4a shows an example set of tiletypes. The self-assembly process starts with a seed-tile in step α_0. In every following step, exactly one tile is added to the assembly. As depicted, the glue strength (the number of black blocks next to the tiles) and the glue labels (the character next to the black blocks) match. The label in the middle can be used for computation.

A *Tile Assembly System* (TAS) has a *temperature* τ. The temperature describes how many glues are required to stably attach two tiles to each other. A stable composite of tiles is called an *assembly*. In this paper, we focus on temperature 1 systems. Most TAS only use two dimensions for simulation. Two-dimensional systems are already capable of universal computation at temperature 2 [4]. Since we are only



(a) The set of tiletypes that is used in the following assembly sequence

Figure 4: An exemplary assembly sequence of four tiles

interested in the geometrical aspect of tile assemblies, we conclude that three-dimensional tiles are of interest and focus on these in all simulations. We already introduced a related approach in [13].

We determined a set of geometrical shapes which are of special interest. Cubes, spheres, ellipsoids, tubes, rectangles and biconcave, flat ellipsoids (erythrocytes) appear useful for in-body application. In the following, we derive algorithms that create sets of tiletypes for the mentioned structures of adjustable sizes.

3 An Algorithm for $n \times n \times n$-Cubes

In this section we present an algorithm that generates sets of tiletypes for different variations of cubes of arbitrary size automatically. The algorithm takes a cube size as an input and outputs a set of tiletypes that assembles the desired shape.

Medical scenarios are among the most prominent areas of application for Nanorobots (e.g. [7, 8, 24]). Experimental research from [2, 15] generated a cube with a controllable lid. It may be used to shelter medical payload from the chemical environment inside the target region. With this level of control, it might be possible to locally apply medication. Through targeted application, the required dosage of medicine may be drastically reduced. The

side-effects of medication might also be reduced, since fewer areas of the body are exposed to the substance. In [12], we analyzed numerous medical scenarios. Since medical payload comes in many shapes, vessels of different, adjustable sizes are of interest. Cubes are especially useful, since they are symmetrical and thus efficient to build.

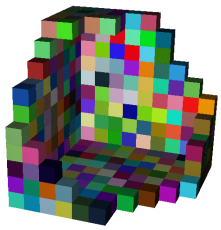
Cubes have already been assembled in DNA-based tile assembly systems [2, 3, 15]. Most of these structures have been designed manually in a complicated process. We present an algorithm that automatically creates a set of tiletypes for either hollow, framed or filled cubes of adjustable size. In the then following engineering step, at least one tile of each of these tiletypes has to be created. Adding the tiles to a shared medium then leads to the automatic self-assembly of the desired structure. The process is hierarchical. In a first step, the components are created, in a second step they are introduced to a medium and self-assemble into the desired structures.

The basic idea behind the algorithm is the creation of a *distinct* tiletype for every position in the defined cube. This procedure allows for fewer errors, since only one possible tile exists for every position in an assembly, i.e., every tile does only fit in exactly one position. As stated above, the number of possible different tiletypes based on DNA molecules is comparably large, thus this property of the algorithm is no restriction for us, since we are only interested in very small constructs.

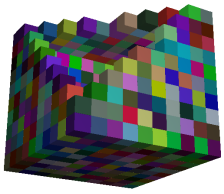
Algorithm 1 displays the computational steps for creating cube tiletypes in pseudocode. In three nested loops, we consider every possible position of a cube. Lines 9 to 11 store in parameter c whether the current position is an inner position ($c = 0$), a wall position ($c = 1$), an edge position ($c = 2$) or a cornerstone position ($c = 3$). The parameter p then controls which position requires a tile. For a parameter $p = 0$ Algorithm1 creates a solid $n \times n \times n$ cube, since every position is filled. Varying the parameter reduces the filling. For $p = 1$ the cube is hollow and for $p = 2$ the cube is just a frame, without even the side walls (only edge and cornerstone tiles).

For every position at which a tile should be placed, we add another tiletype to the output (Line 13). The resulting set of tiletypes may then be simulated in an aTAM simulator which supports three-dimensional tiles. Figure 5 shows a simulation for the set of tiletypes for a $10 \times 10 \times 10$ hollow cube in three different stages and thus depict the assembly progress. The different (randomly assigned) colors visualize the different tiletypes. The glues themselves were omitted, since they are unique for every position. Every tile has a maximum of one glue on each side.

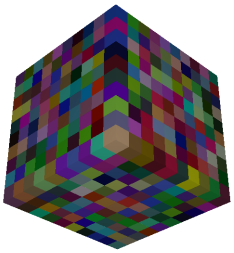
Cubes are limited structures which do not—or only very rarely—naturally appear inside of the human body. While they are the simplest 3D structure most can imagine, they lack qualities required for efficient transportation inside the human bloodstream. Even the intercellular space is a unique environment with challenges a cube is probably not optimal for. Therefore, we propose different structures which are probably better applicable in the following section.



(a) Start building a $10 \times 10 \times 10$ hollow cube



(b) The $10 \times 10 \times 10$ cube with a corner still missing



(c) The fully assembled $10 \times 10 \times 10$ hollow cube

Figure 5: A hollow cube with an edge length of ten is assembled in the aTAM simulation. The simulation is not interactive so three instances were run with the same algorithm. Different coloring is due to a random assignment of colors to tiletypes when displaying.

4 An Algorithm for Arbitrary Predefined Three-Dimensional Shapes

For the creation of arbitrary three-dimensional shapes, we generalized the idea from Algorithm 1. The resulting algorithm can create sets of tiletypes for any given shape. It is also designed for temperature 1 systems. Every tiletype is used exactly once. Since more efficient algorithms exploit symmetries in geometrical figures, a generic (shape-agnostic) algorithm *cannot* be improved to require less tiletypes, in general. This approach always works, but requires the maximum possible amount of tiletypes. We chose temperature 1 systems for the sake of simplicity. They

require few additional tiles that are not part of the main assembly. E.g., efficiently assembled squares are based on a binary counter, limiting their growth.

Due to biological limitations with DNA, we consider the number of possible glues in a TAS limited. Once the DNA strands that constitute the glues are too long, they become more prone to binding with wrong counterparts. We therefore assume that only a constant number k of glues and thereby a constant number of tiles is reasonably implementable.

To assemble big shapes nevertheless, we could use efficient sets of tiletypes that can, e.g., assemble squares with less than $O(n)$ tiletypes. Very good algorithms may be able to generate cubes with only $O(n \log(n))$ tiletypes required, or even less [21]. The algorithm for cubes is special, since it offers different variations of the same shape. In general, not every geometrical figure has a "frame" or a "core" that can be displayed separately. We therefore omitted any specific component for the generalized version of the basic general algorithm.

Algorithm 2 takes a finite subset of points from the $\mathbb{R}^3$ as input. The input structure has to be fully connected. Every cube in the structure has to have at least one direct neighbor in the cardinal directions, as well as above or below. The algorithm then outputs a set of tiletypes for the desired shape. The connectedness-constraint is due to limitations in the aTAM definition.

Figure 6 displays an example application of the generalized algorithm on a voxel-based sphere approximation of radius four. The sphere itself is hollow, as Figure 6b shows. Spheres are naturally occurring structures that might be better suited for moving fluid environments than cubes. They equally distribute pressure from the outside along the whole surface of the shape. The other desired shapes—like ellipsoids or rectangles—may also be assembled as voxel-approximations like the sphere.

We believe that temperature 1 systems suffice for many practical cases and proofs of concept. The idea is to build "nanoscale" constructs. Very efficiently assembled cubes might very well be of micrometers in size. More efficient algorithms are the subject of future work. The algorithm which generates the sets of tiletypes is rather efficient due to its simplicity. The simulator which verifies the output structures on the other hand tends to be slower. Therefore, we analyze the runtime and correctness of the developed software and algorithms.

5 Evaluation of Simulator and Algorithms

The software we developed consists of three major parts, namely (i) a generator implementing algorithms for the synthesis of sets of tiletypes, (ii) the simulation software which verifies the algorithms, and (iii) a component

Algorithm 1: Algorithm to create a set of tiletypes that generates a cube of a given edge length

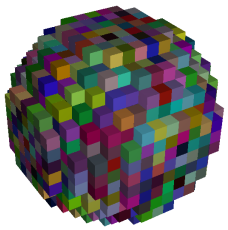
Data: Edge length n, some parameter p
Result: Set of tiletypes T

```
1  x := 0
2  y := 0
3  z := 0
4  T := ∅
5  while x < n do
6      while y < n do
7          while z < n do
8              c := 0
9              for i ∈ {x, y, z} do
10                 if i = 0 ∨ i = n − 1 then
11                     c := c + 1
12             if c ≥ p then
13                 T := T ∪
                     {createPositionTiletype(x, y, z, 1)}
14             z := z + 1
15         y := y + 1
16     x := x + 1
```

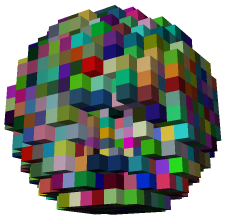
Algorithm 2: Algorithm to create a set of tiletypes which generates a given shape in a Tile Assembly Model

Data: A boolean array *shape* with three dimensions of max lengths n, m, o
Result: Set of tiletypes T

```
1  x := 0
2  y := 0
3  z := 0
4  T := ∅
5  while x < n do
6      while y < m do
7          while z < o do
8              if shape_{x,y,z} then
9                  T := T ∪
                     {createPositionTiletype(x, y, z, 1)}
10             z := z + 1
11         y := y + 1
12     x := x + 1
```



(a) A full hollow sphere produced by the methods for generating arbitrary coherent shapes in the simulator



(b) The hollow sphere from a different angle shown with a hole to verify the Assembly

Figure 6: Simulation of the TAS introduced by Algorithm 2. This creates a sphere with radius eight.

which generates a three-dimensional representation of the assembled structure. We analyze the three parts separately in the following paragraph. We analyze both correctness and runtime.

- The tiletype generator for the 3D aTAM is based on the ISU TAS software [17]. The software implements the mathematical model of Winfree's aTAM rigorously, thus it is working as intended. The previously two-dimensional mathematical objects have been generalized to three dimensions, while keeping the desired behavior intact. To verify the functionality, we tested the software with various sets of tiletypes that have already been evaluated in other papers. Two examples are binary counters and a version of the Sierpinsky Triangle [18]. The sets of tiletypes were modified to fit the 3D context.

  Figure 7 shows the result after several hundred iterations. Only two dimensions were used for the assembly process. The functionality in three dimensions has been verified with algorithms for cubes as displayed in Figure 5.

  The synthesis of the set of tiletypes is dependent on the underlying function used to determine the used positions for the resulting shape. To create each tiletype, only $O(1)$ calculations occur. When using a naive approach to determine the positions that are occupied by tiles, $O(n^3)$ calculations are necessary. The input size n is the biggest size in all of the three dimensions of the given shape. The resulting runtime of the tiletype generation is thus $O(n^3)$. To further improve the runtime, the possible next tiles for every position of the grid may be stored in a d-tree.

- The simulation software is slower than the tileset generation[1]. In every step of the assembly process, one tile is added to the assembly. Since we are only interested in finite structures, the number of required steps is also denoted by n. The possible positions where a tile may be added are kept in a list. The runtime thus depends on the size of the list of possible positions or *frontier* and the number of steps and generally the number of possible tiles for a position, which is always one in our case. All presented cases are limited by n. The resulting runtime is limited by $O(n^3)$.

- The component for the graphical representation requires $O(n)$ steps.

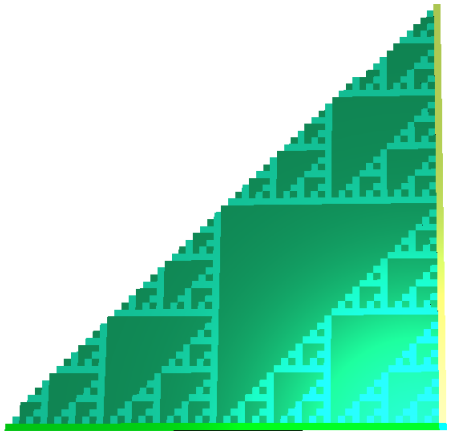


Figure 7: Proof of concept assembly of the Sierpinsky Triangle

Since all three steps are independent of each other, the total runtime is limited by $O(n^3)$ and the software works as intended.

Figure 8 shows the required time for the synthesis of the set of tiletypes for cubes of various sizes[2]. The JVM induces overhead due to garbage collection and allocation when the problem instance increases in size. As an example, the runtime values jump between 2.5 seconds and 8 seconds when the cubes edge length grows larger than 400. Additionally, the PC uses a variable CPU clock and is able to increase the calculation frequency if more calculations are required to save energy when running idle.

[1] The tileset generation's n represents the biggest length in the x, y or Z dimension, while the simulations' n represents the number of tiles in a nanostructure.

[2] The JVM used was OpenJDK version 1.8.0_144 64-Bit Server VM (build 25.144-b01, mixed mode).

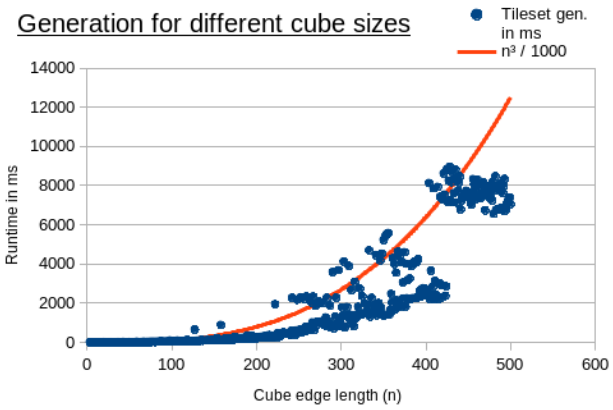


Figure 8: Assembly runtime of cubes of different sizes with respect to the *edge length*

6 Conclusion

The algorithms and the simulator itself have been tested. Both represent first sound steps in the process of automatically designing architectures for nanostructures or more specific nanoscale machines/robots. Other algorithms for better geometrical objects will be introduced in the future and tested in both the aTAM simulator and a more realistic version (the kTAM [27]) which is still in development.

Algorithm 2 is a generalized procedure to generate any connected shape in form of a set of tiletypes. Algorithm 1 exploits symmetries in shapes like squares or cubes to generate variations of the basic structure of arbitrary size. Algorithms for specific shapes may be more efficient than general versions.

The simulation software implements Winfree's aTAM model and is capable of verifying the assembly process of sets of tiletypes. Unlike Xgrow of ISU-TAS [26, 16] the implemented software is capable of simulating growth in three dimensions.

Possible future research focuses on the following aspects:

- Generating efficient algorithms for more complex structures, like red blood cell-like shapes or simple spheres/cubes.
- Increasing the efficiency of algorithms for cubes and rectangles by applying known results from two-dimensional structures.
- Implementing another simulation tool capable of both respecting errors with DNA implementations and using three-dimensional tiles.
- Implementing error-correction measures for arbitrary shapes in the kTAM model.
- Developing a modular principle to generate complex architectures for nanostructures from basic components.
- Defining assembly sequences that prohibit growth of "outer" structures before the "inner" structures are finished.

The often complicated process of designing the tilesets of three-dimensional shapes of specific sizes may now be partly automated. The user simply has to supply a binary description of the target shape and receives the desired output. Only the design of the actual DNA implementation is still necessary.

References

[1] I. F. Akyildiz, F. Brunetti, and C. Blázquez, "Nanonetworks: A new communication paradigm," *Computer Networks*, vol. 52, no. 12, pp. 2260–2279, 2008.

[2] E. S. Andersen, M. Dong, M. M. Nielsen, K. Jahn, R. Subramani, W. Mamdouh, M. M. Golas, B. Sander, H. Stark, C. L. P. Oliveira, J. S. Pedersen, V. Birkedal, F. Besenbacher, K. V. Gothelf, and J. Kjems, "Self-assembly of a nanoscale DNA box with a controllable lid," *Nature*, vol. 459, no. 7243, pp. 73–76, 2009.

[3] F. Becker, E. Remila, and N. Schabanel, "Time optimal self-assembly for 2d and 3d shapes: The case of squares and cubes," in *DNA Computing: 14th International Meeting on DNA Computing, Prague, Czech Republic, June 2-9*. Springer Berlin Heidelberg, 2009, pp. 144–155.

[4] R. Berger, *The undecidability of the domino problem*, ser. Memoirs ; No 1/66. American Mathematical Society, 1966.

[5] F. Büther, F.-L. Lau, M. Stelzner, and S. Ebers, "A formal definition for nanorobots and nanonetworks," in *17th International Conference on Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, O. Galinina, S. Andreev, S. Balandin, and Y. Koucheryavy, Eds., St. Petersburg, Russia, August 28–30, 2017, pp. 214–226. [Online]. Available: https://doi.org/10.1007/978-3-319-67380-6_20

[6] M. Chengde, "Basic DNA structures for self-assembly (c) a DNA DX," 2004. [Online]. Available: https://en.wikipedia.org/wiki/File:Mao-DX-schematic-2.svg

[7] D. C. Ferreira, L. P. Reis, and N. V. Lopes, "A nanocommunication system for endocrine diseases," *Cluster Computing*, pp. 1–18, 2017.

[8] R. A. Freitas, "Current status of nanomedicine and medical nanorobotics," *Journal of Computational and Theoretical Nanoscience*, vol. 2, no. 1, pp. 1–25, 2005.

[9] Y. Ke, L. L. Ong, W. M. Shih, and P. Yin, "Three-dimensional structures self-assembled from dna bricks," *science*, vol. 338, no. 6111, pp. 1177–1183, 2012.

[10] Y. Ke, L. L. Ong, W. Sun, J. Song, M. Dong, W. M. Shih, and P. Yin, "Dna brick crystals with prescribed depths," *Nature chemistry*, vol. 6, no. 11, p. 994, 2014.

[11] J. I. Lathrop, J. H. Lutz, and S. M. Summers, "Strict self-assembly of discrete sierpinski triangles," in *Computation and Logic in the Real World: Third Conference on Computability in Europe*, S. B. Cooper, B. Löwe, and A. Sorbi, Eds., Siena, Italy, June 18-23, 2007, pp. 455–464.

[12] F. Lau, F. Büther, and B. Gerlach, "Computational requirements for Nano-Machines: there is limited space at the bottom," in *4th ACM International Conference on Nanoscale Computing and Communication*, Washington DC, USA, Aug 2017.

[13] F. Lau and S. Fischer, "Embedding space-constrained quantum-dot cellular automata in three-dimensional tile-based self-assembly systems," in *4th ACM International Conference on Nanoscale Computing and Communication (ACM NanoCom'17)*. Washington DC, USA: ACM, Aug 2017.

[14] S. Li, Q. Jiang, S. Liu, Y. Zhang, Y. Tian, C. Song, J. Wang, Y. Zou, G. Anderson, J. Han, Y. Chang, Y. Liu, C. Zhang, L. Chen, G. Zhou, G. Nie, H. Yan, B. Ding, and Y. Zhao, "A dna nanorobot functions as a cancer therapeutic in response to a molecular trigger in vivo." *Nature Biotechnology*, vol. 36, no. 5, pp. 258–264, 2018.

[15] K. Ming-Yang and V. Ramachandran, "DNA self-assembly for constructing 3D boxes," in *Algorithms and Computation: 12th International Symposium*, P. Eades and T. Takaoka, Eds., Christchurch, New Zealand, December 19–21, 2001, pp. 429–441.

[16] M. J. Patitz, "Isu tas," accessed 3. March 2018 URL: http://self-assembly.net/wiki/index.php?title=ISU_TAS.

[17] M. J. Patitz, "Simulation of self-assembly in the abstract tile assembly model with ISU TAS," *CoRR*, vol. abs/1101.5151, 2011.

[18] M. J. Patitz, "An introduction to tile-based self-assembly and a survey of recent results," *Natural Computing*, vol. 13, no. 2, pp. 195–224, 2014.

[19] M. J. Patitz, "An introduction to tile-based self-assembly," in *Proceedings of the 11th International Conference on Unconventional Computation and Natural Computation*, J. Durand-Lose and N. Jonoska, Eds., Orléan, France, September 3-7, 2012, pp. 34–62.

[20] P. W. K. Rothemund, "Folding dna to create nanoscale shapes and patterns," *Nature*, vol. 440, no. 7082, pp. 297–302, Mar 2006.

[21] P. W. K. Rothemund and E. Winfree, "The program-size complexity of self-assembled squares (extended abstract)," in *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*. ACM, 2000, pp. 459–468.

[22] N. C. Seeman, "Nucleic acid junctions and lattices," *Journal of Theoretical Biology*, vol. 99, no. 2, pp. 237 – 247, 1982.

[23] N. C. Seeman, "Structural DNA nanotechnology: An overview," *Methods Mol Biol*, vol. 303, pp. 143–166, 2005.

[24] M. Stelzner, F.-L. Lau, K. Freundt, F. Büther, M. L. Nguyen, C. Stamme, and S. Ebers, "Precise detection and treatment of human diseases based on nano networking," in *11th International Conference on Body Area Networks*, Turin, Italy, December 2016.

[25] H. Wang, "Dominoes and the aea case of the decision problem," in *Computation, Logic, Philosophy: A Collection of Essays*. Springer Netherlands, 1990, pp. 218–245.

[26] E. Winfree, R. Schulman, and C. Evans, "Xgrow," accessed 3. March 2018 URL: http://www.dna.caltech.edu/Xgrow/.

[27] E. Winfree, "Simulations of computing by self-assembly," University of Pennsylvania, 1998.

[28] J. Zheng, J. J. Birktoft, Y. Chen, T. Wang, R. Sha, P. E. Constantinou, S. L. Ginell, C. Mao, and N. C. Seeman, "From molecular to macroscopic via the rational design of a self-assembled 3d dna crystal," *Nature*, vol. 461, no. 7260, pp. 74–77, Sep 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2764300/

AUTHOR BIOGRAPHIES

Florian Lau was born March 21th, 1990 in Lübeck, Germany. He received his B.Sc in computer science in 2013 and his M.Sc. in 2015 respectively. He is currently a phd student at the university of Lübeck. His work is focused on DNA-based self-assembly systems.

Kristof Stahl was born Feburary 24th, 1990 in Pinneberg, Germany. He received his B.Sc in computer science in 2015 and his M.Sc. in 2018 respectively at the University of Lübeck. He is currently working for Ferchau Engineering doing analysis, specification, implementation, documentation and testing in hard-/software development.

Stefan Fischer was born June 13th, 1967. He received his Diploma in Information Systems in 1992 at the University of Mannheim. He received his doctoral degree in Computer Science in 1996 supervised by Prof. Dr. Wolfgang Effelsberg at University of Mannheim. He was Assistant Professor for Information Technology from 1998 to 2001,International University in Germany, Bruchsal. From Sept. 2001 to Oct. 2004 he was Associate Professor for Computer Science at TU Braunschweig and since Nov. 2004: Full Professor at University of Lübeck, Germany