Title:	Active self-assembly and molecular robotics with the nubot model
Name:	Damien Woods
Affil./Addr.	California Institute of Technology
Keywords:	molecular robotics; self-assembly; rigid-body motion
SumOriWork:	2013; Woods, Chen, Goodfriend, Dabby, Winfree, Yin 2013; Chen, Xin, Woods 2014; Chen, Doty, Holden, Thachuk, Woods, Yang

Active self-assembly and molecular robotics with the nubot model

Damien Woods

California Institute of Technology

Years aud Authors of Summarized Original Work

2013; Woods, Chen, Goodfriend, Dabby, Winfree, Yin
2013; Chen, Xin, Woods
2014; Chen, Doty, Holden, Thachuk, Woods, Yang

Keywords

molecular robotics; self-assembly; rigid-body motion

Problem Definition

In the theory of molecular-scale self-assembly, large numbers of simple interacting components are designed to come together to build complicated shapes and patterns. Many models of self-assembly, such as the abstract Tile Assembly Model [6], are cellular automata-like crystal growth models. Indeed such models have given rise to a rich theory of self-assembly as described elsewhere in this encyclopedia. In biological organisms we frequently see much more sophisticated growth processes, where self-assembly is combined with active molecular components that change internal state and use molecular motors that have the ability to push and pull large structures around. Molecular engineers are now beginning to design and build molecular-scale DNA motors and active self-assembly systems [2]. We wish to understand, at a high level of abstraction, the ultimate computational capabilities and limitations of such molecular-scale rearrangement and growth. The nubot model, put forward in [8], is akin to an asynchronous nondeterministic cellular automaton augmented with non-local rigid-body movement. Unit-sized monomers are placed on a 2D triangular grid. Monomers undergo state changes, appear and disappear using local rules, as shown in Figure 1. However, there is also a non-local aspect to the model: rigid-body movement that comes in two forms, movement rules and random agitations.

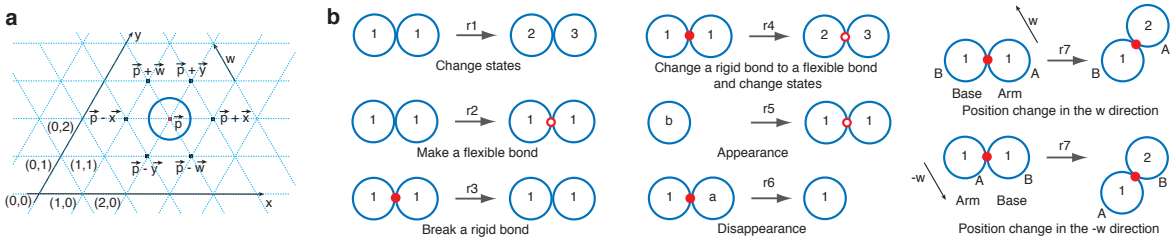

Fig. 1. Overview of the nubot model. (a) A nubot configuration showing a single nubot monomer on the triangular grid. (b) Examples of nubot monomer rules. Rules r1-r6 are local cellular automaton-like rules, whereas r7 effects a non-local movement that may translate other monomers as shown in Figure 2. Monomers continuously undergo agitation, as shown in Figure 3. A flexible bond is depicted as an empty red circle and a rigid bond is depicted as a solid red disk. From [8].

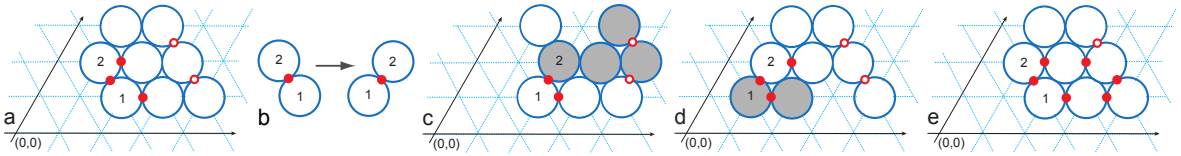

Fig. 2. Movement rule. (a) Initial configuration. (b) Movement rule with one of two results depending on the choice of arm or base. (c) Result if the monomer with state 2 is the arm, or (d) monomer with state 1 is the arm. The shaded monomers are the moveable set. The affect on rigid (filled red disks), flexible (hollow red circles) and null bonds is shown. (e) A configuration for which the movement rule is blocked: movement of 1 or 2 would force the other to move hence the rule is not applicable. From [4].

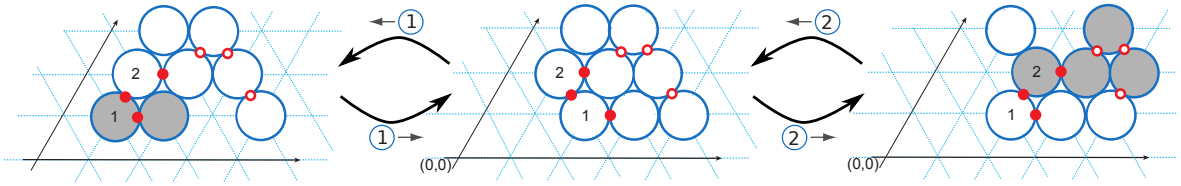

Fig. 3. Example agitations. Starting from the centre configuration, there are 48 possible agitations (8 monomers, 6 directions each) each with equal probability. The right configuration is the result of agitation of the monomer in state 2 in direction →, the left is the result of the agitation of the monomer in state 1 in direction ←. The shaded monomers are the agitation set—monomers that are moved by the agitation. From [3].

A *movement rule* r, consisting of a pair of monomer states A, B, the bond between them and two unit vectors, is a programatic way to specify unit-distance translation of a set of monomers in one step. See Figure 2 for an example. If A and B are in a prescribed orientation, when a movement rule is applied one of them is nondeterministically chosen to move unit distance in a prescribed direction. The rule r is applied in a rigid-body fashion: roughly speaking, if A is to move right it pushes anything immediately to its right and pulls any monomers that are bound to its left which in turn push and pull other monomers, all in one step. The rule may not be applicable if it is blocked (i.e. if movement of A would force B to also move), which is analogous to the fact that a human arm can not push its own shoulder. The other, somewhat related, form of movement is called *agitation*: at every point in time, every monomer on the grid may move unit distance in any of the six directions, at unit rate for each (monomer, direction) pair. An agitating monomer will push or pull any monomers that it is adjacent to, in a way that preserves rigid-body structure and all in one step as shown in Figure 3. Unlike movement, agitations are never blocked. Rules are applied asynchronously and in parallel. Taking its time model from stochastic chemical kinetics, a nubot system evolves as a continuous time Markov process.

For intuition, we describe motion in terms of pushing and pulling. However movement and agitation are actually intended to model a nanoscale environment with diffusion, Brownian motion, convection, turbulent flow, cytoplasmic streaming and other uncontrolled inputs of energy that interact monomers in all directions, moving large molecular assemblies in a random fashion (i.e. agitation) and allowing motors to simply latch and unlatch large assemblies into position (i.e. the movement rule).

Key Results

Assembling simple structures, namely lines and squares, has proven to be a fruitful way to explore the power of the nubot model for a few reasons. Firstly, it helps us develop a number of techniques and intuitions for the model. Secondly, lines and squares get used again and again in more general results that show the full power of the model. Thirdly, the efficiency of assembling simple shapes has been a de facto benchmark problem for a number of self assembly models (although this benchmark often does not give the full story). In a variety of models, such as the abstract Tile Assembly Model, cellular automata and some robotics models, it takes time $\Omega(n)$ to assemble a length n line. In the nubot model this is achieved in merely $O(\log n)$ expected time and $O(\log n)$ states.

Theorem 1 ([8]). *For each $n \in \mathbb{N}$, there is a set of nubot rules $\mathcal{N}_n^{\text{line}}$ such that starting from a single monomer, $\mathcal{N}_n^{\text{line}}$ assembles a length n line in $O(\log n)$ expected time, $n \times 2$ space and $O(\log n)$ states.*

One can trade time for states by giving a slightly slower method with fewer states:

Theorem 2 ([4]). *There is a set of nubot rules $\mathcal{N}^{\text{line}}$ such that for each $n \in \mathbb{N}$, from a line of $O(\log n)$ "binary" monomers (each in state 0 or 1), $\mathcal{N}^{\text{line}}$ assembles a length n line in $O(\log^2 n)$ expected time, $n \times O(1)$ space and $O(1)$ states.*

An $n \times n$ square can be built by growing a horizontal line and then n vertical lines, showing that assembly of squares with nubots is exponentially faster than the $\Theta(n)$ expected time seen in the abstract Tile Assembly Model [1]:

Theorem 3 ([8]). *For each $n \in \mathbb{N}$, there is a set of nubot rules $\mathcal{N}_n^{\text{square}}$ such that starting from a single monomer, $\mathcal{N}_n^{\text{square}}$ assembles a $n \times n$ square in $O(\log n)$ expected time, $n \times n$ space and $O(\log n)$ states.*

The results above, and all of those in [8, 4], crucially make use of the rigid-body movement rule: the ability for a single monomer to control the movement of large objects quickly, and at a time and place of the programmer's choosing. However, in a molecular-scale environment, molecular motion is happening in a largely uncontrolled and fundamentally random manner, all of the time. The *agitation nubot* model does not have the movement rule, but instead permits such uncontrolled random agitation (movement). Although this form of movement is challenging to control in a precise manner, the following result shows we can use it to achieve sublinear expected time growth of a length n line in only $O(n)$ space:

Theorem 4 ([3]). *There is a set of nubot rules $\mathcal{N}_{\text{line}}$, such that for any $\epsilon > 0$, for sufficiently large $n \in \mathbb{N}$, starting from a line of $\lfloor \log_2 n \rfloor + 1$ monomers, each in state 0 or 1, $\mathcal{N}_{\text{line}}$ in the agitation nubot model assembles an $n \times 1$ line in $O(n^{1/3} \log n)$ expected time, $n \times 5$ space and $O(1)$ monomer states.*

For a square we can do much better, achieving polylogarithmic expected time:

Theorem 5 ([3]). *There is a set of nubot rules $\mathcal{N}_{\text{square}}$, such that $\forall n \in \mathbb{N}$, starting from a line of $\lfloor \log_2 n \rfloor + 1$ monomers, each in state 0 or 1, $\mathcal{N}_{\text{square}}$ in the agitation nubot model assembles an $n \times n$ square in $O(\log^2 n)$ expected time, $n \times n$ space and $O(1)$ monomer states.*

This section concludes with three results on general-purpose computation and shape construction with the nubot model. First we have a computability-theoretic result: any finite computable connected shape can be quickly self-assembled.

Theorem 6 ([8]). *An arbitrary connected computable 2D shape of size $\leq \sqrt{n} \times \sqrt{n}$ can be assembled in expected time $O(\log^2 n + t(|n|))$ using $O(s + \log n)$ states. Here, $t(|n|)$ is the time required for a program-size s Turing machine to compute, given a pixel index as a binary string of length $|n| = \lfloor \log_2 n \rfloor + 1$, whether or not the pixel is present in the shape.*

For complicated computable shapes the construction for Theorem 6 necessarily requires computation workspace outside of the shape's bounding box. The next result is of a more resource-bounded style and, roughly-speaking, states that 2D patterns with efficiently computable pixel colours can be assembled using nubots in merely polylogarithmic expected time, while staying inside the pattern's bounding box.

Theorem 7 ([8]). *An arbitrary finite computable 2D pattern of size $\leq n \times n$, where $n = 2^p, p \in \mathbb{N}$, with pixels whose colour is computable on a polynomial time $O(|n|^\ell)$ (inputs are binary strings of length $|n| = O(\log n)$), linear space $O(|n|)$, program-size s Turing machine, can be assembled in expected time $O(\log^{\ell+1} n)$, with $O(s + \log n)$ monomer states and without growing outside the pattern borders.*

The results cited so far can be used to compare the nubot model to other models of self-assembly, and tell us that nubots build shapes and patterns in a fast parallel manner. The next result quantifies this parallelism in terms of a well-known parallel model from computational complexity theory: NC is the class of problems solved by uniform polylogarithmic depth and polynomial size Boolean circuits.

Theorem 8 ([4]). *For each language $L \in$ NC, there is a set of nubots rules $\mathcal{N}_L$ that decides L in polylogarithmic expected time, constant number of monomer states, and polynomial space in the length of the input string of binary monomers (in state 0 or 1). The output is a single binary monomer.*

This result stands in contrast to sequential machines like Turing machines, that cannot read all of an n-bit input string in polylogarithmic time, and "somewhat parallel" models like cellular automata and the abstract Tile Assembly Model, that can not have all of n bits influence a single output bit decision in polylogarithmic time [5]. Thus, adding the nubot rigid-body movement primitive to an asynchronous non-deterministic cellular automaton drastically increases its parallel processing abilities.

Open Problems

Some future research directions are discussed here and in [8, 4, 3]. It remains as future work to look at other topics such as fault tolerance, self-healing, dynamical tasks, or systems that continuously respond to the environment.

The complexity of assembling lines. Theorem 1 states that a line can be grown in expected time $O(\log n)$, space $O(n) \times O(1)$ and $O(\log n)$ states, and Theorem 2 trades time for states to get expected time $O(\log^2 n)$, space $O(n) \times O(1)$ and $O(1)$ states.

What is the complexity (expected time × states) of assembling a line in the nubot model? Is it possible to meet the lower bound of expected time × states = $\Omega(\log n)$? In this problem, the input should be a set of monomers with space × states = $O(\log n)$.

Computational power. Theorem 8 gives a lower bound on the computational complexity of the nubot model. What is the exact power of polylogarithmic expected time nubots? The answer may differ on whether we begin from a small collection of monomers (as in Theorem 8) or a large prebuilt structure. One challenge, for the upper bound, involves finding better Turing machine space, or circuit depth, bounds on computing multiple applications of the movable set on a large nubots grid.

Synchronization and composition of nubot algorithms. Synchronization is a method to quickly send signals using non-local rigid-body motion [8, 4]. The nubot model is asynchronous, but synchronization can be used to set discrete stages, or checkpoints, during a complicated construction. This in turn facilitates composition of nubot algorithms (run algorithm 1, synchronize, run algorithm 2, synchronize, etc.) and many of the results cited here use it for exactly that reason. However, synchronization-less constructions often exhibit a kind of independence where growth proceeds everywhere in parallel, without waiting on signals from distant components. Such systems are highly distributed, easy to analyse and perhaps more amenable to laboratory implementation. Intuitively, this seems like the right way to program molecules. The proof of Theorem 7 does not use synchronization which shows that without it a very general class of (efficiently) computable patterns can be grown and indeed the proof gives methods to compose nubot algorithms without resorting to synchronization. It remains as future work to formalise both this notion of synchronization-less "independence" and what we mean by "composition" of nubot algorithms. What conditions are necessary and sufficient for composition of nubot algorithms? What classes of shapes and patterns can be assembled using without synchronization, or other forms of rapid long-range communication?

Agitation versus the movement rule. Is it possible to simulate the movement rule using agitation? More formally, is it the case that for each nubot program $\mathcal{N}$, there is an agitation nubot program $\mathcal{A}_{\mathcal{N}}$, that acts just like $\mathcal{N}$ but with some $m \times m$ scale-up in space, and a k factor slowdown in time, where m and k are (constants) independent of $\mathcal{N}$ and its input? As motivation, note that every self-assembled molecular-scale structure was made under conditions where random jiggling of monomers is a dominant source of movement! Our question asks if we can *programmably exploit* this random molecular motion to build structures quicker than without it.

Intrinsic universality and simulation. Is the nubot model intrinsically universal? Specifically, does there exist a set of monomer rules U, such that any nubot system $\mathcal{N}$ can be simulated by "seeding" U with a suitable initial configuration? Here the simulation should have a spatial scale factor m that is a function of the number of states in the simulated system $\mathcal{N}$. Is the agitation nubot model intrinsically universal? Our hope would be that simulation could be used to tease apart the power of different notions of movement (for example to understand if nubot-style movement is weaker or stronger than other notions of robotic movement), in the way it has been used to characterize and separate the power of other self-assembly models [7].

Brownian nubots. With nubots, under agitation, or multiple parallel movement rules, larger objects move faster. This is intended to model an environment with uncontrolled and rapid fluid flows. But in Brownian motion, larger objects move slower: what is the power of nubots with such a rate model, for example with rate 1/object size? Although assembly in such a model may be slower than with the usual model many of the same programming principles should apply, and indeed it will still be possible to assemble objects in a parallel distributed fashion.

Acknowledgements. A warm thanks to all of my co-authors on this topic, and especially to Erik Winfree and Chris Thachuk for helpful comments. The author is supported by NSF grants 0832824, 1317694, CCF-1219274 and CCF-1162589.

Cross-References

Combinatorial Optimization and Verification in Self-Assembly
Intrinsic Universality in Self-Assembly
Patterned Self-Assembly Tile Set Synthesis
Randomized Self-Assembly
Robustness in Self-Assembly
Self-Assembly at Temperature 1
Self-Assembly of Fractals
Self-Assembly of Squares and Scaled Shapes
Self-Assembly with Active Components
Self-Assembly with General Shaped Tiles
Temperature Programming in Self-Assembly
Two Handed Self-Assembly

Recommended Reading

1. L. M. Adleman, Q. Cheng, A. Goel, and M.-D. Huang. Running time and program size for self-assembled squares. In *STOC 2001: Proceedings of the 33rd annual ACM Symposium on Theory of Computing*, pages 740–748, Hersonissos, Greece, 2001. ACM.
2. J. Bath and A. Turberfield. DNA nanomachines. *Nature Nanotechnology*, 2:275–284, 2007.
3. H.-L. Chen, D. Doty, D. Holden, C. Thachuk, D. Woods, and C.-T. Yang. Fast algorithmic self-assembly of simple shapes using random agitation. In *DNA20: The 20th International Conference on DNA Computing and Molecular Programming*, volume 8727 of *LNCS*, pages 20–36. Springer, 2014. arxiv preprint: `arXiv:1409.4828`.
4. M. Chen, D. Xin, and D. Woods. Parallel computation using active self-assembly. In *DNA19: The 19th International Conference on DNA Computing and Molecular Programming*, volume 8141 of *LNCS*, pages 16–30. Springer, Sept. 2013. arxiv preprint `arXiv:1405.0527`.
5. A. Keenan, R. Schweller, M. Sherman, and X. Zhong. Fast arithmetic in algorithmic self-assembly. In *UCNC: The 13th International Conference on Unconventional Computation and Natural Computation*, volume 8553 of *LNCS*, pages 242–253. Springer, 2014. Arxiv preprint `arXiv:1303.2416` [cs.DS].
6. E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.
7. D. Woods. Intrinsic universality and the computational power of self-assembly. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2015. Accepted.
8. D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *ITCS'13: Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 353–354. ACM, 2013. Full version: `arXiv:1301.2626` [cs.DS].