Ashish Goel
Friedrich C. Simmel
Petr Sosík (Eds.)

LNCS 5347

DNA Computing

14th International Meeting on DNA Computing, DNA14
Prague, Czech Republic, June 2008
Revised Selected Papers

Springer

Lecture Notes in Computer Science 5347

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Ashish Goel Friedrich C. Simmel
Petr Sosík (Eds.)

DNA Computing

14th International Meeting
on DNA Computing, DNA14
Prague, Czech Republic, June 2-9, 2008
Revised Selected Papers

Springer

Volume Editors

Ashish Goel
Management Science and Engineering,
and Computer Science, Terman 311
Stanford University
Stanford, CA, USA
E-mail: ashishg@stanford.edu

Friedrich C. Simmel
Physics Department
TU Munich
Garching, Germany
E-mail: simmel@ph.tum.de

Petr Sosík
Institute of Computer Science
Faculty of Philosophy and Science
Silesian University
Opava, Czech Republic
E-mail: petr.sosik@fpf.slu.cz

Library of Congress Control Number: 2009936573

CR Subject Classification (1998): F.1, I.2.9, I.2.11, F.2.2, J.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-642-03075-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-03075-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12625284 06/3180 5 4 3 2 1 0

Preface

The 14th international meeting on DNA computation took place in the Czech Republic in Prague, June 2–9, 2008. During the last 14 years the DNA Computing meetings have been the key forum at the boundary between computer science, biochemistry and nanotechnology where the most recent results have been presented and their authors have met. Their scientific program includes mathematical foundations and theoretical study of DNA computing – or biocomputing in general – and recent experimental results in DNA nanotechnology, nanoscience and nanocomputing. It continues to be one of the most exciting interdisciplinary meetings, as exemplified by the diverse nature of contributions in this volume.

The meeting began with tutorial talks by Friedrich Simmel ("Molecular Biology for Computer Scientists"), Nadrian Seeman ("Structural DNA Nanotechnology"), and Yasubumi Sakakibara ("Formal Grammars for DNA Computation and Bioinformatics"). During the meeting, a number of excellent keynote speakers gave an up-to-date overview of different aspects of DNA computing and biochemical information processing. Luca Cardelli talked about "Molecules as Automata," while Niles Pierce gave an exciting talk entitled "Molecular Choreography—Programming Nucleic Acid Self-Assembly and Disassembly Pathways." In a more biological talk, Laura Landweber discussed "RNA-Guided, Epigenetic Programming and Re-programming of Genomic Information in Ciliates," and Ming Li gave an overview of "Modern Homology Search."

The meeting was concluded by a Nanoday with beautiful presentations by Christof Niemeyer, Kurt Gothelf, Andrew Ellington and David Pine.

In total, the meeting was attended by 85 researchers from 14 countries from Asia, North America and Europe. The DNA14 Program Committee received a total number of 59 submissions, of which 25 were presented orally. Their topics included theoretical models of biomolecular computing, demonstrations of biomolecular computing processes, self-assembly systems, DNA nanostructures and nanomachines, biotechnological and other applications of DNA computing and other related themes. This proceedings volume contains improved versions of 15 papers selected from these oral contributions.

We wish to express out gratitude to the members of the Program Committee, the local organizers, the sponsor – Silesian University in Opava – and the Steering Committee who made DNA14 a great success.

November 2008

Ashish Goel
Friedrich Simmel
Petr Sosík

Organization

Program Committee

Ashish Goel (Co-chair)	Stanford University, USA
Friedrich Simmel (Co-chair)	Technical University Munich, Germany
Martyn Amos	Manchester Metropolitan University, UK
Alessandra Carbone	Pierre et Marie Curie, France
Ho-Lin Chen	Stanford University, USA
Mark Daley	University of Western Ontario, Canada
Russell Deaton	University of Arkansas, USA
Rudolf Freund	Technical University of Vienna, Austria
Hendrik Jan Hoogeboom	Leiden University, The Netherlands
Jozef Kelemen	Silesian University, Czech Republic
Thomas LaBean	Duke University, USA
Maurice Margenstern	University of Metz, France
Yongli Mi	Hong Kong University of Science and Technology, Hong Kong
Satoshi Murata	Tokyo Instute of Technology, Japan
Mitsunori Ogihara	University of Rochester, USA
Ion Petre	University of Turku, Finland
John A. Rose	Ritsumeikan APU, Japan
Yasubumi Sakakibara	Keio University, Japan
Lloyd Smith	University of Wisconsin-Madison, USA
David Soloveichik	California Institute of Technology, USA
Petr Sosík	Silesian University in Opava, Czech Republic
Andrew Turberfield	Oxford University, UK
Reidun Twarock	University of York, UK
Ron Weiss	Princeton University, USA
Bernard Yurke	Boise State University, USA
Byoung-Tak Zhang	Seoul National University, Korea

Steering Committee

Lila Kari (Chair)	University of Western Ontario, Canada
Leonard Adleman	University of Southern California, USA (honorary member)
Anne Condon	University of British Columbia, Canada
Masami Hagiya	University of Tokyo, Japan
Natasha Jonoska	University of Southern Florida, USA
Chengde Mao	Purdue University, USA
Giancarlo Mauri	University of Milan-Bicocca, Italy
Satoshi Murata	Tokyo Instute of Technology, Japan

Gheorghe Paun	Romanian Academy, Bucharest and Seville University, Spain
John Reif	Duke University, USA
Grzegorz Rozenberg	University of Leiden, The Netherlands
Nadrian Seeman	New York University, USA
Andrew Tuberfield	Oxford University, UK
Erik Winfree	California Institute of Technology, USA

Local Organizing Committee - Prague

Petr Sosík (chair)	Silesian University in Opava, Czech Republic
Michaela Ačová	Silesian University in Opava, Czech Republic
Ludek Cienciala	Silesian University in Opava, Czech Republic
Lucie Ciencialová	Silesian University in Opava, Czech Republic
Magdalena Chmelařová	Silesian University in Opava, Czech Republic
Alica Kelemenová	Silesian University in Opava, Czech Republic
Sarka Vavrečková	Silesian University in Opava, Czech Republic
Milena Zeithamlová	Action M Agency, Czech Republic

Sponsors

Silesian University in Opava

Table of Contents

Experimental Validation of Signal Dependent Operation in Whiplash PCR

Ken Komiya[1], Masayuki Yamamura[1], and John A. Rose[2,⋆]

[1] Department of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology
{komiya,my}@dis.titech.ac.jp
[2] Institute of Information Communication Technology, Ritsumeikan Asia Pacific University
jarose@apu.ac.jp
http://www.apu.ac.jp/~jarose/

Abstract. Whiplash PCR (WPCR), which implements self-directed operation, programmed within a single DNA molecule, is a potential candidate for both mathematical and biological applications. However, WPCR-based methods are known to suffer from a serious efficiency problem called back-hybridization (BH). Previously, we proposed and partially validated a new *rule-protect* operation to abolish BH. In this work, we experimentally demonstrate the ability of *rule-protect* to drive multi-step WPCR. Successful implementation of isothermal operation at physiological temperatures is an essential benchmark for biological applications. We also propose the use of *rule-protect* for external signalling to control computational operation. Consequently, signal-dependent self-directed operation, which is conceptually new to DNA computing, is achieved. The present architecture, provided with sensing ability, allows a composite system design layering computational reactions, and would be suitable for functioning as the central processing unit of this system.

1 Introduction

In early studies in DNA-based computing, the solution to mathematical problems using pools of DNA molecules for data storage and parallel manipulations was implemented via human-directed reactions [1]. In particular, a computational program was encoded by an experimental protocol requiring human intervention. Subsequently, the feasibility of computations employing DNA molecules to encode both data and instruction sets was investigated, and DNA hairpin formation was utilized in the solution to a combinatorial problem instance for one-time only operation [2]. For successive operations, the use of restriction enzymes on double-stranded (ds) DNA molecules was proposed to implement a DNA-based automaton [3], in which a program was encoded by each reaction mixture involving a set of DNA molecules and enzymes.

In *Whiplash PCR* (WPCR), successive operations are processed according to data and a computational program encoded within a single molecule [4]. Successive state

⋆ To whom correspondence should be addressed.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 1–10, 2009.

transitions are implemented in parallel by the recursive, self-directed polymerase extension of a mixture of single-stranded (ss) DNA molecules. In principle, WPCR is a versatile platform for implementing both mathematical and biological applications. With a library of encoded strands, WPCR has been shown to be capable of solving instances of a variety of combinatorial problems, including those in NP-complete, uniquely in a single solution [5,6]. Moreover, WPCR is also theoretically capable of supporting *in vitro* evolutionary computation [7,8] and programmable protein evolution [9]. However, WPCR's basic process of recursive hybridization and extension is known to suffer from a serious efficiency problem inherent in the system design, called *back-hybridization* (BH) [10]. This effect, which amounts to an unfavorable halting of operation due to the re-formation of previously-targeted hairpin structures, has been shown to render WPCR impractical for actual applications.

To realize highly efficient operation, we previously introduced the *rule-protect* operation to the WPCR architecture [11]. By adding oligonucleotides for priming polymerase extension, a targeted transition rule on each strand is converted into a double-stranded form, rendering it unavailable. When this operation is applied just after each state transition, the hybridized hairpin is displaced, so that successive transitions are performed in a whipping and jacking-up manner, referred to as *Displacement Whiplash PCR* (DWPCR). In DWPCR, BH is expected to be abolished by strand displacement accompanying primer-targeted extension, which simultaneously renders the most recently implemented transition rule unavailable. Note here that, operation timing can be controlled by primer addition as an external signal for regulation. In addition to the expected near-ideal efficiency, DWPCR appears to allow isothermal operation at physiological temperatures, and is thus a promising candidate for potential mathematical and biological applications of WPCR. However, although the essential validity of primer-targeted strand displacement to abolish a hairpin structure was confirmed in [11], the feasibility of successive state transitions driven by *rule-protect* remained unobvious.

In the present work, we demonstrate that a WPCR-based reaction architecture involving the rule-protect operation comprises a highly performable computing system. As rules may also be deactivated via *rule-protect* prior to their implementation, this system also supports switching operation dependent on primer addition. Our architecture, provided with sensing ability, allows signal-dependent self-directed operation, a novel concept in DNA-based computing. The validity of both successive state transitions and rule switching are established via experiments on model hairpin systems. Such an architecture would be suitable for working as a central processing unit to be combined with a control module, which can release oligonucleotides [12] as information signals. Thus, a composite system design, layering computational reactions would be achieved.

2 The Rule Protect Operation

The details of the WPCR-based architecture based on *rule-protect* operation are illustrated in Figs. 1 and 2. Successive state transitions are performed, following both a computational program encoded by a single DNA molecule and a regulatory program implemented by a series of addition of primers. A set of cyclic reactions are applied similarly to standard WPCR, with the addition of *rule-protect* via primer extension, between rounds. It should also be noted that via *rule-protect*, the addition of primers can

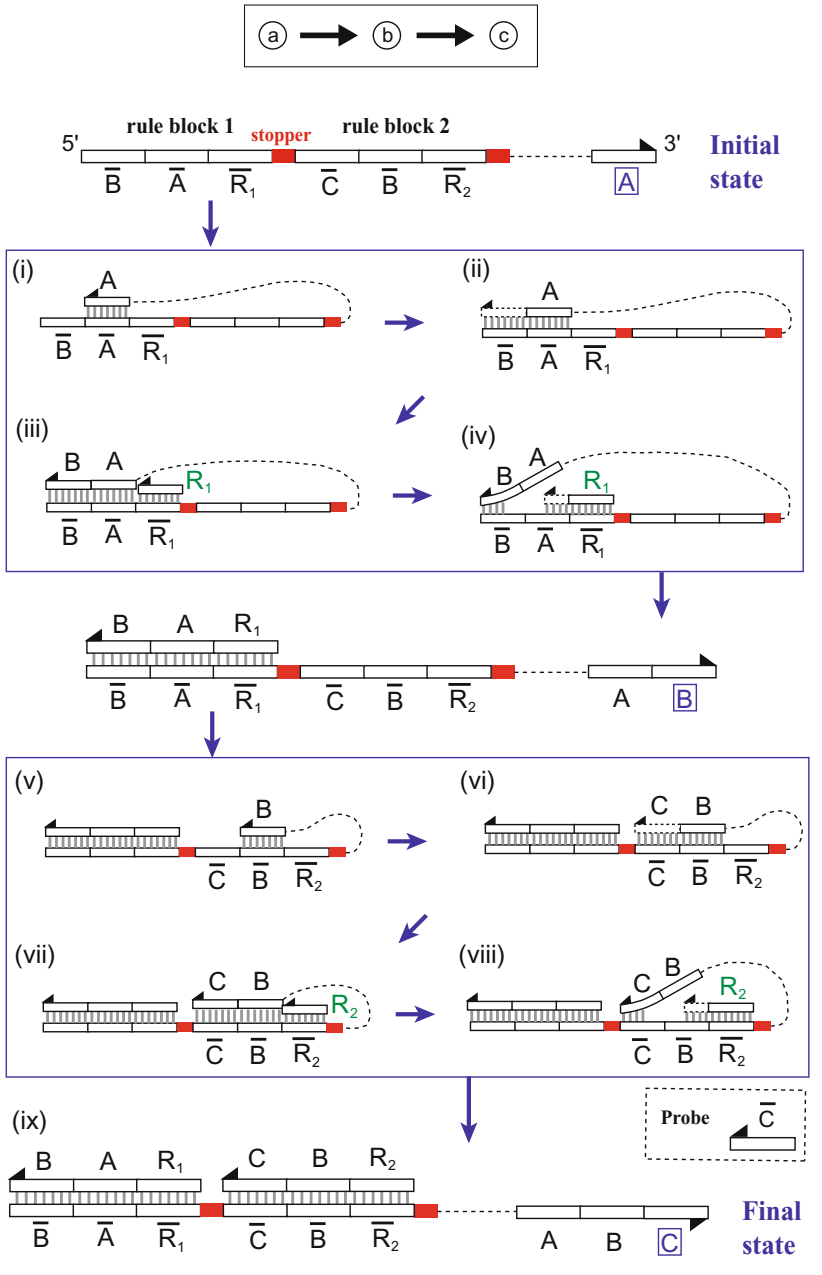


Fig. 1. Application of Rule Protect for Displacement Whiplash PCR: Top panel depicts a DNA strand encoding a DWPCR program for executing the short computational path, $a \rightarrow b \rightarrow c$. After each of the strand's two state transitions (boxed processes (i)-(iv), and (v)-(viii)), the computational process is driven by addition of the corresponding primer. Polymerization then implements *rule-protect*, opening the extended hairpin, and exposing the newly-polymerized head. See text for a detailed description. To aid discussion, a probe for hybridization with sequence *C*, which encodes the final computational state is also shown.

be used not only to drive the basic WPCR process, but also for external signalling to control computational operation. For instance, non-deterministically encoded transition rules on a strand (*e.g.*, two or more encoded rules having the same source state) can be determined by rule-protection. In this case, non-deterministic hairpin formation is abolished via primer-targeted extension prior to hairpin extension, resulting in external control at a fork point in the computation, a process here referred to as *rule-switch* (Fig. 2). Similarly, single rules may simply be switched off via *rule-protect*. As a consequence, the sequences of the input primers can represent input symbols for computation.

For clarity, the reaction architecture is described for a 3-state, 2 step computational implementation, $a \rightarrow b \rightarrow c$. A ssDNA molecule is encoded by its sequence to implement a set of *transition rules*, each of which encodes a transition from computational state x to state y. If X and $\overline{X}$ respectively denote the DNA sequence for state x and its Watson-Crick reverse complement, then the transition rule for executing state transition $x \rightarrow y$ is formed by the catenated pair of DNA sequences, $5' - \overline{Y}\,\overline{X} - 3'$. A rule block is then formed by flanking a sensory sequence at its 3' side for primer binding. The completed transition rule region is formed by the concatenation of all transition rule blocks, separated by *stoppers*. In Fig. 1, the transition rule region contains rule blocks for the transitions, $a \rightarrow b$ and $b \rightarrow c$, along with the sensory sequences for primers R_1 and R_2, respectively. The 3'-most codeword (sequence A in Fig. 1 (top panel)) is the *head*, which encodes the initial state of the strand.

Each round of state transition is implemented via a four-step process of (i) hairpin formation, (ii) polymerase extension, (iii) primer binding to the sensory sequence, $\overline{R_i}$, and (iv) primer extension accompanied by strand displacement. Step (i): successive transitions are initiated via intramolecular hybridization of the 3' head with a complementary state sequence in the transition rules. Step (ii): the transition directed by the hybridized rule block is then executed via polymerase extension of the head, which appends the sequence of the transition target state onto the strand's 3' end. Polymerase extension is terminated automatically when the DNA polymerase molecule encounters the 5' end or the transition block's 5' *stopper*, usually implemented by an adenine triplet sequence (*i.e.*, AAA), combined with the absence of dTTP in the buffer. Step (iii): denaturation of the newly-extended hairpin structure is prompted by primer binding to the sensory sequence in the hybridized rule block. Step (iv): given use of a DNA polymerase with strand-displacement activity, the hairpin structure is eliminated via polymerase extension of the primer, accompanied by conversion of the targeted rule block to the double-stranded form. The strand is then ready for the next round of hairpin formation and extension (process $b \rightarrow c$). This architecture provides a critical improvement for WPCR by abolishing BH. All processes are expected to be performed at high efficiency.

3 Materials and Methods

The feasibility of high-efficiency state transitions rests upon that of the *rule-protect* operation, which employs primer-triggered opening of a BH hairpin structure via strand displacement by DNA polymerase. We investigated the proposed operation in a pair of model experiments: (1) in a single transition, followed by primer-triggered hairpin

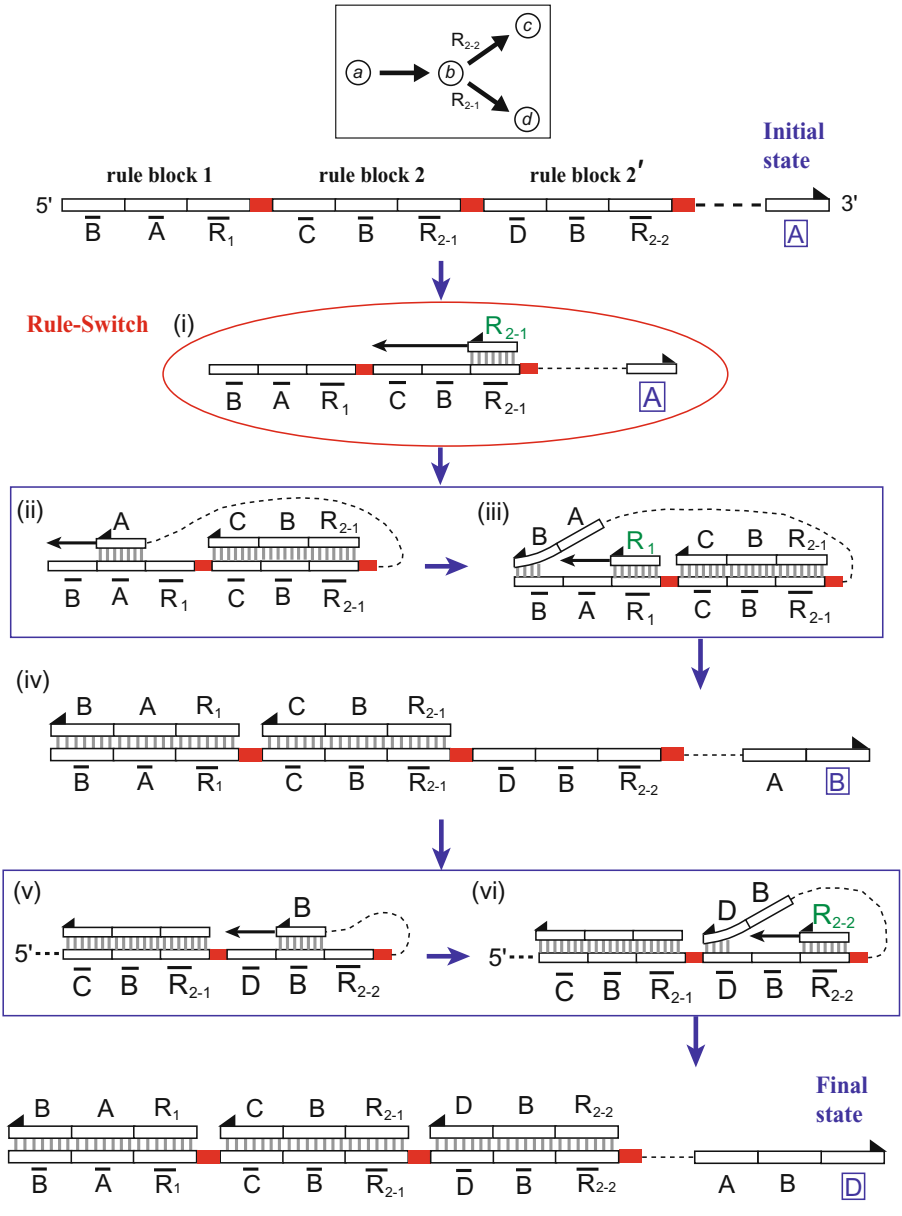


Fig. 2. Application of Rule Protect for Rule Switch: The *rule-protect* operation may be applied during computation, as an external signal to switch between the rules, $b \rightarrow c$ and $b \rightarrow d$. As shown, the selection of rule 2' ($b \rightarrow d$) over rule 2 ($b \rightarrow c$) is implemented by the addition and extension of primer R_{2-1} prior to the first state transition, which implements *rule-protect* at rule block 2 in advance. Here, each DWPCR transition is shown in blue boxes for clarity. Note that selection of rule 2 would be implemented by addition of primer R_{2-2} (not shown); both rules could likewise be switched off via the addition of both primers.

opening to prompt the formation of next target hairpin, and (2) in two successive transitions, followed by probing according to the final state. The feasibility of switching off a rule via *rule-protect* is also examined in the latter experiment. Results were analyzed by polyacrylamide gel electrophoresis (PAGE) with fluorophore-modified primers for fine discrimination of reaction products. The state and primer sequences used in the experiment are as follows (5' to 3'): $\overline{A}$: CCGGTTCTTCGTCTT; $\overline{B}$: TGCTTGTGTGTTGCG; $\overline{C}$: TCTCTCGTGTTTCGG; $\overline{R1}$: CCCTGTGTCTGTTCT; and, $\overline{R2}$: GCTGTTGGTTCCTGT. All DNA strands were purchased from Sigma Genosys, excluding the template for two successive transitions, which was from Nippon EGT.

3.1 Single State Transition

A series of polymerization steps, as essential processes for DWPCR was performed with a 78 nucleotide (nt) template strand (shown in Fig. 3 (A)), with the sequences given above, and a primer, R_1 which was 5'-labeled with the fluorophore, Cy5. The reaction mixture was prepared to become, after the addition of the primer, a 25-μl solution of 1x Thermo Pol buffer (20 mM Tris-HCl, 10 mM $(NH_4)_2SO_4$, 10 mM KCl, 2 mM $MgSO_4$, and 0.1% Triton X-100 (pH 8.8 @ 25°C)), containing 5 pmol template strand, 50 nmol dATP, dCTP, dGTP and 2.4 units of *Bst* DNA Polymerase (New England BioLabs). For the unreacted control (Fig. 3 (A) (i)), DNA polymerase was not added. The reaction was halted at step (ii) or (v) by Phenol-Chloroform-Isoamylalcohol (Pheno-Chlo) treatment. Hairpin extension (Fig. 3 (A) (ii)) was performed by an incubation at 37°C for 15 min on a ND-M01 heat block (Nissen). Afterwards, 7 pmol Cy5-modified R_1 was added to the mixture and again incubated at 37°C for 15 min. For the primer-hybridized intermediate (Fig. 3 (A) (iii)), 0.7 pmol Cy5-modified R_1 was added to the Pheno-Chlo-treated aliquot of 2.5 μl from step (ii). Samples were then run on a 8% (w/v) polyacrylamide gel (mono:bis = 29:1), each 0.5 pmol template in the respective lane, along with 20 bp DNA ladder (Takara Bio). The resulting gel images were obtained by a fluoro-image analyzer FLA-5100 (Fujifilm) upon laser excitation at 635 nm along with detection through the LPR filter (Fujifilm) before DNA staining, and after staining with SYBR Gold (Invitrogen) upon laser excitations at 635, 532, and 473 nm, along with detections through the LPR, LPG, and BPB filters, respectively.

3.2 Two Successive State Transitions, Probing and Switching-Off a Rule

For the complete validation of DWPCR, the feasibility of multi-round state transitions was investigated with an 111 nt template strand (Fig. 1 (top panel)) and primers R_1 and R_2, which were 5' labeled with fluorophores, Cy5 and TAMRA, respectively. The reaction mixture was prepared to become, after the additions of the primers and probe, a 50-μl solution of 1x Thermo Pol buffer, containing 5 pmol template strand, 100 nmol dATP, dCTP, dGTP and 1.6 units of *Bst* DNA Polymerase. For the unreacted control (Fig. 1 (i)), DNA polymerase was not added. Reaction was halted at step (ii), (vi), or (ix) via Pheno-Chlo treatment. Hairpin extension (Fig. 1 (ii)) was performed by an incubation at 37°C for 15 min on a thermal cycler, es gradient S (Eppendorf). Next, 15 pmol of Cy5-modified R_1 was added to the mixture, and it was incubated again at 37°C for 15 min. Afterwards, 15 pmol of TAMRA-modified R_2 was added to this mixture,

which was incubated again at 37°C for 15 min. For the primer-hybridized intermediates (Fig. 1 (iii) and (vii)), 0.9 pmol of Cy5-modified R_1 or TAMRA-modified R_2 was added to the Pheno-Chlo-treated aliquot of 3 μl from step (ii) or (vi), respectively. For the validation of probing according to the final state, 0.9 pmol of a probe which was 5' labeled with the fluorophore, FAM was added to the Pheno-Chlo-treated aliquot of 3 μl from step (ix). PAGE was performed in a manner similar to that described for a single state transition. The resulting gel images were obtained before and after staining with SYBR Gold upon laser excitations at 635, 532, and 473 nm, through the LPR, LPG, and BPB filters, respectively. Another experiment was also performed to validate the feasibility of switching-off a rule. The reaction protocol is the same as described above in this section, except for the use of unmodified primers and their additions in an exchanged order (First, R_2; then, R_1), to render the rule block 2 in Fig. 1 unavailable for transition. All samples were subjected to probing with the FAM-labeled probe before PAGE. The resulting gel image was obtained upon laser excitations at 473 nm, through the BPB filter.

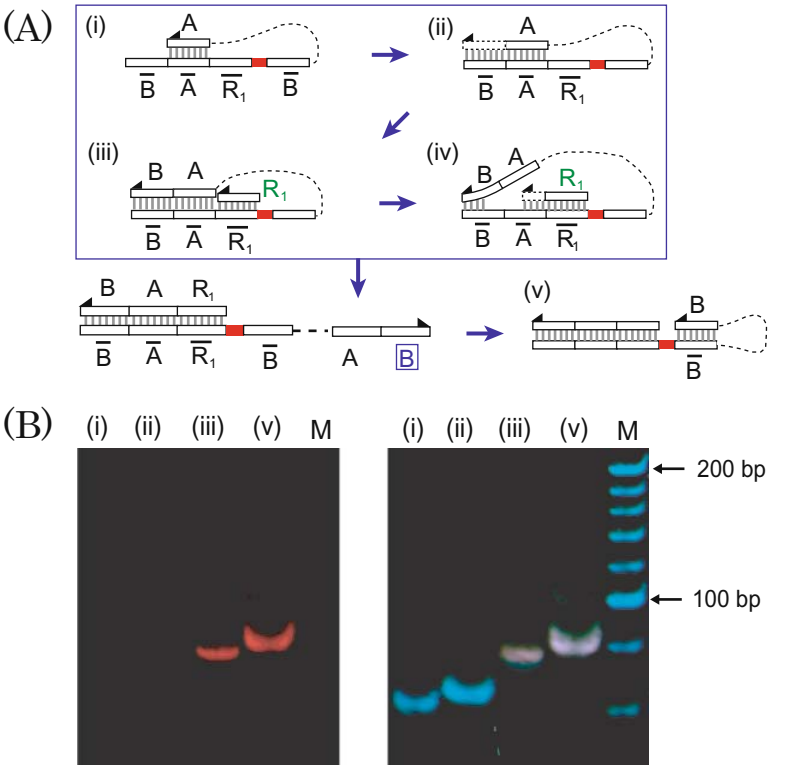


Fig. 3. DWPCR: Single State Transition (A) Experimental system for a single state transition, followed by *rule-protect*, and hairpin formation. (B) PAGE Results. In each panel, lane labels indicate corresponding reaction steps in (A). Left and right images show gel results prior to, and after DNA staining, respectively.

4 Results

Experimental results for one round of DWPCR are shown in Fig. 3 (B). The left image illustrates the obtained gel image before staining for identifying the primer-hybridized strands, shown in pseudo-red color. The right image illustrates that obtained after DNA staining to show all of the product DNA species, generated by merging three pictures each shown in pseudo red, green, and blue colors, respectively. In the lanes of the products from steps (i) to (v), as labeled above each image, the reduced mobilities of the bands lane by lane consistently agreed with the expected increase in the base-paired region of the reaction products depicted in Fig. 3 (A). The complete shift between bands indicated the ideally efficient implementation of the series of polymerization steps, in one round of DWPCR operation.

Experimental results for multiple rounds of DWPCR are shown in Fig. 4. The left and right images illustrate the obtained gel images before and after staining, respectively, generated by merging three pictures. In the left image, red bands which appeared in lanes (iii) and (vi) indicated the product DNA species with hybridized Cy5-labeled R_1 primer before and after primer extension, respectively. Likewise, green bands in lanes (vii) and (ix) indicated those with hybridized TAMRA-labeled R_2 primer before and after primer extension, respectively. Finally, the blue band in lane (ix)+P indicated that with hybridized FAM-labeled probe, which corresponded to the final 2-step transition product. In lanes (vi) and after, the appearance of two or more bands indicated that successive hairpin extension, primer binding and hairpin deformation for the second round of DWPCR were not performed completely. The appearance of the blue band in lane (ix)+P and the complete shift of the upward green band to this single blue band validated the successful implementation of multiple rounds of transition via DWPCR, and ideally efficient probing according to the final state, respectively.

Fig. 4 shows the reaction products of *rule-switch*, corresponding to steps (ii) and (iv) in Fig. 2 with an omission of rule-block 2' on the strands, along with the products as controls from steps (vi) and (ix) in Fig. 1. In lane 2-(iv)+P of the left image, disappearance of the single blue band, which appeared in lane 1-(ix)+P, indicated the successful implementation of *rule-switch* to abolish the hairpin formation for the second transition via *rule-protect*.

5 Discussion

The results presented in this work provide an experimental validation of the fundamental feasibility of the use of the *rule-protect* operation to implement both multi-step state transition and rule-switching. However, the validity of highly efficient multi-step transition still remains unclear. Incomplete reactions, the occurrence of which was indicated by the appearance of two or three bands in lanes (vi) and after in Fig. 4, may be attributed to steric hindrance, reducing the efficiency of primer binding, since we did not insert a sequence for spacing between the rule and head regions. Although the strand-displacement activity of *Bst* DNA polymerase at 37°C was stronger than that of DNA Polymerase I, Klenow Fragment (unpublished data), which was previously used in [11], it may not be sufficient for efficient operation. Optimization of reaction conditions along with quantitative determination of efficiency are deferred to later work.

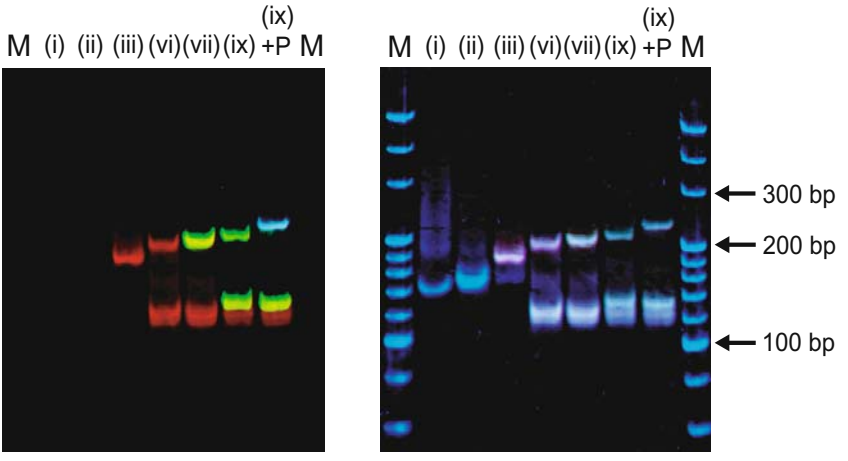


Fig. 4. DWPCR: Two Successive State Transitions and Probing (PAGE Results). In each panel, lane labels indicate corresponding reaction steps in Fig. 1. Left and right images show gel results prior to, and after DNA staining, respectively. See text for a detailed description.

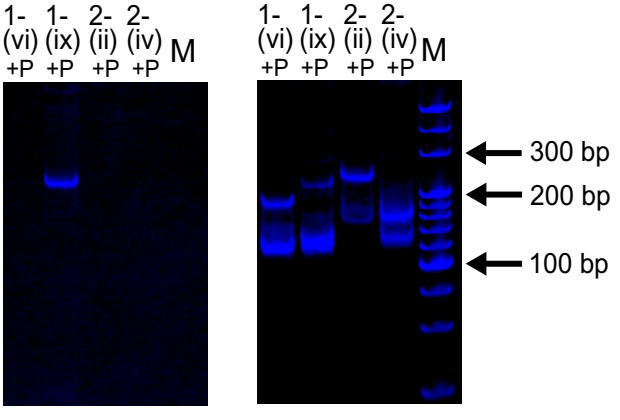


Fig. 5. Rule-Switch (PAGE Results). In each panel, lane labels indicate corresponding reaction steps in Fig. 1 and 2. Left and right images show gel results prior to, and after staining, respectively.

Efficient probing after hairpin deformation via *rule-protect* was also shown in this work. Probe binding to the exposed head region allows the extraction of the strands, which completed the specific transition, out of the reaction mixture containing a pool of differently encoded strands. This operation is directly applicable to the implementation of evolutionary computation. In addition, isothermal operation of multiple rounds of DWPCR at 37°C seems promising for biological applications. The reaction architecture utilizing DNA polymerase for hairpin deformation allows strand encoding without constraints, such as length limitation, uniform stability, and so on. These characteristics may enable evolutionary applications to protein evolution [9]. The versatility of the *rule-protect* operation is noteworthy. This operation can be applied not only to driving

WPCR (*i.e.*, DWPCR), but also to the controlled computation via *rule-switch*, in which a pending target rule of interest is deactivated via primer addition as an external signal, provided either via human manipulation or an output from another computational reaction [12]. The coordinated operation of pools of differently encoded strands according to external signals would realize a highly performable computing system.

Acknowledgements

Financial support for this work was generously provided by a Grant-in-Aid for Scientific Research B (18300100), from the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

References

1. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024 (1994)
2. Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M.: Molecular computation by dna hairpin formation. Science 288, 1223–1226 (2000)
3. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules. Nature 414, 430–434 (2001)
4. Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., Yokoyama, S.: Towards parallel evaluation and learning of boolean μ-formulas with molecules. In: Rubin, H., Wood, D. (eds.) DNA Based Computers III, pp. 57–72 (2000)
5. Sakamoto, K., Kiga, D., Komiya, K., Gouzu, H., Yokoyama, S., Ikeda, S., Sugiyama, H., Hagiya, M.: State transitions by molecules. Biosystems 52, 81–91 (1999)
6. Komiya, K., Sakamoto, K., Kameda, A., Yamamoto, M., Ohuchi, A., Kiga, D., Yokoyama, S., Hagiya, M.: DNA polymerase programmed with a hairpin DNA incorporates a multiple-instruction architecture into molecular computing. Biosystems 83, 18–25 (2006)
7. Wood, D.H., Bi, H., Kimbrough, S.O., Wu, D.-J., Chen, J.: DNA starts to learn poker. In: Jonoska, N., Seeman, N.C. (eds.) DNA 2001. LNCS, vol. 2340, p. 92. Springer, Heidelberg (2002)
8. Rose, J.A., Hagiya, M., Deaton, R.J., Suyama, A.: A DNA-based in vitro genetic program. J. Biol. Phys. 28, 493–498 (2002)
9. Rose, J.A., Takano, M., Hagiya, M., Suyama, A.: A DNA computing-based genetic program for in vitro protein evolution via constrained pseudomodule shuffling. Journal of Genetic Programming and Evolvable Machines 4, 139–152 (2003)
10. Rose, J.A., Deaton, R.J., Hagiya, M., Suyama, A.: Equilibrium analysis of the efficiency of an autonomous molecular computer. Phys. Rev. E 65, Article 021910, 1–13 (2002)
11. Rose, J.A., Komiya, K., Yaegashi, S., Hagiya, M.: Displacement whiplash PCR: Optimized architecture and experimental validation. In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 393–403. Springer, Heidelberg (2006)
12. Seelig, G., Soloveichik, D., Zhang, D., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314, 1585–1588 (2006)

Towards DNA Comparator: The Machine That Compares DNA Concentrations

Fumiaki Tanaka, Takashi Tsuda, and Masami Hagiya

Graduate School of Information Science and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan
fumiaki@dna-comp.org, tsuda@lyon.is.s.u-tokyo.ac.jp,
hagiya@is.s.u-tokyo.ac.jp

Abstract. A DNA comparator, which compares the concentration of target strand with that of reference strand, was developed. To extend the potentiality of autonomous machines consisting of DNA, the DNA comparator was designed by utilizing a difference of kinetics between hybridizations and branch migrations. The performance was evaluated through a fluorescence quenching experiment. As a result, the DNA comparator can detect at least four times higher concentration of target strand compared with that of reference strand. Furthermore, the limitation of this DNA comparator is discussed based on the rate constants and the ratio of branch migration.

1 Introduction

Nowadays, nucleic acids are utilized as nanoscale building blocks constructing intended patterns [1,2] and autonomous machines [3,4,5] (For more information, please refer to review articles such as [6,7]). In particular, enzyme-free machines consisting of nucleic acids have been researched energetically. Seelig *et al.* succeeded in a construction of some logic gates, AND gate, OR gate, NOT gate, and Thresholding gate, by utilizing hybridization and branch migration reactions [4]. This technology is expected to be utilized for the medical diagnosis and treatment at the DNA level such as a control of gene expressions as proposed by Benenson *et al.* [8].

In this paper, to extend the potentiality of autonomous machines made of nucleic acids, we propose a DNA machine called DNA comparator, which compares the concentration of target strand with that of reference strand and outputs single-stranded DNA if the concentration of target is higher than that of reference. This allows us to control DNA machines based on the concentrations of DNA rather than whether a DNA exists or not. The purpose of this paper is to prove the principle of this comparator. To achieve this, the operation of comparator was investigated through a fluorescence quenching experiment. Furthermore, we discuss the limitation of the comparator based on the reaction rate constant and ratio of branch migration.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 11–20, 2009.

2 Materials and Methods

2.1 Sample Preparation

All oligonucleotides were supplied by Sigma-Aldrich Japan and synthesized using PAGE purification except for those modified with a quencher. Oligonucleotides with a quencher were synthesized using HPLC purification. Two different kinds of fluophors, 6-carboxyfluorescein (6-FAM) and Texas Red, were attached to the 5'-end of sequences, while two corresponding quenchers, Black Hole Quencher 1 (BHQ1) and Black Hole Quencher 2 (BHQ2), were attached to the 3'-end of sequences. All oligonucleotides were first dissolved in 10 mM Tris Buffer with a pH of 7.4 and then diluted with 20×SSC buffer containing 3 M NaCl and 0.3 M sodium citrate with a pH of 7.0.

The oligonucleotide concentrations of each sample were determined by absorbance values at 260 nm using extinction coefficients calculated from the nucleotides and dinucleoside phosphates based on a nearest-neighbor approximation [9]. Absorbance values were measured at 90 °C to prevent oligonucleotides from forming secondary structures. Extinction coefficients of FAM, Texas Red, and BHQs (BHQ 1 and BHQ2) at 260 nm were assumed to be 20,960, 14,400, 8,000 L/(mol cm), respectively.

2.2 DNA Comparator

DNA comparator, which is a DNA machine to compare DNA concentrations, was developed by utilizing hybridization and branch migration reactions such as Seelig's DNA logic gate [4]. The purpose of DNA comparator is to determine whether the concentration of target strand is higher than that of reference strand. If the concentration of target is higher, DNA comparator outputs a single-strand, which may be input strand to the other molecular logic gate.

Our DNA comparator consists of three double-strands and two inputs (see Figure 1). One of two inputs is a target strand whose concentration is investigated, while the other is a reference strand whose concentration is the standard. Two of three double-strands have bulge loops in the middle of sequences and toe-hold structures, which two inputs can hybridize to, at 5'-end of the opposite sequences. DNA comparator was designed to work as follows: First, two inputs *reference* and *target* hybridize to the toeholds of *blg_comp1* and *blg_comp2*, respectively. Then, *reference* and *blg_comp1* form a double-strand, while *target* and *blg_comp2* also form a double-strand because of the branch migration reaction. As a result, *bulge1* and *bulge2* are released as single strands and immediately form into a double-strand because the loop domains of *bulge1* and *bulge2* are complementary to each other. Thus, only the extra amount of *bulge2* can hybridize to the toehold of *BHQ2-out_comp*, resulting in the branch migration. As a result, *TRed-output2* is released as the single strand, which can be monitored by the fluorescent intensity of Texas Red. The key point is that the hybridization between *bulge1* and *bulge2* is much faster than the branch migration between *bulge2* and *TRed-output2*&*BHQ2-out_comp*. Thus, if the concentration of *bulge2* is lower than that of *bulge1*, almost all *bulge2* probably hybridize to *bulge1*

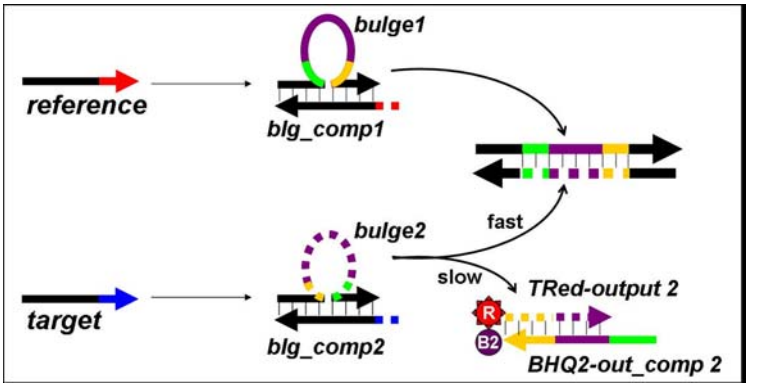


Fig. 1. Principle of DNA Comparator

rather than the toehold of *BHQ2-out_comp*. Therefore, only if the concentration of *target* is higher than that of *reference*, *bulge2* can hybridize to the toehold of *BHQ2-out_comp* and remove *TRed-output2* from *BHQ2-out_comp* by the branch migration.

2.3 Kinetic Measurement

The operation of DNA comparator has been checked by the change of fluorescent intensity at 40 °C using F-2500 Fluorescence Spectrophotometer (HITACHI). The excitation and emission wavelengths for FAM were respectively 494 and 518 nm, while those for Texas Red were respectively 590 and 615 nm.

2.4 Sequence Design

DNA sequences were carefully designed not to hybridize each other except for complementary sequences. The stabilities of every DNA duplexes were calculated by the minimum free energies (ΔG_{min}). The ΔG_{min} was predicted using the program obtained by modifying our previous program [10]. Our program considers intermolecular structures including bulge and internal loops, dangling ends, and stacking pairs rather than intramolecular structures such as hairpin loops. In addition, sequences were designed such that the ΔG_{min} of the toehold domains fell into a narrow range in order to uniform the efficiency of branch migration. For the same reason, sequences were also designed such that the ΔG_{min} of the duplex domains fell into a narrow range. To prevent the unintended secondary structure, sequences were further designed such that they did not have continuous repeats of the same base.

The better sequences were searched according to the above criteria by a hill-climbing algorithm. Eventually, we chose the best sequences among the 20 trials (see Table 1). The ΔG_{min} of any duplexes except for complementaries were at least -4.65 kcal/mol. The deviation of ΔG_{min} of the toehold domains was in the 0.15 kcal/mol range, while that of the duplex domains was in the 0.20 kcal/mol range. Furthermore, obtained sequences had at most three continuous repeats of the same base.

Table 1. Sequence List

sequence name	sequence [a] (5' ⟶ 3')
reference	CCAAACTACTTACGTCTTCTAAGCAACTAACTGATG
bulge1	CCAAACTACTTACGTTGAACATACACCGAGGTTTAGTCCAAACTTCTAAGCAACTAA
blg_comp1	CATCAGTTAGTTGCTTAGAAGACGTAAGTAGTTTGG
FAM1-bulge1 [b]	F-CCAAACTACTTACGTTGAACATACACCGAGGTTTAGTCCAAACTTCTAAGCAACTAA
BHQ1-blg_comp1 [b]	CATCAGTTAGTTGCTTAGAAGACGTAAGTAGTTTGG-B1
target	TATAAGTCAGGTCTCTTTCGTATACCACAATTCCAA
bulge2	TATAAGTCAGGTCTCTTTGGACTAAACCTCGGTGTATGTTCATTTCGTATACCACAA
blg_comp2	TTGGAATTGTGGTATACGAAAGAGACCTGACTTATA
TRed-bulge2 [b]	R-TATAAGTCAGGTCTCTTTGGACTAAACCTCGGTGTATGTTCATTTCGTATACCACAA
BHQ2-blg_comp2 [b]	TTGGAATTGTGGTATACGAAAGAGACCTGACTTATA-B2
TRed-output2	R-TTTGGACTAAACCTCGGTGTA
BHQ2-out_comp2	TGAACATACACCGAGGTTTAGTCCAAA-B2

[a] The F, R, B1, and B2 at the end of sequences denote FAM, Texas Red, BHQ 1, and BHQ 2, respectively.
[b] These sequences were used only to estimate the kinetic constants.

2.5 Estimation of Rate Constant

The rate equation can be represented as the following second-order kinetics.

$$I + OC \xrightarrow{k} IC + O, \tag{1}$$

where I, O, and C represent the input strand, output strand, and complementary strand to input, respectively. Here, "output strand" represents the single strand produced by branch migration rather than the output of DNA comparator. Furthermore, the initial concentration of I is similar to that of OC:

$$I_0 = OC_0, \tag{2}$$

where I_0 and OC_0 are the initial concentrations of I and OC, respectively.

Thus, the rate constant of branch migration was estimated using the following equation:

$$\frac{dO}{dt} = k(\alpha I_0 - O)(\alpha OC_0 - O) \tag{3}$$

$$= k(\alpha I_0 - O)^2, \tag{4}$$

where the parameter α represents the ratio of completion of branch migration. Since all inputs do not complete the branch migration, some double-strands with the toehold probably remain without a branch migration.

Solving the above equation, we obtain the following equation:

$$O = \frac{(\alpha I_0)^2 kt}{1 + \alpha I_0 kt}. \tag{5}$$

To convert the concentration into the fluorescent intensity, the above equation was multiplied by the dF/I_0, where the dF is the difference calculated by the subtracting the fluorescent intensity of quenched state from that of non-quenched state F_0. Therefore, fluorescent intensities as a function t can be estimated by the following equation:

$$f(t) = F_0 + \frac{dF\alpha^2 I_0 kt}{1 + \alpha I_0 kt}. \tag{6}$$

Finally, the rate constants of branch migration were estimated by fitting the above equation to the experimental curves using the fit command of gnuplot.

3 Experimental Results

3.1 Reaction Rate Constant

To confirm the elementary reaction steps and compare their kinetics, we first derived reaction rate constants from the curves of fluorescent intensity versus the reaction time. The parameters are shown in Table 2, while the fitting results are shown in Figure 2. The k_3 was slower than k_1 and k_2. This was probably because the k_3 was the rate constant by separating the double-strand without a bulge, while the k_1 and k_2 were those by separating double-strands with bulges in the middle of sequences (see Table 2). Why was the branch migration with a bulge fast? Once the branch migration progresses at the position of bulge loop, the branching point never goes back because the complementary sequence is apart from the branching point. Thus, the branch migration with a bulge in the middle of sequence was faster than the other without a bulge. This 'loop acceleration' may be useful for the speed-up of branch migration as well as the DNA catalyst technique [11] [5].

On the other hand, the difference of kinetics between k_1 and k_2 was not obvious. Although we designed sequences such that both the toehold and double-helix domains were respectively in the 0.15 kcal/mol and 0.20 kcal/mol ranges, both the rate constant (k) and ratio of branch migration (α) were significantly different. This result showed that the kinetics was not obviously determined by only their thermodynamics. Thus, a design method of sequences to control kinetics need to be further investigated.

3.2 Validation of DNA Comparator

To validate the function of DNA comparator, we investigated whether a higher concentration of *reference* can prevent the branch migration between *bulge2* and double-strand consisting of *TRed-output2* and *BHQ2-out_comp2*. To achieve this, we measured the fluorescent intensity change of Texas Red attached to *TRed-output2* at various concentrations of *reference*. The detailed experimental procedure was as follows:

1. First, 380 μl of 125 nM *TRed-output2* was injected into a micro cell and then the fluorescent intensity began to be monitored.
2. Second, 10 μl of 5 μM *BHQ2-out_comp2* was injected into the cell, where the *BHQ2-out_comp2* hybridized to the *TRed-output2*, resulting in the quenching between TRed and BHQ2.
3. Third, 10 μl mixture of 5 μl of 10 μM double-strand between *bulge1* and *blg_comp1* and 5 μl of 10 μM double-strand between *bulge2* and *blg_comp2* was injected into the cell.
4. Finally, 10 μl mixture of 5 μl of x μM (x is 0, 1.0, 2.5, 5.0, or 10.0) *reference* and 5 μl of 10 μM *target*, where both inputs triggered the reaction by the branch migration.

The result is shown in Figure 3. The top graph in this figure shows the curves of normalized fluorescent intensity versus time (s), while the bottom

Table 2. Reaction Rate Constant

reaction formula [a]	rate constant [b]	ratio of branch migration [c]
ref + F-blg1&B1-blg_cp1 $\xrightarrow{k_1}$ *ref&B1-blg_cp1 + F-blg1*	k_1=3.3·10^5	0.47
target + R-blg2&B2-blg_cp2 $\xrightarrow{k_2}$ *target&B2-blg_cp2 + R-blg2*	k_2=1.3·10^6	0.36
blg2 + R-out2&B2-out_cp2 $\xrightarrow{k_3}$ *blg2&B2-out_cp2 + R-out2*	k_3=3.0·10^4	0.66

[a] Sequence names in reaction formula are slightly modified for simplisity. For example, 'B1-blg_cp1' represents 'BHQ1-blg_comp1' in Figure 1 and Table 1.
[b] This is represented by k in equation 6.
[c] This is represented by α in equation 6.

one does the curve of normalized fluorescent intensity versus the concentrations of *reference*. The fluorescent intensity in the figure was normalized by using $F' = (F - F_{min})/(F_{max} - F_{min})$, where F' and F were respectively the fluorescent intensities after and before the normalization, while F_{min} and F_{max} were respectively the fluorescent intensities after and before the quenching. This figure shows that the *reference* can interrupt the branch migration by *bulge2* because the higher the concentration of *reference* was, the smaller the fluorescent intensity change became. Thus, the comparator worked well as expected. However, the figure also revealed all *reference* did not succeed in preventing the branch migration by *bulge2*. Even if the concentration of *reference* was similar to that of *target* (at $C_t = 10$, where C_t is the total concentration, in the bottom graph in Figure 3), some *bulge2* produced by *target* completed the branch migration with *TRed-output2&BHQ2-out_comp2*. Unfortunately, the branch migration triggered by *target* was approximately four times faster than that by *reference* (see Table 2). For this reason, the single-stranded *bulge2* produced by *target* probably completed the branch migration before it was interrupted by the *bulge1* produced by *reference*.

4 Discussion

4.1 Limitation of Current DNA Comparator

In the previous section, the principle of DNA comparator was proved through a simple experiment. To evaluate the potential of DNA comparator, however, we must consider the limitation of comparator. First, we focus on how concentrations of *target* can be detected by this comparator. The answer of this question mainly depends on the ratio of branch migration. From the result of Table 2, the ratio of the concentration of output strand to that of input strand by two-step branch migration (*target* → *bulge2* → *TRed-output2*) were approximately 0.24 (= 0.36·0.66). Thus, if the concentration of *TRed-output2* is needed for the detection more than 5 % of concentration of *bulge2*, the *target* need to be more than 20.83 % of concentration of *bulge2*. Although this limitation is a serious problem for the application such as medical diagnosis and treatment, the problem may be reduced by using a DNA machine that amplifies an input DNA such as the Seelig's amplifier [12].

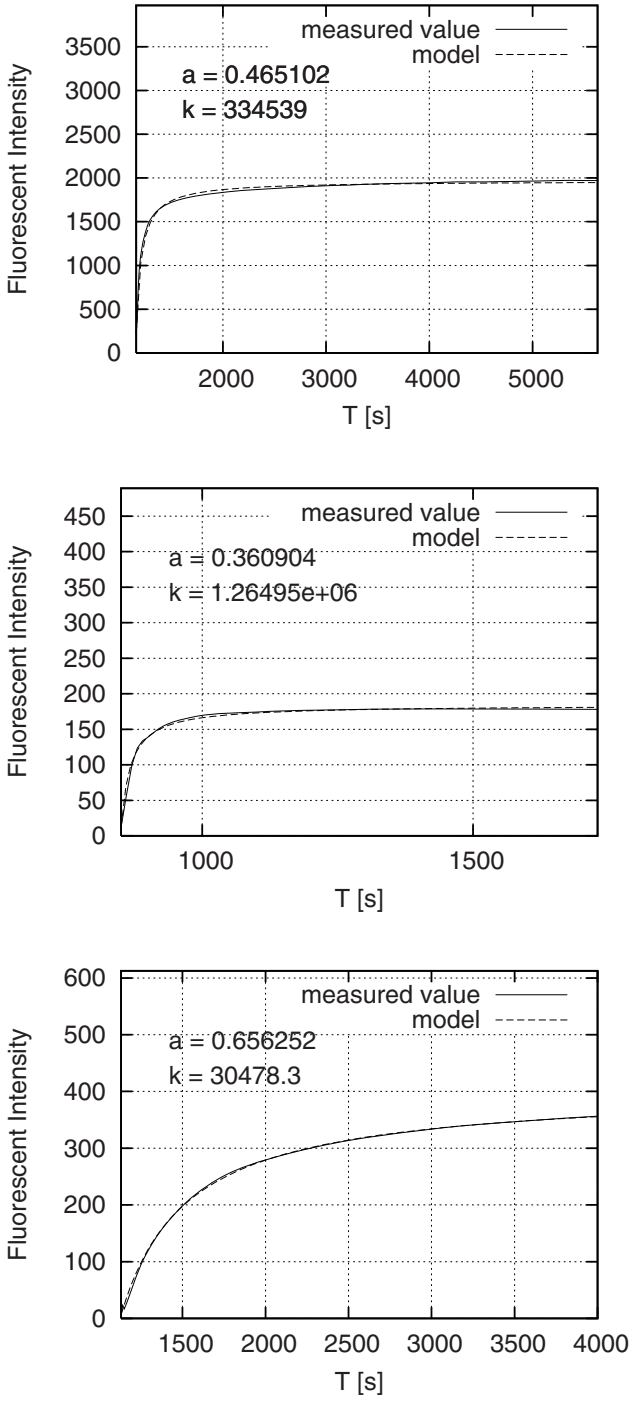


Fig. 2. Fitting Results (From top to bottom, *reference* → *bulge1*; *target* → *bulge2*; *bulge2* → *output2*.)

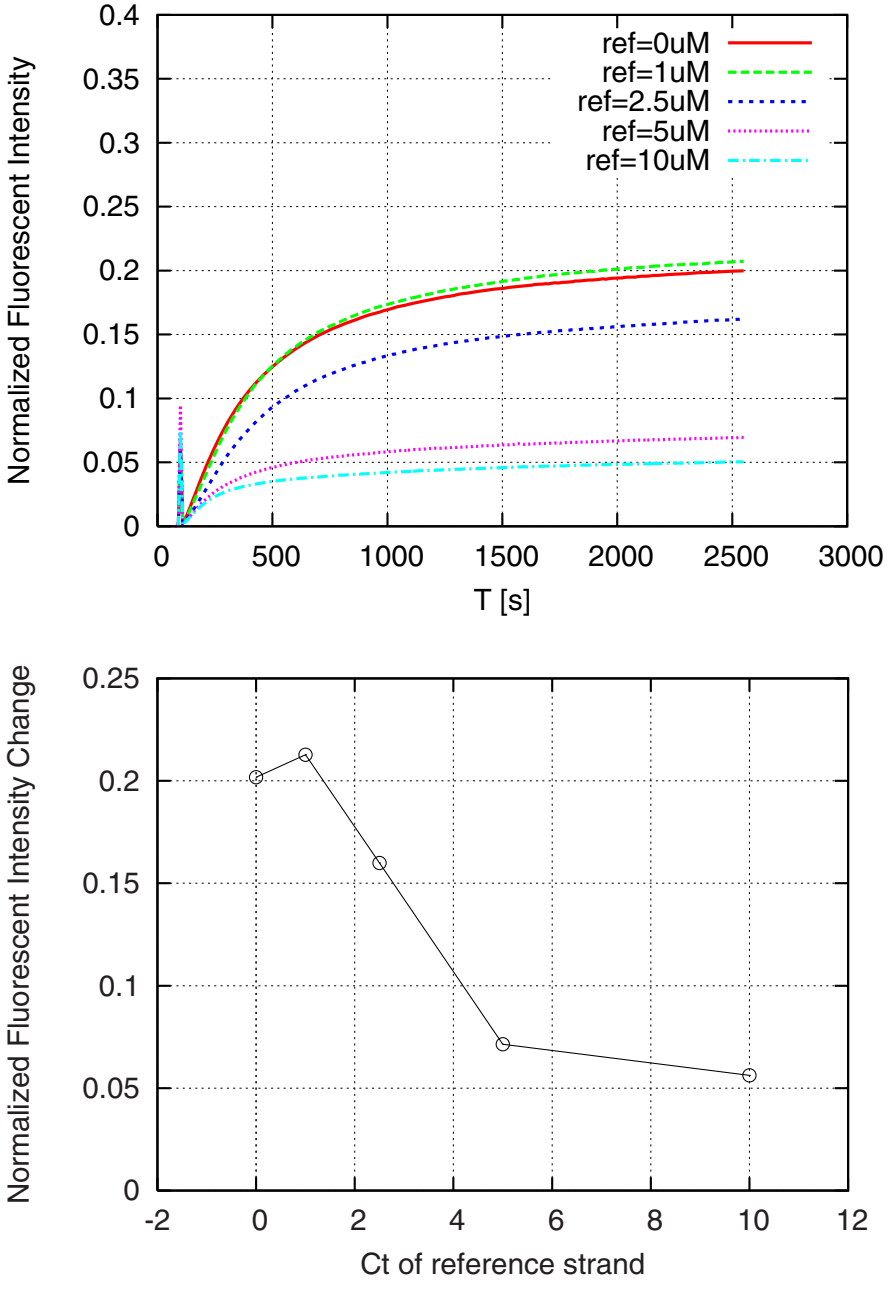


Fig. 3. Result of DNA Comparator

Furthermore, we focus on how difference of concentrations between *reference* and *target* can be detected by the comparator. From the bottom graph in Figure 3, it is difficult to distinguish between when the concentration of *reference*

was 5 μM and when that was 10 μM, where the concentration of *target* was 10 μM. This means that even if the concentration of the *target* was twice as high as that of the *reference*, the comparator probably cannot detect the difference. In contrast, the difference between 10 μM and 2.5 μM of *reference*s could be detected successfully. Therefore, our DNA comparator can detect at least four times higher concentration of *target* compared with that of *reference*.

4.2 Comparison with Traditional DNA Gates

Seelig *et al.* proposed the thresholding gate, which releases an output strand if the concentration of input strand is higher than a threshold value [4]. The thresholding gate worked well and the function of the gate was similar to DNA comparator. Compared with the thresholding gate, the advantage of our comparator is its scalability. As shown in Figure 4, the current comparator can be easily extended to the system with two targets, where the comparator releases *output1* or *output2* according to whether the concentration of *target1* is higher than *target2* or not. Note that both outputs can be utilized as the inputs to the downstream gates. This means that we can switch the behavior of system whether the concentration of *target1* is higher than that of *target2* or not. Therefore, the DNA comparator will be utilized as the switching device based on the concentrations of *targets*.

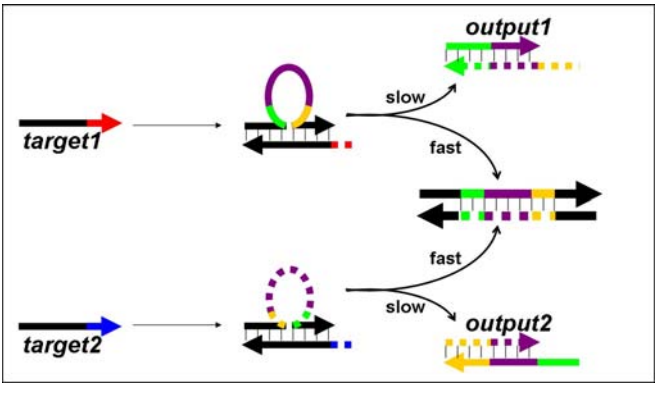


Fig. 4. Extension of DNA Comparator

5 Conclusions and Future Work

DNA comparator, which compares the concentration of *target* with that of *reference*, was developed by utilizing the hybridization and branch migration reactions. The principle, which utilized the difference of kinetics between hybridization (fast) and branch migration (slow), was proved from the curves of fluorescent intensity versus time. The experimental results showed that our DNA comparator can detect at least four times higher concentration of *target* compared with that

of *reference*. In future work, we focus on an experimental validation of the limitation of the comparator and extend it to the comparator that releases two different outputs according to difference of concentrations of two targets.

References

1. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature 440(7082), 297–302 (2006)
2. Erben, C.M., Goodman, R.P., Turberfield, A.J.: Single-molecule protein encapsulation in a rigid DNA cage. Angew. Chem. Int. Ed. Engl. 45(44), 7414–7417 (2006)
3. Yurke, B., Mills Jr., A.P.: Using DNA to power nanostructures. Genetic Programming and Evolvable Machines 4, 111–122 (2003)
4. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)
5. Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318(5853), 1121–1125 (2007)
6. Simmel, F.C., Dittmer, W.U.: DNA nanodevices. Small 1(3), 284–299 (2005)
7. Bath, J., Turberfield, A.J.: DNA nanomachines. Nature nanotechnology 2, 275–284 (2007)
8. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expression. Nature 429(6990), 423–429 (2004)
9. Gray, D.M., Hung, S.H., Johnson, K.H.: Absorption and circular dichroism spectroscopy of nucleic acid duplexes and triplexes. Methods Enzymol. 246, 19–34 (1995)
10. Tanaka, F., Kameda, A., Yamamoto, M., Ohuchi, A.: Design of nucleic acid sequences for DNA computing based on a thermodynamic approach. Nucleic Acids Res. 33(3), 903–911 (2005)
11. Turberfield, A.J., Mitchell, J.C., Yurke, B., Mills, A.P., Blakey, M.I., Simmel, F.C.: DNA fuel for free-running nanomachines. Phys. Rev. Lett. 90(11), 118102 (2003)
12. Seelig, G., Yurke, B., Winfree, E.: Catalyzed relaxation of a metastable dna fuel. J. Am. Chem. Soc. 128(37), 12211–12220 (2006)

Construction of Photon-Fueled DNA Nanomachines by Tethering Azobenzenes as Engines

Xingguo Liang[1], Hidenori Nishioka[1], Nobutaka Takenaka[1], and Hiroyuki Asanuma[1,2,*]

[1] Department of Molecular Design and Engineering, Graduate School of Engineering, Nagoya University, Chikusa, Nagoya 464-8603, Japan
[2] Core Reserach for Evolution Science and Technology (CREST), Japan Science and Technology Agency (JST), Kawaguchi, Saitama 332-0012, Japan
asanuma@mol.nagoya-u.ac.jp

Abstract. Nanoscale DNA tweezers operated by photo-irradiation were constructed by using azobenzene-modified DNA as materials. The azobenzenes that can photoisomerize between *trans* and *cis* form were used as the engines to open and close the tweezers. The work principle is based on the reversible photoregulation of complementary DNA hybridization. When non-substituted azobenzene was used, the tweezers were opened after UV light irradiation (330-350 nm, *cis* form), and closed after visible light irradiation (440-460 nm, *trans* form). More interestingly, the operation reversed when an azobenzene derivative with a *para*-isopropyl group was used: UV light irradiation closed the tweezers and visible light irradiation opened them. As compared with the oligonucleotide-fueled DNA machines, the nanomachines constructed here were "environment-friendly" because no dsDNA waste was produced. Furthermore, the operation can be repeated many times simply by switching the photo-irradiation without any decrease of the cycling efficiency.

Keywords: DNA nanomachine, azobenzene, photoregulation, hybridization.

1 Introduction

Recently, DNA has been considered to be one of the most promising molecules for future applications in nanotechnology and molecular computing [1-5]. A variety of DNA nanomachines such as tweezers, gears, and walkers have been constructed by intelligent sequence design [6-11]. These molecular machines can perform mechanical functions such as scission, directional motion and rolling powered by molecular fuels. Not only satisfying with the scientific curiosity, researchers even have gone further to develop sensors, molecular transporters, and controlled drug delivery systems by using these DNA-based nanomachines [7]. However, most of these DNA machines are fueled with oligonucleotides that hybridize with target sequences and drive the dynamic structural changes. For repetitive operation, the DNA fuel has to be removed from the DNA machine by adding another DNA that is completely or partially

* Corresponding author.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 21–32, 2009.

complementary to the fuel [8]. Usually, the operation efficiency decreases gradually due to the accumulation of the double-stranded DNA waste. For further development of DNA nanotechnology, this problem is required to overcome.

In this study, we developed DNA tweezers powered by light irradiation based on our previously reported technique of photoregulating DNA hybridization-dehybridization [12-14]. Azobenzenes attached on the DNA machine were used as photoswitches to open and close it. The operation can be repeated many times with a constant efficiency because no wastes were produced during the working cycles. Furthermore, a nanomachine that could work on a single supramolecular level was also constructed: Photoswitches were attached directly to the arm of DNA tweezers and the azobenzene-modified part did not separate from the tweezers even at opening state. We proposed a novel concept of molecular machine driven by external stimuli without disturbance of the internal solution conditions during the operation.

2 Results and Discussion

2.1 Photoresponsive DNA Tweezers Involving Non-Substituted Azobenzene

Molecular Design of DNA Tweezers Powered by Photoirradiation. As illustrated in Fig. 1, a photoresponsive DNA machine composed of four strands (**A**, **B**, **C**, and **F**) is designed based on the DNA-fueled "tweezers" reported by Yurke et al [8]. The working principle of the original DNA-fueled molecular tweezers is described as follows (Fig. 1a): Strand **F** is added to close the tweezers by partially hybridizing with strand **B** and **C**, leaving an 8-nt-long overhang. In order to open the closed tweezers, strand **F′** that is completely complementary to strand **F** is added. **F′** first hybridizes with the single-stranded overhang, followed by peeling of strand **F** from the DNA machine through branch migration. For each operation cycle, **F** and **F′** are added alternatively, and **F/F′** duplex is produced as a waste. Obviously, more and more **F/F′** waste accumulates during successive opening and closing operation.

Instead of opening the tweezers by adding strand **F′**, we used an azobenzene-modified oligonucleotide **F** ($\mathbf{F_{8x}}$ or $\mathbf{F_{12x}}$) that can be dehybridized from the strands **B** and **C** by ultraviolet (UV) light irradiation (Fig. 1b). Here, the azobenzenes are tethered onto DNA through D-threoninol linkers. It can be expected that photoisomerization of the multiple azobenzene moieties (**Azo**) to non-planar *cis* form decreases greatly its hybridization ability and opens the tweezers [15-16]. This photoresponsive **F** can be simply recycled with visible (Vis) light irradiation and the *cis*-to-*trans* isomerization closes the tweezers again. Because the photoisomerization is completely reversible and the azobenzene is chemically stable, repetitive operation by simply switching the wavelength of the irradiation light can also be expected. For evaluating the efficiency of opening and closing operation through fluorescence resonance energy transfer (FRET), tetrachlorofluorescein (**TET**, as a fluorophore) and carboxytetramethylrhodamine (**TAMRA**, as an acceptor) were attached at 5′- and 3′-end of **A**, respectively (Fig. 1c). As the efficiency of resonant energy transfer increases abruptly with the decrease of distance between the donor and acceptor, opening and closing of the tweezers can be quantitatively monitored by measuring the fluorescence change of **TET** (emission at around 540 nm, excited at 514.5 nm).

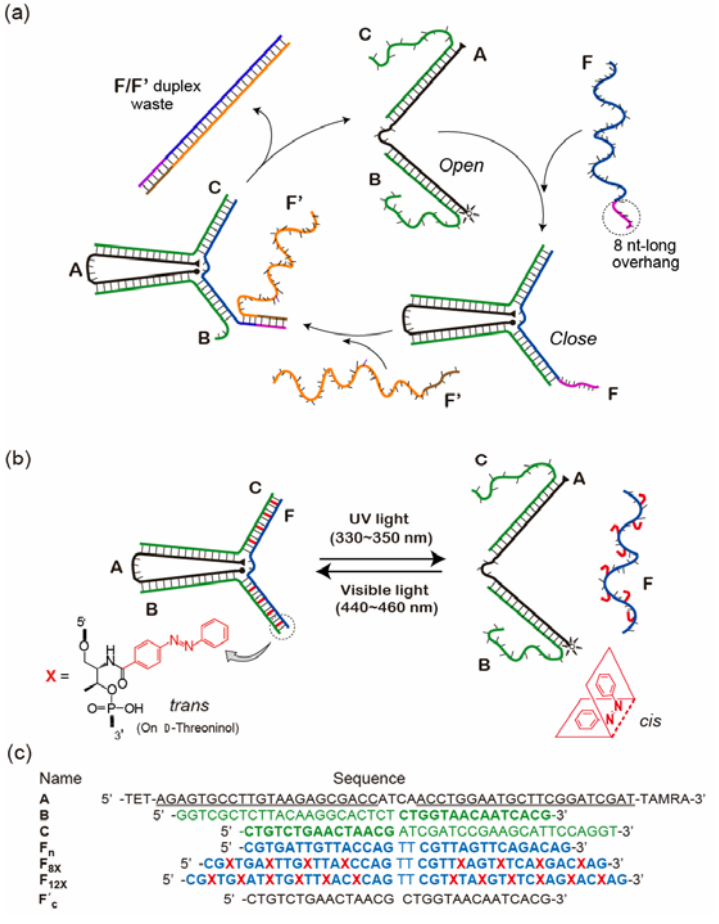


Fig. 1. Schematic illustration of working principles of DNA nano-tweezers. (a) Original tweezers constructed by Yurke et al.; (b) Photoresponsive tweezers involving non-substituted azobenzenes tethered on D-threoninol; (c) Oligonucleotide sequences used in this study. Visible light irradiation (*trans* form) closes the tweezers (left side of (b)), and UV light irradiation (*cis* form) opens them (right side of (b)). The structures of azobenzene moiety (**X** residue) are also show in (b). Underlined letters in **A** indicate the sections that hybridize with **B** and **C**. Complementary sections between **F** ($\mathbf{F_n}$, $\mathbf{F_{8X}}$, and $\mathbf{F_{12X}}$) and **B**, **C** are shown in bold letters. $\mathbf{F_{8X}}$ and $\mathbf{F_{12X}}$ are the photoresponsive oligonucleotides involving 8 and 12 azobenzene moieties, respectively. $\mathbf{F_n}$ is the native DNA without azobenzenes as the control sequence of $\mathbf{F_{8X}}$ and $\mathbf{F_{12X}}$. $\mathbf{F'_c}$ hybridizes to **F** and forms a duplex with a bulge of two thymidines.

According to our previous reports, an efficient photoregulation of DNA hybridization should be carried out at a temperature between the melting temperature (T_m) of the duplex involving *trans*-azobenzenes and that of the duplex involving *cis*-azobenzenes [12-15]. If the temperature is too high, the duplex dissociates in both *trans*- and *cis*-cases; if the temperature is too low, the duplex is formed in both cases. Here, the length of the single-stranded overhangs of **B** and **C** (when the tweezers are

open) that are complementary to **F** was designed to be 15-nt-long (the corresponding parts of DNA-fueled "tweezers" reported by Yurke et al are 24-nt-long), and two extra dTs were inserted in **F** at the middle position to lower the T_m. For efficient photoregulation of the opening and closing of DNA tweezers, as many as 8 or 12 azobenzene residues were introduced into a 32-nt-long strand **F** (designated as $\mathbf{F_{8X}}$ and $\mathbf{F_{12X}}$) [12]. Here, each azobenzene residue was attached on the D-threoninol linker that was inserted into DNA backbone using the standard phosphoramidite chemistry (Fig. 1b). Note that the structure at the position of each azobenzene residue in the DNA duplex is similar to a single bulged base insertion and the azobenzene moiety can be looked as the extra base. Although $\mathbf{F_{8X}}$ and $\mathbf{F_{12X}}$ are much longer than their complementary strand consisting of only natural nucleotides, it has been reported that this does not influence the stability of the formed duplex and the specificity of hybridization when azobenzenes take *trans* form [12]. The photoresponsive DNA tweezers are expected to be operated at a proper temperature range of 40~60 °C, which is between the T_m of *cis* form and *trans* form. The sequences in **A** were designed to hybridize with complementary sequences in **B** and **C** to form two 22-bp-long stiff duplex tweezers that are stable enough at the operation temperature. The four-base single-stranded central region of **A** is used as the hinge of the tweezers.

Firstly, we measured T_ms of $\mathbf{F_{8X}/F'_c}$, $\mathbf{F_{12X}/F'_c}$, and $\mathbf{F_n/F'_c}$ duplexes (with a bulge of two dTs) to evaluate the ability of $\mathbf{F_{8X}}$ and $\mathbf{F_{12X}}$ to photoregulate DNA hybridization (Table 1). Here, $\mathbf{F'_c}$ was used instead of the tweezers at opening state to avoid the influence from the hybridization of **A** and **B**, **C**, which has a T_m of 78.0 °C (Table 1). In both cases, the duplex involving *trans*-azobenzenes has a higher T_m than that of native duplex $\mathbf{F_n/F'_c}$, indicating that introduction of as many as 12 azobenzene moieties did not influence the hybridization ability of the modified oligonucleotide. From the difference in T_m between *trans* and *cis* form (ΔT_m), it can be concluded that $\mathbf{F_{12X}}$ (ΔT_m = 23.1 °C) has higher photoregulation ability than $\mathbf{F_{8X}}$. Thereafter, we mainly used $\mathbf{F_{12X}}$ to construct the photoresponsive DNA tweezers.

Table 1. Melting temperatures of the duplexes

Duplex	T_m / °C [a]		
	trans	*cis*	ΔT_m ($T_{m,\,trs}$-$T_{m,\,cis}$)
$\mathbf{F_{8X}/F'_c}$	76.4	57.4	19.0
$\mathbf{F_{12X}/F'_c}$	73.7	49.6	23.1
$\mathbf{F_n/F'_c}$		72.7	
A/(B,C) [a]		78.0	

[a] 1.0 μM DNA (or 0.5 μM for **A/(B,C)**), pH 6.5 (50 mM Na_2HPO_4), 1.0 M NaCl.

Evaluation of the Working Efficiency of DNA Tweezers Involving Azobenzene Moieties. The photoresponsive tweezers were prepared by mixing stoichiometric quantities of four strands, **A**, **B**, **C**, and **F** in a buffer (50 mM Na_2HPO_4, pH 6.5, 1.0 M NaCl) to a final concentration of 1.0 μM [8]. As shown in Fig. 2, the fluorescence of the solution consisting of **A**, **B**, and **C** (in the absence of **F**) was the strongest (red lines -·-·-), indicating that the tweezers were completely opened. When $\mathbf{F_n}$, the native DNA, was added at a temperature much lower than the T_m of the duplex formed from

F_n, **B** and **C**, the fluorescence became much lower (see solid lines in Fig. 2a and b), showing that the tweezers were closed. Here, a new peak at about 580 nm due to the FRET between **TET** and **TAMRA** appeared, indicating that the distance between **TET** and **TAMRA** became closer. However, the fluorescence spectra did not change much at lower temperatures probably due to the low FRET efficiency between **TET** and **TAMRA** even if the tweezers were completely closed (data not shown).

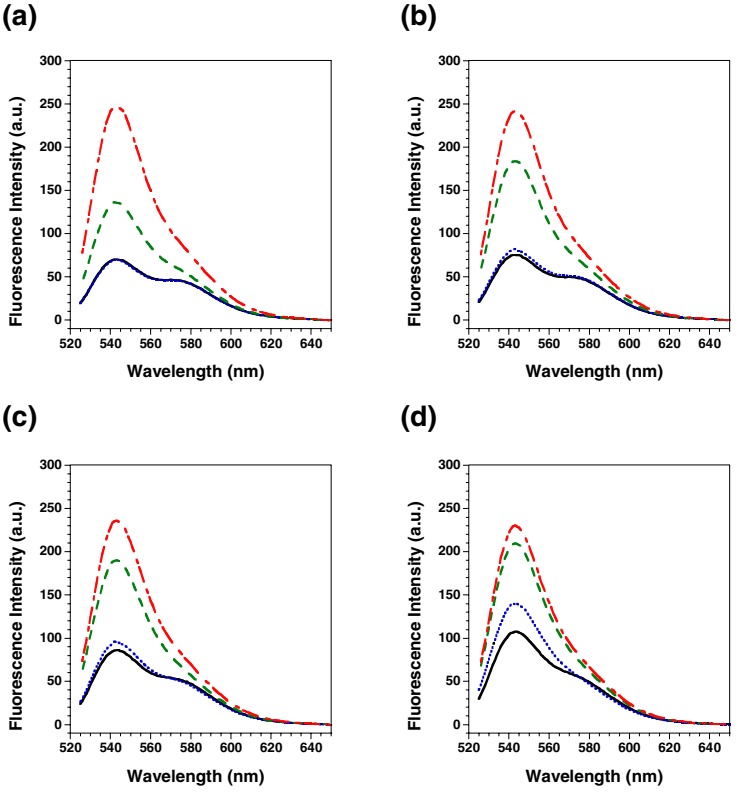


Fig. 2. Closing and opening of the DNA tweezers involving **F_{12X}** with light irradiation at 45 °C (**a**), 50 °C (**b**), 55 °C (**c**) and 60 °C (**d**), monitored by measuring fluorescence spectra. The tweezers are completely open when only strands **A**, **B** and **C** are present in the solution (red lines -·--·-). Upon the addition of strand **F_n**, the fluorescence drops and the tweezers prefer to close (solid lines —). The opening and closing are photoregulated to some extent with UV (green lines ----) and visible light (blue lines ····) irradiation, respectively.

The efficiency for the photoregulation of closing and opening of the tweezers with **F_{12X}** was estimated by measuring the change of fluorescence with the light irradiation at thermo-stable conditions. As shown in Fig. 2b, for example, when the solution consisting of **A**, **B**, **C** and **F_{12X}** was irradiated with visible light (440~460 nm) at 50 °C for 1 minute, the azobenzenes took *trans* form (>95%), and its fluorescence spectra (blue lines ····) was very close to that of the solution consisting of **A**, **B**, **C** and **F_n**

(closed tweezers). On the other hand, when it was irradiated with UV light (330~350 nm) for 5 minutes, the spectra (green lines ----) became close to that of the opening tweezers consisting of only **A**, **B**, and **C** (Fig. 2b). These results revealed that opening and closing of the tweezers were switched simply by light irradiation.

Photoswitched opening and closing of the tweezers were also attained at other temperatures, although the photoregulation efficiency depended greatly on the operation temperature (Fig. 2a, 2c, and 2d). The percentage of opening and closing states was calculated directly from the fluorescence intensity supposing that there is a linear correlation between fluorescence strength and the amount of tweezers of opening state. At 50 °C and 55 °C, about 70% of tweezers were opened after UV light irradiation, and more than 80% of tweezers were closed after visible light irradiation. At 45 °C, only about 40% of tweezers could be opened in the case of *cis* form; at 60 °C, however, the tweezers could not close efficiently in the case of *trans* form. Accordingly, the most efficient photoregulation could be achieved at 50-55 °C (Table 2). These results are consistent with the T_m values of corresponding duplexes: T_ms of duplex $\mathbf{F_{12X}/B(C)}$ in *trans* form and *cis* form are 59 °C and 35 °C, respectively (data not shown).

Table 2. Contents of opening tweezers in the presence $\mathbf{F_{12X}}$ after UV or visible light irradiation at various temperatures

Temperature (°C)	Percentage of the opening tweezers (%)		
	UV	Vis	Δ (UV-Vis)[a)]
45	38.7	2.0	36.6
50	67.0	9.7	57.3
55	73.3	17.6	55.6
60	87.3	45.2	42.1

[a)] Difference in contents of opening state between UV and visible light irradiation.

Another point one may concern is the irradiation time required for opening or closing the tweezers. In the case of visible light irradiation (430~450 nm, 90 mW/cm^2) at 50 °C, we found that more than 90% of the tweezers were closed after one minute. In the case of UV light irradiation (330~350 nm, 0.5 mW/cm^2), 4 minutes were enough to reach the equilibrium, at which about 65% of the tweezers were opened [17]. Thus, the efficient open-close photoswitching could be achieved with UV and visible light irradiation for 4 minutes and 1 minute, respectively. The strong stacking interaction between *trans*-azobenzene and base pairs can be considered as one of the reasons for the time-consuming *trans*-to-*cis* photoisomerization.

Repetitive Opening and Closing of the Photoresponsive DNA Tweezers with Light Irradiation. For a DNA nanomachine, one of the most important functions is whether it can operate successively without losing efficiency. Changes of fluorescence intensity were recorded while the photoresponsive tweezers were cycled ten times between the opening and closing states by alternating irradiation with UV and visible light at 50 °C. The opening and closing efficiency were calculated from the fluorescence change and shown in Fig. 3. For every cycle, about 90% of the tweezers were closed after visible light irradiation for 1 min, and about 65% of the

tweezers were opened after UV light irradiation for 4 minutes (Fig. 3). The cycling efficiency did not decrease at all after opening and closing of the tweezers for ten times. From this point, the photoresponsive tweezers we constructed here are much better than the original DNA-fueled tweezers, whose cycling efficiency decreased by about 40% after 7 cycles due to the successive addition of DNA fuels and the production of waste duplex (Fig. 1a) [8]. In our case, the photoswitching of DNA tweezers was based on the reversible photoisomerization of azobenzene, and no wastes were produced. Once the tweezers were constructed, the concentrations of all the oligonucleotides remained stable during the operation because no extra oligonucleotides were added. Therefore, the photoregulation efficiency does not change with the operation cycles as long as the DNAs involved are not destroyed. No detectable decomposition of the introduced azobenzene residues was observed under the light irradiation conditions we used, although a continuous irradiation with visible light for a long time (e.g. >30 min) was found to destroy the fluorophore **TET** (data not shown). Thus, the opening and closing of photoresponsive tweezers are expected to be photoswitched for lots of cycles without decreasing the working efficiency. In conclusion, an efficient photon-fueled molecular machine that could be repeatedly operated was constructed.

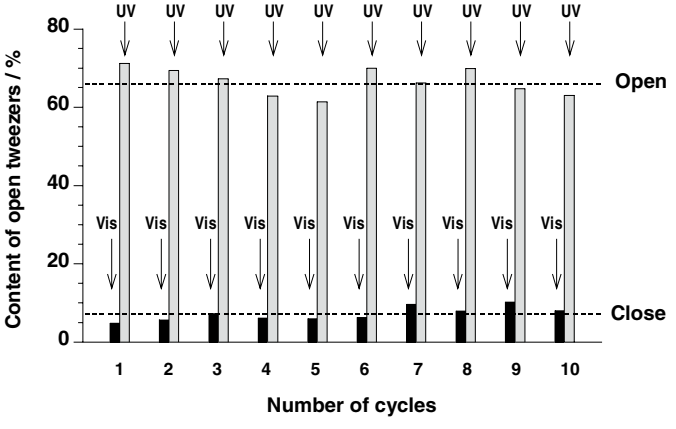


Fig. 3. Repeated opening and closing of photoresponsive molecular tweezers at 50 °C by alternating irradiation with visible light for 1 min and UV light for 4 min. About 55% of tweezers were photoswitched to be open and closed in each cycle.

2.2 Hairpin-Based Photoresponsive DNA Tweezers Involving *para*-Isopropyl-Substituted Azobenzenes

Molecular Design of Smart Photoresponsive DNA Tweezers Involving a Hairpin Structure. Here we aimed to construct more intelligent photon-fueled DNA tweezers by forming a hairpin structure on one of its arms (Fig. 4a). As compared with the tweezers shown in Fig. 1b, strand **B** and strand **C** were replaced by strand **H** and **R**, respectively. In the center part, strand **H** forms a hairpin structure with an 8-bp-long stem and a 4-nt-long loop. The sequence at 5′-side of the hairpin hybridizes with **A** to

form a 22-bp-long duplex, and its 3′-side was designed to be complementary to 5′-side of **R**. Hybridization of **R** with **H** closes the tweezers, and the dehybridization opens them. The photoswitches were introduced to 5′-side of **R**. Obviously, strand **F** is not required here, and no strand is removed from the machine during the operation. Therefore, this DNA machine is possible to work at a very low concentration once the duplex parts **A/R** and **A/H** are formed. Another merit of these novel tweezers is that only one arm can open and close the whole tweezers by photo-induced hybridization-dehybridization so that fewer azobenzenes are required to be introduced.

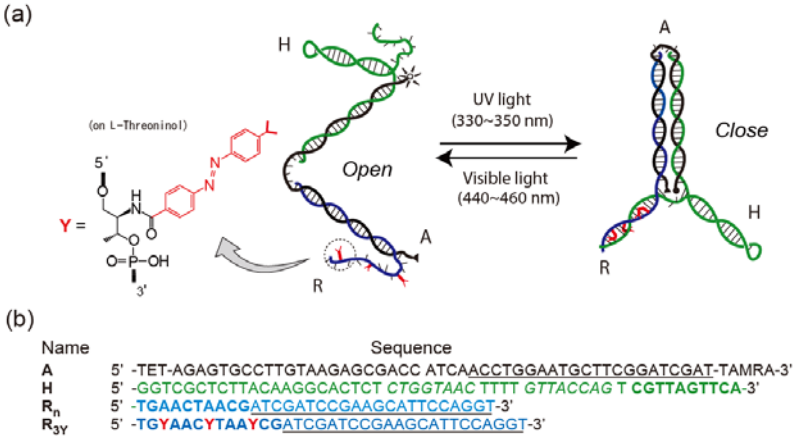


Fig. 4. Schematic illustration of photoresponsive DNA tweezers involving *para*-isopropyl azobenzene (***i*Pr-azo**) on L-threoninol (a) and corresponding oligonucleotide sequences used (b). Visible light irradiation (*trans* form) prefers to open the tweezers, and UV light irradiation (*cis* form) prefers to close them. The structure of azobenzene moiety (**Y** residue) is also show in (a). Underlined letters in **A** indicate the section that hybridizes with **R** ($\mathbf{R_n}$ or $\mathbf{R_{3Y}}$). Italic sections in **H** show the sequences that form hairpin structure with an 8-bp-long stem and a 4-nt-long loop. Complementary parts between **H** and **R** are shown in bold letters.

In contrast with the previously constructed tweezers (Fig. 1b), *para*-isopropyl-substituted azobenzene (***i*Pr-azo**) tethering on L-threoninol, a recently developed photoswitch of reverse type, is used: Visible light irradiation (*trans* form) opens the tweezers and UV light irradiation (*cis* form) closes them [18]. Note that the direction of photoswitch is reversed as compared with the unmodified azobenzene shown in Fig. 1. This photoregulation is based on the following mechanism: The *trans* form destabilizes the duplex formation due to the steric hindrance of bulky *para*-isopropyl group with DNA backbone; whereas the *cis* form prefers to form stable duplex due to groove binding of the hydrophobic *para*-isopropyl group [18]. For decreasing the number of photoswitches, duplex region formed between **R** and **H** was designed as short as 10-bp-long with three ***i*Pr-azo** at **R** strand.

Open and Close DNA Tweezers Involving para-Isopropyl-Substituted Azobenzenes with Light Irradiation. The photoresponsive tweezers were prepared by mixing stoichiometric quantities of three strands, **A**, **H**, and **R** ($\mathbf{R_n}$ or $\mathbf{R_{3Y}}$) in the buffer. As the

control of opening tweezers, strand **C** (see the sequence in Fig. 1c) was first used instead of **H**, and the fluorescence was strongest when **A**, **C**, and $\mathbf{R}_n$ were present in the buffer (red line -·-·- in Fig. 5). We also tried to use **A** and **H** in the absence of **R** as a control of opening state, however, the fluorescence became lower because non-negligible FRET occurred due to the low stiffness of single-stranded part at 3′-side of **A** [8]. When the native DNA $\mathbf{R}_n$ was added to a solution containing **A** and **H** at 10 °C, the fluorescence became much lower, indicating that the tweezers were closed due to the duplex formation between $\mathbf{R}_n$ and **H** (black line in Fig. 5). More than 90% of the tweezers were closed even at 35 °C, and the tweezers were completely open at the temperature higher than 70 °C (data not shown). After $\mathbf{R}_{3Y}$ (with 1 min of visible light-irradiation) was added instead of $\mathbf{R}_n$ at the same temperature, the fluorescence was close to that of the opened state indicating that most of the tweezers were opened because ***i*Pr-azo** took *trans* form (see blue line ····· in Fig. 5). On the contrary, the fluorescence became lower after 5 min of UV light irradiation, although the fluorescence was still much higher than that of the completely closed state (compare blue line with green line ---- in Fig. 5). It can be estimated that about 13% of the tweezers were photoregulated to open and close. This lower efficiency of closing the tweezers may be attributed to the insufficient stabilization effect in *cis* form as we reported recently [18]. When visible light was irradiated for 1 min at this point, the fluorescence increased and the closed tweezers opened again. Note that the operation of the tweezers was reversed as compared with the tweezers shown in Fig. 1b: UV light irradiation closed the tweezers and visible light irradiation opened them.

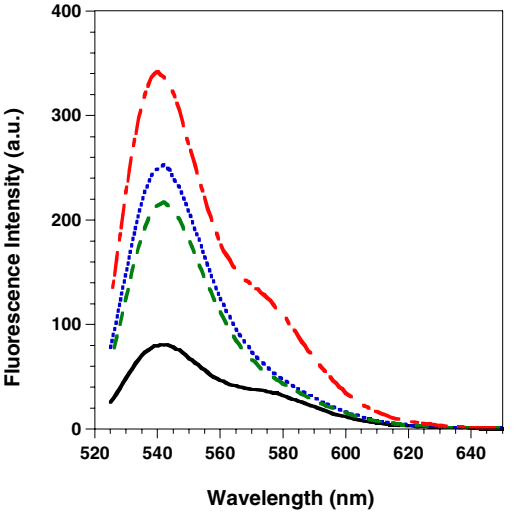


Fig. 5. Change of fluorescence intensity from the DNA tweezers involving $\mathbf{R}_{3Y}$ with light irradiation. The tweezers are completely open when strands **A**, **B**, and $\mathbf{R}_n$ are present in the solution (red line -·-·-). When strand **H** is present instead of **B**, the fluorescence drops and the tweezers are closed (solid line —). When $\mathbf{R}_{3Y}$ is used, the opening and closing are photoregulated to some extent with visible light (blue line ·····) and UV light irradiation (green line ----), respectively. All the photoirradiation and fluorescence measurements were carried out at 10 °C. The concentration of each DNA strand is 0.1 μM.

The tweezers involving *para*-isopropyl-substituted azobenzenes could also be regulated by light irradiation at an extremely low concentration. Since hybridization between **R** and **H** in the DNA tweezers can be regarded as an intra-molecular interaction, the efficiency did not change much when the concentration of each strand varied from 1 μM to 10 nM (data not shown). The low concentration is also helpful for avoiding the dimer formation as Yurke et al. pointed out [8]. From this point, this novel DNA tweezers can be considered as an indeed nanomachine which can work on a single supramolecular level.

3 Conclusions

The DNA tweezers powered by photo-irradiation were constructed by introducing photoresponsive molecules. These nanomachines could be operated successively without decreasing of the working efficiency. By using different photoswitches, the opening and closing of the tweezers can be carried out by using either UV or visible light irradiation. The constructed photoresponsive molecular machines are 'environment-friendly' without producing any waste because they use photons, one of the cleanest energy sources, as the fuel.

The use of DNA-based machines plays an important role in the rapidly developing field of nanobiotechnology. The valuable and promising applications of these nanomachines have bright future perspectives in the development of sensors, molecular transporters, and controlled drug delivery systems other than only satisfying our intellectual scientific curiosity. Our novel concept to operate the DNA-based machine with light irradiation should have an impressive impact in this field and highlight its revolutionary progress for operating molecule machines more freely and precisely. The introduction of azobenzenes to the DNA machine as molecular engines makes it possible to transfer light energy first to chemical one and then to mechanical motion. The use of our photoresponsive molecular machines also opens a new possibility to use them *in vivo*, especially for the DNA tweezers involving hairpin structure. For these purposes, further studies such as improving the operation efficiency and lowering the number of molecular engines by using modified azobenzenes, and improving the thermal stability of *cis*-azobenzene are underway.

4 Experimental Section

Materials. The oligonucleotides involving non-substituted azobenzene residues were supplied by Nihon Techno Service Co., Ltd. (Tsukuba, Japan), and purified by polyacrylamide gel electrophoresis. The oligonucleotides involving *para*-isopropyl azobenzene residues (***i*Pr-azo)** were synthesized as previously reported [18]. The oligonucleotides consisting of only native bases and strand **A** involving **TET** and **TAMRA** were supplied by Integrated DNA Technologies, Inc. (Coralville, USA). Concentrations of all oligonucleotides were determined by UV-Vis spectroscopy analysis within an error margin of 10%. The molecular extinction coefficient (ε) of an azobenzene residue at 260 nm is 1.095×10^4 mol L^{-1} cm^{-1}.

Fluorescence Measurement for Monitoring the Operation of DNA Tweezers. The photoresponsive tweezers were prepared by mixing stoichiometric quantities of DNA

strands in SPSC buffer (50 mM Na_2HPO_4, pH 6.5, 1.0 M NaCl). The solution was added to a 3 mm-square quartz cuvette, and the fluorescence spectra of **TET** were measured with a JASCO FP-6500 fluorescence spectrometer (JASCO, Tokyo, Japan) excited at 514.5 nm. For calculating the content of opened and closed state of the tweezers tethering non-substituted azobenzenes, the fluorescence intensity at 543 nm was used (65.0 a.u. for completely closed state and 242.1 a.u. for completely opened state). The photo-irradiation was carried out with a Xenon light source (MAX-301, Asahi Spectra Co. Ltd, Tokyo, Japan) equipped with an interference filter (9 nm of bandpass) centered at 341.5 nm for UV light irradiation (0.5 mW/cm^2) and an interference filter (9 nm of bandpass) centered at 449.5 nm for visible light irradiation (90 mW/cm^2). During the irradiation, the cuvette containing the tweezers was put in a water bath keeping at a fixed temperature (±2 °C). After the irradiation, the cuvette was immediately moved into the fluorescence spectrometer that was kept at the same temperature to measure the fluorescence. During the fluorescence measurement, further UV or visible light irradiation was not carried out.

Measurement of Melting Temperatures. All the T_ms were measured in SPSC buffer (50 mM Na_2HPO_4, pH 6.5, 1.0 M NaCl). For duplex **F/F′$_c$**, solutions consisting of 1.0 μM of **F′$_c$** and **F$_n$**, **F$_{8X}$**, or **F$_{12x}$** were used. For measuring the T_ms of DNA solution consisting of strands **A**, **B**, **C**, and **F**, a final concentration of 0.5 μM for each strand was used. After keeping at 90 °C for 3 min, the solution was gradually cooled to 10 °C (1.0 °C/min). Melting curves were obtained at a heating rate of 1.0 °C/min. Peak temperatures in the derivative curves (dA/dT) were designated as T_ms. UV spectra and UV-melting profiles (260 nm) were recorded by a JASCO model V-530 spectrometer equipped with a programmable temperature-controller.

Acknowledgments

This work was supported by Core Research for Evolution Science and Technology (CREST), Japan Science and Technology Agency (JST). Partial support was provided by a Grant-in-Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology, Japan, and The Mitsubishi Foundation (for H. A.) is also acknowledged.

References

1. Seeman, N.C., Lukeman, P.S.: Nucleic Acid Nanostructures: Bottom-Up Control of Geometry on the Nanoscale. Rep. Prog. Phys. 68, 237–270 (2005)
2. Shih, W.M., Quispe, J.D., Joyce, G.F.: A 1.7-Kilobase Single-Stranded DNA that Folds into a Nanoscale Octahedron. Nature 427, 618–621 (2004)
3. Mirkin, C.A., Letsinger, R.L., Mucic, R.C., Storhoff, J.J.: A DNA-Based Method for Rationally Assembling Nanoparticles into Macroscopic Materials. Nature 382, 607–609 (1996)
4. Chen, J.H., Seeman, N.C.: Synthesis from DNA of a Molecule with the Connectivity of a Cube. Nature 350, 631–633 (1991)
5. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and Autonomous Computing Machine Made of Biomolecules. Nature 414, 430–434 (2001)

6. Seeman, N.C.: From Genes to Machines: DNA Nanomechanical Devices. Trends Biochem. Sci. 30, 119–125 (2005)
7. Beissenhirtz, M.K., Willner, I.: DNA-Based Machines. Org. Biomol. Chem. 4, 3392–3401 (2006)
8. Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: A DNA-Fuelled Molecular Machine Made of DNA. Nature 406, 605–608 (2000)
9. Beyer, S., Simmel, F.C.: A Modular DNA Signal Translator for the Controlled Release of a Protein by an Aptamer. Nucleic Acid Res. 34, 1581–1587 (2006)
10. Shin, J.S., Pierce, N.A.: A Synthetic DNA Walker for Molecular Transport. J. Am. Chem. Soc. 126, 10834–10835 (2004)
11. Tian, Y., Mao, C.D.: Molecular Gears: A Pair of DNA Circles Continuously Rolls against Each Other. J. Am. Chem. Soc. 126, 11410–11411 (2004)
12. Asanuma, H., Liang, X.G., Nishioka, H., Matsunaga, D., Liu, M.Z., Komiyama, M.: Synthesis of Azobenzene-Tethered DNA for Reversible Photo-Regulation of DNA Functions: Hybridization and Transcription. Nat. Protocols 2, 203–212 (2007)
13. Asanuma, H., Ito, T., Yoshida, T., Liang, X.G., Komiyama, M.: Photoregulation of the Formation and Dissociation of a DNA Duplex by Using the cis-trans Isomerization of Azobenzene. Angew. Chem. Int. Ed. 38, 2393–2395 (1999)
14. Asanuma, H., Takarada, T., Yoshida, T., Liang, X.G., Komiyama, M.: Enantioselective Incorporation of Azobenzene into Oligodeoxyribonucleotide for Effective Photoregulation of Duplex Formation. Angew. Chem. Int. Ed. Engl. 40, 2671–2673 (2001)
15. Asanuma, H., Liang, X.G., Yoshida, T., Komiyama, M.: Photocontrol of DNA Duplex Formation by Using Azobenzene-Bearing Oligonucleotides. Chem. BioChem. 2, 39–44 (2001)
16. Liang, X.G., Asanuma, H., Kashida, H., Takasu, A., Sakamoto, T., Kawai, G., Komiyama, M.: NMR Study on the Photoresponsive DNA Tethering an Azobenzene: Assignment of the Absolute Configuration of Two Diastereomers and Structure Determination of Their Duplex in the trans-Form. J. Am. Chem. Soc. 125, 16408–16415 (2003)
17. Liang, X.G., Nishioka, H., Takenaka, N., Asanuma, H.: A DNA Nanomachine Powered by Light Irradiation. Chem BioChem. 9, 702–705 (2008)
18. Liang, X.G., Takenaka, N., Nishioka, H., Asanuma, H.: Molecular Design for Reversing the Photoswiching Mode of Turning ON and OFF DNA Hybridization. Chem. Asian J. 3, 553–560 (2008)

Operon Structure Optimization by Random Self-assembly

Yusuke Nakagawa[1], Katsuyuki Yugi[1], Kenji Tsuge[2], Mitsuhiro Itaya[2], Hiroshi Yanagawa[1], and Yasubumi Sakakibara[1,3]

[1] Department of Biosciences and Informatics, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522, Japan
yasu@bio.keio.ac.jp
[2] Institute for Advanced Biosciences, Keio University, 403-1 Daihoji, Tsuruoka, Yamagata, 997-0017, Japan
[3] Institute for Bioinformatics Research and Development (BIRD), Japan Science and Technology Agency (JST)

Abstract. Synthetic biology is an emerging research area that aims to investigate natural biological phenomena and reconstruct complex artificial biological systems. Recent development of genetic engineering such as multiple gene assembly method accelerates the synthetic biology study. Ordered gene assembly in *Bacillus subtilis* (OGAB method) is to assemble multiple genes in one step using an intrinsic *B.subtilis* plasmid transformation system and enables to reconstitute sets of relevant genes. The OGAB method assembles multiple DNA fragments with a fixed order and orientation and constructs an operon structure in a resultant plasmid. However, the optimal order and orientation to reconstitute a set of genes are generally not trivial and depends on several factors in host bacteria, where the "optimal" means the efficiency of biosynthesis induced by transfered genes in a metabolic pathway. We propose a method to apply self-assembly technique to optimization problem of operon structure. Self-assembly of multiple genes generates all possible orders of genes on operon structure. The number of generated orders on operon structure becomes the factorial of the number of multiple genes. All generated orders of multiple genes are then introduced into *E.coli* cells and most prominent colony for biosynthesis is extracted. We show some preliminary experiment to construct more efficient orders for five genes in the carotenoid biosynthetic pathway, and found a new order that is more efficient than previous studies for gene order.

1 Introduction

Assembling a number of relevant genes into one plasmid is an essential technique for synthetic biology study to reconstruct biosynthesis process engaged by multiple genes in a metabolic pathway. A classical recombinant protocol for multiple gene assembly is to progressively introduce each gene one by one. A novel gene assembly method, OGAB developed by Tsuge et al. [1], offers one-step assembly of multiple DNA fragments with high efficiency. This one-step assembly

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 33–40, 2009.

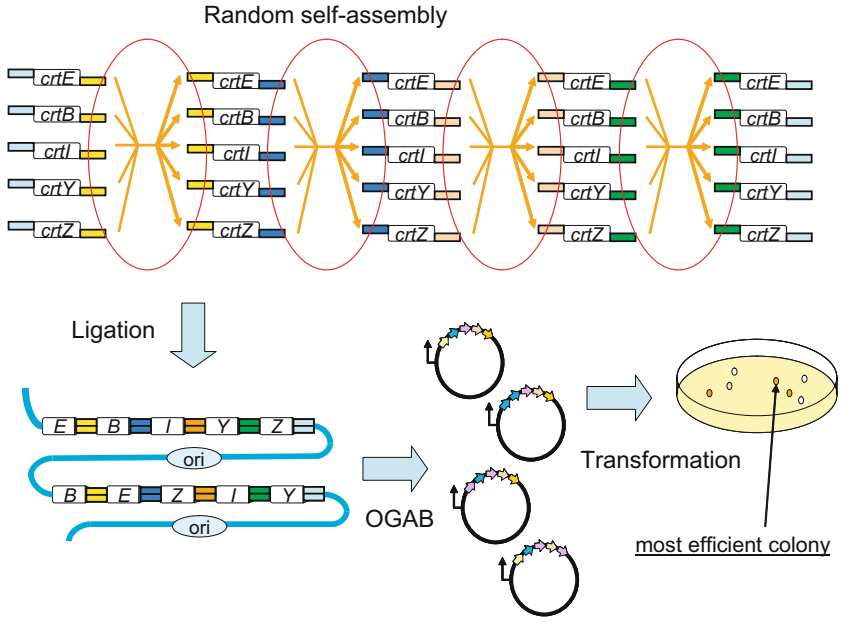


Fig. 1. Optimization by random self-assembly and OGAB method for five genes in carotenoid biosynthesis

feature enables applications of recently advanced and more algorithmic methods developed in DNA-based computing.

On the other hand, engineering of carotenoid biosynthesis transformed in *E.coli*, conducted by Nishizaki et al. [2], is a typical application of OGAB method. They found that the order of biosynthesis genes on plasmid affects the final amount of zeaxanthin synthesized in transformed *E.coli*. The production efficiency of metabolic biosynthesis generally depends on several factors such as activity of biosynthetic enzymes, mRNA expression levels of biosynthesis genes, and growth rate of host bacteria *E.coli*. Previous studies reveal that these factors are significantly affected and changed by the order of genes on operon structure of plasmid. For example, it is observed that the rank order of mRNA expression levels in each plasmid corresponded to the sequential order of those genes of carotenoid biosynthesis on the operon structure [2]. Thus, Nishizaki et al. investigated only five circularly permuted orders of biosynthesis genes among all possible orders. The number of all possible orders of five biosynthesis genes becomes the factorial of the number of multiple genes, for example, $5! = 120$ orders for five genes, so that it is not effective to examine all combinations manually. Our main contribution in this paper is to propose a method to apply DNA computing method, that is, a random self-assembly technique to optimization problem of the order of biosynthesis genes.

In the theory of DNA computing, the models of computation based on *self-assembly* are well recognized to be of great importance in that they can provide one of the most basic frameworks for DNA-based computing paradigms [3,4].

In fact, Adleman's ground-breaking experimental work ([3]) that solved a small instance of the Hamiltonian Path Problem (HPP) is a typical example of DNA computation based on the self-assembly principle. A most important feature of self-assembly computation is that random self-assembly of well-encoded DNA fragments generates an exponential number of all possible combinations of the fragments which correspond to all candidates of solutions and efficient extraction of an optimal solution among the pool solves computationally intractable (NP-hard) problems. Our strategy to apply DNA computing method to solving optimization problem to determine the order of biosynthesis genes consists of three steps: (i) random self-assemblies of DNA fragments for biosynthesis genes with well-designed sticky ends, (ii) one-step assembly of multiple DNA fragments using OGAB method to construct plasmid vectors with various orders of self-assembled genes to generate all possible orders, and (iii) transformation of *E.coli* cells with various plasmid vectors and extraction of most prominent colony for biosynthesis from plate (See Figure 1).

2 Methods

2.1 Random Self-assembly of Operon Structure

A general schema of self-assembly computation model is outlined as follows:

1. design a finite set of basic units for assembly computation,
2. put all those basic units with sufficiently high concentration into one test tube, to create a random pool of shuffled self-assemblies,
3. perform screening mechanism to extract only necessary assembly of basic units,
4. detect whether or not there is an assembly with desired constraints.

Our aim to use self-assembly computation is to generate various operon structures with randomly shuffled orders of biosynthetic genes. Operon structure on plasmids (genome) is generally defined as one transcriptional unit consisting of a sequence of ribosome-binding site (RBS) and open reading frames (ORFs) following one promoter, where an open reading frame is composed of start codon, gene-coding region, and stop codon (See Figure 2).

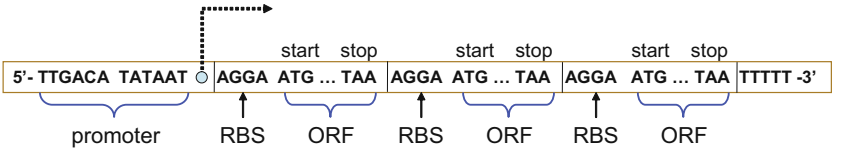


Fig. 2. An operon structure

In order to apply self-assembly computation to randomly generating operon structures, a basic unit corresponds to a RBS and a double-stranded ORF with both sticky ends. The type II restriction enzyme DraIII that recognizes two

separate recognition sites, 5'-CACNNNGTG-3', is employed so that various different sticky ends of three bases can be designed, where N represents any base (A, C, G, or T). DraIII cleaves a double-stranded DNA as follows:

$$5' - \texttt{CACNNN} \uparrow \texttt{GTG} - 3'$$
$$3' - \texttt{GTG} \uparrow \texttt{NNNCAC} - 5'$$

where CAC and GTG are two recognition sites and ↑ represents cleavage site. To implement self-assembly graph shown in Figure 3 (above) to generate randomly shuffled orders of five genes in carotenoid biosynthesis, we designed each sticky end as 3'-ACT for node 1, 3'-GTT for node 2, 3'-ATG for node 3, 3'-CTA for node 4, 3'-TGA for node 5 (as shown in Figure 3 (below)).

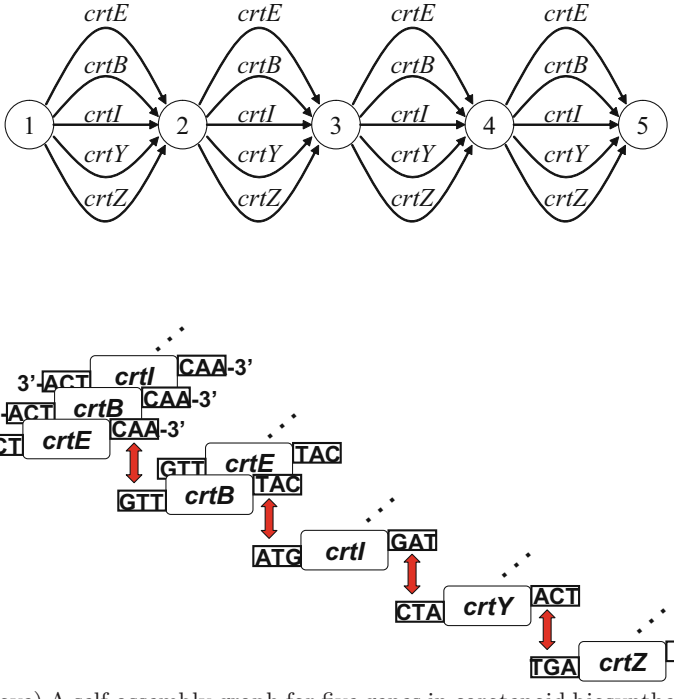


Fig. 3. (Above) A self-assembly graph for five genes in carotenoid biosynthesis. (Below) Basic units with well-designed sticky ends.

2.2 Reconstitution of Multiple Genes by OGAB Method

A novel gene assembly method [1], called OGAB method, was developed to reconstitute sets of genes with one-step assembly of multiple DNA fragments with high efficiency. The key principle of OGAB method is based on unique mechanisms of *B.subtilis* for DNA uptake and plasmid establishment. The *B.subtilis* mechanism circularizes linearized plasmids when homologous sequences are present. That is, the highly recombinogenic single-stranded DNAs form double-stranded DNA by pairing. Then, the partial double-stranded molecule is repaired to give an intact double-stranded circular DNA that starts replication as a plasmid [1,5].

We apply OGAB method to constructing circular plasmids with randomly shuffled orders of five carotenoid biosynthetic genes. Five fragments *crtE*, *crtB*, *crtI*, *crtY*, and *crtZ* are ligated to form high-molecular-weight DNA with a tandem repeat unit. According to the random self-assembly, the order and orientation of these five fragments are randomly shuffled. The linear DNA molecule with more than two tandemly aligned units at *ori*-subsequence transforms *B.subtilis*, and preferentially establishes the plasmid, termed pGETS109RAND, *in vivo*. Note that we employ the plasmid pGETS109 of *E.coli-B.subtilis* shuttling vector constructed in the previous study [1].

2.3 Transformation of *E.coli* Cell and Extraction of Best Colony

Constructed plasmid vectors with various orders of self-assembled genes are transfered into *E.coli* by electroporation. These transformed *E.coli* are cultivated on LB plate and then various colonies are grown. Then, our important strategy for detection of optimal solution is that most prominent colony with exhibiting stronger color is extracted. This strategy also contributes to filtering out operons with repeated same genes and hence incomplete set of five genes.

3 Experiments

We applied our random self-assembly method to searching optimal order of five genes *crtE*, *crtB*, *crtI*, *crtY*, *crtZ* for synthesis of zeaxanthin in carotenoid biosynthetic pathway. In the following preliminary experiment, we tested only the first two genes, *crtE* and *crtB*, to see the feasibility of our random self-assembly method.

3.1 Carotenoid Biosynthetic Pathway

Carotenoids are found in bacteria, fungi, and higher plants, and act as photoprotecting agents. Improvements of production of carotenoids were made by transforming bacteria with gene clusters encoding carotenoid biosynthetic genes. The five biosynthetic genes, *crtE*, *crtB*, *crtI*, *crtY*, and *crtZ*, were extracted from the natural carotenoid cluster of *Pantoea ananatis* [2,6]. Actually, these five genes were amplified from the plasmid pCAR25 which was originally constructed by Misawa [6]. (See also Table 1 in Appendix to design appropriate primers for random self-assembly.)

3.2 Random Self-assembly for Carotenoid Biosynthetic Gene Operon

Each random ligation of the first two genes, *crtE* and *crtB*, followed by the fixed order of three genes *crtI*, *crtY*, and *crtZ* for five carotenoid biosynthetic genes was assembled as one transcriptional unit (plasmid) in a polycistronic manner by OGAB method. Therefore, four different plasmids had to be constructed, that is, *crtE-crtB-crtI-crtY-crtZ* denoted by pCrtP-EBIYZ, *crtB-crtE-crtI-crtY-crtZ* denoted by pCrtP-BEIYZ, *crtB-crtB-crtI-crtY-crtZ* denoted by pCrtP-BBIYZ, *crtE-crtE-crtI-crtY-crtZ* denoted by pCrtP-EEIYZ. Since the

third plasmid pCrtP-BBIYZ and the fourth pCrtP-EEIYZ do not contain the complete set of carotenoid biosynthetic genes, these transformants will fail to synthesize and produce zeaxanthin.

Next, these constructed plasmids were transferred into *E.coli* to assay the production of zeaxanthin. These transformants were then cultivated for 48 hours on LB plates at 37°C. The growth of various colonies was observed, and two distinct colonies that significantly exhibit colors were extracted as shown in Figure 4 (right). Each plasmid from two selected colonies was digested by the restriction enzyme

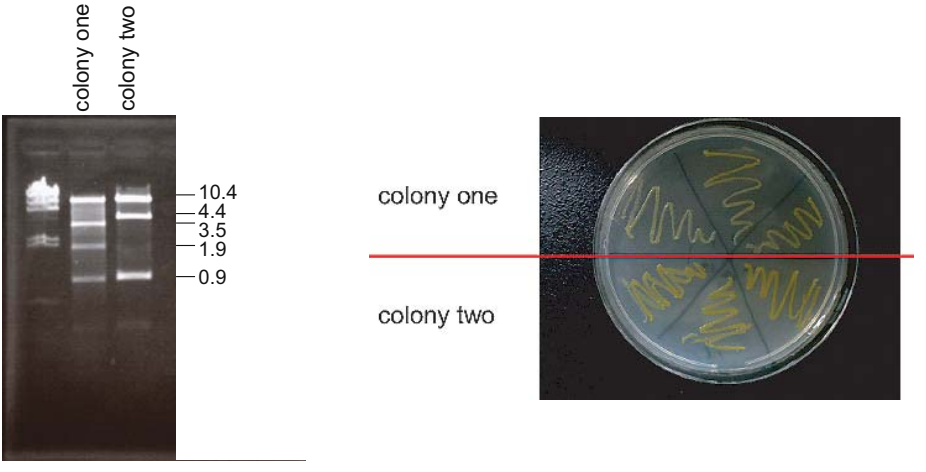


Fig. 4. (Left) The ligation products were confirmed by restriction mapping. The second lane from the left corresponds to the plasmid pCrtP-EBIYZ and the third lane corresponds to pCrtP-BEIYZ. (Right) Two colonies that significantly exhibit colors were extracted and re-cultivated.

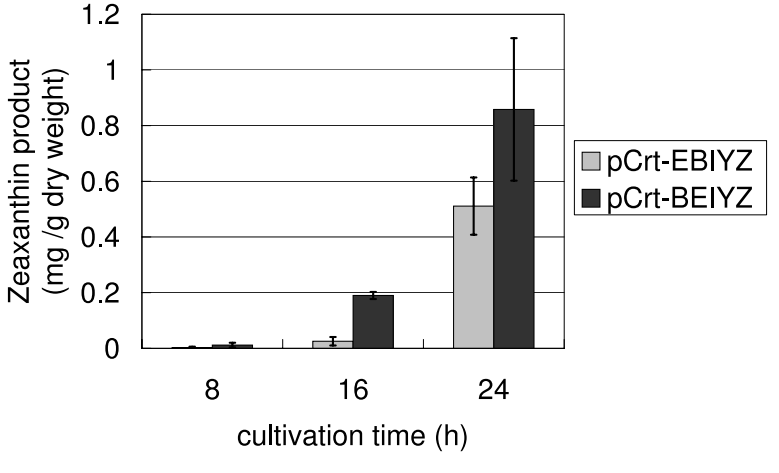


Fig. 5. HPLC result of Zeaxanthin production

BglI and the resultant fragments were purified by agarose gel electrophoresis and confirmed by restriction mapping. The result is shown in Figure 4 (left). The second lane from the left in Figure 4 (left) is identical to the plasmid pCrtP-EBIYZ and the third lane is identical to pCrtP-BEIYZ. Therefore, we confirmed that our self-assembly to generate random operons was succeeded.

3.3 Extraction of Prominent Colony for Carotenoid Biosynthesis

The zeaxanthin production in the *E.coli* transformants with each of two pCrtP plasmids was quantified using high-performance liquid chromatography (HPLC). As shown in Figure 5, the higher zeaxanthin content of mg/g (dry weight) was obtained in the transformant with pCrtP-BEIYZ. More concretely, synthesis of zeaxanthin with pCrtP-BEIYZ plasmid was 1.7-fold higher than that of zeaxanthin with pCrtP-EBIYZ for 24h cultivation. This order of pCrtP-BEIYZ is different from the order of (P_{crtE}-*crtE*-*crtB*-*crtI*-*crtY*-*crtZ*) that corresponds to the zeaxanthin biosynthetic pathway and was previously found as most efficient order [2].

4 Conclusion

We proposed a method to apply self-assembly technique to optimization problem of operon structure. Our strategy consists of three steps: (i) random self-assemblies of DNA fragments for biosynthesis genes with well-designed sticky ends, (ii) one-step assembly of multiple DNA fragments using OGAB method to construct plasmid vectors with different orders of self-assembled genes to generate all possible orders, and (iii) transformation of *E.coli* cells with various plasmid vectors and extraction of most prominent colony for biosynthesis from plate. We presented a preliminary experiment of our method for optimizing operon structures of five genes in carotenoid biosynthesis, showed that our method is effective to improve the efficiency of zeaxanthin production, and found a new order that is more efficient than previous studies for gene order. Since our preliminary experiment is only on shuffling the first two genes among five carotenoid biosynthetic genes, we are now conducting random shuffling orders of all five genes.

Acknowledgements

This work is supported in part by Grant program for bioinformatics research and development of Japan Science and Technology Agency, and Grant-in-Aid for Scientific Research on Priority Area “Comparative Genomics” No. 17018029 from the Ministry of Education, Culture, Sports, Science and Technology of Japan. This work was also performed in part through Grant-in-Aid for Young Scientists (Start-up) No. 18800046, Japan Society for the Promotion of Science (JSPS), grant of Keio Leading-edge Laboratory of Science and Technology (KLL) specified research projects and grant of Keio Engineering Foundation.

References

1. Tsuge, K., Matsui, K., Itaya, M.: One step assembly of multiple dna fragments with a designed order and orientation in *Bacillus subtilis* plasmid. Nucleic Acids Research 31, 133 (2003)

2. Nishizaki, T., Tsuge, K., Itaya, M., Doi, N., Yanagawa, H.: Metabolic engineering of carotenoid biosynthesis in *Escherichia coli* by ordered gene assembly in *Bacillus subtilis*. Applied and Environmental Microbiology 73, 1355–1361 (2007)
3. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024 (1994)
4. Yokomori, T., Sakakibara, Y., Kobayashi, S.: A magic pot: Self-assembly computation revisited. In: Brauer, W., Ehrig, H., Karhumäki, J., Salomaa, A. (eds.) Formal and Natural Computing. LNCS, vol. 2300, pp. 418–429. Springer, Heidelberg (2002)
5. Tsuge, K., Itaya, M.: Recombinational transfer of 100-kilobase genomic DNA to plasmid in *Bacillus subtilis* 168. Journal of Bacteriology 183, 5453–5458 (2001)
6. Misawa, N., Nakagawa, M., Kobayashi, K., Yamano, S., Izawa, Y., Nakamura, K., Harashima, K.: Elucidation of the *Erwinia uredovora* carotenoid biosynthetic pathway by functional analysis of gene products expressed in *Escherichia coli*. Journal of Bacteriology 172, 6704–6712 (1990)

Appendix

Table 1. Oligonucleotide sequences used as primers

Primer	Sequence
*crt*E-1st-F	5'- tagcacACTgtgttataaggacagcccga*atg*ac -3'
*crt*E-2nd-F	5'- tagcacACTgtgttataaggacagcccga*atg*ac -3'
*crt*E-3rd-F	5'- tagcacACTgtgttataaggacagcccga*atg*ac -3'
*crt*E-1st-R	5'- tagcacAACgtg*tta*actgacggcagcgag-3'
*crt*E-2nd-R	5'- tagcacAACgtg*tta*actgacggcagcgag-3'
*crt*E-3rd-R	5'- tagcacAACgtg*tta*actgacggcagcgag-3'
*crt*B-1st-F	5'- tagcacGTTgtgatgctggaggatctgat*atg*aa -3'
*crt*B-2nd-F	5'- tagcacGTTgtgatgctggaggatctgat*atg*aa -3'
*crt*B-3rd-F	5'- tagcacGTTgtgatgctggaggatctgat*atg*aa -3'
*crt*B-1st-R	5'- tagcacCATgtg*cta*gagcgggcgc-3'
*crt*B-2nd-R	5'- tagcacCATgtg*cta*gagcgggcgc-3'
*crt*B-3rd-R	5'- tagcacCATgtg*cta*gagcgggcgc-3'
*crt*I-1st-F	5'- tagcacATGgtgcgttaaagagcgactac*atg*aaac -3'
*crt*I-2nd-F	5'- tagcacATGgtgcgttaaagagcgactac*atg*aaac -3'
*crt*I-3rd-F	5'- tagcacATGgtgcgttaaagagcgactac*atg*aaac -3'
*crt*I-1st-R	5'- tagcacTAGgtg*tca*tatcagatcctccagca-3'
*crt*I-2nd-R	5'- tagcacTAGgtg*tca*tatcagatcctccagca-3'
*crt*I-3rd-R	5'- tagcacTAGgtg*tca*tatcagatcctccagca-3'
*crt*Y-4th-F	5'- tagcacCTAgtgcttaagtgggagcggct*atg* -3'
*crt*Y-4th-R	5'- tagcacTCAgtg*tta*acgatgagtcgtcataatggc-3'
*crt*Z-5th-F	5'- tagcacTGAgtgtctctaccggagaaatt*atg*ttgtg -3'
*crt*Z-5th-F	5'- tagcacAGAgtg*tta*cttcccggatgcgg-3'

Underlined nucleotides are DraIII recognition sites, capital-letter nucleotides indicate sticky ends generated by DraIII digestion, and the start and stop codons (complementary sequences) are indicated as italic letters.

Isothermal Reactivating Whiplash PCR for Locally Programmable Molecular Computation

John H. Reif and Urmi Majumder

Department of Computer Science,
Duke University, Durham, NC, USA
{reif,urmim}@cs.duke.edu

Abstract. Whiplash PCR (WPCR), due to Hagiya *et al.* [1], is a novel technique for autonomous molecular computation where a state machine is implemented with a single stranded DNA molecule and state transition is driven by polymerase and thermal cycles. The significance of WPCR computation lies in the fact that while other forms of autonomous molecular computing such as tiling assembly [2] or Benenson automata [3] operate based on global rules, it is possible to execute multiple WPCR machines, each holding its own distinct program, in parallel. However, since each transition requires a thermal cycle, multi-step WPCR machines are laborious and time-consuming. Hence they limit program execution to only a few steps. To date, no WPCR protocol has been developed which is *both* autocatalytic (self-executing) and isothermal (with no change in temperature). Here we describe such a protocol for computing with WPCR which uses a combination of strand displacement and DNA polymerization. Our designs include (1) a protocol where transition rules cannot be reused in subsequent computing, a feature that is crucial for reducing back-hybridization (2) a protocol where rules can be reused using an auxiliary strand displacement event, (3) a reusable rule protocol that prevents back-hybridization [1]. We also compute its state transition likelihood and rate and present a DNA sequence design of a 3-state machine and an experimental verification plan.

1 Introduction

1.1 Need for an Autocatalytic and Isothermal Protocol for WPCR

A primary challenge in nanoscience is the design of synthetic molecular devices that run autonomously (meaning that the program executes without any external mediation over multiple work cycles) and programmable (meaning that the machine's behavior can be modified without completely redesigning the system). In the last few years, the idea of constructing complex devices at the molecular scale using synthetic materials, such as synthetic DNA, has gone from theoretical concept to experimental reality. One such autonomous molecular computing device is called a Whiplash PCR (WPCR) machine [1], equivalent to a restricted class of finite automata. In this machine, the current state is encoded at the $3'$ end of a DNA single strand while the remainder of the strand encodes the

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 41–57, 2009.

state transition rules. The machine works as follows: using appropriate thermal cycles, the current state anneals to the correct transition rule and, next, a polymerase extends the $3'$ end of the strand to copy the next state from the encoded transition rule. The only limitation of this system is that it can execute only a single step before it requires significant environmental changes (thermal cycles) before continuing the computation. This paper presents a protocol that converts a WPCR system into an autonomous computing device, thus eliminating the need for external mediation to enable further steps. This capability is important if we wish to use these machines outside fully equipped laboratories.

1.2 Importance of Locally Programmable Molecular Computation

Although existing autonomous molecular computing devices e.g. DNA based tiling assembly [2] and restriction enzyme based automata [3] are computationally quite powerful [4], they are not capable of executing distinct programs in parallel[1]. In contrast, many complex molecular mechanisms found in the cell are more flexible and can perform a diverse set of tasks simultaneously. Whiplash PCR [5,6,7,8,1,9] allows parallel execution because each machine holds its own program and thus multiple, distinct molecular programs can run simultaneously in the same reaction tube.

1.3 Previous Methods for WPCR Computing Devices and Their Limitations

Hagiya *et al.* first proposed and experimentally demonstrated (only for a limited number of steps) a WPCR machine [1]. It was not easy to increase the number of steps because of a phenomenon called *back-annealing* (also known as back-hybridization), where a hairpin with a longer double stranded (ds) DNA region is preferentially formed over one with a shorter ds-DNA region. Sakamoto *et al.* suggested a modified protocol [9] where successive transitions were carried on at the same temperature, thus preventing back-hybridization. They also proposed a protocol for preventing *out-of-frame annealing* which happens because portions of two adjacent sequences constitute the sequences complimentary to a third state sequence. However, this work did not significantly increase the number of steps the WPCR machine could execute before stalling at an intermediate step. Later, Rose *et al.* proposed a scheme using targeted PNA_2/DNA triplex formation [7]. This triplex region destabilized the hairpin structure, thus preventing back-hybridization. Although this protocol can be isothermal, it is not autocatalytic. Further, it has been not yet been implemented and its role in increasing the number of steps a WPCR machine can execute before stalling is not clear. A recent paper by Rose *et al.* [10] proposed isothermal (no thermal cycling required) conditions for the functioning of a WPCR machine using strand displacement techniques. However, one still needs to add a rule protection strand

[1] Tiling assembly can be made to do multiple programs in parallel if we start with a universal cellular automata tile set with different seed rows. However, it is not very practical to generate such a large tile set.

after each polymerization step to drive the computation forward. Hence if we wish to overcome this problem and also allow flexibility of applications, we need to design an autocatalytic (a system that reactivates itself) and isothermal (no thermal cycles) protocol for WPCR that would allow us to operate the machine in places other than a laboratory.

1.4 Our Contribution

Our main contribution in this paper is the design of a WPCR machine that eliminates the need for any other external mediation (such as thermal cycles) and, thus, the resultant machine is both *isothermal* and *autocatalytic*. By isothermal we mean that the temperature of the reaction mixture is constant throughout computation. However, our design requires an additional preparation stage which precedes the computation stage and this step is not isothermal. This is also in contrast with the definition of isothermal by Sakamoto *et al.* [9] who defined isothermal as a design where the denaturation of the previous state and annealing of the next state occur at the same temperature. This design still needs external thermal control for multiple state transitions. By autocatalytic, we mean that, once the protein enzyme is introduced in the solution, it drives computation on its own. This protocol can again be contrasted against the protocol for PNA-mediated WPCR [7] where, after each polymerization step, the mixture needs to be washed by bis-PNA. Our isothermal and reactivating protocol is partially inspired by our previous work where we used a combination of DNA polymerization and strand displacement to minimize errors in computational tile assembly [11].

The key idea in our design of an isothermal, reactivating WPCR (IR-WPCR) machine is to use primer extension of a secondary strand to dehybridize the $3^{'}$ end of the WPCR strand after the next state is copied. Thus, the $3^{'}$ end of the WPCR strand is now free to bind to the complement of the new current state. This action essentially eliminates the thermal cycling required by the original WPCR machine to execute a state transition. We give three versions of IR-WPCR: (1) a protocol where state transition rules are no longer available for computation after the next state is copied (Section 3), (2) a protocol in which a rule can be made "reusable" by using an auxiliary strand that restores the original state of the secondary primer (Section 4), (3) a protocol that allows reuse of transition states while preventing back-hybridization (Section 5). The protocol presented in Section 5 is the most significant protocol in this paper: isothermal reactivating WPCR where the states are reusable and yet it is capable of preventing back-hybridization using a type of WPCR that we call folding WPCR. A primary assumption in all the above-mentioned protocols is that the concentration of the WPCR strand is such that two or more copies of WPCR strands do not interact among each other. We further compute the state transition rate for IR-WPCR (Section 6). We also estimate the rate at which states are made reusable in the second protocol. Additionally, we present a DNA sequence design of a 3 state machine and an experimental verification plan (Section 7). It should be remembered, however, that this example is meant to demonstrate the isothermal

and autocatalytic aspects of the protocol and not the computational power of a WPCR device.

2 Original Whiplash PCR System

In the original WPCR machine, the transition table is encoded on a single stranded DNA W as $S - a_1 - b_1 - S - a_2 - b_2 - \ldots - S - a_n - b_n$ where each pair $a_i - b_i$ represents the transition from state a_i to state b_i. The stopper sequence S isolates one state transition rule from another. The $3^{'}$ end of the same strand encodes the current state. For the description of the rest of the protocol, refer to Figure 1. We represent the stopper sequence S as a black square in the figure. Without loss of generality (w.l.o.g.), let us assume that the current state of the machine is a_i^*. Following the transition table, a_i can transition to b_i. Once a_i^* hybridizes with a_i (Figure 1: State S1) in W, polymerase extends the $3^{'}$ end of W to copy b_i (Figure 1: State S3). The polymerase halts after transcribing the bases complementary to b_i because of S which is often implemented by omitting one of the bases in the solution. For instance, in our experiments described later in the paper, we use T as the chosen base. Using appropriate thermal cycling, W

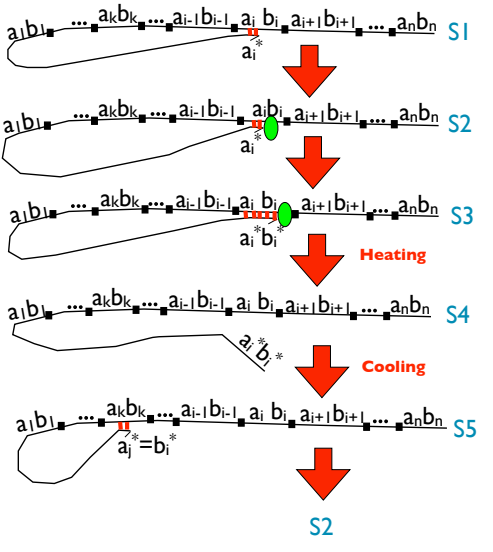


Fig. 1. *Schematic of the protocol for the original Whiplash PCR machine*: (S1)Initial state of the WPCR strand W with current state being a_i^*, (S2) Polymerase binds to the $3^{'}$ end of W (bearing the current state), (S3) Next state b_i^* is copied at the head of W by primer extension, (S4) The mixture is heated so that W loses its hairpin structure, (S5) The solution is cooled so that the head of W can bind to the new current state $b_i^* = a_j^*$ encoded at the $3^{'}$ end of the strand and the whole state transition repeats again beginning with State S2

is then denatured. Consequently, it loses the hairpin structure (Figure 1: State S4). Once the mixture is cooled, the $3^{'}$ end of newly extended W (now bearing b_i^* as the current state) hybridizes with another section of itself which encodes the appropriate transition rule (in this rule $a_j = b_i$ is the current state and b_j is the next state) (Figure 1: State S5). Although input is not part of the description of the WPCR machine, one can easily supply input as part of the initial state and update the encoding of the transition table to include inputs in the manner $S - a_i - I_i - b_i$ for the i^{th} transition rule. However, back-hybridization as mentioned in Section 1 is a serious problem with the original WPCR system and hence we propose a new protocol in Section 3 to minimize such erroneous state transitions.

3 IR-WPCR with Non-reusable Rules

In IR-WPCR with non-reusable rules, computation comprises of the following steps after the $3^{'}$ end of W binds to current state in rule R_i: (a) as with the original WPCR protocol, copying the next state at the $3^{'}$ end of the WPCR strand W, (b) dislodging a secondary primer sequence P_i, which is specific to the transition rule R_i from its initial position triggered by the primer extension on W, (c) subsequent hybridization of P_i to its final position in rule R_i and (d) dislodging of $3^{'}$ end of W by primer extension of P_i, allowing the $3^{'}$ end of W to bind to the new transition rule (so that it can hybridize with the new current state) by the primer extension of P_i. Observe that (b) and (d) act like a logical toggle switch allowing for an autocatalytic reaction.

In this version of WPCR, each rule is encoded as a 7-tuple $< x_i, y_i, z_i, a_i, b_i, w_i, y_i >$ where a_i still represents the current state and $b_i w_i y_i$ represents the next state where the b_i in IR-WPCR is not the same as b_i in the original WPCR strand. Rather, the original b_i is now divided into 3 subsequences b_i, w_i and y_i; we will describe a DNA design of a small IR-WPCR machine in Section 7. The

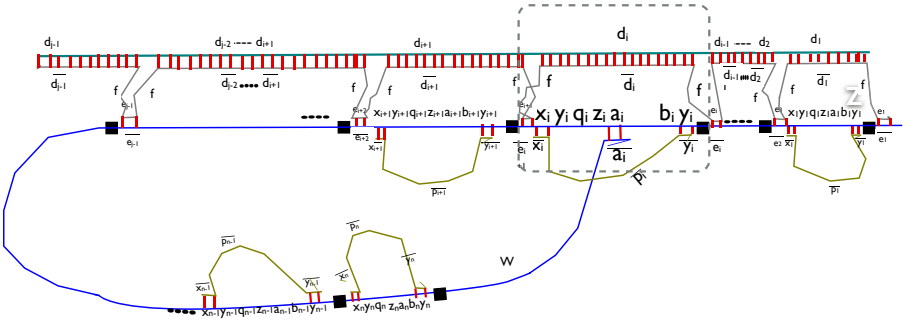


Fig. 2. Complete WPCR Strand for isothermal and autocatalytic program execution (Rule R_i on focus)

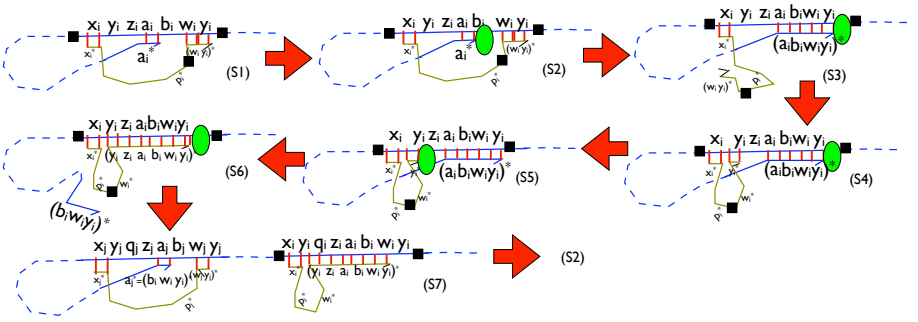


Fig. 3. *Evaluation Stage for non-reusable rules IR-WPCR protocol with the focus being only on the transition rule R_i to which the current state is hybridized*: (S1) WPCR strand W with protection strand P_i encoded as $(x_i p_i y_i)^*$ partially hybridized with rule R_i. Also the $3^{'}$ end of W, bearing the current state a_i^* is hybridized to a_i of R_i. (S2) Polymerase binds to the $3^{'}$ end of W (S3) Polymerase extends a_i^* to copy $b_i w_i y_i$, thus displacing $w_i^* y_i^*$ of P_i from $w_i y_i$ of rule R_i located further away from x_i in R_i. (S4) y_i^* of P_i binds to y_i located next to x_i in R_i. (S5) Polymerase binds with the $3^{'}$ end of P_i (S6) $3^{'}$ end of P_i is extended by the polymerase to copy $z_i a_i b_i w_i y_i$, thus displacing $3^{'}$ end of W which has the new current state $a_j = b_i w_i y_i$. (S7) $3^{'}$ end of W bearing a_j^* binds to the a_j in rule R_j and the process repeats starting with the polymerase binding to the $3^{'}$ end of W as shown in State S2.

other regions in this tuple are required for destabilizing the $3^{'}$ end of the strand once the next state is copied at the end of it. In this machine, the transition table with n rules is encoded on a single stranded DNA as $S{-}x_1{-}y_1{-}z_1{-}a_1{-}b_1{-}w_1{-}y_1{-}\ldots{-}S{-}x_i{-}y_i{-}z_i{-}a_i{-}b_i{-}w_i{-}y_i{-}\ldots{-}S{-}x_n{-}y_n{-}z_n{-}a_n{-}b_n{-}w_n{-}y_n$. The $3^{'}$ end of the single strand still encodes the current state as in original WPCR. We also tether the transition table portion of W to another stable nanostructure to prevent formation of any undesired secondary structure (Figure 2).

3.1 Computing with a Non-reusable Rules IR-WPCR Strand

Suppose we have the single strand in the form shown in Figure 2 prior to the addition of polymerase. In Section 3.2, we will discuss how we can obtain this particular secondary structure. W.l.o.g we will assume that the $3^{'}$ end of the single strand encodes for the complement of the state a_i in rule R_i. For clarity, we will refer to a figure that focuses only on the events at R_i (Figure 3).

Once a_i^* binds to a_i in R_i (Figure 3 :State S1) in presence of polymerase (Figure 3: State S2), the next state $b_i w_i y_i$ is copied at the $3^{'}$ end of W, thus dehybridizing the $(w_i y_i)^*$ portion of the protection strand P_i encoded as $(x_i p_i w_i y_i)^*$ (Figure 3: State S3). The y_i^* portion of P_i is now free to hybridize with the y_i portion on the rule R_i that is closer to x_i (Figure 3: State S4). The $3^{'}$ end of P_i, in presence of polymerase (Figure 3: State S5), then extends up to the stopper sequence S (shown in black filled squares in the figure), thus displacing the $3^{'}$ end of W (Figure 3: State S6). This rule site is now completely unavailable for

further hybridization and hence this protocol is called non-reusable rules IR-WPCR. The new current state $a_j^* = (b_i w_i y_i)^*$ at the $3^{'}$ end of W then binds to a_j which is the current state for transition rule R_j (Figure 3: State S7). At this stage, the next step of the computation starts with the polymerase binding to the head ($3^{'}$ end) of W, encoding the current state a_j^* (Figure 3: State S2). Hence, the state machine operates without thermal cycles and uses only polymerase to facilitate denaturation of the $3^{'}$ of W from the old rule.

3.2 Preparing a Non-reusable Rule IR-WPCR Strand for Computation

This section will describe how to obtain the secondary structure of the WPCR strand W as shown in Figure 2. In order to ensure that the hairpin structure is stable, we tether the single strand with another nanostructure which has extended ds-DNA regions. For clarity, we again focus only on rule R_i for the description of the preparation stage (Figure 4). Since we use a secondary primer to drive state transition, we need to ensure that the former is correctly hybridized at the onset. This is because there are two competing sections in each rule R_i (i.e. y_i) where the secondary primer can bind. We can use either a simple or a complex preparation protocol for this purpose. The complex preparation protocol ensures that the initial hybridization state of every secondary primer (each corresponding to a rule R_i) is the desired one through a series of protection/deprotection steps as described in Figure 4 while the simple preparation protocol takes advantage of more energetically favorable hybridization as discussed below.

In the *simple preparation protocol*, once we have guaranteed that the secondary structure of W is that of a hairpin, we directly introduce the protection strands P_i for each rule R_i into the solution. Since the $w_i y_i$ section is longer than

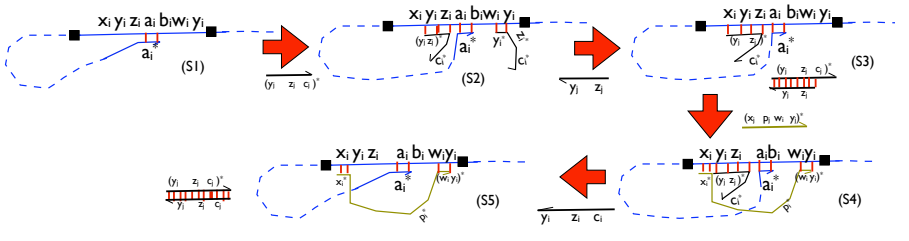


Fig. 4. *Complex Preparation Module with respect to only rule R_i:* (S1) WPCR strand W tethered to support (not shown in figure) (S2) $(y_i z_i c_i)^*$ is added to the solution. One copy binds to the y_i near x_i and another binds to y_i further away from it (S3) The copy of $(y_i z_i c_i)^*$ that binds to the y_i in R_i further away from x_i is removed by the addition of $y_i z_i$. The duplex thus formed is then removed from the solution using magnetic beads (not shown here) (S4) Protection strand P_i encoded as $(x_i p_i w_i y_i)^*$ is introduced and it hybridizes with the x_i and free $w_i y_i$ of rule R_i (S5) The copy of $(y_i z_i c_i)^*$ that is bound to the y_i in R_i nearer to x_i is removed by the addition of $y_i q_i z_i$. Here too, the duplex is later removed using magnetic beads.

the y_i region near x_i for each R_i the protection strand for R_i acquires the configuration shown in Figure 3: State S1 with high probability. For more guaranteed hybridization of the protection strand with its corresponding transition rules, one may adhere to the *complex protection protocol* which has been presented in Figure 4.

3.3 Handling Inputs

Inputs in the IR-WPCR system can be handled in a manner very similar to the original WPCR system [1]. Each input symbol that is part of the transition table, can be encoded between the current state and next state of a rule while the symbols in the external input string are encoded uniquely (so as to enable the machine to determine which input symbol to use for the next transition) and is ligated at the $3^{'}$ end of W at the start of the corresponding transition. However, there is a catch, in the sense that if the input strands hybridize to the rules independent of the current state of the WPCR strand then they can bind long before the current state of the machine is that of a particular transition rule. Consequently, the resultant computation will be erroneous. On the other hand, if the current state of a transition rule is bound to a tertiary protection strand (bearing complement of current state and input symbol for that particular transition rule) then free floating inputs have to bind to the tertiary protection strand first and polymerase has to fill in the rest of the bases before the current state is available to participate in state transition. This protocol will work for finite state machines that can be represented as directed acyclic graphs (i.e. machines that do not revisit transitions).

3.4 Limitation

The only limitation of non-reusable rules IR-WPCR is that a rule can only be used once. One way to avoid this problem is to have several redundant copies of each rule encoded in W. However, we propose a more elegant solution using strand displacement in Section 4.

4 IR-WPCR with Reusable Rules

The above mentioned protocol works very well for reducing back-hybridization since it makes a rule unavailable after a transition. To address the limitation of the design, this section describes a protocol that uses an additional strand A_i to displace P_i after primer extension (Figure 5(Right)). This protocol comprises of the same steps: (a) to (d) as the non-reusable rules IR-WPCR protocol (See Section 3). Additionally it has a final step (e) where an auxiliary strand A_i (present in the reaction mixture) changes the secondary structure of P_i. This strand A_i can partially hybridize with the extended section of P_i and force P_i to return to its original "protection" state, permitting further computation using this transition rule. In a manner, the rule is "reset" after the state transition

takes place. Hence this protocol is called Reusable Rules IR-WPCR. In the following section we describe how a state transition occurs in this new protocol. Its preparation stage is the same as that of non-reusable rules IR-WPCR (See Section 3.2).

4.1 Computing with a Reusable Rule IR-WPCR Strand

The first part of evaluation is the same as that in non-reusable rules IR-WPCR protocol (Figure 5(Right): State S1-State S6). However, unlike the other protocol, in this method, the solution additionally contains a large concentration of A_i, which is encoded as $w_i y_i z_i a_i b_i$. Hence once P_i is extended, its w_i^* region is used an a toehold by A_i to displace the former. However, the $(w_i y_i)^*$ portion at the end of extended P_i is still free to hybridize with its complementary region on R_i (Figure 5(Right): State S7). This step ensures that secondary structure of P_i is "reset" enabling R_i to participate in computation again. The last stage of evaluation is the same as in non-reusable rules IR-WPCR where the new state corresponds to $b_i w_i y_i = a_j$ for some j and hence the $3^{'}$ end of P binds to rule R_j (Figure 5(Right): State S8).

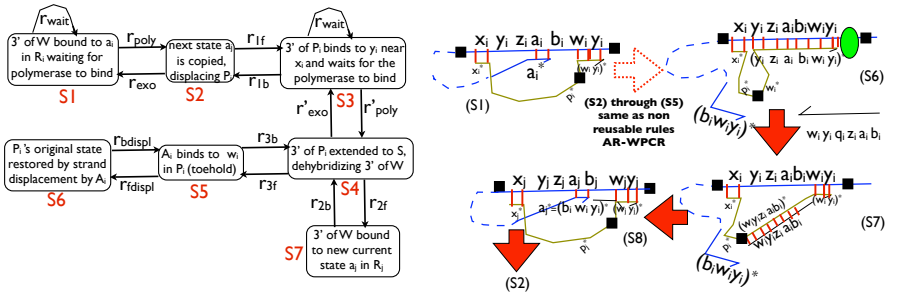


Fig. 5. *(Left) Continuous time Markov Chain for rule R_i in the reusable rules IR-WPCR protocol that does not prevent back-hybridization, (Right) Evaluation Stage in IR-WPCR with reusable rules with the focus being only on the transition rule to which the current state is hybridized*: (S1)-(S6) Same as IR-WPCR protocol with non-reusable rules (S7) A_i encoded $(w_i y_i z_i a_i b_i)$ present in the solution displaces $(w_i y_i z_i a_i b_i)^*$ region of the protection strand P_i so that the configuration of the latter can be reset (S8) $3^{'}$ end of W bearing a_j^* binds to the a_j in rule R_j and the process repeats again starting with the polymerase binding to the $3^{'}$ end of W (State S2)

This additional step adds a number of new behaviors to the computation. For example, after each state transition, the portion on P_i between the ends hybridized with R_i has an additional ds-DNA region encoded as $w_i y_i z_i a_i b_i$, and also gets longer each time R_i is used. The longer strand may prevent reusability of a rule after some number of steps due to steric hindrance. Note that a copy of A_i is consumed each time R_i is used so the solution must have an excess of A_i. P_i must also contain a stopper sequence to prevent P_i from being extended in an undesired direction.

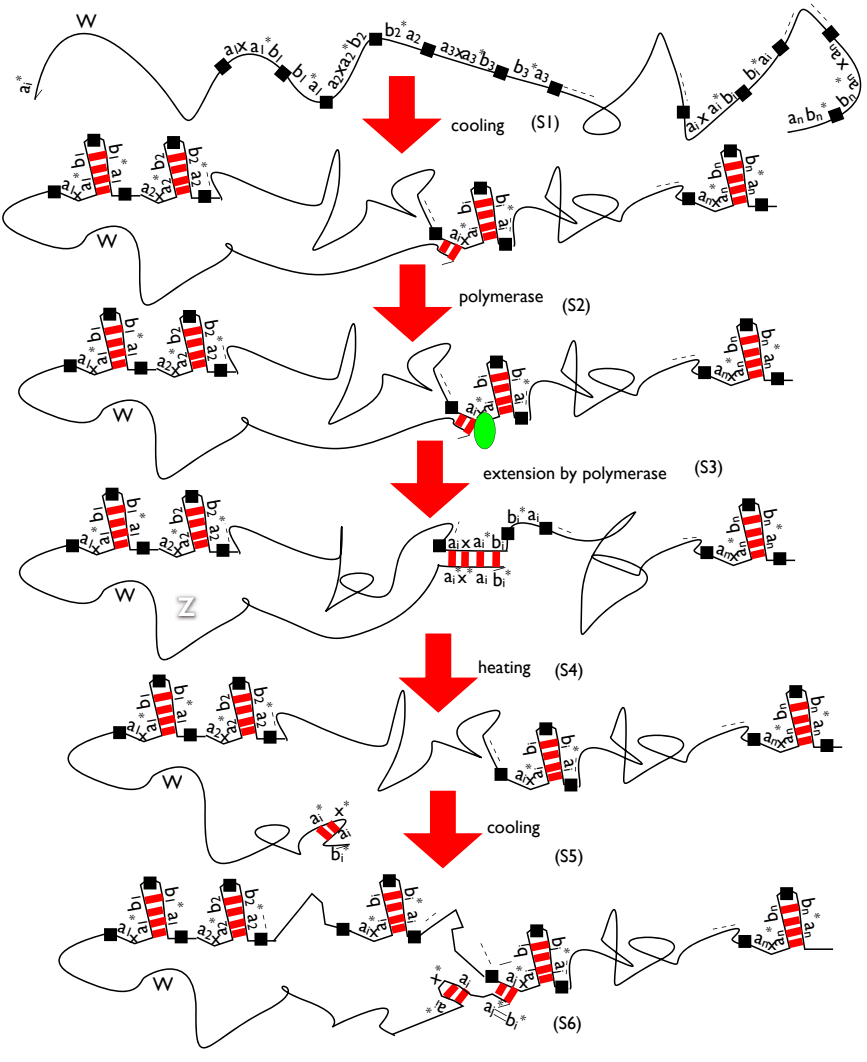


Fig. 6. Schematic of the protocol for the folding Whiplash PCR machine that uses thermal cycling for state transition yet prevents backhybridization using hairpin loops: (S1) Initial state of the WPCR strand W, (S2) The solution is cooled such that the next state in each rule hidden in a hairpin loop with current state of the machine being a_i^*, (S3) Polymerase binds to the $3^{'}$ end of W (bearing the current state), (S4) Next state b_i^* is copied at the head of W by primer extension and hairpin loop is opened, (S5) The mixture is heated so that W loses its hairpin structure (It may even open up the individual hairpin loops in each rule, not shown here), (S6) The solution is cooled so that the head of W can bind to the new current state $b_i^* = a_j^*$ encoded at the $3^{'}$ end of the strand and the whole state transition repeats again beginning with State S2. Note that the next state in each rule is hidden in a stem loop as is the old current state encoded at the $3^{'}$ end of the WPCR strand. This stem loop formation is key to preventing back-hybridization in this protocol.

4.2 Limitations of a Reusable Rule IR-WPCR Machine

Reusable rules IR-WPCR suffers from all the limitations of the original WPCR, such as back-hybridization and out-of-frame annealing. Consequently, previously

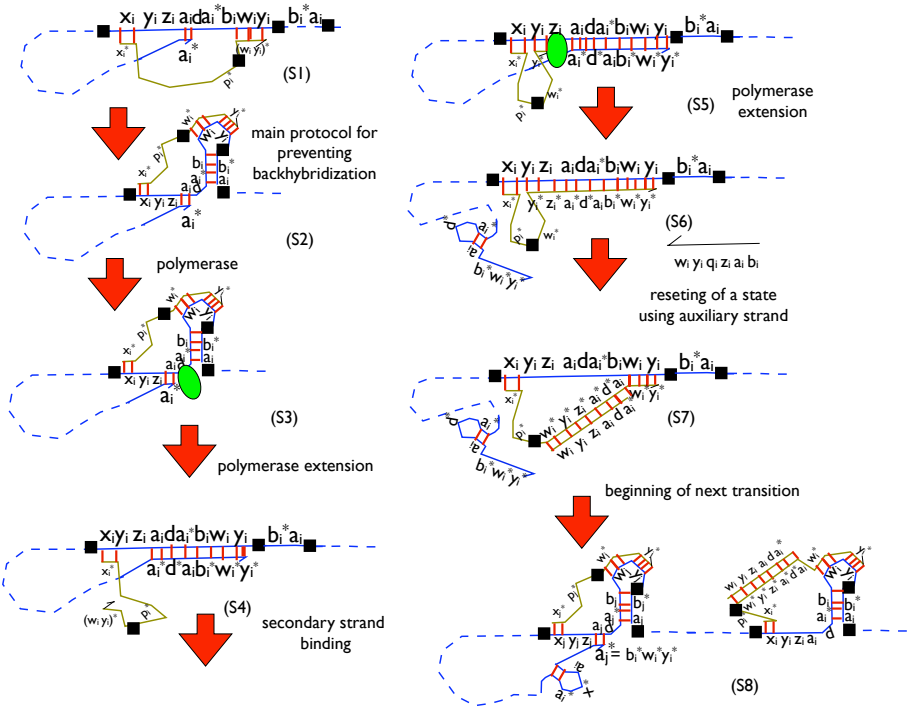


Fig. 7. *Evaluation Stage for reusable rules IR-WPCR protocol (without thermal cycling) that prevents back-hybridization using folding PCR with the focus being only on the transition rule R_i to which the current state is hybridized*: (S1) WPCR strand W with protection strand P_i encoded as $(x_i p_i y_i)^*$ partially hybridized with rule R_i. Also the $3^{'}$ end of W, bearing the current state a_i^* is hybridized to a_i of R_i. (S2) The temperature of the solution is such that the b_i part of the next state $b_i w_i y_i$ forms part of a stem loop. (S3) Polymerase binds to the $3^{'}$ end of W. (S4) Polymerase extends a_i^* to copy $b_i w_i y_i$, thus displacing $w_i^* y_i^*$ of P_i from $w_i y_i$ of rule R_i located further away from x_i in R_i. Furthermore it opens the stem loop in which part of the next state was hidden, (S5) y_i^* of P_i binds to y_i located next to x_i in R_i. Polymerase binds with the $3^{'}$ end of P_i (S6) $3^{'}$ end of P_i is extended by the polymerase to copy $z_i a_i d(a_i)^* b_i w_i y_i$, thus displacing $3^{'}$ end of W which has the new current state $a_j = b_i w_i y_i$. (S7) A_i encoded $(w_i y_i z_i a_i b_i)$ present in the solution displaces $(w_i y_i z_i a_i b_i)^*$ region of the protection strand P_i so that the configuration of the latter can be reset. Furthermore, at the $3^{'}$ end of the WPCR strand the old state a_i forms part of a hairpin loop because of the solution temperature. (S8) The next state of R_i is reset to its stem loop configuration and $3^{'}$ end of W bearing a_j^* binds to the a_j in rule R_j and the process repeats starting with the polymerase binding to the $3^{'}$ end of W as shown in State S2.

mentioned protocols [9,7] to avoid these problems (e.g: PNA mediation for back-hybridization) can be incorporated into our system at the expense of losing full autonomy. Thus, for finite state machines that can be represented as a directed acyclic graph, non reusable rules IR-WPCR machines may be recommended. For encoding general purpose state transitioning system, reusable rules IR-WPCR will probably be better. There is, however, a more general yet complex machine that builds on the previously described machines and a new protocol which we call folding WPCR (as described in Figure 6 only due to space constraints) that allows reuse of transition rule while handling backhybridization.

5 IR-WPCR Machine That Prevents Back-Hybridization

Refer to Figure 7 for the isothermal WPCR with reusable states protocol that uses folding WPCR to prevent back-hybridization. In isothermal Whiplash PCR using reusable states, for rule i, the current state is a_i while the next state is $b_i w_i y_i$. However, in the isothermal WPCR that also uses folding WPCR to prevent back-hybridization, the state encoding is $x_i y_i z_i a_i d(a_i)^* b_i w_i y_i S(b_i)^* a_i$. Under suitable temperature, $(a_i)^* b_i w_i y_i S(b_i)^* a_i$ forms a stem loop and the protection strand P_i hybridizes with x_i and $w_i y_i$. W.l.o.g., assume that the current state is a_i and thus the $3^{'}$ end of the WPCR strand is binds to rule i. When the polymerase binds to this end, it opens up the stem loop hiding part of the next state encoding in this rule. The $3^{'}$ end of the WPCR strand is extended as far as the first stopper sequence in this rule. This event, in turn, displaces the $3^{'}$ end of the protection strand. The latter now binds to its second best match in rule i, y_i. Polymerase now binds to the $3^{'}$ end of the protection strand and it extends to displace the $3^{'}$ end of the WPCR strand which has the new current state encoded in it. This marks the completion of a state transition. An auxiliary strand already in solution, then resets state i to its original configuration. However, the only difference between this protocol and the isothermal, reactivating WPCR with reusable states that cannot prevent back-hybridization is that the former uses folding WPCR. In other words, at the end of any state transition (say i for instance), the $3^{'}$ end of the WPCR strand not only has $(b_i)^*(w_i)^*(y_i)^*$ encoded in it, but also has $(a_i)^* d a_i$ preceding the $(b_i)^*(w_i)^*(y_i)^*$ encoding. This folds into a loop and hides $(a_i)^*$. Similarly, in rule i, the next state b_i is hidden in a loop after being copied at the $3^{'}$ end of the strand. This prevents back-hybridization.

6 Probability and Rate of State Transition in IR-WPCR Method

The WPCR machine is a stochastic system and hence assuming the correctness of the original WPCR system, we estimate the likelihood of a single state transition in the IR-WPCR machine. We base our calculation on the corresponding continuous time Markov Chain in Figure 5(Right). Based on this Markov Chain we can also compute the probability and rate of "reseting" a state. We, however, first need to explain all the rates shown in the Markov Chain (Figure 5(Left)).

6.1 Rate of Polymerization

In Figure 5(Left) r_{poly} corresponds to the rate at which the next state of length l_1 bases is copied and $r^{'}_{poly}$ corresponds to the effective rate at which P_i (l_2 bases) is extended. If the polymerase can extend N_{bases} bases/sec, then $r_{poly} = \frac{N_{bases}}{l_1}$ while $r^{'}_{poly} = \frac{N_{bases}}{l_2}$. r_{wait} is the mean DNA polymerase and strand encounter rate, given by $r_{wait} = \frac{N_{poly}}{N_{strands}} v_t$ where N_{poly} is the number of units of polymerase in the solution and $N_{strands}$ corresponds to the total number of strands (each P_i in a W counts for a separate strand). Additionally $v_t = \frac{1}{\frac{1}{r_{poly}} + \frac{1}{r_{poly}}}$ denotes the mean number of distinct extensions/sec by 1 unit of polymerase under optimal conditions using excess target and primer [7]. Furthermore, r_{exo} ($r^{'}_{exo}$) is the effective rate of exonuclease activity. It is generally very small and is given by $r_{exo} = \frac{k_{exo}}{l_1} \frac{N_{poly}}{V}$ and $r^{'}_{exo} = \frac{k_{exo}}{l_2} \frac{N_{poly}}{V}$ where k_{exo} for $\phi 29$ (our chosen polymerase) can be calculated from primer extension experiments [12]. Here, V is the total volume of the solution. Since target/primer is in excess, the exonuclease activity is limited by the concentration of the polymerase.

6.2 Rate of Hybridization

For hybridization events, the rates r_{1f}, r_{2f} and r_{3f} are not concentration dependent since all the components are part of the same nanostructure. Hence $r_{if} \propto \sqrt{hl_i}$ where hl_i is the length of the hybridization segment [13] for $i = 1, 2, 3$ under optimal conditions, thus neglecting effect of temperature and salt concentration. The rate of dehybridization (r_{1b}, r_{2b} and r_{3b}) is given by $k_f e^{-G_{se_i}}$ where $G_{se_i} = (\frac{c}{T} - 11) hl_i$, $c = 4000$ is a constant and T being the temperature of the solution in K and k_f is the forward rate constant [14].

6.3 Rate of Strand Displacement

Finally, for estimating r_{fdispl} and r_{bdispl} we model strand displacement as a 1D random walk. After the toehold hybridization let G denote the free energy of the 3 strand complex, G_l the free energy after one base pair migration to the left and G_r the same for migration to the right. Moreover, let $\Delta G_r = G_r - G$ and $\Delta G_l = G_l - G$ where the change in free energy can be computed from the nearest neighbor model. If p_r and p_l are the probability for right and left directional migration, then $p_r \propto exp(\frac{-\Delta G_r}{RT})$ and $p_l \propto exp(\frac{-\Delta G_l}{RT})$. Further, the mean time for a single base migration is about 100 μ sec [15]. Hence, $r_{fdispl} = \frac{c}{\sum_{i=1}^{l_{fdispl}} \frac{p_{r_i}}{p_{r_i} + p_{l_i}}}$ and $r_{bdispl} = \frac{c}{\sum_{i=1}^{l_{bdispl}} \frac{p_{l_i}}{p_{r_i} + p_{l_i}}}$ where $c = 10000$ is a constant, l_{fdispl} and l_{bdispl} denote the number of bases displaced in the forward and backward direction respectively. A_i displaces P_i to the right and corresponds to r_{fdispl}.

6.4 Likelihood and Rate of a State Transition

Based on the Markov Chain in Figure 5(Left), the probability of a state transition is given by $\left(\frac{r_{poly}}{r_{poly}+r_{wait}}\right) \cdot \left(\frac{r_{1f}}{r_{1f}+r_{exo}}\right) \cdot \left(\frac{r'_{poly}}{r_{wait}+r_{1b}}\right) \cdot \left(\frac{r_{2f}}{r_{2f}+r'_{exo}}\right)$ and the rate of state transition, given that $3'$ end of W is bound to a_i in R_i and the polymerase is bound at the end of it, is given by $r_{poly}\left(\frac{r_{1f}}{r_{1f}+r_{exo}}\right) \cdot \left(\frac{r'_{poly}}{r_{wait}+r_{1b}}\right) \cdot \left(\frac{r_{2f}}{r_{2f}+r'_{exo}}\right)$. On the other hand, the probability that a state will be "reset" or made reusable after participating in the computation is given by $\left(\frac{r_{poly}}{r_{poly}+r_{wait}}\right) \cdot \left(\frac{r_{1f}}{r_{1f}+r_{exo}}\right) \cdot \left(\frac{r'_{poly}}{r_{wait}+r_{1b}}\right) \cdot \left(\frac{r_{3f}}{r_{3f}+r'_{exo}}\right) \cdot \left(\frac{r_{fdispl}}{r_{fdispl}+r_{3b}}\right)$ while the corresponding rate is $r_{poly}\left(\frac{r_{1f}}{r_{1f}+r_{exo}}\right) \cdot \left(\frac{r'_{poly}}{r_{wait}+r_{1b}}\right) \cdot \left(\frac{r_{3f}}{r_{3f}+r'_{exo}}\right) \cdot \left(\frac{r_{fdispl}}{r_{fdispl}+r_{3b}}\right)$, given that the current state of W is a_i and polymerase is bound to its $3'$ end.

7 DNA Design of IR-WPCR Computing on a 3 State Machine

In this section we provide a concrete example of the execution of the IR-WPCR protocol on a 3 state machine. We also describe how one can verify computation on this state machine using gel electrophoresis and the Fluorescence Resonance Energy Transfer (FRET) method.

7.1 Encoding of the WPCR Strand

Suppose we have a 3 state machine: $s_1 \longrightarrow s_2 \longrightarrow s_3$. For simplicity, we assume that the input is part of the state description. Thus we have to encode only two transition rules viz. $s_1 \longrightarrow s_2$ and $s_2 \longrightarrow s_3$ in the WPCR strand W $(x_1-y_1-z_1-a_1-b_1-w_1-y_1-S-x_2-y_2-z_2-a_2-b_2-w_2-y_2-S-S'-a_1^*)$ Here S (6 bases) is the stopper sequence (Our chosen polymerase tends to ignore and continue polymerizing if stopper sequences are shorter than this length.) and S' is the spacer sequence that allows W to form a hairpin structure easily. Observe that a_1 is the encoding for state s_1, while $b_1 - w_1 - y_1$ (or a_2) is the encoding for s_2 and finally $b_2 - w_2 - y_2$ is the encoding for s_3. Each state is used for priming and, hence, comprises of at least 15 bases.

We use polymerase $\phi 29$ to drive the computing, because of its excellent strand displacement capability [16]. However, it has to be used in very low concentration because of its high fidelity in higher concentrations. Further, if any state is used as a molecular beacon for FRET experiments (e.g: $b_2w_2y_2$) the length has to be at least 24 bases. Besides W, we need to consider P_1's DNA design. P_1 binds to the x_1 and w_1y_1 of rule R_1 before extension and, hence, at least three bases are necessary for each section to ensure stable hybridization. It should be remembered that, since R_2 is the last transition rule visited by the hairpin, it is not necessary to include P_2 or A_2 for R_2. This compact hairpin structure is more suitable for gel analysis.

We can perform two distinct experiments E_1 and E_2 to verify whether both state transitions are executed. This is essential because, in IR-WPCR protocols, once the polymerase is added to the solution, in theory, the reaction goes to completion. Thus, it is not easy to probe the state of W after every state transition. Hence, E_1 corresponds to the experiment when W encodes for $s_1 \longrightarrow s_2$ and $s_2 \longrightarrow s_3$ while, in E_2, W encodes for $s_1 \longrightarrow s_2$. Since we have only a three state machine these experiments together allow us to study the secondary structure of W. It should be noted that for E_2 we do not need either P_1 or A_1 since $s2$ is the final state for this experiment. Sequences designed for E_1 and E_2 are shown in Figure 8.

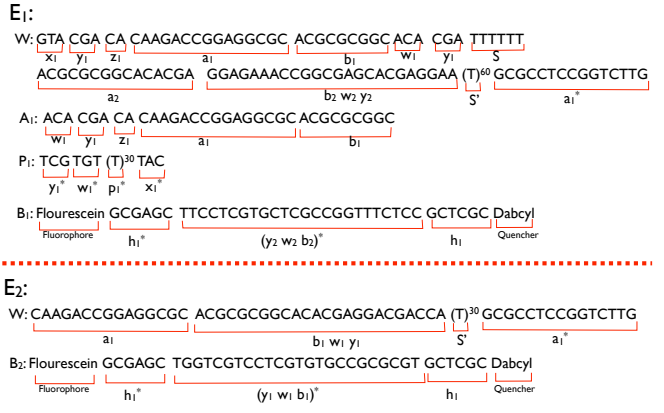


Fig. 8. DNA Sequences for experiments E_1 and E_2

7.2 Verification of Computation with FRET Analysis

Before introducing $\phi 29$ for E_1 we add P_1 to W and anneal the mixture to $30°C$ so that W can attain the hairpin structure and P_1 can bind to rule R_1. Auxiliary strand A_1 is added in the next step. Once a small volume of very low concentration (about $0.05\mu M$) $\phi 29$ is added to the solution at $30°C$, the resultant mixture is ready to be analyzed in about 30 minutes [16]. Gel electrophoresis may be performed to analyze size of W before and after adding $\phi 29$. However, the inference may not be conclusive.

To perform FRET analysis, we use the molecular beacon [17] technique. For E_1 we use an extended $b_2w_2y_2$ region (about 24 bases in length) that is bound to a single strand B_1 encoded as $h(b_2w_2y_2)^*h^*$. There is a fluorophore and a quencher at the two ends of B_1. h and h^* are complementary to each other and each is about 6 bases long. The idea of a molecular beacon is that, when W copies the next state in R_2, it will displace B_1. Consequently, B_1 is released and the h portion of B_1 binds to its h^* portion, resulting in a hairpin. Consequently, there is a detectable drop in fluorescence signal. Similarly for E_2, the molecular beacon B_2 $(h(b_1w_1y_1)^*h^*)$ is hybridized to $b_1w_1y_1$ region of R_1 in W. Here too, as soon as the next state in R_1 is copied by$\phi 29$, B_2 is released, forms a hairpin and the existing fluorescence is quenched.

8 Conclusion

In summary, WPCR is an useful model of computation for running distinct programs in parallel. However, existing protocols are not simultaneously isothermal and autocatalytic, thus limiting the number of steps the machine can execute as well as its flexibility. Here, we presented three such protocols of for computing with WPCR. The key idea is to use strand displacement and polymerization of a secondary priming sequence to dehybridize the $3^{'}$ of the WPCR strand once the next state is copied eliminating the need for a thermal cycle. One immediate future direction is to verify with laboratory experiments and computer simulation the proposed design of the 3 state machine (Section 7) and compare the yields with that of the original WPCR implementation [9,1].

References

1. Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., Yokoyama, S.: In: Rubin, H., Woods, D.H. (eds.) DNA Based Computers III, pp. 55–72. American Mathematical Society (1999)
2. Winfree, E., Liu, F., Wenzler, L., Seeman, N.: Nature 394(6693), 539–544 (1998)
3. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Nature 414, 430–434 (2001)
4. Soloveichik, D., Winfree, E.: Theoretical Computer Science 244, 279–297 (2005)
5. Matsuda, D., Yamamura, M.: Cascading whiplash PCR with a nicking enzyme. In: Hagiya, M., Ohuchi, A. (eds.) DNA 2002. LNCS, vol. 2568, pp. 38–46. Springer, Heidelberg (2003)
6. Nishikawa, A., Hagiya, M.: In: Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.) Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 960–966. IEEE Press, Los Alamitos (1999)
7. Rose, J.A., Deaton, R.J., Hagiya, M., Suyama, A.: PNA-mediated whiplash PCR. In: Jonoska, N., Seeman, N.C. (eds.) DNA 2001. LNCS, vol. 2340, pp. 104–116. Springer, Heidelberg (2002)
8. Winfree, E.: Whiplash PCR for $o(1)$ computing. Technical Report 1998.23, Caltech (1998)
9. Sakamoto, K., Kiga, D., Komiya, K., Gouzu, H., Yokoyama, S., Ikeda, S., Sugiyama, H., Hagiya, M.: State transitions by molecules. Biosystems 52, 81–91 (1999)
10. Rose, J.A., Komiya, K., Yaegashi, S., Hagiya, M.: Displacement whiplash PCR: Optimized architecture and experimental validation. In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 393–403. Springer, Heidelberg (2006)
11. Majumder, U., LaBean, T., Reif, J.:DNA 13. LNCS, vol. 287, pp. 195–214. Springer, Heidelberg (1987)
12. Saturno, J., Blanco, L., Salas, M., Esteban, J.: J. of Bio. Chem. 270(52), 31235–31243 (1995)
13. Hames, B.D., Higgins, S.J.: Gene Probes 2. Oxford University Press, Oxford (1995)
14. Winfree, E.: Simulation of computing by self-assembly. Technical Report 1998.22, Caltech (1998)
15. Sahu, S., Wang, B., Reif, J.H.: A framework for modeling DNA based molecular systems. In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 250–265. Springer, Heidelberg (2006)
16. Sahu, S., LaBean, T.H., Reif, J.H.: A nanomotor powered by polymerase performing motions on DNA tracks (manuscript, 2008)
17. Tyagi, S., Kramer, F.: Nat Biotechnol. 14(3), 303–308 (March 1996)

DNA as a Universal Substrate for Chemical Kinetics
(Extended Abstract)

David Soloveichik, Georg Seelig, and Erik Winfree

California Institute of Technology, Pasadena, CA, USA
dsolov@caltech.edu, seelig@dna.caltech.edu, winfree@caltech.edu

Abstract. We show that a DNA-based chemical system can be constructed such that it closely approximates the dynamic behavior of an arbitrary system of coupled chemical reactions. Using strand displacement reactions as a primitive we explicitly construct reaction cascades with effectively unimolecular and bimolecular kinetics. Our construction allows for individual reactions to be coupled in arbitrary ways such that reactants can participate in multiple reactions simultaneously, correctly reproducing the desired dynamical properties. Thus arbitrary systems of chemical equations can be compiled into chemistry. We illustrate our method on a chaotic Rössler attractor; simulations of the attractor and of our proposed DNA-based implementation show good agreement.

1 Introduction

Chemical reaction equations and mass action kinetics provide a powerful mathematical language for describing and analyzing chemical systems. For well over a century, mass action kinetics has been used to model chemical experiments, in order to predict and explain the evolution of the various species over time, and to elucidate the dynamical properties of the system under investigation. Chemistry exhibits complex behavior like oscillations, limit cycles, chaos or pattern formation, all of which can be explained by the corresponding systems of coupled chemical reactions [1,2,3]. While the use of mass action kinetics to describe existing chemical systems is well established, the inverse problem of experimentally implementing a given set of chemical reactions has not been widely considered. Many systems of coupled chemical equations appear to not have realizations in known chemistry.

Here we propose a method for implementing an arbitrary system of coupled chemical reactions using nucleic acids. We develop an explicit implementation of unimolecular and bimolecular reactions which can be combined into arbitrarily coupled reaction networks. In a formal system of chemical reactions such as

$$\begin{aligned} A &\xrightarrow{k_1} B \\ A + B &\xrightarrow{k_2} C + D \\ C &\xrightarrow{k_3} \end{aligned}$$

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 57–69, 2009.

a species may need to participate in multiple reactions, both as a reactant and/or as a product (species A, B or C) and these reactions need to progress at rates determined by the rate constants (k_1, k_2 and k_3). This imposes onerous constraints on the chemical properties of the species participating in these reactions. For example, it is likely hard to find a physical implementation of the chemical reaction equations using small molecules, since small molecules have a limited set of reactivities. Information-bearing biopolymers such as proteins or nucleic acids provide a more promising physical substrate for implementing arbitrary chemical reactions. Nucleic acids have the unique advantage that interactions between different single-stranded species can be programmed since sequence determines reactivity through Watson-Crick base pairing.

In our DNA implementation, we assign each formal species (e.g., A, B, C, D) to a set of DNA molecules. In some instances it may be possible to map a formal species to a single oligonucleotide but more generally a single formal species will correspond to several DNA species in order to reproduce the correct kinetics. Effective interactions between the species are mediated by an additional set of DNA complexes. Since the underlying chemistry involves aqueous-phase nucleic acid hybridization and strand exchange reactions, arbitrarily large rate constants and concentrations cannot be attained. However, any system of coupled chemical reactions can be scaled to use smaller rate constants and concentrations without affecting the kinetics except by a scaling factor (see Section 6). While our constructions are purely theoretical at this point, they are based on realistic assumptions and provide a roadmap for future experiments.

In the next section we describe strand displacement reactions that will serve as the basic building block for our construction. In the following section we show how to implement arbitrary unimolecular reactions, and then extend our construction to cover bimolecular reactions. In the final section we give a demonstration of our approach by describing the implementation of a system due to Willamowski and Rössler [4] with 3 species and 7 reactions exhibiting chaotic concentration fluctuations. Numerical simulations of the original formal system and our DNA-based chemical reactions using realistic rate constants and concentrations are in good agreement.

2 Cascades of Strand Displacement Reactions

We use strand displacement reactions as the basic primitive for our constructions (Fig. 1). Strand displacement has been found to be a flexible method for designing complex behaviors with nucleic acids including motors, logic gates, and catalysts [5,6,7,8]. Although a strand displacement reaction involves multiple elementary steps, including a random walk process, it is well modeled as a second-order process for a wide range of reaction conditions [9,10]. The effective rate constant of the second-order process is governed by the degree of sequence complementarity between the toeholds on the single-stranded species and on the partially double-stranded species [10].

We have recently used strand displacement cascades to construct DNA-based logic circuits [6,8]. Here we use some of the nomenclature and ideas from that

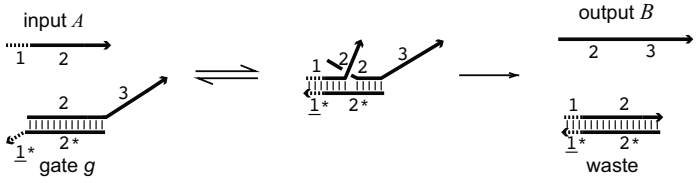


Fig. 1. Strand displacement reactions. The 3′ end of each strand is indicated by an arrow. Functional domains are numbered and the star indicates complementarity. We use the underline notation $\underline{1}^*$ to indicate that this domain may not be completely complementary to domain 1. The reaction between input strand A and gate g is initiated at the toehold (dashed, domain 1^*). The reaction then proceeds through multiple short-lived intermediates and leads to the release of an output strand B and the formation of a chemically inert double-stranded waste product. Kinetically, the overall reaction is well approximated as being bimolecular, i.e., $A + g \xrightarrow{k} B$, where we omit the inert waste product. The value of the rate constant k depends on reaction conditions (salt, temperature), length and sequence composition of the toehold as well as the degree of complementarity between the toeholds on the strand and gate (domains 1 and $\underline{1}^*$). In practice, toehold domains are typically 2–8 nucleotides long, and the domains undergoing strand displacement are typically 20–30 nucleotides long.

work. Fig. 2 shows a two-stage strand displacement cascade where an input single-stranded nucleic acid species (strand) initiates a strand displacement cascade between two complexes (gates) leading to the release of an output strand. In strand displacement cascades, a strand is functionally inactive before its release from a gate and becomes active upon becoming completely single-stranded. For example, intermediate strand o cannot react with translator gate t before it is released from gate g because its toehold domain 3, which is required for initiating the reaction with t, is double-stranded. Similarly, output Bs cannot initiate a downstream strand displacement cascade until it is released from translator gate t because its toehold domain 4 is double-stranded. However, upon the addition of free As, intermediate strand o is released through strand displacement, which then causes the release of output Bs. The release of strand Bs makes it capable of initiating other strand displacement cascades in turn. Note that the binding of a toehold domain to its complement is transient unless a strand displacement reaction can be initiated because the toehold domains are short. Thus, for example, the 3 domain of input As does not block the 3^* domain of translator gate t.

An input or output strand has two regions: a recognition region which can participate in a strand displacement reaction, and a history region which cannot. The sequence of the history region (e.g., domain 7 on strand Bs) is determined by the translator gate from which the strand was released. All strands with the same recognition region react equivalently and we do not distinguish between them. For example, any strand with recognition region 1-2-3 is called As and any strand with recognition region 4-5-6 is called Bs. Since there are no sequence constraints (i.e., complementarity or equality) between the recognition region of

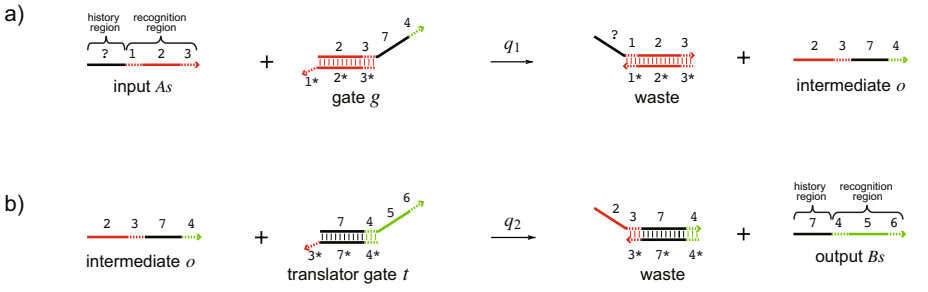


Fig. 2. Two-stage strand displacement cascade. Functional domains are numbered and all toehold domains are dashed. Different recognition regions are shown in different color. Input or output strands with identical recognition regions react equivalently and are therefore grouped into the same species. For example, As is any strand with recognition region 1-2-3, and Bs is any strand with recognition region 4-5-6, irrespective of their history regions. The two-stage cascade shown produces Bs with history region 7. Note that the sequences of the recognition regions of input and output strands As and Bs (1-2-3 and 4-5-6) may be completely unrelated to one another and therefore such a two-stage strand displacement cascade can link any input with any output species. **a)** Input strand As binds to gate g and by a strand displacement reaction releases the intermediate strand o. **b)** The intermediate strand o binds translator gate t and by a strand displacement reaction releases the output Bs.

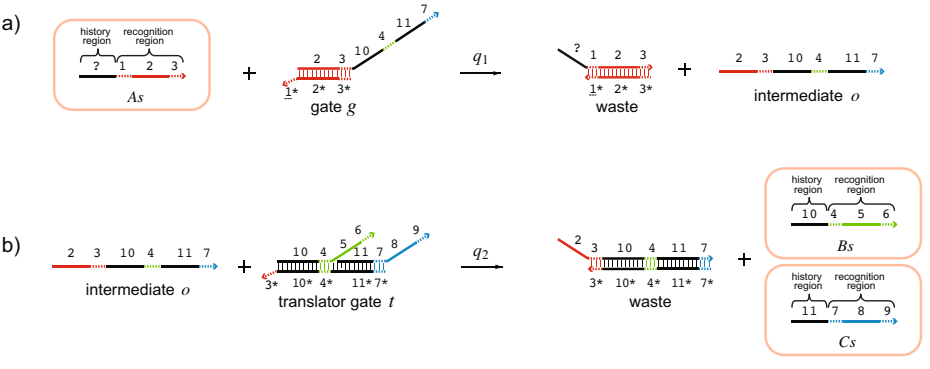


Fig. 3. Molecular implementation of the unimolecular reaction $A \rightarrow B + C$. Orange boxes highlight the DNA species As, Bs, and Cs that correspond to the formal species A, B, and C. Rate constant q_1 can be reduced by decreasing the complementarity between domains 1 and $\underline{1}^*$. The sequences of the recognition regions of input and output strands As, Bs, and Cs (regions 1-2-3, 4-5-6, and 7-8-9, respectively) may be completely unrelated to one another. The regime for desired unimolecular kinetics (concentrations of g, t and rate constants q_1, q_2) is described in the text. **a)** Input strand As binds to gate g and by a strand displacement reaction releases the intermediate strand o. **b)** The intermediate o binds translator gate t and by a strand displacement reaction releases the outputs Bs and Cs.

the input strand As and the output strand Bs, arbitrary chains of such two-step cascades can be linked together. This is possible for two-step cascades as shown (see "full translator" in Ref. [6]); however, a one-step cascade would force a part of the recognition region of the output strand to have sequence equality with the input strand, complicating the sequence design process. We call the second gate a translator gate to emphasize its role in translating the input to the appropriate output. A two-step strand displacement cascade may output multiple strands if we attach two outputs to translator gate t and extend the intermediate strand o using one more distinct history region (as is shown in Fig. 3). Again no sequence constraints exist between the input and the output strands.

In the design of systems of coupled two-step cascades, nucleic acid sequences need to be constructed to avoid unintended interactions. For instance, we can first design all recognition regions to have maximally independent sequences, and then for every translator gate, design maximally independent history regions of its output strands. Then a gate can react with only one recognition region (g-type gates) or intermediate strand (translator gates), ensuring the specificity of interactions. In addition, all sequences must have minimal secondary structure, such as hairpin loops, because such structure can inhibit the desired interactions.

3 Arbitrary Unimolecular Reactions

As a first step we will implement the basic monomolecular reaction $A \xrightarrow{k} B$, such that A and B are single-stranded nucleic acid species with completely independent recognition regions. As we will show, the appropriate monomolecular kinetics can be obtained as a limiting case of the reaction kinetics for a two-step strand displacement cascade:

$$A \xrightarrow{k} B \quad \Rightarrow \quad \begin{cases} As + g \xrightarrow{q_1} o \\ o + t \xrightarrow{q_2} Bs \end{cases}$$

We use the notation As and Bs to mean the implementation of formal species A and B by DNA strands with recognition regions unique for A and B, respectively. We do not include inert waste products when writing the chemical reaction equations. We now discuss the conditions required to make the implementation valid. First, we assume that all non-designed interactions are negligible. We will work in a regime where the concentrations $[g]$ and $[t]$ are in large excess of $[As]$ and $[o]$ so that they remain effectively constant at initial values $[g]_0$ and $[t]_0$ respectively. Then the two-step strand displacement cascade becomes equivalent to a pair of monomolecular reactions:

$$\begin{aligned} As &\xrightarrow{q_1[g]_0} o \\ o &\xrightarrow{q_2[t]_0} Bs \end{aligned}$$

By varying the toehold strength of gate g which determines rate constant q_1, or the initial concentration $[g]_0$, we set $q_1[g]_0$ equal to the formal rate constant k and

attain $d[As]/dt = -k[As]$ as desired. To also ensure that $d[Bs]/dt = k[As]$, we make $q_2[t]_0$ large enough that intermediate strand $[o]$ settles to its quasi-steady-state value $q_1[g]_0[As]/(q_2[t]_0)$ on a much faster time scale than that on which $[As]$ changes. Then $d[Bs]/dt = q_2[t]_0[o] \approx q_1[g]_0[As] = k[As]$ as desired. To make the quasi-steady-state approximation hold in this example, we can increase the relative toehold strength of gate t compared to gate g, or use a much larger initial concentration $[t]_0$ than $[g]_0$.

While experimentally, it may be useful to vary the degree of toehold complementarity affecting q_1 or concentration of gates $[g]_0$ to tune the effective rate constant, for simplicity throughout this paper we control reaction kinetics by tuning toehold strengths, while treating all gates as being present at the same high concentration ξ. Thus we set q_1 equal to k/ξ.

The same scheme can be extended to more complex unimolecular reactions. Reactions with more than one product species (e.g., $A \rightarrow B + C$ or $A \rightarrow 2B$) including catalytic (e.g., $A \rightarrow A + B$) and autocatalytic reactions (e.g., $A \rightarrow 2A$) can be constructed using a translator gate t that releases multiple strands as in Fig. 3. Removing the translator gate altogether allows for unimolecular decay reactions (e.g., $A \rightarrow$). Fractional product stoichiometry (e.g., $A \rightarrow (1/3)B + C$) can be realized using a mixture of translator gates with some fraction having incomplete output strands. For example, reaction $A \rightarrow (1/3)B + C$ can be implemented if 2/3 of translator gates t in Fig. 3 are missing the 7-8 domains. Fractional product stoichiometries are equivalent to multiple reactions in which the same reactants produce different products, where the products are in integer stoichiometries. E.g. the two reactions $A \xrightarrow{2k/3} C$ and $A \xrightarrow{k/3} B + C$ are kinetically equivalent to a single reaction $A \xrightarrow{k} (1/3)B + C$. Conversely, all reactions with the same reactants but different products can always be combined into one reaction with possibly fractional product stoichiometries.

Arbitrary sets of unimolecular reactions can be coupled together by reusing the same recognition region in multiple reactions. Each reaction corresponds to a distinct two-step strand displacement cascade. For example, the system

$$\begin{aligned} A &\xrightarrow{k_1} B + C \\ B &\xrightarrow{k_2} 2B \end{aligned}$$

can be implemented with gate-mediated reactions

$$A \xrightarrow{k_1} B + C \quad \Rightarrow \quad \begin{cases} As + g_1 \xrightarrow{k_1/\xi} o_1 \\ o_1 + t_1 \longrightarrow Bs + Cs \end{cases}$$

$$B \xrightarrow{k_2} 2B \quad \Rightarrow \quad \begin{cases} Bs + g_2 \xrightarrow{k_2/\xi} o_2 \\ o_2 + t_2 \longrightarrow 2Bs \end{cases}$$

where unlabeled rate constants are much larger than k_1/ξ and k_2/ξ and initial concentrations $[t_i]_0$, $[g_i]_0 = \xi$ are high enough to remain effectively constant. The expressions for the DNA gate-mediated reactions in terms of formal rate

constants are obtained from the above analysis. As described in the previous section, the different two-step strand displacement cascades do not have significant undesired interactions. Thus each reaction should proceed without interference from the others except through the desired coupling of input and output strands.

4 Arbitrary Bimolecular Reactions

Consider the basic bimolecular reaction $A + B \xrightarrow{k} C$. Since a reaction between an input strand and a gate can be viewed as being bimolecular, it provides a possible implementation of this reaction. As before, A is mapped to strand As, but now B would have to be mapped to a gate. To emphasize that a gate is mapped to a formal species B we call the gate Bg. As in the case of unimolecular reactions, we can use the translator gate t to ensure sequence independence between recognition regions of As and Cs. The corresponding gate-mediated reactions therefore are:

$$A + B \xrightarrow{k} C \quad \Rightarrow \quad \begin{cases} As + Bg \xrightarrow{k} o \\ o + t \longrightarrow Cs \end{cases}$$

We set the unlabeled rate constant to be very large and the initial concentration of the translator gate $[t]_0 = \xi$ to be big enough to remain effectively constant. Then using the quasi-steady-state approximation for the intermediate strand o as in Section 3 we obtain the desired effective bimolecular reaction rate $k[As][Bg]$.

Having said that, this naive implementation has severe shortcomings. Since strand As must directly bind gate Bg, their sequences are not independent. Thus, gate Bg can react only with input As and cannot participate in reactions with other strand species. Further, it is not always possible to uniquely assign reactants to a gate or a strand. One such example is the following system:

$$\begin{aligned} A &\xrightarrow{k_1} B \\ A + B &\xrightarrow{k_2} C \end{aligned}$$

If we combine the implementation of monomolecular reactions developed in the previous section with the proposed bimolecular scheme, in the resulting system species B is mapped to two different forms, a strand Bs and a gate Bg_2:

$$A \xrightarrow{k_1} B \quad \Rightarrow \quad \begin{cases} As + g_1 \xrightarrow{k_1/\xi} o_1 & \text{(i)} \\ o_1 + t_1 \longrightarrow Bs & \text{(ii)} \end{cases}$$

$$A + B \xrightarrow{k_2} C \quad \Rightarrow \quad \begin{cases} As + Bg_2 \xrightarrow{k_2} o_2 & \text{(iii)} \\ o_2 + t_2 \longrightarrow Cs & \text{(iv)} \end{cases}$$

The concentrations of strand form Bs and gate form Bg_2 are entirely independent, and therefore the DNA reactions do not implement the desired formal chemical system.

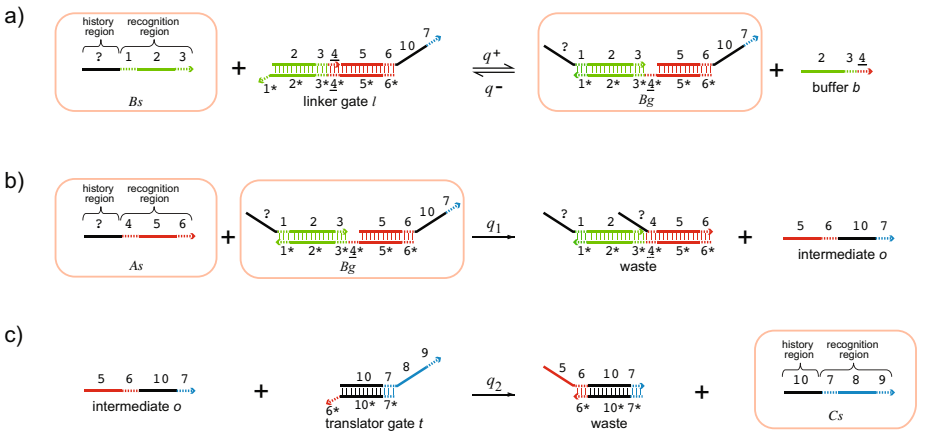


Fig. 4. Molecular implementation of the bimolecular reaction $A+B \rightarrow C$. Orange boxes highlight the DNA species As, Bs, and Cs that correspond to the formal species A, B, and C. Rate constant q_1 can be reduced by decreasing the complementarity between domains 4 and $\underline{4}^*$. The sequences of the recognition regions of input and output strands As, Bs, and Cs (regions 1-2-3, 4-5-6, and 7-8-9, respectively) are completely unrelated to one another. The regime for desired bimolecular kinetics (concentrations of l, b, t and rate constants q^+, q^-, q_1, q_2) is described in the text. **a)** Input strand Bs reversibly binds to the linker gate l forming the activated gate Bg, i.e., $B + l \rightleftharpoons Bg + b$. **b)** Input strand As binds to the activated gate complex Bg and irreversibly releases intermediate strand o through strand displacement. **c)** The intermediate strand o binds translator gate t and by a strand displacement reaction releases the output Cs.

However, if the two forms of B could be interchanged into one another on a time scale that is fast compared to the other reactions in the system, the correct behavior can be restored. We can link the two species Bs and Bg_2 through a fast reversible reaction

$$Bs \underset{k^-}{\overset{k^+}{\rightleftharpoons}} Bg_2$$

such that the two species achieve pseudoequilibrium. Then the formal species B exists in two different forms: $B = \{Bs, Bg_2\}$ and the total concentration of B is $[B] = [Bs] + [Bg_2]$. Let $f(Bg_2) = [Bg_2]/[B]$ be the fraction of B in gate form Bg_2. Under the pseudoequilibrium assumption, $f(Bg_2) = (k^+ + k^-)/k^+$ is a constant. Since the second formal reaction can only use the gate form Bg_2 as a reactant, and not all of B, we scale the rate constant of reaction (iii) by $1/f(Bg_2)$ so that the new rate constant is $k_2/f(Bg_2)$. Then the effective rate of the implementation of $A + B \xrightarrow{k_2} C$ is $(k_2/f(Bg_2))[As][Bg_2] = k_2[A][B]$ as desired. We can easily extend this idea to create a pseudoequilibrium between strand Bs and gates Bg_i for multiple reactions i.

We realize the above reaction establishing pseudoequlibrium between Bs and Bg_2 using a linker gate shown in Fig. 4(a). Strand Bs and buffer strand b

reversibly compete with each other via strand displacement reactions in a toehold exchange process [8]. Thus the reaction establishing pseudoequilibrium is implemented with gates as follows:

$$Bs \underset{k^-}{\overset{k^+}{\rightleftharpoons}} Bg_2 \quad \Rightarrow \quad Bs + l \underset{q^-}{\overset{q^+}{\rightleftharpoons}} Bg_2 + b$$

For the correct first-order kinetics $Bs \underset{k^-}{\overset{k^+}{\rightleftharpoons}} Bg$, the linker gate l and the buffer strand b must be in excess, such that their concentrations remain effectively constant. Then $k^+ = q^+[b]_0$ and $k^- = q^-[l]_0$ where $[b]_0$ and $[l]_0$ are the initial concentrations of the buffer and linker strands respectively. For simplicity we will use $[b_0] = [l]_0 = \xi$ and $q^+ = q^-$.

Lastly, we need to confirm the absence of unintended cross-reactions when implementing multiple coupled bimolecular reactions. As in the simple strand displacement cascades described in Section 2, gates can only react with specific recognition regions or intermediate strands. The exception to this rule is the reaction of gate Bg with the buffer strand b. Gate form Bg can react with any strand with accessible domains ...3-4. Because without loss of generality we can assume that there is only one formal reaction $A + B \rightarrow$ (see discussion of fractional product coefficients in Section 3), and domains 3 and 4 are unique to Bs and As respectively, nothing other than the correct buffer strand can react here.

5 Systematic Construction

In this section we take the ideas developed above and describe a systematic algorithm for compiling arbitrary unimolecular and bimolecular reactions into DNA gate-mediated chemistry. This algorithm is used in the next section to implement a Rössler attractor chaotic chemical system.

Without loss of generality we assume that every reaction has a unique combination of reactants. For example, the pair of reactions $A+B \overset{k_1}{\rightarrow} C$ and $A+B \overset{k_2}{\rightarrow} D$ are combined into a single reaction $A+B \overset{k_1+k_2}{\longrightarrow} (k_1/(k_1+k_2))C+(k_1/(k_1+k_2))D$ (see the discussion of fractional product coefficients in Section 3). Let i index reactions and $X_j \in \{A, B, C, \dots\}$ index species. Let $f(X_j s)$ be the fraction of X_j in strand form $X_j s$. Similarly let $f(X_j g_i)$ be the fraction of X_j in gate form $X_j g_i$.

Consider any unimolecular formal reaction i. Write the reaction as $X_1 \overset{k}{\rightarrow} \alpha_2 \cdot X_2 + \cdots + \alpha_n \cdot X_n$, where $0 < \alpha \leq 1$. We implement this reaction by a two-step strand displacement cascade (Fig. 3), modeled by the DNA gate reactions below (where we omit inert waste products, and combine all strands with the same recognition regions into a single species).

$$\begin{aligned} X_1 s + g_i &\overset{k'}{\longrightarrow} o_i \\ o_i + t_i &\longrightarrow \alpha_2 \cdot X_2 s + \cdots + \alpha_n \cdot X_n s. \end{aligned}$$

Product fractions α_j are set by preparing translator gate t_i with α_j fraction of complete and $1-\alpha_j$ incomplete output strands for $X_j s$ as discussed in Section 3. Unlabeled rate constants as well as the initial concentrations $[g_i]_0 = [t_i]_0 = \xi$ are as high as possible. Rate constant k' is set to $\frac{k}{\xi f(X_1 s)}$ by varying the degree of complementarity of the toehold on gate g_i with the toehold on strand $X_1 s$. Note that by following the argument of Section 3, and using the fact that $[X_1] = [X_1 s]/f(X_1 s)$, the effective rate of this reaction is $k'[X_1 s]\xi = k[X_1]$ as desired.

Consider any bimolecular formal reaction i. Write the reaction as $X_1 + X_2 \xrightarrow{k} \alpha_3 \cdot X_3 + \cdots + \alpha_n \cdot X_n$, where $0 < \alpha \leq 1$. We implement this reaction by a linker gate mechanism combined with the two-step strand displacement cascade (Fig. 4) and is modeled by the DNA gate reactions below (where we again omit inert waste products, and combine all strands with the same recognition regions into a single species).

$$\begin{aligned} X_1 s + l_i &\rightleftharpoons X_1 g_i + b_i \\ X_2 s + X_1 g_i &\xrightarrow{k'} o_i \\ o_i + t_i &\longrightarrow \alpha_3 \cdot X_3 s + \cdots + \alpha_n \cdot X_n s \end{aligned}$$

Product fractions α_j are set by preparing translator gate t_i with α_j fraction of complete and $1-\alpha_j$ incomplete output strands for $X_j s$ as before. Unlabeled rate constants are as high as possible, with the forward and reverse rates of the first reaction being equal. Rate constant k' is set to $\frac{k}{f(X_2 s) f(X_1 g_i)}$ by varying the degree of complementarity of the toehold on $X_1 g_i$ with the toehold on strand $X_2 s$. The initial concentrations $[l_i]_0 = [b_i]_0 = [t_i]_0 = \xi$ are as high as possible. Following the argument of Section 4, and using the facts that $[X_2] = [X_2 s]/f(X_2 s)$ and $[X_1] = [X_1 g_i]/f(X_1 g_i)$, we see that the effective rate of this reaction is $k'[X_2 s][X_1 g_i] = k[X_1][X_2]$ as desired.

With the above construction, determining $f(X_j s)$ and $f(X_j g_i)$ is easy: for every i, j, $f(X_j s) = f(X_j g_i) = 1/(m+1)$ where m is the number of bimolecular reactions in which X_j appears as the first reactant.

The sequences of the DNA components can be designed as follows. First, for all formal species design maximally independent recognition regions with minimum secondary structure. Then, for each formal reaction, design the history regions for all products of that reaction to be maximally independent and have minimum secondary structure. At this point all auxiliary DNA species are fully specified. Significant unintended interactions between auxiliary species participating in different formal reactions cannot occur by the arguments in Sections 2 and 4. The system is initiated by adding appropriate starting amounts of the formal species in single-stranded form with arbitrary history regions.

6 Example

We illustrate our method of using DNA-based chemistry to implement arbitrary formal systems of coupled chemical equations on the chaotic system due

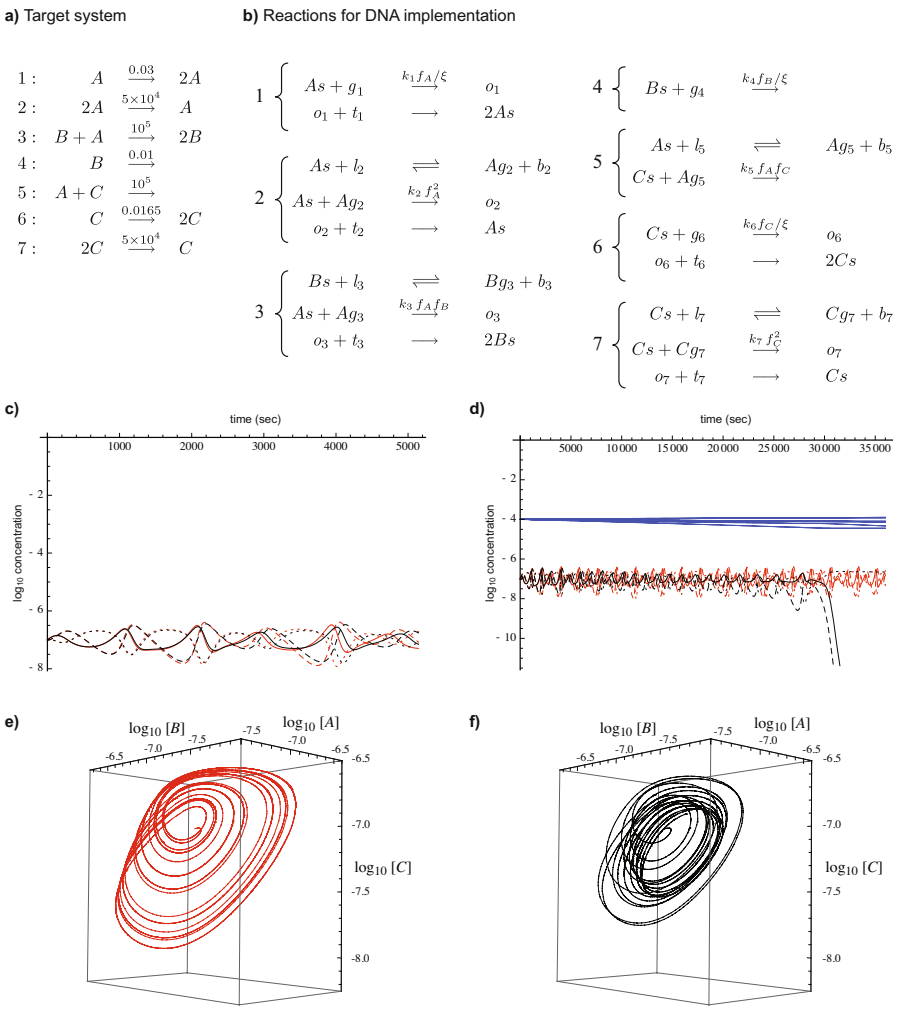


Fig. 5. Rössler attractor example. **(a)** The formal chemical reaction system to be implemented. **(b)** Reactions modeling our DNA implementation. Each bracket implements the formal reaction with the number indicated. Here k_1 through k_7 are the original rate constants for reactions 1 through 7 as in (a). Multiplicative factors $f_A = 1/f(As) = 1/f(Ag_2) = 1/f(Ag_5) = 3$, $f_B = 1/f(Bs) = 1/f(Bg_3) = 2$, $f_C = 1/f(Cs) = 1/f(Cg_7) = 2$. We use initial concentration of the gates and buffer strands $\xi = 10^{-4}$. Unlabeled rate constants are 10^5. **(c)** Plot of the log-concentrations of A (solid), B (dashed), C (dotted) for the original system (red), as well as their modeled concentrations (black). **(d)** Longer time plot showing also the log-concentrations of g_i (blue, decreasing) and b_i (blue, increasing). **(e,f)** Trajectories of the original system and DNA implementation in the 3-dimensional phase-space (first 5 hours).

to Willamowsky and Rössler [4]. We start with the following formal reactions, where the rate constants are from Ref. [11]:

$$
\begin{aligned}
1: &\quad A \xrightarrow{30} 2A \\
2: &\quad 2A \xrightarrow{0.5} A \\
3: &\quad B + A \xrightarrow{1} 2B \\
4: &\quad B \xrightarrow{10} \\
5: &\quad A + C \xrightarrow{1} \\
6: &\quad C \xrightarrow{16.5} 2C \\
7: &\quad 2C \xrightarrow{0.5} C
\end{aligned}
$$

The strange attractor for the concentrations of A, B, and C is in the range of about 0–40.

First we scale this system into a regime realistic for DNA-based chemistry which constrains reaction rates and concentrations. Second order rate constants for strand displacement reactions can be approximately in the range 0–$10^6/M/s$, with their value determined by the degree of toehold complementarity [10]. Typical experimental concentrations are on the order of 0–$10^{-3}M$. Similar to experimental implementations of other dynamical chemical systems, a flow reactor may be used to replenish the stock of unreacted gates and remove waste to maintain the appropriate reaction conditions [3]. This may make it possible to use lower gate concentrations.

Clearly, by scaling all rate constants by the same factor we simply speed up or slow down the system without affecting the dynamical behavior. We can scale the concentrations at which the chaotic system operates by scaling the bimolecular rate constants differently from the unimolecular ones. In general if $[X_j](t)$ are solutions to differential equations arising from a set of unimolecular and bimolecular reactions, then $\alpha[X_j](t)$ are solutions to the differential equations arising from the same set of reactions but in which we divide all bimolecular rate constants by α. We first slow down the system by multiplying all rate constants by 10^{-3}, and then use concentration scaling factor $\alpha = 10^{-8}$, obtaining the rate constants in Fig. 5(a).

Applying our construction yields a DNA implementation governed by the equations in Fig. 5(b). Simulations confirm (Fig. 5(c, d)) that the DNA implementation behaves very close to the formal system (a) until the depletion of linker gates l_i and the buildup of buffer strands b_i sufficiently alters the effective rate constants, gradually decoupling the gate implementation from the target system.

7 Conclusion

We have proposed a method for approximating an arbitrary system of coupled unimolecular and bimolecular chemical reactions using DNA-based chemistry. Our construction takes advantage of cascades of strand displacement reactions [6], and elementary techniques of approximation in chemical kinetics. Each formal species

occurring in the system of chemical reactions is represented as a set of strands and gates. The multiform representation is necessary because it is not always possible to find a single DNA species that is capable of participating in all reactions involving a given formal species. However, the different forms are constructed to be in equilibrium with each other and thus participate in kinetics as if they were a single species, up to a scaling of rate constants.

While we have taken care to provide a systematic algorithm for compiling a set of chemical reactions into DNA, in practice it may often be possible and preferable to reduce the complexity by optimizing the construction for the particular system of interest. For example, in many cases complete sequence independence between strands may not be necessary, possibly allowing one to eliminate some translator gates. Similarly, pseudoequilibrium linkage is unnecessary if mapping a species directly to a strand or gate does not cause problems.

For simplicity in our systematic construction rate constants are set by the degree of sequence complementarity between toehold domains. However, there are many other degrees of freedom available such as the relative concentrations of linker gate and buffer strand for bimolecular reactions. Probably in practice, toehold domains provide a rough order of magnitude control over formal rate constants, while adjusting concentrations of auxiliary species allows fine-tuning them.

Acknowledgments. This work was supported by NSF Grant 0728703. GS acknowledges support from the Swiss National Science Foundation and the Burroughs Wellcome Fund. We thank D. Zhang, J. Schaeffer, and M. Magnasco for useful discussions.

References

1. Gavalas, G.R.: Nonlinear Differential Equations of Chemically Reacting Systems. Springer, Heidelberg (1968)
2. Scott, S.K.: Chemical Chaos. Oxford University Press, Oxford (1991)
3. Epstein, I.R., Pojman, J.A.: An Introduction to Nonlinear Chemical Dynamics: Oscillations, Waves, Patterns, and Chaos. Oxford University Press, Oxford (1998)
4. Willamowski, K.D., Rössler, O.E.: Irregular oscillations in a realistic abstract quadratic mass action system. Z. Naturforsch A 35, 317–318 (1980)
5. Yurke, B., Turberfield, A.J., Mills Jr., A.P., Simmel, F.C., Neumann, J.L.: A DNA-fuelled molecular machine made of DNA. Nature 406(6796), 605–608 (2000)
6. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)
7. Seelig, G., Yurke, B., Winfree, E.: Catalyzed relaxation of a metastable DNA fuel. Phys. Rev. Lett. 90, 118102–118111 (2006)
8. Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318(5853), 1121 (2007)
9. Green, C., Tibbetts, C.: Reassociation rate limited displacement of DNA strands by branch migration. Nucleic Acids Research 9, 1905–1918 (1981)
10. Yurke, B., Mills, A.P.: Using DNA to Power Nanostructures. Genetic Programming and Evolvable Machines 4(2), 111–122 (2003)
11. Gaspard, P.: Encyclopedia of Nonlinear Science. In: "Rössler Systems", pp. 808–811. Routledge (2005)

A Simple DNA Gate Motif for Synthesizing Large-Scale Circuits

(Extended Abstract)

Lulu Qian and Erik Winfree

Computer Science, Computation & Neural Systems, and Bioengineering
California Institute of Technology, Pasadena, CA 91125, USA

Abstract. The prospects of programming molecular systems to perform complex autonomous tasks has motivated research into the design of synthetic biochemical circuits. Of particular interest to us are cell-free nucleic acid systems that exploit non-covalent hybridization and strand displacement reactions to create cascades that implement digital and analog circuits. To date, circuits involving at most tens of gates have been demonstrated experimentally. Here, we propose a DNA catalytic gate architecture that appears suitable for practical synthesis of large-scale circuits involving possibly thousands of gates.

1 Introduction

DNA-based catalysts [1,2,3] and logic gates [4,5,6] have been proposed as general-purpose components for synthesizing chemical circuits [7,8,6] with applications in medical therapeutics, nanotechnology, and embedded control of chemical reactions. Progress in this direction will depend upon three future advances: (1) Developing input/output interfaces between the DNA circuits and biomedically relevant molecules [9], DNA nanomachines [10,11], and general chemistries [12,13]; (2) Developing DNA circuit construction techniques that scale up so that large and interesting circuits can be systematically created; and (3) Extending the DNA programming methodology beyond well-mixed solutions to include spatial structures at the molecular [14,15,16] and macroscopic scales [17,18,19].

In this paper we focus on the second challenge. We introduce a DNA catalytic gate motif suitable for scaling up to large circuits, along with an abstract circuit formalism that aids the design and understanding of circuit behavior. To illustrate the potential of this approach, we show how to implement arbitrary feedforward digital logic circuits, arbitrary relay circuits, and analog circuits exhibiting a variety of temporal dynamics. Thanks to the modular design of the gate motif, sequence design is straightforward. Furthermore, we argue that synthesis and preparation of circuit components can be parallel and scalable. Our estimates suggest that circuits involving thousands of distinct gates may be possible.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 70–89, 2009.

2 A Simple Catalytic Gate with a Threshold

Each catalytic gate may be represented abstractly as a node with one or more wires connected to the left and right sides (Fig. 1a). Each wire will correspond to a single-stranded DNA molecule with a specific sequence, called a signal strand, which may be absent (an inactive wire) or present at some concentration (an active wire). Each node will correspond to a gate strand that may bind to or release signal strands on demand. As described below, an active wire on one side of a gate can catalyze the exchange of activity on wires on the other side. Furthermore, there may be a threshold that must be exceeded before catalysis occurs. While the catalytic gates are intrinsically symmetric, we will typically configure them asymmetrically, with an input side and an output side. When connected into circuits involving many interacting catalytic gates, complex circuit behavior can be obtained.

The DNA implementation of the elementary gate illustrated in Fig. 1a is shown in Fig. 1bc. The three wires correspond to free signal strands S_2TS_3 ("fuel"), S_1TS_3 ("output"), and S_3TS_4 ("input"), while the node indicates complexes involving the gate base strand $T'S'_3T'$. The state of the network is indicated by writing the amounts of each species in appropriate locations: the (relative) concentrations of signals on the wires, and the (relative) concentrations of gate:signal complexes within the nodes at the ends of the corresponding wires. The concentration of a threshold complex is written as a negative number in the location for the corresponding gate:signal complex for the signal it is absorbing. Thus, Fig. 1a specifies the gate:output complex, fuel signal strand, and input signal strand of Fig. 1b with respective concentrations $10x$, $10x$, and $1x$; the threshold gate absorbing the input strand, shown in Fig. 1c, is present at $0.5x$; where $1x$ is a standard concentration, perhaps 30 nM. With these initial concentrations, the input strand will first overcome the threshold and then act catalytically to facilitate the equilibration of the output strand and the fuel strand to approximately $5.1x$ each. (This level can be estimated by noting that by symmetry the two wires will have similar activity, and that the total concentration on each wire and within the gate remains constant. Detailed formulas are provided in Section 3.)

The reaction mechanism is a simplified version of the entropy-driven catalytic gate introduced in [3,20]. Fig. 1b shows reactions between the three signal strands and the gate complexes involving those signal strands. Fig. 1c shows the threshold gate that absorbs the input strand. Example sequences are shown in Fig. 1d. The fundamental operation is toehold exchange, a toehold-mediated strand displacement reaction that results in a free right-side signal strand replacing a bound left-side signal strand, or visa-versa. For example, input signal strand S_3TS_4 can bind to gate:output complex $T'S'_3T'$:S_1TS_3 via toehold domain T, unbiased three-strand branch migration within the S_3 recognition domain will lead to a state where either input strand S_3TS_4 or output strand S_1TS_3 is bound by no more than the short toehold domain T, and that signal strand will quickly fall off. (For simplicity, we ignore the abortive attempts, and only consider those reactions in which strand replacement occurs.) At this

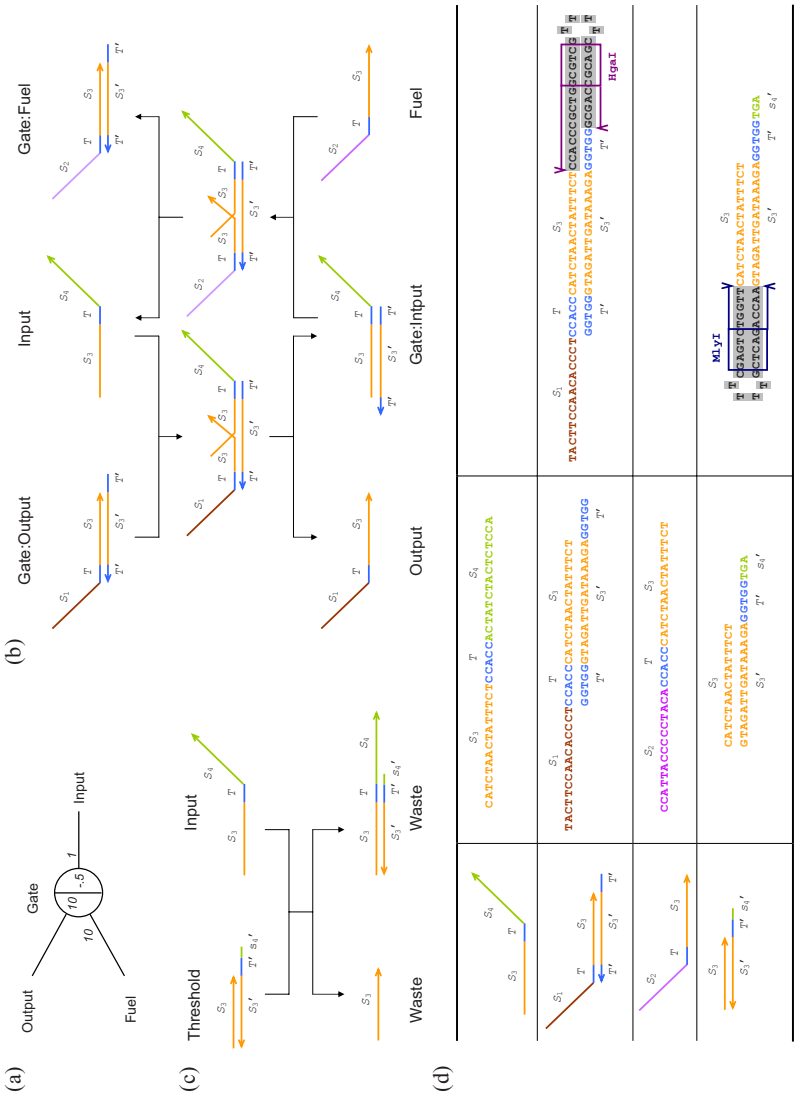


Fig. 1. The DNA motif for catalytic "seesaw" gates. (a) Abstract gate diagram. (b) The DNA gate motif and reaction mechanism. S_1, S_2, S_3, S_4 are the recognition domains; T is the toehold domain; T' is the Watson-Crick complement of T, etc. Arrowheads mark the $3'$ ends of strands. Black lines indicate elementary reactions via alternating strand displacement and toehold exchange. All reactions are reversible and unbiased; arrowheads indicate the dominant flows for the initial concentrations shown in (a). (c) The threshold motif and reaction mechanism. The toehold is extended by three bases (s_4', the complement of the first three $5'$ bases of S_4), providing an increased rate constant relative to the gate itself. (d) Example sequences. Gate complexes and signal molecules are shown in their operational form (second column). To prepare gates from single strands of DNA, hairpin precursor molecules (third column) are cleaved by restriction enzymes to create functional gates. Recognition domain sequences are 15 nt and the toehold domain sequence is 5 nt.

point, the gate base strand has its left toehold revealed, and its right toehold hidden: toehold exchange has been achieved, yielding the release of an output strand and the production of a gate:input complex. An analogous process allows the fuel strand to similarly displace the input strand from the new gate:input complex, completing a catalytic cycle that has the net effect of exchanging one left-side signal strand in solution (the fuel) for another left-side signal strand (the output). Note, however, that a free left-side signal strand (e.g. the fuel) cannot directly replace a bound left-side signal strand (e.g. the output bound to the gate). It can only do so in the presence of a right-side signal strand (e.g. the input) by an indirect sequence of events: the right-side signal strand displaces the bound left-side signal strand, and in turn the other left-side signal strand displaces the now-bound right-side strand. Thus, the right-side signal strand has catalyzed the exchange of the two left-side signal strands, which cannot exchange in its absence. This back-and-forth motion is the inspiration for our name for this motif: "seesaw gates".

In the above discussion, recognition domains S_1, S_2, and S_4 played no active role in the mechanism. Rather, these domains allow the signal strands to interact with other gates and thus dictate connectivity within a circuit. For example, output strand S_1TS_3 could also serve as a right-side signal strand for another gate with base strand $T'S_1'T'$, not shown here. However, while bound to base strand $T'S_3'T'$, signal strand S_1TS_3 cannot act as a catalyst for the downstream gate because its toehold domain is sequestered. Of course, if it is released into solution by a strand displacement reaction, then it is free to act upon its target gate $T'S_1'T'$. In summary, each gate base strand consists of a single recognition domain identifying the gate, flanked by two toehold domains, only one of which is exposed at any given time; while each signal strand consists of two recognition domains identifying the two gates it connects, separated by a central toehold domain that is sequestered and thus inert when the signal strand is bound to a gate base strand.

In addition to providing an ON/OFF switch for catalysis, threshold behavior can be helpful for reliable circuit function by cleaning up after leaky reactions. Fig. 1c shows a threshold gate that serves as a competitive inhibitor of the signal strand S_3TS_4. Because of the slightly longer toehold on the threshold gate, the signal strand will react faster with the threshold gate than with the original gate complex. (Toehold-mediated strand displacement reaction rates depend exponentially on toehold length, for short toeholds [21]. In this paper, we conservatively assume a 10-fold speed-up specific to the targeted signal strand, which is less than the ratio between the maximal rate constant for DNA hybridization and the measured rate constant for 5 nt toeholds [21]. In gates with more than one input, there will be some crosstalk if both inputs have a threshold, because the threshold for the first input can absorb the second input signal strand and visa versa, albeit at a 10-fold slower speed. Thus, after one input exceeds its own threshold, it will continue to be absorbed by the other threshold – at the same rate with which it catalyzes the exchange of fuel and output signal. For the simulations presented in this paper we neglect this crosstalk, but simulations including the crosstalk reactions show

qualitatively similar results, albeit less ideal. The problem also could be avoided by designing circuits in which only gates with a single input are allowed to have a threshold.) Once the signal strand has reacted with threshold gate, it will never be released because all relevant toeholds are sequestered – effectively, inert waste has been produced. Only after all the threshold gates have been used up, then the remaining signal strands can react (perhaps catalytically) with the original gate.

This consistent and modular motif makes systematic construction of circuits logically straightforward, as discussed below, and further makes laboratory procedures for synthesizing gates and circuits plausible to carry out on a large scale. A substantial difficulty with our previous work [6,3] was that each gate substrate was a complex of multiple strands that had to be separately annealed together, and each complex had to be purified to remove excess single-stranded species and malformed gate substrates. Here, we aim to simultaneously prepare all gate complexes together in a single test tube; to do so, we must ensure that different gate species do not interact, and that the strands needed to form a given gate complex find each other efficiently. For our solution, we draw inspiration from the observation [22,23] that mixtures of hairpin molecules, when annealed, are likely to form non-interacting intramolecular hairpins even if at room temperature there exist lower free energy states involving intermolecular complexes. This occurs because the intramolecular hairpins are typically stable at some moderately high temperature, above the melting temperature of the intermolecular complexes – thus, during annealing, the hairpins form first and become kinetically trapped. The implication for gates is that if each gate species can be synthesized initially as a hairpin precursor, annealing all such gate precursors in a single reaction will result in a high yield of properly formed non-interacting molecules. Fig. 1d shows our realization of this scheme. After annealing, incubation with appropriate restriction enzymes removes the now-undesired linker subsequence, resulting in a well-formed complex of two strands. The entire solution could be purified by gel, since all gates are the same size; all threshold gates could be purified similarly. One way or another, making circuit function robust to sloppy parallel synthesis methods will be crucial to scaling up existing DNA circuits to hundreds or thousands of gates.

3 Abstract Circuit Formalism and Function

The abstract network representation introduced in Fig. 1a facilitates concise reasoning about circuits involving many interacting catalytic gates. In general, a circuit consists of a number of gate nodes and a number of wires between gate nodes. Each wire connects exactly two gates (Fig. 2b). Each gate consists of a left side and a right side, and it may connect to any number of wires on each side (Fig. 2a). Virtual gates (dotted lines) are gates whose total concentration is zero; they are used solely as syntactic sugar to provide a consistent naming scheme for wires that are connected on just one side, such as the input and output wires. When it is necessary to make the distinction, gates with non-zero concentration are referred to as real gates (solid lines). A seesaw gate circuit then consists of

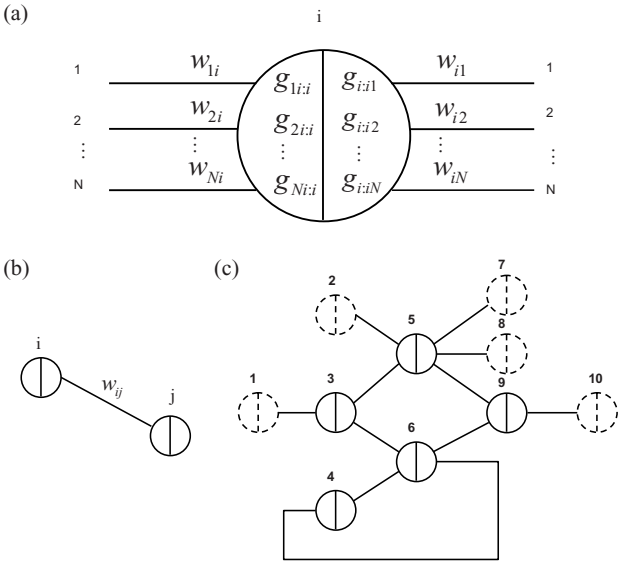


Fig. 2. Abstract diagrams for seesaw gate circuits. (a) The general form of a gate node. Each gate may be connected to many wires on each side, potentially all N nodes in the network, including itself. (b) The general form of a wire. Each wire is specifically connected on its left end to the right side of a gate node, and connected on its right end to the left side of a gate node. (c) An example circuit with five real gates, five virtual gates, and eleven wires. Each wire is identified by the two gates it connects; thus the virtual gates serve to provide full names to their incident wires.

a number of gate nodes connected by wires between their left and right sides (Fig. 2c).

The implementation of any such circuit diagram using DNA molecules is straightforward. (1) For a circuit with N gates, begin by chosing a set of N sufficiently distinct m-mer sequences $S_1, S_2, \ldots, S_N$. Sequence length m must increase (approximately logarithmically) with N; the value $m = 15$ was used in Fig. 1. Use the same univeral toehold sequence T in all cases. (2) To each gate i, associate the gate base strand $T'S_iT'$. To each wire ij, associate the wire signal strand S_iTS_j. This determines the sequences for all molecules in the circuit. (3) Based on the desired initial concentrations of each signal strand, gate complex, and threshold, synthesize the appropriate hairpin molecules, cleave them with restriction enzymes, and purify the desired complexes. (4) To run the system, add the appropriate input signal strands and observe (e.g. by fluorescence) the target output signal strands.

For standard nomenclature, a wire between gates i and j, with gate i on the left, is called w_{ij}, representing the free signal strand S_iTS_j. Likewise, a gate i base strand $T'S_i'T'$ bound to the left end of wire ij signal strand S_iTS_j is referred to as $g_{i:ij}$; similarly $g_{ki:i}$ is used when the right end of signal strand S_kTS_i is bound to gate i. Note that $g_{i:ii}$ and $g_{ii:i}$ refer to distinct complexes: in the former

case, the signal strand is bound on its left end, leaving the gate's left toehold accessible; while in the latter case the signal strand is bound on its right end, leaving the gate's right toehold accessible. The threshold complex for wire ij into gate j is called $th_{ij:j}$, and similarly for $th_{i:ij}$. In a slight abuse of notation, in the formulas below, w_{ij} refers both to the signal strand molecule itself and to the concentration of that species, and similarly for the gate and threshold complexes. Finally, note that although each gate has an identifying number corresponding to its base strand sequence, circuit diagrams may be drawn without any numerical annotation as the names are not relevant to circuit function.

Because all components are in a standard form, the set of chemical reactions modelling the toehold exchange steps and threshold absorption steps can be written concisely. For all $i, j, k \in \{1, 2, \ldots, N\}$,

$$w_{ji} + g_{i:ik} \underset{k_0}{\overset{k_0}{\rightleftharpoons}} g_{ji:i} + w_{ik} \tag{1}$$

$$w_{ij} + th_{ij:j} \overset{k_1}{\rightarrow} waste \overset{k_1}{\leftarrow} th_{i:ij} + w_{ij} \tag{2}$$

where the variables refer to the molecular species. Using standard mass action chemical kinetics, this gives rise to a system of ordinary differential equations (ODEs) for the dynamics. In the following, w_{ij} and similar terms refer to the concentrations of the respective species, rather than to the species themselves.

$$\begin{aligned}\frac{d\,w_{ij}}{dt} = {} & k_0 \left(\sum_{n=1}^{N} w_{ni} \cdot g_{i:ij} + w_{jn} \cdot g_{ij:j} - w_{ij} \cdot g_{ni:i} - w_{ij} \cdot g_{j:jn} \right) \\ & -k_1 \left(w_{ij} \cdot th_{i:ij} + w_{ij} \cdot th_{ij:j} \right)\end{aligned} \tag{3}$$

$$\frac{d\,g_{i:ij}}{dt} = k_0 \left(\sum_{n=1}^{N} w_{ij} \cdot g_{ni:i} - w_{ni} \cdot g_{i:ij} \right) \tag{4}$$

$$\frac{d\,g_{ij:j}}{dt} = k_0 \left(\sum_{n=1}^{N} w_{ij} \cdot g_{j:jn} - w_{jn} \cdot g_{ij:j} \right)$$

$$\frac{d\,th_{i:ij}}{dt} = -k_1 \cdot w_{ij} \cdot th_{i:ij} \tag{5}$$

$$\frac{d\,th_{ij:j}}{dt} = -k_1 \cdot w_{ij} \cdot th_{ij:j}$$

These dynamics have conserved quantities for each gate node i and for each signal wire ij:

$$g_{i:ij} - th_{i:ij} + w_{ij} + g_{ij:j} - th_{ij:j} \overset{def}{=} c_{ij} \tag{6}$$

$$\sum_{n=1}^{N} g_{ni:i} + g_{i:in} \overset{def}{=} c_i$$

$$\frac{d\,c_i}{dt} = \frac{d\,c_{ij}}{dt} = 0$$

Furthermore, the equilibrium (if it is obtained) enforces a simple relationship between the free and bound forms of the signal strands, namely that their ratio with respect to a particular gate must be identical for all wires connected to that gate. This follows immediately from Eq. 1 and the detailed balance equation $k_0 \cdot w_{ji} \cdot g_{i:ik} = k_0 \cdot g_{ji:i} \cdot w_{ik}$. To wit, for each gate i, at equilibrium all wires achieve the ratio

$$\frac{w_{1i}}{g_{1i:i}} = \frac{w_{2i}}{g_{2i:i}} = \cdots = \frac{w_{Ni}}{g_{Ni:i}} = \frac{w_{i1}}{g_{i:i1}} = \frac{w_{i2}}{g_{i:i2}} = \cdots = \frac{w_{iN}}{g_{i:iN}} \stackrel{def}{=} r_i \tag{7}$$

As an example, we can calculate the equilibrium concentrations for a single real gate, i, connected only to virtual gates, as in Fig. 1a. The equilibrium wire concentration $w_{ij}(\infty)$ depends upon the gate ratio and the initial concentrations on that wire:

$$\begin{aligned} w_{ij}(\infty) &= \frac{w_{ij}(\infty)}{g_{i:ij}(\infty) + w_{ij}(\infty)}(g_{i:ij}(\infty) + w_{ij}(\infty)) \\ &= \frac{r_i}{1 + r_i}(g_{i:ij}(0) + w_{ij}(0) - th_{i:ij}(0)) = \frac{r_i}{1 + r_i} c_{ij} \end{aligned}$$

The equilibrium gate ratio r_i can be calculated from the conserved quantities of Eqs. 6 and the equilibrium constraints of Eq. 7:

$$c_{ij} = w_{ij}(1 + 1/r_i) \qquad\qquad c_i = (1/r_i) \sum_j w_{ji} + w_{ij}$$

from which it follows with a little algebra that

$$\frac{r_i}{1 + r_i} = 1 - \frac{c_i}{\sum_j c_{ji} + c_{ij}}.$$

For the concentrations given in Fig. 1a, we can work out $\frac{r_i}{1+r_i} = 1 - \frac{10}{10+10+1-0.5} \approx 0.51$, so at equilibrium $w_{13} = w_{23} \approx 10x \times 0.51 = 5.1x$, as claimed earlier.

The uniformity of components in seesaw gate circuits also facilitates their simulation. Using a general purpose mass action chemical kinetics simulator written by David Soloveichik in Mathematica, we have written routines for concisely representing, constructing, and simulating models of seesaw gate circuits.

4 Feedforward Digital Logic Circuits

Digital logic has two compelling features for circuit construction: first, it has proven to be very expressive for the synthesis of a wide range of desired behaviors; and second, it is intrinsically robust to a variety of manufacturing and operational defects. The basic principle underlying digital logic is that an intrinsically analog signal carrier may be considered simply to be either ON or OFF if at each stage of computation, signals are either pushed toward the ideal ON value or pushed toward the ideal OFF value. This is called signal restoration,

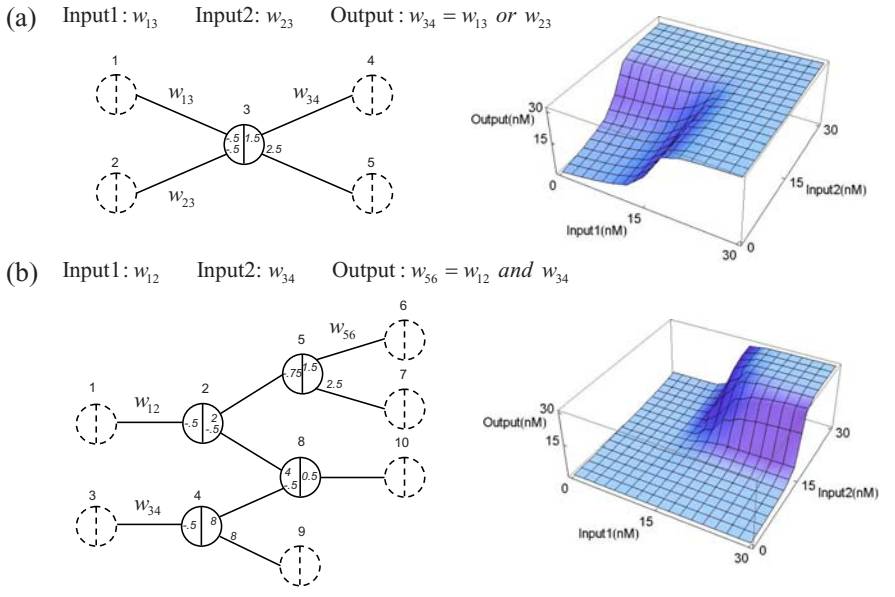


Fig. 3. The circuit diagram and input/output behavior of boolean logic gates. (a) A two-input OR gate. (b) A two-input AND gate. Simulations are performed with the reference concentration $1x = 30$ nM.

because if noise or device imperfections slightly corrupt a signal, that deviation from ideal behavior is cleaned up (perhaps not completely) by subsequent processing without altering the interpretation of the signal as ON or OFF. In this section we will consider a digital abstraction in which signal concentrations below $0.1x$ are considered OFF, while signal concentrations above $0.9x$ are considered ON. Intermediate signal levels – in the transition region – are considered a fault because proper behavior of logic gates is no longer guaranteed for input levels in the intermediate range.

Feedforward digital circuits are an important class of combinational circuits, i.e. memoryless circuits in which the inputs uniquely determine the outputs without need to re-use any parts of the circuit. It is well known that any boolean function can be computed (usually quite efficiently) by a well-designed feedforward circuit built from AND, OR, and NOT gates. Interestingly, with a small cost in circuit size and a minor change in representation, called "dual-rail logic", AND and OR by themselves suffice. We will therefore provide constructions for these basic operations, in addition to signal fan-in, fan-out, routing, and thresholding that may be required to paste the computing gates together into large circuits.

The OR gate (Fig. 3a) is a simple extension of the basic catalyst (Fig. 1a) in which there are two input wires, 13 and 23, that serve as catalysts for the release onto output wire 34, driven by exchange with the "power supply" wire 35. Under the assumption that the threshold gate reaction rate constant, k_1, is

10 times faster than the gate reaction rate constant k_0, clean digital behavior is seen in the model. For larger ratios k_1/k_0, the sharpness of the threshold increases correspondingly.

The AND gate is a little trickier. Essentially, the idea here is similar to what is done with transistors: control flow from a power supply using two regulatory gates in series, and only if both gates are active will the output be affected. However, in our case there is no actual flow, just a sloshing back and forth of the signals on the seesaw gates. (Recall that there is a constant total concentration of a given signal strand either free in solution or bound to the gate at either end.) That said, consider the circuit shown in Fig. 3b. First, if input signals 12 and/or 34 are OFF (i.e. between $0x$ and $0.1x$), then they will be absorbed by their respective thresholds. Now consider that if input 34 is ON (i.e. between $0.9x$ and $1x$), then it can catalyze the power supply 49 to push strands onto the wire 48. Gate 8 serves as a signal reflector, effectively routing a signal emanating from the right side of a gate to arrive at the right side of another gate. (Recall that wires can only connect left sides to right sides, so this cannot be achieved directly.) Once the input to gate 8 exceeds its threshold, the right-side catalyst will be released, in turn catalytically releasing strand 28 to impinge on gate 2. But, if input 12 remains OFF, there will be no catalysis and therefore no signal will be pushed onto wire 25. On the other hand, if input 12 is also ON, the signal on 28 will provide power to push signal onto 25, which will in turn exceed the threshold for gate 5 and catalyze the production of output 56. The final case we must consider is if input 34 is OFF but input 12 is ON. In this case, there is at most $0.5x$ input 12 remaining after absorption by the threshold; some of it will end up bound to gate 2 after having pushed some 25 onto the output wire; however, at most a total of $0.5x$ can be pushed onto 25, and this will be absorbed by the gate 5 threshold. Because the total signal on gate 2 must remain at $2x$, we see that all signal strands will remain bound to gate 2, the wires 12 and 25 will be empty, and no output 56 will be produced. This is of course considered OFF by the digital abstraction. Full simulations bearing out these intuitive considerations are shown.

Fan-in and fan-out are handled easily using these constructions. More than two inputs (additional fan-in) to an OR gate simply requires additional wires connecting to the seesaw gate node, slight adjustments to the initial concentrations, and placing the threshold in a separate subsequent gate (to avoid the threshold crosstalk problem). More than one output (fan-out) from an OR gate is correspondingly simple: connect more output wires. Fan-out from an AND gate is analogous, but additional fan-in poses problems: a longer sequence of regulatory gates in series would require the initial power supply concentration to be considerably larger. Thus it is preferable to construct a multi-input AND gate as a binary tree of two-input AND gates.

Because NOT gates appear to be difficult to implement directly (although we have not yet outlawed the possibility), the dual-rail convention is used. The following procedure may be used to convert a single-wire digital logic circuit (that may use AND, OR, NOT, NAND, and NOR) into an equivalent dual-rail

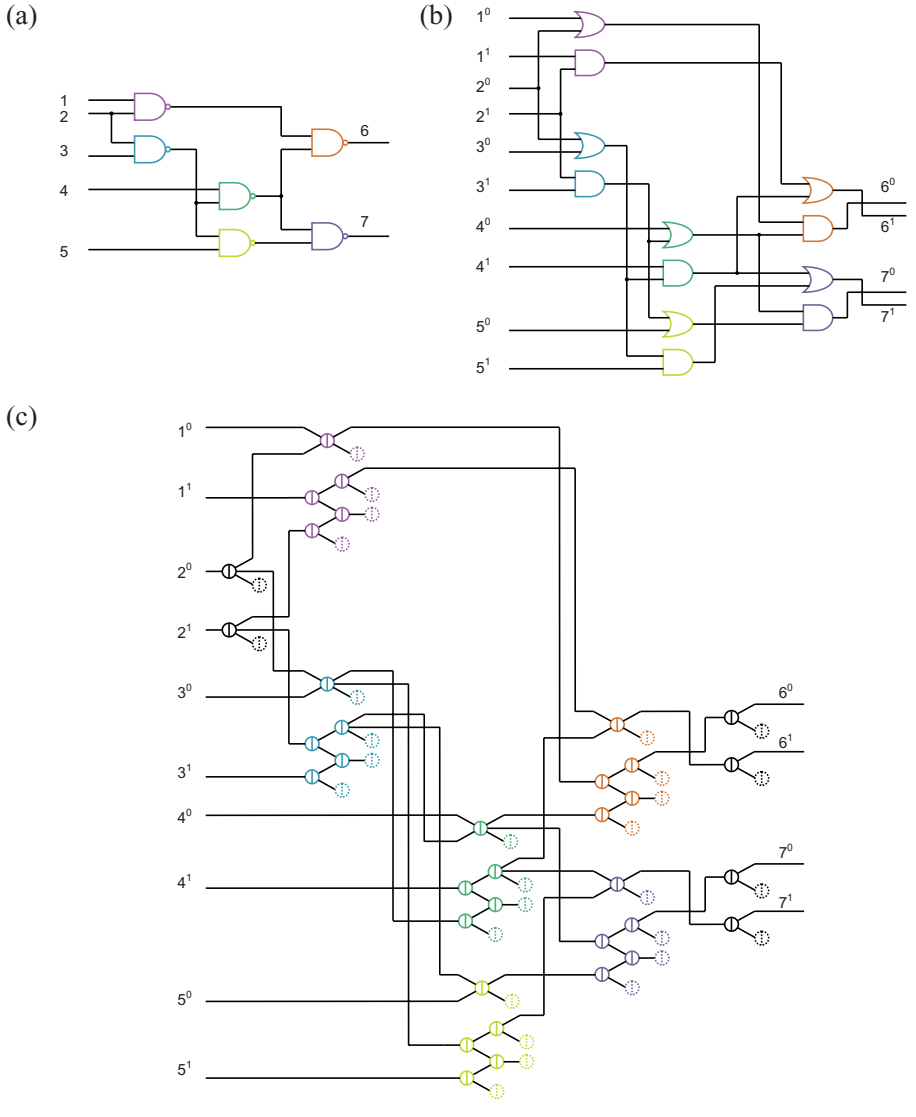


Fig. 4. Compiling boolean logic circuits. (a) A sample circuit with 6 gates. (b) Translation into an equivalent dual-rail circuit with 12 gates. (c) Translation into an equivalent seesaw gate circuit with 36 gates.

circuit (which uses only AND and OR), as illustrated in Fig. 4ab. In the new circuit, which will contain at most twice as many gates, each wire z is replaced by two wires, z^0 and z^1. If neither new wire is ON, this indicates that the logical value of z has not been computed yet; if z^0 is ON, this indicates that the logical value of z must be OFF; while if z^1 is ON, this indicates that the logical value of z must be ON. (If both z^0 and z^1 are ON, then the circuit is experiencing

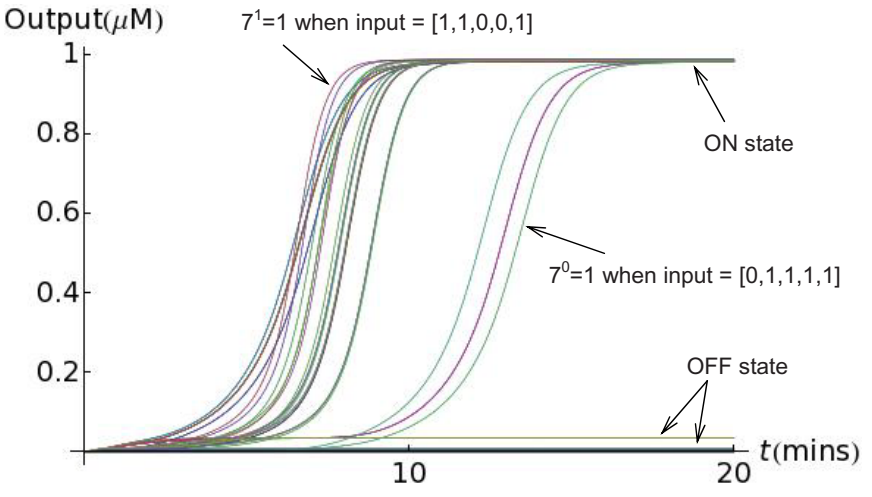


Fig. 5. Simulation results for all 32 possible input vectors. The concentrations of all four dual-rail output species are shown as a function of time. Delays vary with the input, depending the shortest decision path through the network.

a fault.) With this representation, each original two-input boolean logic gate can be implemented using one AND gate and one OR gate. XOR can be easily implemented in dual-rail logic as well, although it requires 4 OR gates and 2 AND gates. In terms of DNA molecules, the difficulty implementing NOT gates derives from the need to detect and act on the absence of an input strand, but the action must not take place before the computation has reached this point in the circuit. One possible solution could be to use a clock signal to initiate the action, or lack of it, at the appropriate time. This is effectively what dual-rail logic does, except that explicit clocks are unnecessary because every signal value (once computed) is represented by the presence of an appropriate molecule that, in itself, can serve as a timing signal as well.

To demonstrate these concepts, we compiled a small circuit (5 inputs, 6 NAND gates, 2 outputs) from its original netlist specification (Fig. 4a), to the equivalent dual-rail circuit consisting of only AND and OR gates (Fig. 4b), and then to its implementation as a network of seesaw gates (Fig. 4c). Because the thresholds on downstream gates drain the outputs of upstream gates and thus shift the equilibrium, for successful composition it was necessary to adjust the initial concentrations shown for isolated subcircuits in Fig. 3. It was also necessary to provide fan-out gates for inputs that were used more than once, and to add final read-out gates to properly drain the adjusted AND and OR subcircuits. The corresponding system of ODEs describing the network's mass action kinetics were then simulated for all 32 possible input combinations. (The compiler and simulator was written in Perl and Mathematica.) As shown in Fig. 5, in every case both outputs reached either a clear "0" or "1" concentration level, which was verified to be correct. This provides concrete evidence that the digital logic subcircuits compose well.

5 Relay Contact Circuits

In his seminal 1940 Master's Thesis [24], Claude Shannon established a systematic symbolic approach to the analysis and design of digital circuits. A prevalent technology at the time was relay contact circuits, in which input switching signals (A, B, C, ...) opened or closed electrical contacts in a network, either allowing current to flow through the network, or not. Like circuits made of AND, OR, and NOT gates, relay contact circuits can concisely implement arbitrary boolean functions. (Here we consider only circuits of primary relays – those directly controlled by external input signals.) To illustrate the flexibility of the seesaw gate motif, we provide a general method to compile relay contact circuits down to equivalent seesaw gate circuits.

The basic primitives for constructing relay contact networks are simple. A circuit with one switch (Fig. 6a) corresponds to a single seesaw gate. The current signal is represented by the free signal strands of the seesaw gate, and the switching signal is represented by the input catalyst strands. Signal is produced on the output wire if and only if the switching signal A is ON. Two switches in series perform an AND operation (Fig. 6b). Since the connectivity of seesaw gates is oriented, we make each wire always connect two different sides of the gates in the design. The two seesaw gates corresponding to the circuit "A and

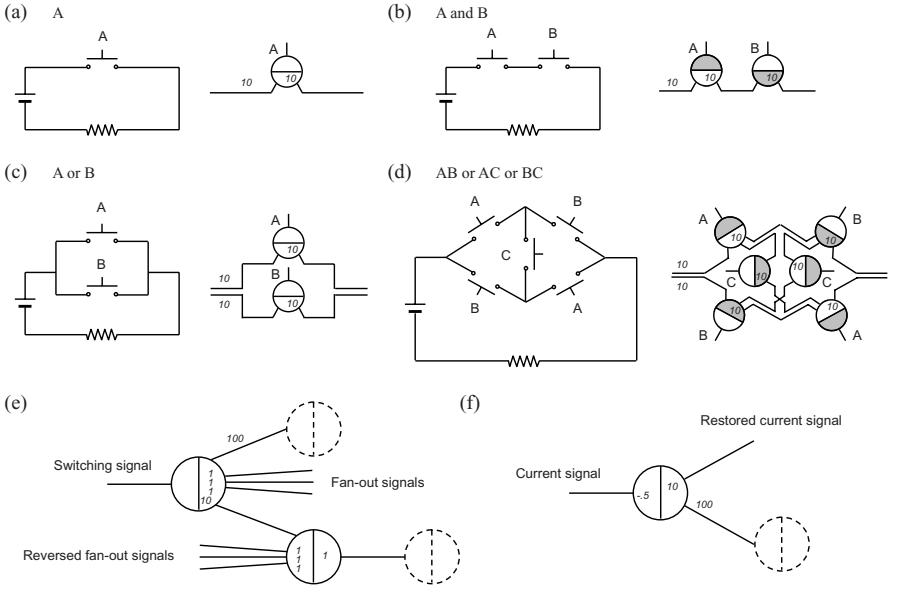


Fig. 6. Implementation of relay circuits. (a) A simple circuit with current source (battery) and controlled device (denoted by a resistor), and the corresponding seesaw gate circuit. (b) AND logic. The "left" side of each seesaw gate is shaded. (c) OR logic. (c) A more complex circuit. (e) Input signal fan-out for both left-side and right-side wires. (f) Restoration for output and intermediate signals.

B" therefore must have different sides connected. Two switches in parallel perform an OR operation (Fig. 6c). The only innovation here is that two distinct input and output wires are used. Fan-out at the current source can provide such a signal, and an OR gate at the current drain can consolidate the output into a single wire, if desired.

For more complex circuits, such as the one in Fig. 6d, a number of additional issues arise. First, there may be two possibilities for the current direction through a given relay (such as C). We use a pair of seesaw gates in such cases. The current flow from A to C to A will go through the seesaw gate C on the left while the current flow from B to C to B will go through the other one on the right. This provides the basis for a general construction. Each original contact relay is replaced by a pair of seesaw gates, one prepared for each possible current flow direction. Output of each gate is routed to all other connected gates in the given flow direction. Thus, for each input setting, if there exists a connected path in the relay circuit, then there is a directed path of catalytically active gates in the seesaw gate circuit. Second, because we need different switching signal strands for different seesaw gates representing the same logical input (such as A, B, and C in Fig. 6d), and we also may need to connect them to either the "left" or the "right" side of the gate (as for C in Fig. 6d), we make use of a signal fan-out unit producing any number of signals in both the forward and reversed directions (Fig. 6e). Finally, if we use a uniform gate:output concentration (say $10x$), the current signal will decrease as it passes through each seesaw gate. Thus, we need to restore the current at any place the signal becomes weak. This can be achieved by an amplifier with a threshold (Fig. 6f). Following this procedure, any relay contact circuit can be implemented with seesaw gates.

6 Analog Time-Domain Circuits and Feedback

The behavior of seesaw gate circuits is intrinsically analog. Following the approach of [3], we construct amplifier cascades with initial quadratic growth and with initial exponential growth. More complex temporal dynamics can also be synthesized, such as a pulse generator.

The amplifier shown in Fig. 7a is a two-stage feedforward cascade. Input signal 12 catalytically pushes strands onto wire 24, which exhibits linear growth with time. Signal 24 also serves as catalysts for the release onto output wire 46, which in turn grows quadratically with time.

The amplifier shown in Fig. 7b is a one-stage feedback cascade. Initially, input signal 12 catalytically releases strands onto output wire 22. However, signal strand 22 contains the recognition domain of gate 2 on both left side and right side, and therefore can play the role of the input signal once released, binding to the gate:output complex by its right side. Thus, strand 22 is also catalytically active in releasing additional copies of itself, resulting in exponential growth of the signal.

The pulse generator shown in Fig. 7c illustrates a non-amplifying temporal dynamic. The basic idea is that the input strand 45 initially releases a large amount of output 24, but this is slowly absorbed by gate 2. The large gate 2

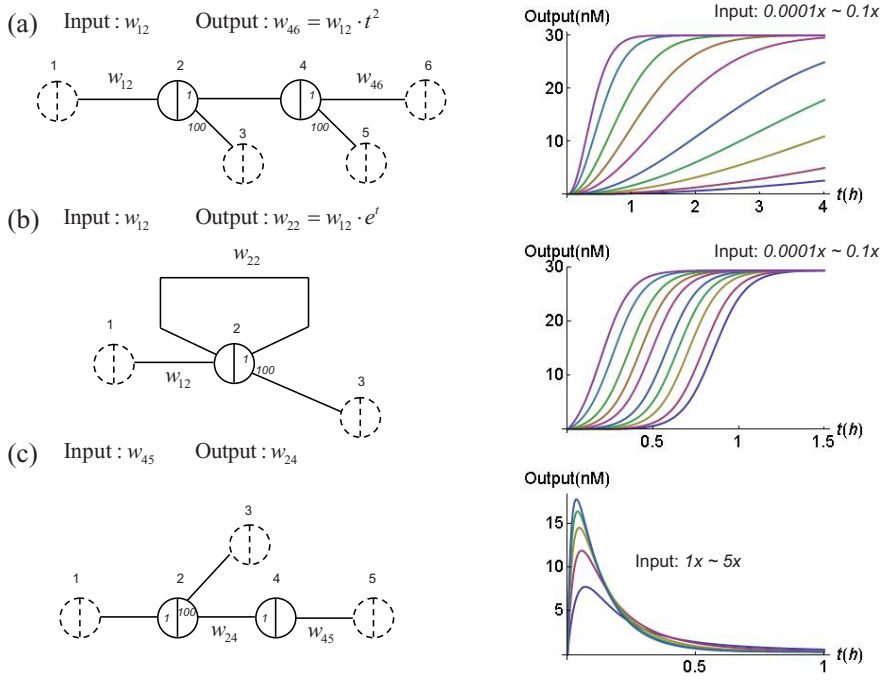


Fig. 7. Analog time-domain circuits. (a) A catalytic cascade that exhibits initially quadratic growth. Temporal trajectories are shown for a serious of exponentially decreasing initial input concentations. (b) A positive feedback circuit that exhibits initially exponential growth. A series of exponentially decreasing input concentrations yields a series of trajectories with linearly increasing half-completiong times. (c) A pulse-generating circuit. Pulse amplitude depends on the input concentration. Simulations use $1x = 30$ nM.

concentration ensures that signal on wire 24 returns to a small value at equilibrium, because the gate-to-wire ratio for wire 24 must be the same as that for 23, which cannot be higher than $0.01x$.

7 Discussion

This project was inspired by the remarkable success of scaffolded DNA origami [15,25,26] for programming the self-assembly of hundreds of DNA strands into a single target structure. Self-assembly was extraordinarily reliable despite that DNA sequences could not be optimized to avoid undesired binding interactions and that unpurified DNA strands were used, for which the strand concentrations were not precisely known and many molecules were presumably incomplete or damaged. We were therefore looking for an analogous design for hybridization-based DNA catalysts and circuits that would require minimal sequence design effort and work well even with unpurified strands and unreliable concentrations.

Does our proposed seesaw gate motif live up to hopes and expectations as a DNA circuit component suitable for scaling up to large and complex circuits? We see some encouraging features, some concerns, and some clear challenges.

First, design of large feedforward digital circuits looks promising. At the highest level, abstract specifications for circuit function can be expressed concisely using existing hardware description languages such as Verilog [27,28] and VHDL [29], then compiled down to a gate level netlist specifying elementary gates (AND, OR, NOT, NOR, NAND, XOR) and their connectivity. Thus, the sheer complexity of large-scale circuit design can be managed by off-the-shelf tools. The next step is compiling the digital logic netlist down to the seesaw gate circuit abstraction, using the constructions described above for dual-rail logic. This is straightfoward if no circuit size optimizations are attempted. To achieve the final step of designing molecules, we must assign sequences to each gate base strand. For this purpose, a single large set of sufficiently distinct domain sequences would enable construction of any circuit containing up to the given number of seesaw gates.

Can we design sufficiently many domain sequences to scale up to circuits with many gates? We consider two design criteria: (1) signal strands should not have secondary structure and should not interact with each other, and (2) strand displacement should be unable to proceed if the invading sequence is not an exact match. The first criterion can be satisfied by standard DNA sequence design methods [30,31]; here we take the easy approach by using a three letter code (A, C, and T) for the signal strands, thus ensuring that problematic secondary structures and interactions are unlikely [32,33,34]. The gate base strand will therefore consist of (A, G, and T). A universal 5 nt toehold sequence may be used for all nodes, since specificity of interactions arises from uniqueness of the recognition domain sequences. Because of the complete independence of domains within the seesaw gate motif, no system-level conflicts are generated when strand sequences are generated by concatenation. The second criterion may be formalized as combinatorial sequence constraints. For example, we could require that at least 30% of bases are different for any two distinct recognition domains; as each mismatch impedes branch migration speed by a factor of roughly 10 [35,36], even 5 mismatches will dramatically reduce crosstalk. Additionally, we require that mismatches are spread out, so that when the wrong signal strand interacts with a gate, it will quickly encounter difficulties and dissociate; specifically, the longest run of matches must be less than 35% of the domain length. Finally, to prevent synthesis errors and ensure comparable melting temperatures, we require that there are no more than 4 A's or T's in a row, no more than 3 C's or G's in a row, and that sequences have between 30% and 70% GC-content. (Cf. constraints 1,7,8 of [37].) Using a "sphere-packing" technique [38], we have found sets of codes of sizes 76, 559, 5181, and $\gg 11,000$ for recognition domains of lengths 10, 15, 20, and 25, respectively, confirming the theoretically expected exponential growth in codebook size. This is enough to construct some interesting circuits. The caveat is that we are not certain that molecules designed using these criteria will work consistently in the laboratory.

Can so many distinct sequences be synthesized and handled in the laboratory? Promising synthesis techniques include the Agilent SurePrint inkjet microarray spotter, which is advertized to be capable of synthesizing roughly 250,000 distinct sequences per slide [39,40]; based on typical DNA array densities and slide dimensions, this corresponds to ≈ 0.040 fmol per spot. If each spot provides a $0.25x$ concentration (the minimal increment we use) in our chosen reaction volume, then a gate complex using $6x$ would require the use of 24 spots. If we define the complexity of a seesaw gate circuit to be the total initial concentration of strands, in $1x$ units, then the Agilent technology would allow us to create circuits of complexity up to $62,500x$. (This corresponds to either $\approx 15,000$ OR gates or ≈ 4500 AND gates for the concentrations used in Fig. 5.) NimbleGen has a similar technology, capable of producing 350,000 distinct 85-mers [41], and we can expect technologies capable of synthesizing longer strands in the near future. The hairpins in Fig. 1d are up to 91 nucleotides long, which we consider to be synthesizable. After synthesis, linkers attaching strands to the slide can be cleaved, and a mixture of all strands can be collected in a single tube, annealed to form hairpins, digested with restriction enzymes to produce gates, and then gel-purified to eliminate non-functional molecules. Thus, all molecules for the entire circuit are synthesized and processed in parallel in a single tube.

Once designed and synthesized, will they work? The first question is speed. If the maximum total concentration for reliable DNA gate operation is 100 μM (perhaps optimistic), then $1x$ would be 1.6 nM for a $62,500x$ circuit, and the slowest reaction half-times (the effective gate delay) would be 3 days if $k_0 = 10^4$ /M/s and $k_1 = 10^6$ /M/s. This is too slow. Either one must resign oneself to smaller circuits, for which the concentrations can be higher, or one must find a way to speed up hybridization reactions in dilute complex mixtures. PERT [42,43] claims to be such a method; in which case, $k_0 = 10^8$ and $k_1 = 10^{10}$ may be possible. This would reduce the 3 days to 30 seconds.

The second question is whether the computation will be correct. For feedforward digital circuits, thresholding and signal restoration – the digital abstraction – is expected to provide some robustness to variations in concentrations, to leak, and to minor crosstalk. However, experimental exploration of seesaw gate circuits will be needed to evaluate the potential for producing reliable function in practice. We anticipate that there will be many unforeseen difficulties.

In summary, we have proposed a new catalytic DNA gate that appears to be suitable for scaling up to analog and digital circuits incorporating thousands of gates with a reasonable expectation that adequate speed and reliability could be achieved.

However, a limitation of this work is that the circuit constructions we described all function by completely depleting key fuel species, hence each circuit preparation can be used only once. An important goal is to implement sequential circuits containing buffers, flip-flops, resets, and clocks that orchestrate the re-use of circuit elements and can process time-varying input signals. We do not know at this point whether this limitation is essential to the seesaw gate motif, or whether we have just not yet been clever enough to see how to do it. Similarly, we do not yet

have a characterization of the class of analog dynamics that can be achieved, although it appears to be a rich space of behaviors. On the practical side, interfaces between DNA circuits and other chemical reactions will be necessary if DNA circuits are to serve as embedded controlers for molecular events.

Acknowledgements

The authors thank Dave Zhang for discussion of the catalytic mechanism, Marc Riedel for providing example netlists from logic synthesis benchmarks, Virgil Griffith for suggesting useful techniques for DNA sequence design, Ho-Lin Chen and Shuki Bruck for suggesting the connection to relay circuits, David Soloveichik for Mathematica code for simulating chemical reaction networks, and Georg Seelig, Bernard Yurke, and everyone else for discussions and support. This work has been supported by NSF grant no. 0728703 and HFSP award no. RGY0074/2006-C.

References

1. Tang, J., Breaker, R.R.: Rational design of allosteric ribozymes. Chem. Biol. 4, 453–459 (1997)
2. Turberfield, A.J., Mitchell, J.C., Yurke, B., Mills Jr., A.P., Blakey, M.I., Simmel, F.C.: DNA fuel for free-running nanomachines. Physical Review Letters 90(11), 118102–118104 (2003)
3. Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318, 1121–1125 (2007)
4. Stojanovic, M.N., Mitchell, T.E., Stefanovic, D.: Deoxyribozyme-based logic gates. Journal of the American Chemical Society 124, 3555–3561 (2002)
5. Hagiya, M., Yaegashi, S., Takahashi, K.: Computing with hairpins and secondary structures of DNA. In: Chen, J., Jonoska, N., Rozenberg, G. (eds.) Nanotechnology: Science and Computation, pp. 293–308. Springer, Heidelberg (2006)
6. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314, 1585–1588 (2006)
7. Penchovsky, R., Breaker, R.R.: Computational design and experimental validation of oligonucleotide-sensing allosteric ribozymes. Nat. Biotechnol. 23(11), 1424–1433 (2005)
8. Macdonald, J., Li, Y., Sutovic, M., Lederman, H., Pendri, K., Lu, W., Andrews, B.L., Stefanovic, D., Stojanovic, M.N.: Medium scale integration of molecular logic gates in an automaton. Nano Letters 6, 2598–2603 (2006)
9. Yashin, R., Rudchenko, S., Stojanovic, M.N.: Networking particles over distance using oligonucleotide-based devices. Journal of the American Chemical Society 129, 15581–15583 (2007)
10. Seeman, N.C.: An overview of structural DNA nanotechnology. Mol. Biotechnol. 37, 246–257 (2007)
11. Bath, J., Turberfield, A.J.: DNA nanomachines. Nature Nanotechnology 2, 275–284 (2007)
12. Gartner, Z.J., Liu, D.R.: The generality of DNA-templated synthesis as a basis for evolving non-natural small molecules. Journal of the American Chemical Society 123, 6961–6963 (2001)

13. Gothelf, K.V., LaBean, T.H.: DNA-programmed assembly of nanostructures. Organic & Biomolecular Chemistry 3, 4023–4037 (2005)
14. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539–544 (1998)
15. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature 440, 297–302 (2006)
16. Yin, P., Choi, H.M.T., Calvert, C.R., Pierce, N.A.: Programming biomolecular self-assembly pathways. Nature 451, 318–322 (2008)
17. Turing, A.M.: The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society (part B) 237, 37–72 (1953)
18. Zhabotinsky, A.M.: A history of chemical oscillations and waves. Chaos 4, 379–386 (1991)
19. 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, 74–82 (2000)
20. Zhang, D.Y., Winfree, E.: Control of DNA strand displacement kinetics using toehold exchange (in preparation)
21. Yurke, B., Mills Jr., A.P.: Using DNA to power nanostructures. Genetic Programming and Evolvable Machines 4, 111–122 (2003)
22. Dirks, R.M.: Analysis, design, and construction of nucleic acid devices. PhD thesis, California Institute of Technology (2005)
23. Bois, J.S.: Analysis of interacting nucleic acids in dilute solutions. PhD thesis, California Institute of Technology (2007)
24. Shannon, C.E.: A symbolic analysis of relay and switching circuits. Technical Report Master's Thesis, Massachussetts Institute of Technology (1940)
25. Qian, L., Wang, Y., Zhang, Z., Zhao, J., Pan, D., Zhang, Y., Liu, Q., Fan, C., Hu, J., He, L.: Analogic china map constructed by DNA. Chinese Science Bulletin 51, 2973–2976 (2006)
26. Douglas, S.M., Chou, J.J., Shih, W.M.: DNA-nanotube-induced alignment of membrane proteins for NMR structure determination. Proc. Nat. Acad. Sci. USA 104, 6644–6648 (2007)
27. Thomas, D.E., Moorby, P.R.: The Verilog Hardware Description Language. Kluwer, Dordrecht (1991)
28. Golze, U.: VLSI Chip Design with the Hardware Description Language VERILOG. Springer, Heidelberg (1996)
29. Shahdad, M., Lipsett, R., Marschner, E., Sheehan, K., Cohen, H., Waxman, R., Ackley, D.: VHSIC hardware description language. IEEE Computer 18, 94–103 (1985)
30. Brenneman, A., Condon, A.: Strand design for biomolecular computation. Theor. Comput. Sci. 287, 39–58 (2002)
31. Bishop, M.A., D'Yachkov, A.G., Macula, A.J., Renz, T.E., Rykov, V.V.: Free energy gap and statistical thermodynamic fidelity of DNA codes. Journal of Computational Biology 14, 1088–1104 (2007)
32. Mir, K.U.: A restricted genetic alphabet for DNA computing. In: Landweber, L.F., Baum, E.B. (eds.) DNA Based Computers II. DIMACS, vol. 44, pp. 243–246. American Mathematical Society, Providence (1998)
33. Braich, R.S., Chelyapov, N., Johnson, C., Rothemund, P.W.K., Adleman, L.M.: Solution of a 20-variable 3-SAT problem on a DNA computer. Science 296, 499–502 (2002)

34. Faulhammer, D., Cukras, A.R., Lipton, R.J., Landweber, L.F.: Molecular computation: RNA solutions to chess problems. Proc. Nat. Acad. Sci. USA 97(3), 1385–1389 (2000)
35. Panyutin, I.G., Hsieh, P.: Formation of a single base mismatch impedes spontaneous DNA branch migration. Journal of Molecular Biology 230, 413–424 (1993)
36. Panyutin, I.G., Hsieh, P.: Kinetics of spontaneous DNA branch migration. Proc. Nat. Acad. Sci. USA 91, 2021–2025 (1994)
37. Kao, M.-Y., Sanghi, M., Schweller, R.T.: Randomized fast design of short DNA words. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1275–1286. Springer, Heidelberg (2005)
38. King, O.D.: Bounds for DNA codes with constant GC-content. Electronic Journal of Combinatorics 10, R33 (2003)
39. Agilent Technologies. SurePrint technology (web page), http://www.chem.agilent.com/scripts/generic.asp?lpage=557
40. Agilent Technologies. Multi-pack gene expression microarrays (web page), http://www.chem.agilent.com/scripts/generic.asp?lpage=51683
41. NimbleGen Systems, Inc. Array synthesis (web page), http://www.nimblegen.com/technology/manufacture.html
42. Kohne, D.E., Levison, S.A., Byers, M.J.: Room temperature method for increasing the rate of DNA reassociation by many thousandfold: The phenol emulsion reassociation technique. Biochemistry 16, 5329–5341 (1977)
43. Goldar, A., Sikorav, J.-L.: DNA renaturation at the water-phenol interface. Eur. Phys. J. E. 14, 211–239 (2004)

Tiamat: A Three-Dimensional Editing Tool for Complex DNA Structures

Sean Williams[1], Kyle Lund[2,4], Chenxiang Lin[2,4], Peter Wonka[3], Stuart Lindsay[2,4,5], and Hao Yan[2,4]

[1] Department of Computer Science
University of California,
Davis, CA, 95616
[2] Center for Single Molecule Biophysics, The Biodesign Institute
Arizona State University
Tempe, AZ, 85287
[3] School of Computing and Informatics
Arizona State University
Tempe, AZ, 85287
[4] Department of Chemistry and Biology
Arizona State University
Tempe, AZ, 85287
[5] Department of Physics and Astronomy
Arizona State University
Tempe, AZ, 85287

Abstract. We present the development of a new graphical user interface driven molecular modeling, editing and visualization tool called Tiamat. Tiamat addresses the challenge of how to efficiently model large and complex DNA nanostructures. We describe the three major components of our system. First, we discuss design guidelines and data structures that form the basis of flexible and large-scale editing. Second, we explain a semi-automatic sequence generator that combines user input with random sequence generation to efficiently label the molecules in the DNA structure. Third, we outline the visualization techniques including a simplification algorithm that are used to render large designs. The results demonstrate how Tiamat was used to generate large and complex designs.

Keywords: Structural DNA nanotechnology, DNA tile, 3D DNA structure display, DNA sequence design.

1 Introduction

The concept of the immobile branched DNA junctions [1], proposed by Seeman in 1982, led to the use of DNA as a tool for constructing nanoscale arrays and objects [2,3]. The designs of most DNA nanoarrays are constrained by the helical repeat in DNA of 10.5 bases. Designing 1-dimensional (1D), 2-dimensional (2D) and 3-dimensional (3D) structures using a simple graphics program is possible using multiples of 10.5 bases in designing the crossover points. When trying

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 90–101, 2009.

to make very complex structures, a DNA modeling program is needed to see precisely how the geometrical design is going to fold. Another key aspect to designing DNA nanostructures is the generation of a random sequence with minimized sequence symmetry where there is no secondary structure that can be formed by the DNA.

Already in 1985, Seeman produced a command line driven Fortran program for creating relatively simplistic drawings of DNA structures [4]. The lack of a graphical interface in the program made the direct observation of the 3D aspect of the DNA structure difficult. Alternatively, the jacks and straw modeling of DNA was used for making simple structures, but as the structure increased in complexity this became very difficult to handle [5]. Recently, Birac and Seeman developed GIDEON, a 3D DNA rendering program [6]. GIDEON is a useful program that allows a user to design complex and precise DNA nanostructures. The program makes it possible to see the designed structure in top, side and front views. GIDEON has the ability to predict the actual distortions that will occur to the DNA structure. Overall, GIDEON is a great tool for the design of DNA structures. In this paper we build on this successful previous work while addressing some of the limitations that make it difficult to efficiently generate large designs. The first limitation is that the sequence generation is not integrated with the modeler.

The generation of DNA sequences has been predominantly prepared using a Fortran program called SEQUIN [7]. SEQUIN has the ability to join DNA helices together and then generate a group of sequences one at a time. The joining of DNA helices together requires the user to sketch a 2D line drawing of the structure. The structure is then put into the program by designating arm lengths and linking the arms together. Once the structure has been implemented in the program the user can generate the sequence for the structure by comparing new sequences to sequences that already exist in the structure to avoid sequence homology. The sequence generator is not random and the user has to decide if they like the sequence or if they want the computer to generate another sequence. Also, it is cumbersome to input very large structures into a separate, command-line tool.

The second limitation of previous work is that the design of data structures and visualization strategies were not flexible enough to allow for large and complex designs. The large number of molecules in a single design might become confusing for the designer and can also overwhelm the graphics hardware. In this paper we present our modeling program Tiamat to overcome these limitations. Tiamat is a 3D modeling program designed specifically for the creation of large and complex DNA structures. This paper contains three major contributions. First, a design interface focused on flexibility in modeling structures. The structures allow the modeling of general designs. Second, an integrated random sequence generator that checks sequences directly against the provided model to prevent the formation of secondary structures. Third, a fast and dynamic level of detail algorithm to simplify display of very large structures while still displaying important information.

2 Methodology

This section will describe the important design and implementation considerations for working with structural DNA, with overall structure sizes ranging from dozens to tens of thousands of bases. In section 2.1, we will discuss the data structures usable for structural DNA modeling and their impact on interactivity. In section 2.2, we will discuss random sequence generation directly applied to a DNA model. In section 2.3, we will discuss a dynamic level of detail algorithm for very large DNA structures.

2.1 Data Structures

In this section, we will discuss various approaches to storing and interacting with the data that describes the components of a DNA structure. A natural starting place is an undirected graph [8], a data structure in which the data is stored in vertices that can be arbitrarily connected by edges, with the caveat that if an edge exists between vertices A and B, an edge also exists between B and A. With this in mind, there exists a more important question of what a vertex represents.

The high level approach is to have each vertex represent a DNA helix. Each vertex would store the five-prime sequence (the three-prime sequence can be inferred to be the complement of the five-prime sequence), with four potential edges: the top and bottom of the five-prime strand, and the top and bottom of the three-prime strand. By dividing a helix – converting it into two helices with the bottom of the top helix connected to the top of the bottom helix – junctions to other helices could be added with little trouble. In this approach, helix-centric operations are easy, such as calculating the physical deformation caused by junctions. However, base-centric operations, such as the creation of arbitrarily-shaped probes, are more difficult.

The low level approach is to have each vertex represent a base. Each vertex would store its own Watson-Crick type (e.g. cytosine), and its edges would connect to its neighboring bases: the base up the strand, the base down the strand, and its paired base. With this model, the creation of junctions and other deviations from a standard helix are accomplished by reassigning the edges of the involved bases. This model makes base-centric operations easy, while making helix-centric operations difficult.

There are benefits and drawbacks to both choices, so the strengths of each implementation are based on the necessity of its associated features. In a base-centric design some elements of higher level structure can be inferred from connectivity; that is, by traversing along up and down edges one can determine all the bases that compose one strand. However, if two helices are linked by a junction, then a traversal of all edges – which finds the entire helix a base belongs to – will treat the linked helices as one. In this sense, a base-centric design throws away most high level information, making helix-centric operations virtually impossible.

The base-centric and helix-centric designs each have one associated feature under consideration. The helix-centric feature desired was a calculation of the

physical deformation a structure would undergo given a configuration of junctions. This stems from the physical requirement that five-prime and three-prime connections of bases all be the same length, since they must all be composed of the same phosphate chains. An algorithm that could reasonably solve this problem would have to perform some sequence of rotations and translations on helices to align them into a valid arrangement; this algorithm cannot be done without helix-centric information.

The base-centric feature desired was a method for creating arbitrarily shaped strands. If a strand is represented in data only as a sequence, its graphical representation derives from well-established biological rules: a double helix with a known rotation per base pair, distance between base pairs, and so on. Creating an arbitrarily shaped strand – for example, a straight line – cannot be done directly, and would at best require a hybrid implementation in which arbitrary strands are base-centric and standard helices are helix-centric. The primary drawback of such a hybrid approach is that it would be too complicated for any straightforward implementation, effectively crippling its usability.

The final decision was ultimately reached by a desire for functionality over aesthetics. That is, the most important use for a structural DNA modeling tool is to create structures to be reproduced in real life by ordering the strands that will create the structure via self-assembly. Calculating the physical distortions a structure will undergo in Tiamat has relatively little theoretical value, while being given the maximum flexibility in modeling decisions increases the range of structures Tiamat can be used to design. Thus, Tiamat employs a base-centric representation of the structure.

2.2 Sequence Generation

In this section, we will discuss the generation of random sequences directly applied to a DNA structure. Self-assembly dictates that, if the sequences of each strand are assigned correctly, then those strands would assemble into the original structure. However, the sequences must be chosen carefully to avoid the formation of secondary structures, in which unintended Watson-Crick bonds occur and destroy part or all of a structure. To prevent secondary structure formation, Tiamat recognizes three constraints that can be applied to an otherwise randomly generated sequence: unique sequence limit, repetition limit, and GC percentage.

The unique sequence limit refers to the shortest subsequence that must appear only once in the structure. For example, with a unique sequence limit of 5, if a strand contains the sequence ATGACT, then ATGAC and TGACT may not appear anywhere else in the structure (for example, the sequence ATGACC cannot appear anywhere else), though sequences containing ATGA, TGAC, and GACT may appear elsewhere (for example, ATGAGT may appear somewhere else).

The repetition limit refers to the longest sequence of bases that can all be the same. For example, if the repetition limit is 5, then CCCCC may not appear in the structure, though CCCCA or TCCCC can.

The GC percentage refers to the minimum percent of bases in the structure that must be either guanine or cytosine. This places no upper limit on the

number of guanine or cytosine that can occur, and for reasons that will be explained below, the results will tend toward more occurrences of guanine and cytosine than specified.

Generating a valid sequence is a discrete optimization problem [9,10], in which all constraints must be satisfied but any solution that fulfills all the constraints is valid. Exhaustive search approaches to this problem are not feasible, as the size of the search space can easily be shown to be 4^N. That is, an algorithm based on assigning sequences in order until one turns out to meet all the criteria will, for large problem instances, take years to solve. There is a characteristic of this problem that makes it solvable in a short amount of time, however: the size of the set of valid configurations is relatively large with respect to the size of the set of possible configurations. It is therefore reasonable to use a stochastic algorithm to solve this problem.

Our algorithm is inspired by sampling algorithms such as Metropolis-Hastings and Simulated Annealing. We start by assigning all bases a random type. Then, for each base, verify that it meets all of the constraints. If a base is found to violate a constraint, randomly change one of the offending bases. We design a fitness function of the design that includes all constraints and make a random decision on keeping or discarding a change based on changes in the fitness function. The problem with this algorithm is that it lacks a stopping condition; if the constraints are too tight for a solution to be possible, this algorithm will continue forever. A rather inelegant but effective solution to this problem is to note that the algorithm will find a valid solution very quickly if one exists, and to simply put a timeout on the algorithm. That is, if a solution is not found within some number of seconds, to assume a solution does not exist and report the failure to the user. It should also be noted that the minimum valid strength of the constraints is governed by the size of the structure; a very large structure has more connected bases than a very small structure, and must therefore allow more repetitions and longer repeating subsequences in order for a valid solution to exist.

It should also be noted that, in order to reduce the execution time of the algorithm, the random assignment of bases should take into account the GC percentage constraint. For example, if it is required that 70% of bases be guanine or cytosine, it is counterproductive to assign 50% of bases a guanine or cytosine type. However, a sequence of randomly generated numbers lying on a given mean will form a Gaussian distribution around that mean; thus, in order to ensure a solution is more likely to be found, the percentage of guanines and cytosines should be greater than the required percentage by at least one standard deviation. The result of this shift is that more bases than required will be assigned guanine or cytosine, but this is a relatively trivial drawback for its performance increase.

2.3 Visualization

In this section, we will discuss a dynamic level of detail algorithm for very large DNA structures. Levels of detail refers to a process for simplifying the geometry

of a model in order to accelerate the rendering of the model [11]. This is typically accomplished through coarsening the polygonal mesh, and selecting an amount of coarseness to represent the mesh based on the camera's distance from the mesh. The general level of detail algorithms work directly on 2-manifold smooth surfaces.

Therefore, the first and most obvious approach is to simplify the geometry of the structure as much as possible while maintaining the same visual elements (see Fig. 1). Since it is still desired in the long run to make nice-looking images, it makes sense to create two drawing modes. The simpler mode exists to draw bases as very simple spheres and connect them with lines; this mode looks coarse, but is much faster to draw. The more complex mode draws bases using smooth spheres and connects them using smooth cylinders; this mode looks good, but takes a long time to draw. The draw time difference is significant primarily because editing tools are frustrating to use if user instructions are not carried out at interactive frame rates. With the lower resolution view, even relatively slow computers can be used to create structures, then switched to high resolution drawing only when design of a structure is complete.

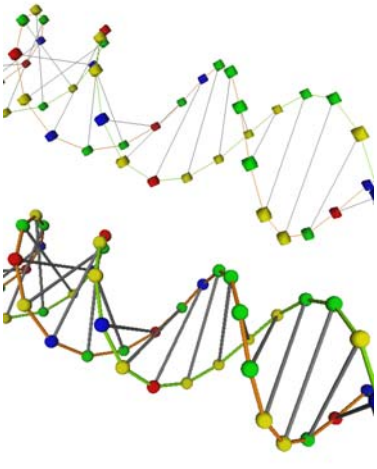


Fig. 1. The same helix rendered in two details: edit mode (top) and render mode (bottom)

Even by coarsening the geometry, a direct rendering will still be very complex for large structures (see Fig. 2). It is difficult to extract useful information from this visualization: while the camera is far away from a helix, the position and orientation of that helix are the only important pieces of information. A better approach is to represent the helices using a proxy geometry, such as straight lines (see Fig. 3, and 4). This is particularly helpful in designing and visualizing large complex DNA nanostructures such as DNA Origami structures recently developed by Rothemund [12].

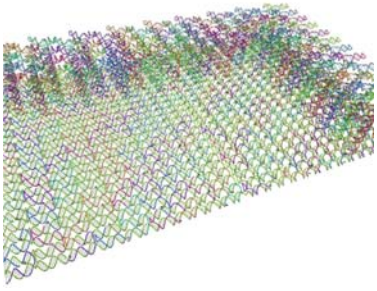


Fig. 2. A very large structure; this view is both visually confusing and has a slow frame rate

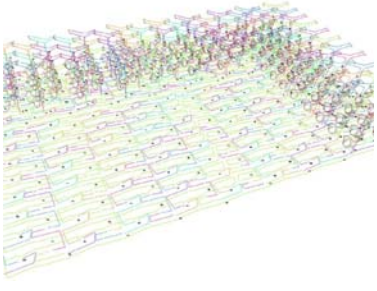


Fig. 3. A very large structure with some helices reduced to single lines

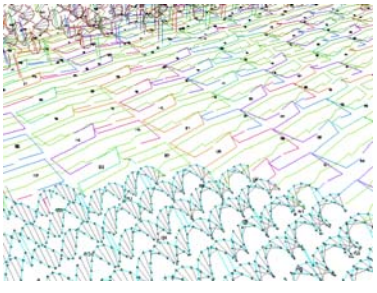


Fig. 4. The cutoff between level of detail zones

Converting a helix to a line requires four pieces of information: the direction of the backbone axis, the origin of the backbone axis, the height of the helix, and the bounds at which a helix remains simple (e.g. has no junctions or sharp turns).

The information about a helix is thrown away once the helix is created; see section 2.1 for a detailed discussion on this topic. To reverse engineer the helix direction, we take the position of the twenty-first base minus the position of the first base, and normalize that. The backbone direction can also be approximated

as the average of the vectors from the first to the tenth and the first to the eleventh base, though this approximation is only employed for helices shorter than twenty-one bases. This can finally be applied to even shorter, eight-base groups by using a ratio of approximately 3:1.

To determine the origin of the backbone axis, one of the bases from the second pair is projected into the plane of the first pair; these three points now uniquely define a circle. If the three points are specified as $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, the center of that circle, which is also the center of the helix, is defined by (4). To determine the height of the cylinder, the same calculation is done for the last base pair in the helix; the length of the vector pointing from the first center point to the last center point is the height of the helix.

$$\alpha = \frac{|\mathbf{P}_2 - \mathbf{P}_3|^2(\mathbf{P}_1 - \mathbf{P}_2) \cdot (\mathbf{P}_1 - \mathbf{P}_3)}{2|(\mathbf{P}_1 - \mathbf{P}_2) \times (\mathbf{P}_2 - \mathbf{P}_3)|^2} \tag{1}$$

$$\beta = \frac{|\mathbf{P}_1 - \mathbf{P}_3|^2(\mathbf{P}_2 - \mathbf{P}_1) \cdot (\mathbf{P}_2 - \mathbf{P}_3)}{2|(\mathbf{P}_1 - \mathbf{P}_2) \times (\mathbf{P}_2 - \mathbf{P}_3)|^2} \tag{2}$$

$$\gamma = \frac{|\mathbf{P}_1 - \mathbf{P}_3|^2(\mathbf{P}_2 - \mathbf{P}_1) \cdot (\mathbf{P}_2 - \mathbf{P}_3)}{2|(\mathbf{P}_1 - \mathbf{P}_2) \times (\mathbf{P}_2 - \mathbf{P}_3)|^2} \tag{3}$$

$$\mathbf{C} = \alpha\mathbf{P}_1 + \beta\mathbf{P}_2 + \gamma\mathbf{P}_3 \tag{4}$$

The final and most difficult challenge is determining the bounds of a simple helix (see Fig. 5). For a simple helix, one can traverse from a base back to itself by going across, down, across, and down. (Note that, since the two strands of a helix are anti-parallel, the down direction of one strand is the up direction of its complimentary strand. Thus, going down one turn on a strand then down on its compliment will traverse in a circle.) In the case of a junction, going across and down sends a traversal into the other helix, so going across and down again will find a base in a different helix. In the case of sharp turns, in order to keep a physically possible shape, "filler" bases must be added on the wide side of the turn, so again a circular traversal will not return to its starting point. Given all this, a section of a helix can be simplified to one line if and only if a circular traversal is possible for each base on that segment.

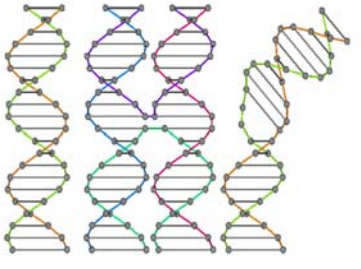


Fig. 5. Three cases for circular traversal: first, a simple helix; second, a junction; third, a sharp turn; of these cases, the first can be represented by one cylinder, the second by four cylinders, and the third by two cylinders

Given all this information, during the drawing routine, if a base is far enough away from the camera, traverse up and down from that base to find either where the helix stops being simple (this can be precalculated) or where the simple helix gets too close to the camera, draw a properly colored line between the two ends, and mark that all bases in between have been drawn.

3 Results

To validate that Tiamat is a functional tool and useful in structural DNA nanotechnology a previously designed 4x4 DNA tile[3e] was re-modeled using Tiamat (see supplemental information for strand sequences generated). The design was to use a single 4x4 DNA tile to grow into a large 2D array with the sequence generated through Tiamat. The structure was made according to the design by Yan et al. [3e] incorporating four 4-arm branched junctions and the adjoining junctions are orthogonal to each other and each end has a five base sticky-end overhang. The design uses the corrugated design which allows for the tiles to form large 2D arrays instead of folding on themselves and forming tube structures. Figure 6A is a Tiamat output that shows the design of the 4x4 tile, and Figure 6B is the joining of four tiles to make the beginning stages of a 2D array.

We used Atomic Force Microscopy (AFM) and native polyacrylamide gel electrophoresis (PAGE) to verify if the 4x4 structure and the self-assembled arrays designed by Tiamat formed correctly. Native PAGE (see Figure 7) was used to verify that the tile structure did not form any other undesired structures (usually shown as upper bands above the single intact band) and any break down structures (lower band indicating unstable dissociations) when annealed at stoichiometric amounts. Figure 8 shows AFM images taken of the 2D crystals that were formed using the Tiamat design. The images prove that Tiamat was able to design the structure and then generate a sequence that could fold into the rationally designed array structures.

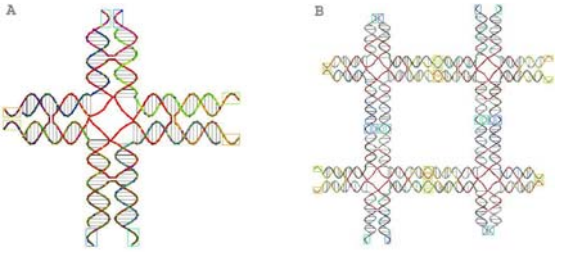


Fig. 6. Tiamat model of the 4x4 DNA tile and a model of the joining of four DNA tiles together

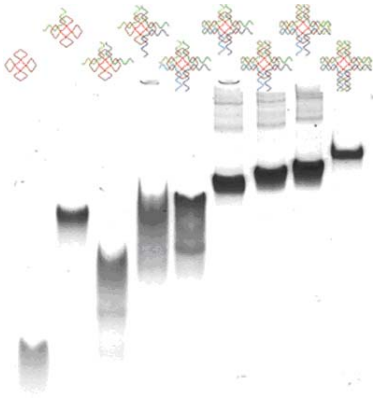


Fig. 7. Native PAGE gel of the formation of the 4x4 DNA tile. Tiamat models are used to represent each lane. The complete tile is from a purified sample of the 4x4 tile.

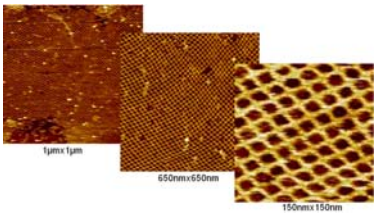


Fig. 8. AFM images of the 2D crystals formed by the 4x4 DNA tile

4 Discussion

As demonstrated by the above examples and figures, we have shown that Tiamat is a novel and useful program for the designing of DNA structures and modeling those structures for feasibility. The main novelty of Tiamat is the possibility to efficiently model large and complex DNA structures. Future implementation and improvement of Tiamat may include more features of DNA modeling such as functions to estimate thermodynamic parameters for a designed DNA nanostructure.

Tiamat is written in C++ using Microsoft Foundation Classes (MFC) for windowing and interface, and OpenGL for rendering. Therefore, it only runs on Microsoft Windows operating systems; specifically, Windows 2000, XP, and Vista. Tiamat requires only OpenGL 1.0, so it can run on virtually any available video card.

Supplemental information includes a manual for the Tiamat software, detailed model and sequences of the 4x4 DNA tile and experimental methods used in this work. The Tiamat program and manual can be downloaded free from http://chemistry.asu.edu/faculty/hao_yan.asp.

Acknowledgements

This work is supported by a grant from NIH to S.L. and H.Y.; H.Y. also would like to thank funding supports from NSF, ONR and AFOSR. We thank members from Hao Yan's group for testing the use of the Tiamat program.

References

1. Seeman, N.C.: Nucleic Acid Junctions and Lattices. J. Theo. Biol. 99, 237–247 (1982)
2. (a) Kallenbach, N.R., Ma, R.-I., Seeman, N.C.: An Immobile Nucleic Acid Junction Constructed from Oligonucleotides. Nature 305, 829–831 (1983) (b) Chen, J., Seeman, N.C.: The Synthesis from DNA of a Molecule with the Connectivity of a Cube. Nature 350, 631–633 (1991) (c) Zhang, Y., Seeman, N.C.: The Construction of a DNA Truncated Octahedron. J. Am. Chem. Soc. 116, 1661–1669 (1994) (d) Fu, T.-J., Seeman, N.C.: DNA Double Crossover Structures. Biochemistry 32, 3211–3220 (1993) (e) Shih, W.M., Quispe, J.D., Joyce, G.F.: A 1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron. Nature 427, 618–621 (2004) (f) Goodman, R.P., Schaap, I.A.T., Tardin, C.F., Erben, C.M., Berry, R.M., Schmidt, C.F., Turberfield, A.J.: Rapid chiral assembly of rigid DNA building blocks for molecular nanofabrication. Science 310, 1661–1665 (2005) (g) Erben, C.M., Goodman, R.P., Turberfield, A.J.: A Self-Assembled DNA Bipyramid. J. Am. Chem. Soc. 129, 6992–6993 (2007)
3. (a) Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and Self-Assembly of Two-Dimensional DNA Crystals. Nature 394, 539–544 (1998) (b) LaBean, T., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J.H., Seeman, N.C.: The Construction of DNA Triple Crossover Molecules. J. Am. Chem. Soc. 122, 1848–1860 (2000) (c) Shen, Z., Yan, H., Wang, T., Seeman, N.C.: Paranemic Crossover DNA: A Generalized Holliday Structure with Applications in Nanotechnology. J. Am. Chem. Soc. 126, 1666–1674 (2004) (d) Mao, C., Sun, W., Seeman, N.C.: Designed Two-Dimensional DNA: Holliday Junction Arrays Visualized by Atomic Force Microscopy. J. Am. Chem. Soc. 121, 5437–5443 (1999) (e) Yan, H., Park, S.H., Ginkelstein, G., Reif, J.H., LaBean, T.H.: DNA templated Self-assembly of Protein Arrays and Highly Conductive Nanowires. Science 301, 1882–1884 (2003) (f) Liu, D., Wang, M., Deng, Z., Walulu, R., Mao, C.: Tensegrity: Construction of Rigid DNA Triangles with Flexible Four-Arm DNA Junctions. J. Am. Chem. Soc. 126, 2324–2325 (2004) (g) He, Y., Chen, Y., Liu, H., Ribbe, A.E., Mao, C.: Self-Assembly of Hexagonal DNA Two-Dimensional (2D) Arrays. J. Am. Chem. Soc. 127, 12202–12203 (2005); (h) He, Y., Tian, Y., Ribbe, A.E., Mao, C.: Highly Connected Two-Dimensional Crystals of DNA Six-Point-Stars. J. Am. Chem. Soc. 128, 15978–15979 (2006); (i) 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. J. Am. Chem. Soc 127, 17590–17591 (2005) (j) Ke, Y., Liu, Y., Zhang, J., Yan, H.: A Study of DNA Tube Formation Mechanisms Using 4-, 8-, and 12-Helix DNA Nanostructures. J. Am. Chem. Soc. 128, 4414–4421 (2006)
4. Seeman, N.C.: The Interactive Manipulation and Design of Macromolecular Architecture Utilizing Nucleic Acid Junctions. J. Mol. Graphics 3, 34–39 (1985)

5. Seeman, N.C.: Physical Models for Exploring DNA Topology. J. Biomol. Struct. Dynam. 5, 997–1004 (1988)
6. Birac, J.J., Sherman, W.B., Kopatsch, J., Constantinou, P.E., Seeman, N.C.: GIDEON, A Program for Design in Structural DNA Nanotechnology. J. Mol. Graphics Model 25, 470–480 (2006)
7. Seeman, N.C.: De Novo Design of Sequences for Nucleic Acid Structure Engineering. J. Biomol. Struct. Dynam. 8, 573–581 (1990)
8. Cormen, et al.: Introduction to Algorithms
9. Nocedal, J., Wright, S.: Numerical Optimization, Springer Series in Operations Research and Financial Engineering
10. Chong, E.K.P., Żak, S.H.: An Introduction to Optimization, 2nd edn
11. Luebke, D., Reddy, M., Cohen, J.D., Varshney, A., Watson, B., Huebner, R.: Level of Detail for 3D Graphics. The Morgan Kaufmann Series in Computer Graphics
12. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature 440, 297–302 (2006)

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

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 102–111, 2009.

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

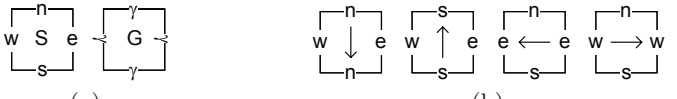


Fig. 1. $\mathbb{S}_{cpf}$ uses a special start (S) and goal (G) tile (a) and four "working" tiles (b)

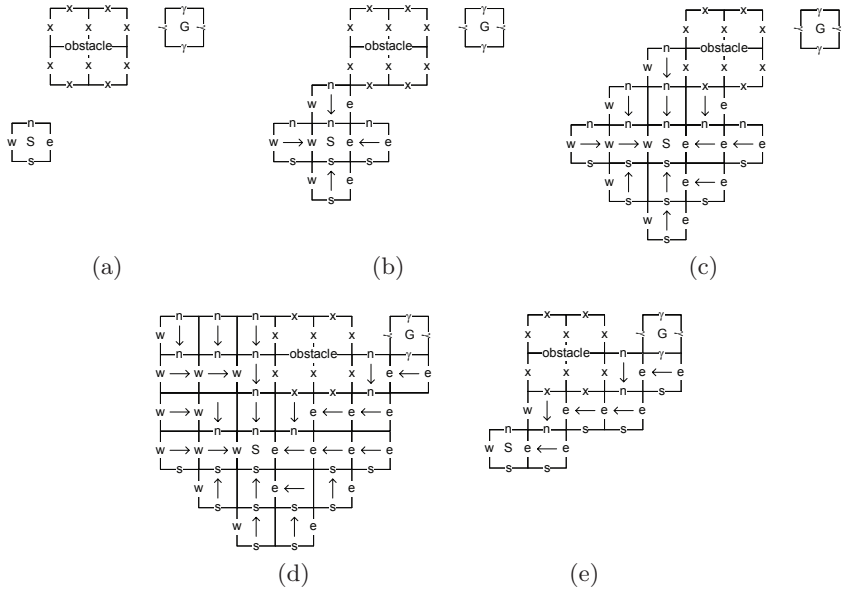


Fig. 2. An example execution of $\mathbb{S}_{cpf}$. $\mathbb{S}_{cpf}$ works at temperature 2 (a-d) to build possible paths and then at temperature 3 (e) to prune unsuccessful paths. Only binding domains that are exposed or attached have been labeled.

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)

Polyomino-Safe DNA Self-assembly via Block Replacement

Chris Luhrs*

Stanford University
cluhrs@cs.stanford.edu

Abstract. The standard abstract model for analyzing DNA self-assembly, aTAM, assumes that single tiles attach one by one to a larger structure. In practice, tiles may attach to each other forming structures called polyominoes and then attach to the assembly using bonds from multiple tiles. Such polyominoes may cause errors in systems designed with only aTAM in mind. In this paper, we first present a formal definition of when one tile system is a "block replacement" of another. Then we present a block replacement scheme for making any system that admits non-trivial block replacement polyomino-safe. In addition, we present a smaller block replacement scheme that makes the Chinese Remainder counter polyomino-safe and prove that the question of whether a system is polyomino-safe (or other similar properties) is undecidable. Finally, we show that applying our polyomino-safe system produces self-healing systems when applied to most self-healing systems.

1 Introduction

Nanotechnology presents obvious and enormous potential. Manipulating objects on that scale explicitly is infeasible though. As a result, the discipline of nano-scale self-assembly has arisen as a means to harness nanotechnology's promise. In a self-assembly model, small components attach to each other using simple local rules, producing large complicated shapes. DNA has two properties that make it a natural tool for self-assembly. First, strands of DNA naturally store strings of data that can be used to identify themselves. Second, for every strand of DNA there is a complementary strand that will attach. Thus, DNA provides a means to generate local rules in which two pieces will want to attach if they have complimentary strands of DNA. In addition, the lab techniques for manipulation of DNA are already well-developed because of their many other applications. As a result of these factors, DNA self-assembly has been used in many nanoscale applications including as a means to perform computation [1,2], to produce patterns [3,4], and to produce nano-scale machines [5,6,7,8,9,10].

One particularly well-studied type of DNA self-assembly is the tile model. Rectangular DNA molecules have been formed that have a piece of single-stranded DNA on each side [11]. An abstract version of their behavior, the

* Department of Computer Science. Research supported in part by NSF grants 1104592 and 1097249.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 112–126, 2009.

asynchronous Tile Assembly Model (aTAM), was introduced by Rothemund and Winfree [12]. In this model, we think of each molecule as a *tile* with each of the strands as a *glue*. Each glue has an affinity for itself called its strength (which can be controlled in the lab by changing the length of the strand). In aTAM, assembly starts from a structure consisting of a single tile called the *seed*. One by one, tiles attach to the existing structure under the constraint that a tile may only attach to the system at a location if its glues that match the structure at that location exceed the *temperature*, a parameter of an aTAM system (corresponding to the actual temperature of the solution in which the assembly is taking place). Study of aTAM has led to many interesting tile systems such as counters [13,12,14,15] and systems capable of Turing-universal computation and producing arbitrary computable shapes [16,3,17].

While aTAM is a valuable tool for analyzing tile systems, it does not perfectly match behavior in the laboratory. For example, a tile may attach with a strength lower than the temperature. Such "insufficient attachments" will fall off much more quickly than attachments at the temperature but can persist if another nearby attachment locks them in. The experimental rate for such problems is somewhere between 1% and 10% [18], and a great deal of research has gone into schemes to minimize the effects of these errors [19,20,21,5,18,22,23]. Another problem that can occur is a large portion of an assembly may fall off in the middle of construction. This behavior can cause problems when an assembly intended to grow in one direction is not designed to grow uniquely from other directions. Systems resilient to this problem (called self-healing systems) have been developed for several interesting assembly problems [24,25].

Implicit in most of these error-reduction schemes is the concept of *block replacement*. In block replacement, we start with a tile system that forms some shape we want but has other undesirable properties such as susceptibility to the kinds of errors we described above. We then construct a new system by replacing each tile in the old system by a larger set of tiles. These tiles assemble into a rectangle that functions like the tile they are replacing in the new scheme, and at the same time they locally prevent the problems of the original system from occurring. For example, the block might halt its own formation when an insufficient attachment occurs giving the error time to fall off before it gets locked into the assembly [19,20] or be only able to grow in a desired subset of directions [25]. While the inuition is clear, a precise definition of what should be called a block replacement seems to be absent.

Another assumption aTAM makes is that tiles attach to the existing structure one by one. In practice, there are large numbers of each tile floating around in solution and if two tiles have enough attraction between them to attach independent of the existing structure they will do so. This new supertile or *polyomino* can use the glues from both tiles and may be able to attach to the super-structure in places where neither tile could attach individually. Such a model of assembly was first discussed by Aggarwal et al. [15]. In that paper, they propose what they call the q-tile assembly model, which permits supertiles consisting of q or fewer tiles. In that paper, polyominoes are seen as a tool to potentially produce

more efficient tile systems. Similarly, Demaine et al. proposed doing DNA self-assembly in stages [26]; different sets of tiles would be allowed to assemble in different test tubes, and then those test tubes would be mixed so that resultant polyominoes could attach to each other. By doing so, they produced tile systems that could theoretically produce shapes more efficiently than those in single-stage self-assembly. Polyominoes are not a strictly positive effect though. aTAM assumes single tiles attach one by one, but polyominoes may attach using their shared glues in places none of the individual tiles could attach by themselves. As a result, the intended assembly may be derailed. Tile systems not prone to such problems are called *polyomino-safe* and were first discussed by Winfree [24]. In his paper, he proposes a 5×5 block replacement scheme that would make a class of tile systems he calls *transformable* polyomino-safe. The primary requirement for a system to be transformable is that each side of a tile is always either used to attach that tile to the existing assembly or always used as an attachment for future tiles. While many natural tile systems have this property, not all do. In particular, any system that wants to be able to regenerate from more than a single location cannot have this property.

Polyomino tile attachments are not only a possibility, they are frequent enough that some experiments have depended on them [21]. Thus, we need a way to produce polyomino-safety in as much generality as possible.

1.1 Our Results

We begin our paper by presenting the definitions of aTAM, its extension for polyominoes, and block replacement. While there is an intuitive understanding of what a block replacement scheme should do in the literature, a formal expression of this intuition is subtle, and we know of no paper giving a rigorous definition.

Then we develop a 6×6 (slightly larger than Winfree's scheme for transformable systems) block replacement scheme that guarantees polyomino-safety. Our system will work for a class of tile sets we call *block admissible*. This restriction is rather minor as any system failing this requirement not only cannot have a polyomino-safe block replacement scheme but can have no non-trivial block replacement scheme whatsoever. We then show that the Chinese remainder counter is not polyomino-safe and present a 3×1 polyomino-safe block replacement scheme for it. We also show that determining whether a given tile system is polyomino-safe is undecidable. The proof is more generally applicable and can be applied to other important properties of a tile system (such as whether a system is self-healing). Finally, we show that our polyomino-safe block replacement scheme preserves self-healing for any block admissible self-healing tile system.

2 Definitions

The tile assembly model was originally developed by Rothemund and Winfree [12]. Informally, a tile is a square with glues on each side. When the glues of two

tiles on corresponding sides match the tiles will want to attach. Here, we present a slightly modified version from the standard.

Formally, let Σ be a set of glues containing a distinguished glue *null*. Let δ denote the set of four directions $\{N, S, E, W\}$, with the inverse of a direction defined naturally. Associate each direction with a unit vector $v_N = (0, 1)$, $v_S = (0, -1)$, $v_E = (1, 0)$, and $v_W = (-1, 0)$ respectively. A tile t is defined by its four glues, one for each direction in δ, denoted $\sigma_i(t)$ for each i in δ, drawn from Σ. We define a *tile system* as a tuple $< T, s, g, \tau >$. Here, T is a set of tiles, s is a distinguished seed tile, g is the *glue function* from $\Sigma \times \Sigma$ to the non-negative integers, and τ is the *temperature*, a positive integer. We assume $g(x, y) = 0$ for $x \neq y$ (glues only attach to themselves) and that $g(null, null) = 0$ (*null* is inert). The standard aTAM model has s in T, but in this paper we will not include s in T. This modification may correspond to the seed being much rarer than the rest of the tiles or the seed being generated by some special unique process. We will use this assumption to preclude two large polyominoes both of which contain the seed from interacting with each other.

A *configuration* for a tile system is a map from $\mathbb{Z} \times \mathbb{Z}$ to $T \cup \{s\} \cup \{empty\}$. Let C and D be two configurations such that C matches D except at (x, y) where C is *empty* and D is some tile $t \in T$. t is *attachable* to C at (x, y) if the sum of its glue functions with the surrounding tiles is at least the temperature: $\sum_{d \in \delta} g(\sigma_d(t), \sigma_{d^{-1}} C((x, y) + v_d)) \geq \tau$. If this is the case we write $C \rightarrow D$. Define a sequence (possibly infinite) of configurations $\{C_i\}$ to be an *assembly sequence* if $C_i \rightarrow C_{i+1}$. We say D is *derivable* from C (denoted $C \rightsquigarrow D$) if there is an assembly sequence beginning at C whose limit is D. The set of *reachable* configurations is the set of configurations derivable from the configuration that is s at $(0, 0)$ and *empty* elsewhere.

We now define the *polyomino Tile Assembly Model* (pTAM) to reflect the possibility of polyominoes interacting during assembly. As in aTAM a tile configuration is a mapping from $\mathbb{Z} \times \mathbb{Z}$ to $T \cup \{s\} \cup \{empty\}$. We define the set of *reachable* configurations in pTAM recursively as follows. All configurations that are a tile from T in a single coordinate or s at $(0, 0)$ and *empty* elsewhere are reachable. Two configurations C and D are *compatible* if at least one of them is *empty* at each coordinate. For two compatible configurations the *composition* is the configuration that takes on C or D's value wherever one of them is non-*empty* and is *empty* elsewhere. Given two reachable, compatible configurations their composition is reachable if

$$\sum_{(x,y)} \sum_{d \in \delta} g(\sigma_d C(x, y), \sigma_{d^{-1}}(D(x, y) + v_d) \geq \tau.$$

In other words, two polyominoes can attach if they overlap nowhere and the sum of the matching glue strengths everywhere they are adjacent exceeds τ. Since all single tile configurations are reachable and the attachment function for pTAM agrees with aTAM in that case, the reachable configurations under aTAM are a subset of the reachable configurations under pTAM. We say that a tile system is *polyomino-safe* if the reachable configurations under aTAM are exactly the

reachable configurations under pTAM containing s. That is, we allow arbitrarily complicated polyominoes to form as long as their formation does not alter what configurations are reachable from the seed.

Many papers add a desired property (for example self-healing or reliable stochastic assembly) to a tile system by replacing single tiles with blocks of tiles. The intuition for what such systems should do is very natural, but we know of no paper that writes out a formal definition of what such a construction should achieve. We propose a definition of what it means for a system to be a *block replacement* of another system here.

For two tile systems $X =< T, s, g, \tau >$ and $Y =< T', s', g', \tau' >$ a (m, n)-*blowup function* ϕ is a function from $T \cup \{s\}$ to $(T' \cup \{s'\})^{m \times n}$. Let Φ map a configuration C of X to a configuration $\Phi(C)$ of Y produced by saying that the $m \times n$ block of tiles starting at $(mx + a, ny + b)$ (for some fixed offset (a, b)) is $\phi(C(x, y))$. We say Y is an (m, n)-*block replacement* of X under ϕ if:

1. The image of all reachable configurations in X under Φ are reachable in Y.
2. For any reachable configuration D of Y there are configurations D' in Y and C reachable in X such that $D \rightsquigarrow D'$, $D' = \Phi(C)$, and there is at least one non-*empty* square in D in every block corresponding to a non-*empty* square of C.

Here, the first constraint says that each reachable configuration of X maps to a reachable configuration of Y. The second constraint is in some sense an inverse of the first. It would be too strong a requirement for every reachable configuration of Y to be the image of a reachable configuration of X. Tiles attach one by one in Y, but every tile in X corresponds to multiple tiles in Y. Thus, there must be intermediate configurations that do not correspond perfectly to any configuration of X. Hence, we allow ourselves to grow a configuration of Y until it agrees with something in X. Letting Y grow until it could reach any configuration that matches a configuration of X would be too weak a constraint though. In the extreme case, Y could grow in some arbitrary fashion so long as its terminal configurations match those of X. This case would not match our intuition that Y should have essentially the same growth dynamics as X but with each tile being replaced by several. We would not want to say one system emulated another if the second grows in some completely different direction, and the two systems's assemblies only converged much later. Thus, our definition restricts Y to grow to match X only by finishing blocks it had started.

There may be room for variation in the above definition, but our definition fits with all examples of block replacement we know of in the literature. With the above definition, it may be the case that two systems work the same way when starting from the seed but have divergent behavior when starting from different beginnings. A stronger concept would be a block replacement where if both systems start out in analagous positions they continue analgously regardless of whether or not the original positions were reachable from the seed. We define Y to be a *strong block replacement* of X if:

1. The image of all reachable configurations in X are reachable in Y, and for all configurations C in X, the image of all configurations derivable from C under Φ are derivable from $\Phi(C)$ in Y.
2. For all configurations C in X, for any configuration D such that $\Phi(C) \rightsquigarrow D$ there are C' in X and D' in Y such that $C \rightsquigarrow C'$, $D \rightsquigarrow D'$, $\Phi(C') = D'$, and there is at least one non-empty square in D in every block corresponding to a non-empty square in C'.

3 Universal Block Replacement for Polyomino Safety

In this section we present a strong $(6, 6)$-block replacement scheme for polyomino safety at temperature 2. There are two fundamental barriers to producing such a scheme that cannot be directly overcome. The first problem is the seed. Regardless of what scheme we use, the whole system must be able to form from the seed. Thus, any sub-assembly of the system must be a potential polyomino, so if two sub-assemblies attaching to each other is a potential problem in the original system there is little we can do to fix it. Our approach to this problem is to treat the seed as a special tile that only starts assembly and does not appear in the solution as stated in the definitions. The second problem is that some tile systems can never have a non-trivial block replacement.

3.1 Block Admissibility

There are essentially two reasons a tile system cannot have a non-trivial (larger than $(1, 1)$) block replacement scheme. First, a system can never require a tile to attach using glues on opposite sides (i.e. north and south, or east and west). If such an attachment were required no non-trivial block replacement scheme would be possible. Consider a situation in the original tile system where a tile will attach using its north and south glues. In any block replacement longer than 1 in this dimension, there is no single tile where there is enough information to determine if this tile should attach: the first tile that attaches in this block must attach using either only the north or only the south face, which would be an error if the other side is not present.

The second problem is when two different tiles might be able to attach at the same location but using glues from different sides (e.g one tile could attach using its north and east glues while another tile could attach using its north and west glues). In the original system either one tile or the other would attach first, locking the other tile out. In a block replacement, this process would not be atomic. The first tile for one of the blocks can start attaching on one side or corner while tiles for the other block start attaching somewhere else. The result is that neither of the blocks can form completely, breaking block replacement.

Because no system having either of these two properties can have a meaningful block replacement we call systems where tiles never attach using opposite glues and never allow two different tiles to simultaneously be able to attach at the same location *block admissible*.

Definition 1. *A tile system is block admissible if:*

1. *No tile can ever attach in a way that requires both its north and south glues or its east and west glues.*
2. *It is never possible for two tiles to attach at the same location using glues from different directions.*

In fact, block admissibility is a sufficient condition for polyomino-safe block replacement.

3.2 The Polyomino-Safe Block Replacement Scheme

The workhorse of our construction is the 6×6 block presented in figure 1(a). All glues are unique in the interior of a block. On a face of a block, the glues match the glues on the face of another block if and only if those two faces had the same glue in the original system. If the glue in the original system was strength 1 all glues on the face are strength 1. If it was strength 2, we also make the glue on the third tile strength 2. If the tile was inert we make all the glues strength 0. The block has two useful properties. First, the whole block can form from any complete (non-inert) face and a single tile attached to that face. Second, it is easy to check that the system has no polyominoes larger than size 2 (this even applies for the polyominoes that cross blocks using the strength 2 glues on faces between blocks), and none of these polyominoes have two faces on the exterior of the block. Because the seed block will have no face to grow from, we must use a different construction for the block corresponding to the seed, which we show in figure 1(b). The full block can assemble if the seed tile s is present, but no polyomino of size larger than 2 can assemble otherwise. We handle the glues on the faces as we did previously.

Theorem 1. *The system described above is a polyomino-safe strong* $(6, 6)$*-block replacement scheme for any block admissable tile system.*

Proof. Call the original system X and the system produced by our transformation Y. We first verify that Y is a strong block replacement scheme for X. Given only the seed tile of our block replacement scheme the whole seed block can form, so the image of the seed tile is reachable. Thus, we may start with a configuration of X and its image in Y, and proceed by induction on the length of the assembly sequence. Consider an assembly sequence in X, and assume the image of the kth configuration is derivable. Consider the $(k+1)$th tile attachment and the image of the location where it would attach in Y. The entire faces corresponding to whatever tiles it used to attach are present. It either attached using a single strength 2 glue or two strength 1 glues. If it used a strength 2 glue in X the face it used has a strength 2 glue in Y, so a first tile of the block can attach and the rest of the block can attach using that face. Similarly, if two strength 1 glues were used in X then a first tile in that block can attach at the corner between the two faces and either face is sufficient for the rest of the

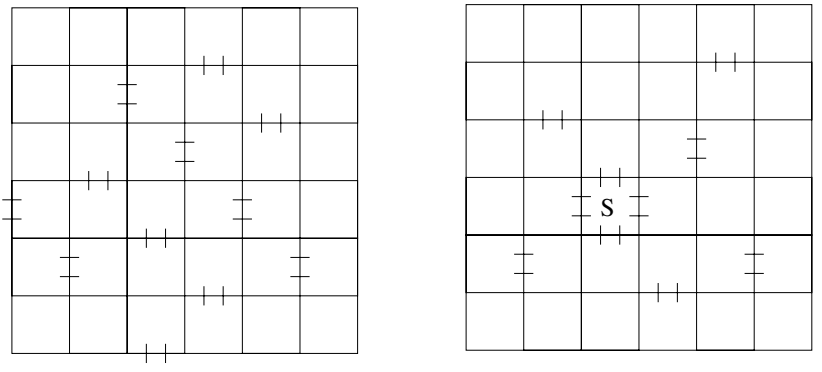


Fig. 1. (a) The basic structure for the polyomino-safe block replacement scheme where the original block had strength 2 glues in the south and west and strength 1 glues in the north and west. (b) The block corresponding to the seed in our replacement scheme with the seed of the new system labelled s here. Two lines indicate strength 2 glues, and all other glues are strength 1.

block to attach. Thus, the images of all configurations derivable from a configuration in X are derivable from its image in Y, and the images of all reachable configurations are reachable.

The second part of the block replacement definition can be broken into two pieces; first, we show that no erroneous tile (a tile that precludes the assembly from being extended to the image of some derivable configuration) can ever attach, and then we show that we can complete any assembly to be the image of a derivable configuration without introducing new blocks. For the first, consider the first time an erroneous tile attaches. All glues internal to blocks are unique in the assembly, so the error must occur on the border of some block. All tiles currently in the configuration are correct, so the error tile must attach using only the faces of valid blocks. But then the tile must either use a strength 2 glue from a face corresponding to a strength 2 tile in X or two strength 1 glues from faces corresponding to strength 1 glues. In either case, the tile must be part of a block corresponding to a tile that could have attached in that location in X. Also, since there is only one place for the first tile of a block to attach given the sides it attaches from, this tile cannot be erroneous because it does not match tiles that have already attached in the block. For the second part, a tile can only attach if a tile in its block is already present or using glues from other blocks whose preimage tiles would allow its preimage tile to attach. Thus, any assembly sequence in Y induces an assembly sequence in X by adding a tile in the original system the first time a tile from its block attaches. We can then extend our configuration by completing blocks one by one in this assembly sequence's order. The result is the image of the final configuration in the assembly sequence of the original system, and no tiles from outside blocks that already had a tile were needed.

We finally are ready to establish polyomino safety. By inspection, there are no polyominoes larger than two tiles that do not contain the seed. Thus, it is sufficient to show none of these dominoes can attach to the assembly in an erroneous location. Consider the first error generated by a polyomino attachment. Since all glues internal to a block are unique a polyomino must still attach at the appropriate coordinates in a block. The error cannot attach using glues from any tiles in the same block because all previous tiles are correct, and if one tile in a block is correct completing the rest of the block must also be correct. Thus, the only way a first error could occur is if a polyomino is using glues on the boundary of two blocks. But none of our dominoes has more than one face on a boundary, so a single tile could attach in those places any time a polyomino could. Thus, there can be no first polyomino error completing the proof.

3.3 Higher Temperatures

Our construction above can easily be extended to higher temperatures. Regardless of the temperature, there are only two ways a tile can attach in a block admissible system: using a glue with strength equal to the temperature from a single tile or using the combination of two glues that share a corner. In the first case, we can put the first attachment in the middle of a face of our block replacement. In the second case, the block must start assembling at the corner. Given a tile t in a system of temperature τ we can provide a $(6, 6)$-block replacement as follows:

1. For all directions with glue strength at least τ place a strength τ glue on the outer face of the third tile on that face.
2. For all directions with glue strength s less than τ place a strength s glue on the first and sixth tiles of that face.
3. Place a strength τ glue everywhere the temperature 2 polyomino-safe block replacement scheme had a strength 2 glue in the interior of the block.
4. Place a strength $\lceil \frac{\tau}{2} \rceil$ everywhere the temperature 2 polyomino-safe block replacement scheme had a strength 1 glue.

Theorem 2. *The system described above is a strong polyomino-safe block replacement scheme.*

Proof. The proof proceeds as above, using the facts that any block can form completely given one of its tiles and a face, the first tile in a block will only attach if the preimage of its block could attach in the original system, there are no polyominoes without the seed of size larger than 2, and none of the polyominoes has more than one face on a boundary between blocks.

4 Complexity Properties

Ideally, we would like an algorithm to verify if a tile system is polyomino-safe. Unfortunately the problem is undecidable.

Theorem 3. *Determining if a given tile assembly system is polyomino-safe undecidable.*

Proof. aTAM is strong enough to emulate a Turing machine by producing the entire tape after each step [3]. Take a polyomino-safe implementation of a universal Turing machine, and add a structure that is not polyomino-safe that only attaches to a specific state tile. Determining whether a Turing machine will reach this state is undecidable, so determining if a polyomino-safety violation can occur is undeciable as well.

The problem is still undecidable even given a finite final structure because the potential problem can be produced by a Turing machine that is not part of the normal assembly and assembles as a polyomino. Note that there is nothing special about polyomino-safety in the above proof: it would apply to any property of an assembly system that is only present if a certain tile attaches. In particular:

Theorem 4. *Deterimining if a given tile assembly system is self-healing from a given configuration is undecidable.*

5 Polyomino Safety in Existing Systems

Many existing systems are already polyomino-safe. For example, consider rectilinear systems (like the Sierpinski tile system [3]) as shown in figure 2 in which tiles form an L using strength 2 glues and fill in the L with strength one glues. Arbitrarily long polyominoes can form from either of the leg tiles. There is never any place for them to attach inside the system though, so all rectilinear systems are polyomino-safe. Similarly, a basic binary counter as described in Adleman et al. [13] can easily be verified as polyomino-safe. Polyomino-safety does not always come for free though.

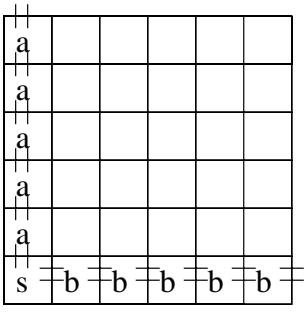


Fig. 2. The basic rectilinear system in which long strings of the same tile are formed on the west and east edges from the seed s. The interior is then filled in with strength 1 glues.

5.1 The Chinese Remainder Counter

Consider the Chinese remainder counter system, described as follows. Let p_1, $\ldots p_n$ be distinct primes. Column i of the counter consists of tiles $a_1^i \ldots a_{p_i}^i$ such that there is a strength 3 glue between the north face of a_k^i and the south face of a_{k+1}^i for k from 0 to $p_i - 1$ and a strength 2 glue from the north of $a_{p_i}^i$ to a_1^i as shown in figure 3. The east and west glues of each tile are strength 1 matching with the west and east glues on all tiles in the adjacent columns. We take the row of tiles $a_1^1 \ldots a_1^n$ as our "seed" (a minor modification of the system can allow for a single-tile seed [25]). At temperature 3 each column can count from 1 to p_i by itself, stalling at transitions from p_i to 1. If the column on either side has a tile in the next row though, it can use the additional strength 1 glue to roll over from p_i to 1 again. Thus, this system will grow until all columns are simultaneously on the p_i to 1 transition, which happens at row $\prod p_i$. The Chinese remainder counter is not polyomino-safe. Single-width columns of tiles up to size p_i can form by themselves. As all glues between adjacent columns are the same, such polyominoes of length 3 or greater may attach at any point to which an adjacent column has progressed, whether or not that string of tiles is appropriate. Thus, the Chinese remainder system is unpredictable in the presence of polyominoes.

We could of course use the 6×6 construction to make the system polyomino-safe. However, in this particular case we can do much better, using a $(1, 3)$-block replacement scheme where we reduce the temperature to 2 and replace each tile with three tiles as illustrated in figure 4. For tiles between 1 and $p_i - 1$ the middle north and south glues have strength 2, and the glue from p_i to 1 has strength 1. We can think of this construction as adding an additional buffer tile on the left and right of each of the old blocks and then lowering the temperature of the system to 2 (making north/south glues 2 normally and 1 on the rollover from p_i to 1). The block replacement still permits long chains of tiles to form in the center of each column. However, these chains can no longer cause problems.

Theorem 5. *The tile system described above is a strong polyomino-safe* $(1, 3)$*-block replacement scheme.*

Proof. The proof for strong block replacement is very similar to the one in Theorem 1, and we omit it here. For polyomino-safety, we see by inspection that the only polyominoes that can form without the seed are chains of middle blocks such that there are no p_i to 1 attachments. Consider a first polyomino

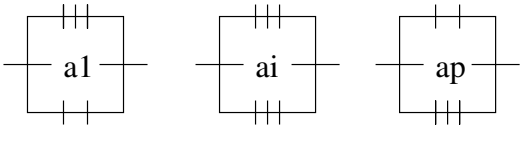


Fig. 3. The first, ith $(1 < i < p)$, and pth, tiles in a column corresponding to the prime p. The east and west glues of each tile match the west and east glues of the adjacent columns.

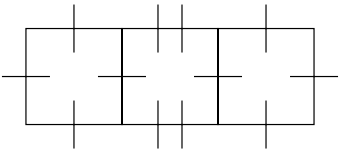


Fig. 4. The polyomino-safe (1,3)-block replacement scheme for the Chinese remainder counter. The middle north (south) glue is strength 1 instead for the step from p_i to 1.

error in some assembly sequence. Every tile uniquely determines what the tile above it and below it is if there is such a tile. In fact, since there are no p_i to 1 attachments in these polyominoes we know that if one tile in a polyomino belongs in the location it attached all the other tiles must also belong where they attached. But the east and west glues of a middle tile are unique to the block they are part of, so any attachment using one must put that tile in its correct place. Similarly, any time a north or south strength 2 glue is used that tile must also be correct because those glues are sufficient to determine if a tile should attach in the original system. All that remains is the strength one north or south glues. However, using the block replacement property if there were a tile for the north glue to attach to then all blocks below it must have already started forming, and the polyomino would be correct. Finally, a strength 1 glue from the south is not sufficient for the polyomino to attach, and the system is polyomino-safe.

Note that our system reduces the temperature to 2, a desirable property. Normally, a $(1, 2)$-block replacement is used to lower the Chinese Remainder counter's temperature, so our polyomino-safe system represents only a 50% increase in size over that basis.

6 Self-healing and Block Replacement

Occasionally, a large portion of an assembly can get knocked out of a tile system. A system is self-healing if it can rebuild itself correctly and completely after such an event provided one of a set of relatively small pieces is intact. Essentially, there are two aspects of a system being self-healing. For two configurations C and D, $C \preceq D$ if C is *empty* everywhere D is *empty*, and $C(x, y) = D(x, y)$ whenever $C(x, y)$ is not empty. A tile system is *immutable* if for all reachable configurations D, $C \preceq D$ and $C \rightsquigarrow E$ imply $D(x, y) = E(x, y)$ when $D(x, y)$ and $E(x, y)$ are both not *empty*. Thus, a system is immutable if whenever a tile attaches no other tile can take its place if it falls off. A tile system is *progressive* for a configuration B if for all reachable D, when $B \preceq C \preceq D$ there is E such that $C \rightsquigarrow E$ and E is not *empty* wherever D is not empty. Thus, a system is progressive from B if any tiles that fall off can be recovered as long as B remains. A system is progressive from a set of configurations $\mathcal{B}$ if it is progressive from all $B \in \mathcal{B}$. A system is *self-healing* from $\mathcal{B}$ if it is immutable and progressive from $\mathcal{B}$. We now prove that a strong block replacement of a self-healing system is self-healing provided all tiles in a block have unique internal glues.

Note that an immutable system can never allow two different tiles to be able to attach in the same location. Thus, a self-healing system is block admissible if it never uses glues on opposite sides of a tile to attach. Block admissibility is sufficient to turn a system that is self-healing into a system that is self-healing *and* polyomino-safe.

Theorem 6. *A strong block replacement of a tile system that is block admissible and self-healing from $\mathcal{B}$ is self-healing from $\Phi\mathcal{B}$ (the image of $\mathcal{B}$ under the block replacement) if the tiles within a block all have unique glues.*

Proof. Let X be the old system and Y be the new system. Progressiveness from $\Phi\mathcal{B}$ follows immediately from the ability to imitate any assembly sequence in X with a sequence in Y. For immutability consider some reachable configuration D and $C \preceq D$. Consider the first attachment of a tile t to C that does not match D. This attachment can use no glues within its block because all glues are unique, and only the one correct tile can use them at any location. If the attachment uses only glues from outside its block the preimage of t's block in X must have been able to attach in that location in X. Then since X is immutable, and any assembly sequence in Y induces a sequence in X, t must come from the same block that was present there in D. Since each tile in a block can only attach at one location in that block t must match the tile in D, and Y is immutable.

In particular, our $(6, 6)$-block replacement scheme meets the requirements of the theorem, allowing us to make block admissible self-healing systems polyomino-safe as well.

7 Open Problems

One potential improvement to our results would be a smaller polyomino-safe block replacement scheme. It is clear that a 2×2 block replacement is too small, but we have no proof that a 3×3 scheme cannot exist. The primary challenge to creating polyomino-safe block replacements is spreading out the necessary strength 2 glues, so that they do not allow polyominoes that are too big to form. A 6×6 scheme allows every strength 2 glue to be isolated from every other, bounding the polyomino size by 2. It is possible that a smaller polyomino-safe block replacement exists that allows larger polyominoes to form. Another worthwhile question is how to address systems that are neither polyomino-safe nor block admissible.

References

1. Barish, R., Rothemund, P., Winfree, E.: Two computational primitives for algorithmic self-assembly: Copying and counting. Nano Lett. 5, 2586–2592 (2005)
2. Winfree, E., Yang, X., Seeman, N.: Universal computation via self-assembly of DNA: Some theory and experiments. In: Landweber, L.F., Baum, E.B. (eds.) DNA Based Computers II. DIMACS, vol. 44, pp. 191–213. American Mathematical Society, Providence (1998)

3. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology, Computation and Neural Systems Option (1998)
4. Rothemund, P.: Folding DNA to create nanoscale shapes and patterns. Nature 440, 297–302 (2006)
5. Soloveichik, D., Winfree, E.: Complexity of compact proofreading for self-assembled patterns. In: Carbone, A., Pierce, N.A. (eds.) DNA 2005. LNCS, vol. 3892, pp. 305–324. Springer, Heidelberg (2006)
6. Yurke, B., Turberfield, A., Mills Jr., A., Simmel, F., Neumann, J.: A DNA-fuelled molecular machine made of DNA. Nature (406), 605–608 (2000)
7. Shin, J.S., Pierce, N.: A synthetic DNA walker for molecular transport. J. Am. Chem. Soc. 126, 10834–10835 (2004)
8. Sherman, W., Seeman, N.: A precisely controlled DNA biped walking device. Nano Letters 4, 1203–1207 (2004)
9. Yin, P., Yan, H., Daniel, X., Turberfield, A., Reif, J.: A unidirectional DNA walker moving autonomously along a linear track. Angewandte Chemie 43, 4906–4911 (2004)
10. Tian, Y., He, Y., Chen, Y., Yin, P., Mao, C.: A DNAzyme that walks processively and autonomously along a one-dimensional track. Angewandte Chemie 117, 4429–4432 (2005)
11. Winfree, E., Liu, F., Wenzler, L., Seeman, N.: Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539–544 (1998)
12. Rothemund, P., Winfree, E.: The program-size complexity of self-assembled squares. In: Symposium on Theory of Computing (STOC), Portland, Oregon, United States, pp. 459–468. ACM Press, New York (2000)
13. Adleman, L., Cheng, Q., Goel, A., Huang, M.: Running time and program size for self-assembled squares. In: ACM Symposium on Theory of Computing, pp. 740–748 (2001)
14. Cheng, Q., Goel, A., Moisset, P.: Optimal self-assembly of counters at temperature two. In: Proceedings of the first Conference on Foundations of nanoscience: self-assembled architectures and devices (April 2004)
15. Aggarwal, G., Cheng, Q., Goldwasser, M., Kao, M.Y., de Moisset Espanes, P., Schweller, R.: Complexities for generalized models of self-assembly. SIAM Journal on Computing 34, 1493–1515 (2005)
16. Lagoudakis, M., LaBean, T.: 2-D DNA self-assembly for satisfiability. In: Winfree, E., Gifford, D.K. (eds.) DNA Based Computers V. DIMACS, vol. 54, pp. 141–154. American Mathematical Society, Providence (2000)
17. Baryshnikov, Y., Coffman, E., Momcilovic, P.: DNA-based computation times. In: Proceedings of the Tenth International Meeting on DNA Based Computers, Milano, Italy (June 2004)
18. Winfree, E., Bekbolatov, R.: Proofreading tile sets: Error-correction for algorithmic self-assembly. In: Chen, J., Reif, J.H. (eds.) DNA 2003. LNCS, vol. 2943, pp. 126–144. Springer, Heidelberg (2004)
19. Chen, H., Goel, A.: Error free self-assembly using error prone tiles. In: [27], pp. 62–75
20. Reif, J., Sahu, S., Yin, P.: Compact error-resilient computational DNA tiling assemblies. In: [27], pp. 293–307
21. Schulman, R., Winfree, E.: Programmable control of nucleation for algorithmic self-assembly. In: [27], pp. 319–328 (Extended abstract; preprint of the full paper is cond.mat/0607317 on arXiv.org)

22. Chen, H., Cheng, Q., Goel, A., Huang, M., Moisset, P.: Invadable self-assembly: Combining robustness with efficiency. In: ACM-SIAM Symposium on Discrete Algorithms (SODA) (2004)
23. Chen, H., Goel, A., Luhrs, C.: Dimension augmentation and combinatorial criteria for efficient error-resistant DNA self-assembly. In: Symposium on Discrete Algorithms (2008)
24. Winfree, E.: Self-healing tile sets. Nanotechnology: Science and Computation, 55–78 (2006)
25. Chen, H., Goel, A., Luhrs, C., Winfree, E.: Self-assembling tile systems that heal from small fragments. In: DNA 13 (2007)
26. Demaine, E.D., Demaine, M.L., Fekete, S.P., Ishaque, M., Rafalin, E., Schweller, R.T., Souvaine, D.L.: Staged self-assembly: Nanomanufacture of arbitrary shapes with $O(1)$ glues. In: Garzon, M.H., Yan, H. (eds.) DNA 2007. LNCS, vol. 4848, pp. 1–14. Springer, Heidelberg (2008)
27. Ferretti, C., Mauri, G., Zandron, C. (eds.): DNA 2004. LNCS, vol. 3384. Springer, Heidelberg (2005)

Robust Self-assembly of Graphs

Stanislav Angelov[1], Sanjeev Khanna[2], and Mirkó Visontai[3]

[1] Google, Inc., New York, NY 10011, USA
angelov@google.com
[2] Department of Computer and Information Science, University of Pennsylvania
Philadelphia, PA 19104, USA
sanjeev@cis.upenn.edu
[3] Department of Mathematics, University of Pennsylvania
Philadelphia, PA 19104, USA
mirko@math.upenn.edu

Abstract. Self-assembly is a process in which small building blocks interact autonomously to form larger structures. A recently studied model of self-assembly is the Accretive Graph Assembly Model whereby an edge-weighted graph is assembled one vertex at a time starting from a designated seed vertex. The weight of an edge specifies the magnitude of attraction (positive weight) or repulsion (negative weight) between adjacent vertices. It is *feasible to add* a vertex to the assembly if the total attraction minus repulsion of the already built neighbors exceeds a certain threshold, called the assembly temperature. This model naturally generalizes the extensively studied Tile Assembly Model.

A natural question in graph self-assembly is to determine whether or not there exists a sequence of feasible vertex additions to realize the entire graph. However, even when it is feasible to realize the assembly, not much can be inferred about its likelihood of realization in practice due to the uncontrolled nature of the self-assembly process. Motivated by this, we introduce the *robust* self-assembly problem where the goal is to determine if every possible sequence of feasible vertex additions leads to the completion of the assembly. We show that the robust self-assembly problem is co-NP–complete even on planar graphs with two distinct edge weights. We then examine the tractability of the robust self-assembly problem on a natural subclass of planar graphs, namely grid graphs. We identify structural conditions that determine whether or not a grid graph can be robustly self-assembled, and give poly-time algorithms to determine this for several interesting cases of the problem. Finally, we also show that the problem of counting the number of feasible orderings that lead to the completion of an assembly is #P-complete.

1 Introduction

Self-assembly is a process in which small building blocks interact autonomously to form larger structures. The self-assembly approach is especially suitable for building molecular scale objects with nano-scale features. Several representative applications and practical models of self-assembly are discussed in [1,2,3,4,5,6,7,8].

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 127–143, 2009.

Rothemund and Winfree [9] proposed the *Tile Assembly Model* to formalize and facilitate the theoretical study of the self-assembly process. This model extends the tiling models based on Wang tiles [10]. In their work, the building blocks, namely the DNA tiles, are abstracted as oriented unit squares. Each side of a tile has a glue type and a (non-negative) strength associated to it. An assembly starts from a designated *seed* tile and can be augmented by a tile if the sides of the tile match the glue types of its already assembled neighbors, and the total glue strength is no less than a threshold parameter τ, referred to as the *temperature* of the assembly.

Reif, Sahu, and Yin [11] introduced a generalization of the Tile Assembly Model, to one on general graphs, called the *Accretive Graph Assembly Model*. The accretive graph assembly is a *sequential* process where a given weighted graph is assembled one vertex at a time starting from a designated seed vertex. The weight of each positive (resp. negative) edge specifies the magnitude of attraction (resp. repulsion) between the adjacent vertices. It is *feasible to add* a vertex to the assembly if the total attraction minus the total repulsion of the already built neighbors is at least the temperature τ. Here, *accretive* suggests the monotone property of the process, i.e., once a vertex is added it cannot be removed later (cf. the *Self-Destructive Graph Assembly Model* [11] and the *Kinetic Tile Assembly Model* where tiles can fall off [12,13]).

The Accretive Graph Assembly Model addresses some of the deficiencies of the Tile Assembly Model. For example, it models repulsion and allows the assembly of general graph structures. A central problem in this model is the *Accretive Graph Assembly Problem* (`AGAP`): Given a weighted graph, a seed vertex, and an assembly temperature τ determine if there is a sequence of feasible vertex additions that builds the graph. Among other results, Reif et al. [11] showed that `AGAP` is NP-complete for graphs with maximum degree 4 and for planar graphs (`Planar AGAP`) with maximum degree 5. Subsequently, Angelov, Khanna, and Visontai [14] improved these results by giving a dichotomy theorem which completely characterized the complexity of `Planar AGAP` on graphs with maximum degree 3 and only 2 possible edge weights. Specifically, it was shown that whenever the allowed edge weights and τ satisfied a simple set of inequalities the problem is NP-complete, and poly-time solvable otherwise.

A drawback of the Accretive Graph Assembly Model is that even when there exists a feasible order of vertex additions to build the graph, its realization in practice may require a careful control over the order of assembly. Such control is arguably hard to implement at the molecular level, and perhaps, even in conflict with the notion of *self*-assembly. To alleviate this drawback, Reif et al. [11] considered a probabilistic variant of the model where at any point of time, the vertex to be build is chosen uniformly at random from the set of all vertices that can be added at that time to the partial assembly. Note that assembly still proceeds by adding one vertex at a time (cf. *insufficient attachment* in [12,13]). One of the main problems in this, so-called Stochastic Accretive Graph Assembly Model is to determine the probability of a graph system being assembled. One approach to estimating this probability is to consider the ratio of the number of

orderings that assemble the input graph to the total number of feasible maximal orderings. Reif et al. [11] showed that the problem of counting the number of ways a given *subgraph* can be assembled is #P-complete, and inferred that determining the probability of assembly of the subgraph is also #P-complete.

However, the following example shows that the number of orderings that assemble a graph against all possible ways to assemble a maximal subgraph can be arbitrary far from the actual probability of assembly. Consider the following graph with seed vertex s, a special vertex t, and two sets of vertices U and V, each of size n. The vertices in U are connected to s with edges with weight $\tau+1$, and to t with edges with weight -1. The vertices in V are connected only to t with edges with weight τ and there is an edge (s,t) with weight τ. Here τ is the assembly temperature. It is easy to see that starting from s, if the first vertex that is built is t, we can complete the remaining vertices in $(2n)!$ possible ways. On the other hand, if we build first any vertex from U, we make t infeasible. Furthermore, there are $n!$ orderings that cannot be extended with additional vertices and do not build the whole graph. Thus, the probability of assembly is exactly $\frac{1}{n+1}$. On the other hand, the ratio of feasible orderings that complete the graph to all possible ways to assemble a maximal subgraph is essentially 1.

Our Results and Techniques. We introduce a new accretive graph self-assembly problem that captures the uncontrolled nature of the self-assembly process: Given a graph G, does G assemble *robustly*, i.e., with probability 1? We refer to this problem as `Robust AGAP` and characterize its complexity as follows.

Theorem 1. `Robust AGAP` *with* 2 *weights is* co-NP*–complete on planar graphs. Moreover, when the number of weights is* 3*,* `Robust AGAP` *is* co-NP*–complete even on graphs with maximum degree* 3*.*

We use ideas developed in [11,14] along with several new combinatorial gadgets. The use of gadgets allows us to follow the same general framework while optimizing various parameters of the problem by finding equivalent gadgets for each case. We note that NP-completeness of `AGAP` on a family of instances does not imply that the corresponding `Robust AGAP` is co-NP–complete. It is easy to construct NP-hard instances of `AGAP` that admit a poly-time decision algorithm for `Robust AGAP`. Also, note when the number of allowed weights is one or the maximum degree is at most two, `Robust AGAP` is trivially solvable in poly-time.

In light of Theorem 1, it is natural to consider `Robust AGAP` with two weights on some subclasses of planar graphs. Towards this end, we study the tractability of `Robust AGAP` with two weights on grid graphs. The setting, with a positive weight w_p and a negative weight w_n modeling attraction and repulsion, respectively, is a natural analog of the Tile Assembly Model. We systematically analyze the complexity of `Robust AGAP` for all possible relationships between w_p, w_n, and the assembly temperature τ. We obtain the following partial characterization.

Theorem 2. `Robust AGAP` on grid graphs *is poly-time solvable when either* $\tau \leq w_p + 2w_n$ *or* $\tau > 2w_p + w_n$*.*

Finally, we strengthen a result in [11] by showing #P-hardness results for counting problems in the context of self-assembly. We omit the details from this version of the paper.

Theorem 3. *The problem of counting the number of ways an instance of* AGAP *can be assembled, namely* #AGAP*, is* #P*-complete.*

Organization. We begin by defining AGAP and Robust AGAP. In Section 3, we show hardness of Robust AGAP using reduction from AGAP by introducing modular gadgets. We also show hardness of Robust AGAP on planar graphs via a new reduction from DNF tautology. In Section 4, we study a related problem to tile assembly in the presence of repulsion, namely Robust AGAP on grid graphs.

2 Preliminaries

We adopt the *Accretive Graph Assembly Model* introduced in [11]. An *accretive graph assembly system* is a quadruple $\langle G, v_s, w, \tau \rangle$, where $G = (V, E)$ is undirected weighted simple connected graph, $v_s \in V$ is the seed vertex, $w : E \to \mathbb{Z}$ is a weight function on the edges, and $\tau \in \mathbb{N}$ is the temperature of the assembly. The assembly process is the following. Initially, the assembly consists of v_s only. The process is a *sequential* attachment of vertices to the assembly, i.e., vertices are built one by one. Given a partially assembled graph and $v \in V$, let $\Gamma(v)$ be the set of already built neighbors of v in G. Now, v can be built iff $\sum_{u \in \Gamma(v)} w(u, v) \geq \tau$. The model is *accretive* because once a vertex is built it cannot be detached from the assembly. For $u, v \in V$ we will use $u \prec v$ to denote that u has already been built when vertex v is built. Note that $\prec$ is an irreflexive, antisymmetric, and transitive relation. We consider the following problems:

Definition 1 (Accretive Graph Assembly Problem (AGAP)). *Given an accretive graph assembly system* $\langle G = (V, E), v_s, w, \tau \rangle$*, determine if* G *can be* assembled sequentially *(in short,* assembled*) starting from the seed vertex* v_s*, and provide a feasible order of assembly,* $v_s = v_{\pi(1)} \prec v_{\pi(2)} \prec \ldots \prec v_{\pi(n)}$*, if one exists. Here,* π *is a permutation of* $\{1, \ldots, n\}$ *and* $n = |V|$*. The* AGAP *problem restricted to planar graphs is referred to as* Planar AGAP*. When there are at most* k *different edge weights in* G*, we denote the problem as* k-Wt. AGAP*. Similarly, when the maximum degree of* G *is* d*, we use* d-Deg. AGAP*.*

AGAP and Planar AGAP are NP-complete [11]. Furthermore, Planar AGAP (hence AGAP) with maximum degree 3 and two distinct weights is NP-complete [14]. Note, even when an instance G of (Planar)AGAP can be assembled, a careful control over the order in which vertices are built may be required to assemble G. To deal with such situations, we introduce the notion of *robust* self-assembly.

Definition 2 (Robust AGAP). *Given an accretive graph assembly system with underlying graph* G*, determine if every* partial *feasible order of assembly of* G *can be extended to a* full *feasible order of assembly of* G*.*

`Robust AGAP` is in co-NP since given any ordering π, we can check in polynomial-time that π is a partial feasible assembly of a strict subset of V that cannot be extended to include additional vertices.

We also consider `Robust AGAP` on *grid graphs* due to its close connection with tile assembly with repulsion.

Definition 3 (Grid Graph). *An $m \times l$ grid graph $G_{m,l} = (V_{m,l}, E)$ is a graph such that its vertices can be arranged in an $m \times l$ rectangular (integer) grid with edges between vertices with ℓ_1 distance* 1.

3 Hardness of Robust Self-assembly

Planar 3SAT. In our hardness results, we will mostly use a reduction from `Planar 3SAT` similar to [11,14]. Lichtenstein proved that `Planar 3SAT`, i.e., `3SAT` with the restriction that the *identifying graph* is planar, remains NP-complete [15]. The identifying graph of a `3SAT` formula ϕ is the following graph G. Vertices of G correspond to literals and clauses of ϕ. There is an edge between a literal vertex and a clause vertex if the literal participates in the clause in ϕ, and there is an edge between every literal and its complement. Middleton showed that deciding the satisfiability of a `Planar 3SAT` formula with a modified identifying graph (see Fig. 1) obeying the following restrictions is still NP-complete [16]:

(1) There is a cyclic path, called the *loop* (the dashed circle denoted by L in Fig. 1), that can be drawn in the plane such that it passes between all pairs of complementary literals, but does not intersect any other edges of G.
(2) The formula ϕ contains only clauses in which the literals are either all positive or all negative.

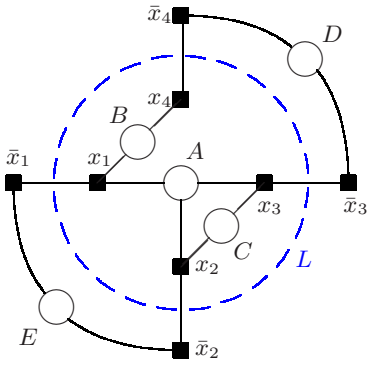
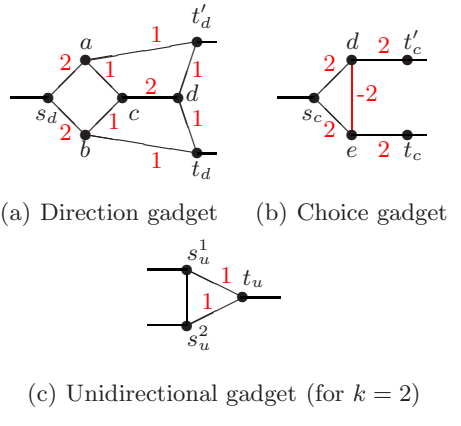


Fig. 1. The identifying graph for the formula $A \wedge B \wedge C \wedge D \wedge E = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2)$



(a) Direction gadget (b) Choice gadget

(c) Unidirectional gadget (for $k = 2$)

Fig. 2. Gadgets for $w_p = 2$, $w_n = -2$, and $w_o = 1$, and temperature $\tau = 2$. Edges without annotation have weight $w_p = 2$.

(3) G can be arranged so that interior (resp. exterior) clauses have positive (resp. negative) literals.
(4) Let $C(\ell)$ denote the set of clauses in which a literal ℓ participates, then $|C(\ell)| \leq 2$ for all ℓ in ϕ.

We assume the loop to be *directed*. This provides a natural (cyclic) ordering of the variables. For x and y we use the notation $xy \in L$ to denote that y succeeds x in L, e.g., $x_1x_2 \in L$, but $x_1x_3 \notin L$ in Fig. 1.

Gadgets. In our hardness constructions, we use modular composition of basic graph gadgets as outlined below. In parentheses, we give the identifying vertices of the gadgets (omitting any additional vertices for clarity). We first describe the gadgets when there are 3 distinct edge weights ($w_p \geq \tau$, $w_n < 0$, and $0 < w_o < \tau$) and then show the required modifications for 2 distinct edge weights ($w_p \geq \tau$ and $w_n < 0$ only). Note that we have maximum degree 3 in the first case, and maximum degree 5 in the latter. Also, the gadgets are planar and $\tau > 1$.

Direction gadget (s_d, t_d, t'_d) [14]**:** The gadget (see Fig. 2(a)) properties are as follows. Note, to realize the gadget, we require $2w_o \geq \tau$.

– If s_d is built, we can complete the gadget: i.e., $s_d \prec \{a, b\} \prec c \prec d \prec \{t_d, t'_d\}$.
– If t_d and t'_d are built, we can complete the gadget: i.e., $\{t_d, t'_d\} \prec d \prec c \prec \{a, b\} \prec s_d$.
– If only t_d or t'_d are built, but not both, we cannot build s_d via the gadget.

Choice gadget $(s_c, t_c,$ [14]**:** The gadget (see Fig. 2(b)) properties are as follows. Note, to realize the gadget, we require $w_p + w_n < \tau$ and $2w_p + w_n \geq \tau$.

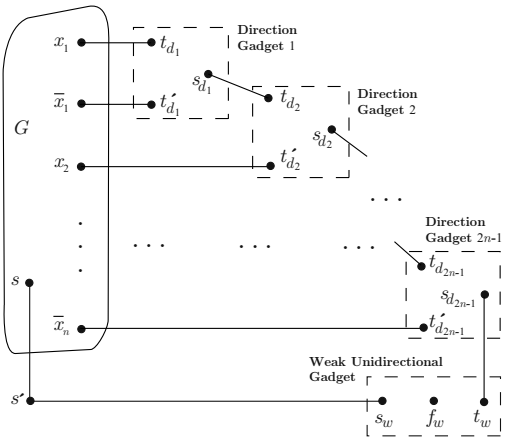
– If s_c is built, we can build either t_c or t'_c but not both via the gadget. For example, building d before e makes the net contribution to e from (s_c, e) and (d, e) equal to 0 which is less than $\tau = 2$.
– If only t_c (resp. t'_c) is built, we cannot build t'_c (resp. t_c) via the gadget. This property follows from a similar argument to the one above.

Unidirectional gadget $(s^1_u, \ldots, s^k_u, t_u)$**:** The gadget (see Fig. 2(c)) properties are as follows. Note, to realize the gadget, we require $kw_o \geq \tau$ $(k > 1)$.

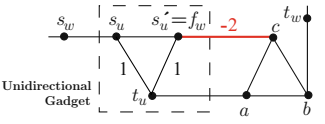
– If s^1_u or s^2_u are built we can build t_u.
– If only t_u is built, we cannot build s^1_u nor s^2_u via the gadget.

Weak unidirectional gadget (s_w, t_w, f_w)**:** The gadget (see Fig. 3(b)) properties are as follows. Note, to realize the gadget, we require $w_p + w_o + w_n < \tau$, $2w_o \geq \tau$, and $2w_p + w_n \geq \tau$.

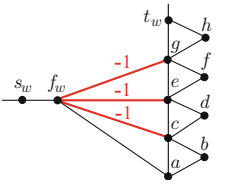
– If s_w is built but not t_w then in any feasible order of assembly for the gadget, we can build t_w, i.e., $s_w \prec s_u \prec s_{u'} = f_w \prec t_u \prec a \prec b \prec \{c, t_w\}$.
– If t_w is built before f_w, then there is an order of assembly in which f_w is made infeasible, i.e., cannot be built. For example, consider the order of assembly: $t_w \prec a \prec b \prec c$. The contribution to f_w from (c, f_w) is -2 which cannot be offset by the weights of (s_u, f_w) and (t_u, f_w).



(a) Composition of graph G induced by a formula ϕ (as in Theorem 4) with $2n-1$ direction gadgets and a weak unidirectional gadget.



(b) Weak unidirectional gadget for graphs with maximum degree 3 and three possible edge weights, e.g., $\{2, 1, -2\}$ for $\tau = 2$.



(c) Weak unidirectional gadget for graphs with maximum degree 5 and two possible edge weights, e.g., $\{2, -1\}$ for $\tau = 2$.

Fig. 3. Template for co-NP–hardness reductions for 3-Deg. 3-Wt. Robust AGAP and 5-Deg. 2-Wt. Robust AGAP and the corresponding weak unidirectional gadgets. Edges without annotation have weight equal to the temperature τ.

The gadgets above can also be constructed using only two edge weights (e.g., $w_p = 2$ and $w_n = -1$ at $\tau = 2$) by increasing the maximum degree to 5. For the Direction and Unidirectional gadgets, we model an edge (u, v) of weight 1 by creating a triangle adding vertex w and setting $(u, v) = -1$, $(u, w) = 2$, and $(w, v) = 2$. For the Weak unidirectional gadget, we can use the construction given in Fig. 3(b). In general, for $\tau > 1$, we require the following edge weight constraints to realize each gadget:

- Direction gadget: $2w_p + 2w_n \geq \tau$.
- Choice gadget: $w_p + w_n < \tau$ and $2w_p + w_n \geq \tau$.
- Unidirectional gadget: $2w_p + 2w_n \geq \tau$.
- Weak unidirectional gadget: $2w_p + 3w_n < \tau$ and $2w_p + w_n \geq \tau$.

In Section 3.2, we will also use the *Asymmetric gadget*.

Asymmetric gadget $(s_a, t_a; w_s \geq 0, w_t \geq 0)$**:** The gadget (see Fig. 5) property is that starting from s_a (resp. t_a) and building all vertices of the gadget except t_a (resp. s_a) the net (weight) contribution to t_a (resp. s_a) is w_t (resp. w_s).

3.1 Robust AGAP Is Co-NP–Complete

To show our hardness results, we reduce AGAP to Robust AGAP. Given an assembly system on graph G, we construct an instance H of Robust AGAP such that there is a maximal ordering that does not assemble all of H iff G can be

assembled. For the purpose, we will compose G with direction gadgets and one weak unidirectional gadget (identified by s_w, t_w, and f_w) such that if all of G can be assembled then the vertex f_w can be made infeasible (via t_w). But, if G cannot be assembled, H robustly assembles (via s_w).

For the basis of our reductions we will use the following result shown in [14].

Theorem 4 ([14]). *Given a* `Planar 3SAT` *formula ϕ, there is an instance of* `3-Deg. 3-Wt. Planar AGAP` *(also an instance of* `5-Deg. 2-Wt. Planar AGAP`*) where the underlying graph $G = (V, E)$ has a subset of vertices $V' \subset V$ satisfying:*

(i) V' consists of the seed vertex and the literals of ϕ: V'=$\{s, x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$,
(ii) each vertex in V' has degree 2,
(iii) G can be assembled iff all vertices in V' can be built, and
(iv) all vertices in V' can be built iff ϕ is satisfiable.

Furthermore, G consists of carefully composed choice and direction gadgets only.

We now show that `Robust AGAP` is co-NP–complete.

Theorem 5. `3-Deg. 3-Wt. Robust AGAP` *is* co-NP*–complete.*

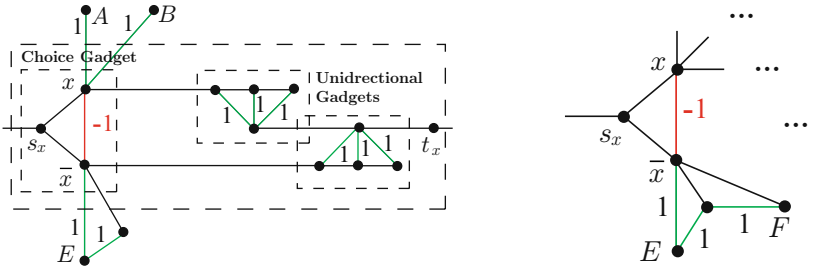
Proof. W.l.o.g., we show the proof for $\tau = 2$ and weights $\{2, 1, -2\}$. By using the same gadgets with different weights, the argument extends to any $\tau > 1$.

We use reduction from an `AGAP` instance with graph G, seed vertex s, edge weights $\{2, 1, -2\}$ and formula ϕ containing literals $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ (as in Theorem 4). From G, we obtain graph H in the following way (see Fig. 3(a)). We use $2n-1$ direction gadgets where the ith gadget is identified by t_{d_i}, t'_{d_i}, and s_{d_i}. The first copy is connected to x_1 and $\bar{x}_1$ by edges (x_1, t_{d_1}) and $(\bar{x}_1, t'_{d_1})$. For $i > 1$, the ith gadget is connected to the $(i-1)$th gadget by an edge $(s_{d_{i-1}}, t_{d_i})$ and to G by an edge (y, t'_{d_i}), where $y = x_{\lfloor \frac{i}{2} \rfloor + 1}$ for even i, and $y = \bar{x}_{\lfloor \frac{i}{2} \rfloor + 1}$ otherwise. The last direction gadget is connected to a weak unidirectional gadget (identified by s_w, t_w, and f_w) by an edge $(s_{d_{2n-1}}, t_w)$. Finally, an additional vertex s' is set to be the seed vertex and is connected by edges (s', s) and (s', s_w). All connecting edges have weight equal to 2.

We now prove that there is a feasible maximal ordering H that does not assemble all of H iff G can be assembled. We will use the fact that $s_{d_{2n-1}}$ can be built without t_w being built iff G can be assembled (from Theorem 4 and properties of direction gadgets). Furthermore, if $s_{d_{2n-1}}$ is built, we can build all of G via the direction gadgets.

(only-if) Suppose G cannot be assembled. Then in any feasible order of assembly of H, $s' \prec s_w \prec f_w \prec t_w \prec s_{d_{2n-1}}$. Now, once $s_{d_{2n-1}}$ is built, we can assemble all vertices of G corresponding to literals via the direction gadgets. Therefore, we can assemble all of G and thus all of H.

(if) On the other hand, if G can be assembled then we can build $s_{d_{2n-1}}$ and therefore t_w before s_w and f_w. Using the properties of the weak unidirectional gadget, we conclude f_w can be made infeasible.



(a) Gadget replacing a variable x, participating in clauses A, B, E along the loop L.

(b) Fragment: Literal participating in two clauses with two literals each.

Fig. 4. Construction for 6-Deg. 3-Wt. Robust AGAP on planar graphs for edge weights $\{-1, 1, 3\}$ and temperature $\tau = 3$. Edges without annotation have weight 3.

Using the equivalent gadgets for the case when there are only 2 possible edges weights, we obtain the next corollary.

Corollary 1. 5-Deg. 2-Wt. Robust AGAP *is* co-NP*–complete.*

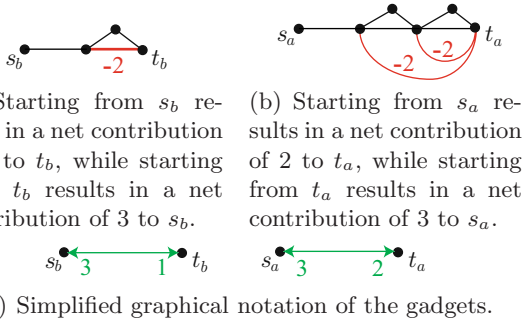
3.2 Robust AGAP on Planar Graphs Is Co-NP–Complete

Note that in the previous section the constructed graph H is *not* planar regardless of G being planar. We show that Robust AGAP on planar graphs is co-NP–complete by using reduction from DNF tautology. We construct a planar graph that robustly self-assembles iff the underlying formula is a tautology.

Let formula ϕ be a Planar 3SAT formula. Then $\bar{\phi}$ is a DNF formula that has the same identifying graph as ϕ up to a permutation of the variables. Let the loop L induce the ordering of the variables $x_1, \ldots, x_n$ and recall that each clause has either 2 or 3 literals. Given $\bar{\phi}$ and its identifying graph, we modify the graph similarly to the constructions given in [11,14] but using different gadgets. We then connect all clauses (preserving planarity) such that if any one clause is built we can build the remaining clauses and all other vertices. In the end, we show that the graph robustly self-assembles iff $\bar{\phi}$ is a tautology.

We now describe the details of the construction below for temperature $\tau = 3$ and draw the edge weights from the set $\{3, 1, -1\}$. For every variable x and its negation $\bar{x}$, we replace the edge $(x, \bar{x})$ in the graph (Fig. 1) with the gadget depicted in Fig. 4(a). For x and y, $xy \in L \backslash \{x_n x_1\}$, we connect the corresponding gadgets with edge (t_x, s_y) with weight $w(t_x, s_y) = 3$. The gadget ensures that unless all literal vertices adjacent to a clause are built (i.e., the clause is satisfied) then for each variable x at most one of x or $\bar{x}$ can be built following the ordering induced by L. Formally, let i be the largest index such that x_i or $\bar{x}_i$ is built. Then, exactly one of x_j or $\bar{x}_j$, for all $j \leq i$, are built if there is no clause with all literals already built.

We connect each literal ℓ to the clauses it participates as follows. If a clause $A \in C(\ell)$ has two literals, and ℓ is induced by the variable with smaller index,



(a) Starting from s_b results in a net contribution of 1 to t_b, while starting from t_b results in a net contribution of 3 to s_b.

(b) Starting from s_a results in a net contribution of 2 to t_a, while starting from t_a results in a net contribution of 3 to s_a.

(c) Simplified graphical notation of the gadgets.

Fig. 5. Asymmetric gadgets for $\tau = 3$ and weights $\{-2, 3\}$. Graphically, we will represent the gadgets with bidirectional edge with corresponding weights at end-points.

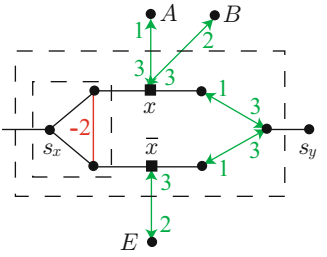


Fig. 6. Composition of gadgets for 9-Deg. 2-Wt. Robust AGAP on planar graphs for edge weights $\{-2, 3\}$ and $\tau = 3$.

then ℓ and A are connected with an edge of weight 2. Otherwise, ℓ and A are connected with an edge of weight 1. We simulate edge weight 2 by a triangle with two edges of weight 1 and one with weight 3 (see Fig. 4(a)). If a literal is connected to two clauses in this manner, then the induced two triangles share the edge with weight 3 adjacent to the literal (see Fig. 4(b)).

Finally, we want to connect all clauses with paths of edges with weight 3 such that if one clause is built then we can build the remaining clauses, planarity is preserved, and the maximum degree of the resulting construction is low. Consider the clauses with only positive literals (similarly negative). A clause $A = x_a \wedge x_b \wedge x_c$ $(a < b < c)$ is contained in clause $B = x_i \wedge x_j \wedge x_k$ $(i < j < k)$ iff $i < a < c < j$ or $j < a < c < k$. Note that the relation is transitive. Let $p(A)$ denote the parent of A, i.e., the clause B that contains A such that there is no other clause C such that C contains A and B contains C. Note that clauses are properly nested and thus preserve planarity.

Connect all clauses with a common parent in a binary tree where edges have weight 3 and clauses are the leaves. *W.l.o.g.*, this tree preserves planarity. Furthermore, it ensures that if a clause can be built then all clauses with the same parent can be built. We then connect the root of the tree with the parent clause of the leaves. We introduce vertices, r_p and r_n, corresponding to the null parents of the clauses with positive and negative literals, respectively, and an edge (r_p, r_n) of weight 3. Note that since we did not connect x_n and x_1 above, this edge also preserves planarity.

The maximum degree of the above construction is 6. Each clause has three edges due to connections with literals (if the clause has two literals, one literal contributes two edges). One edge connects the clause to its parent and to the root of at most two trees of contained clauses. The literals have degree at most 6 and all other vertices have degree at most 5.

For graph G, a maximal order of assembly has the following properties.

- For each variable x, the vertex s_x is built. Therefore, vertex x or $\bar{x}$ (or both) is also built.
- Assume there is a built clause and let C be the first such clause in the ordering. Then all literals participating in C are built before C.
- If a clause is built then all clauses are built: Recall all clauses are connected by weight 3 edges and adjacent to only positive edges.
- If all clauses are built then all vertices are built: Since all clauses and all s_x's are built, each literal receives contribution of at least $3+(-1)+1 \geq 3$. After the literals are built, all of the remaining vertices can be built.
- If no clause is built then exactly one of x and $\bar{x}$ is built, for each variable x. Such a partial assembly corresponds to a certificate that $\overline{\phi}$ is not a tautology.

Hence, we obtain the following theorem.

Theorem 6. `6-Deg. 3-Wt. Robust AGAP` on planar graphs *is* co-NP*–complete.*

Using $\tau = 3$ and only two edge weights, we can modify the above construction but the resulting maximum degree will be 9 (see Figs. 5 and 6).

Corollary 2. `9-Deg. 2-Wt. Robust AGAP` on planar graphs *is* co-NP*–complete.*

4 `Robust AGAP` on Grid Graphs

Grid graphs are of particular interest due to their correspondence to the Tile Assembly Model. For completeness, we mention that `AGAP` on grid graphs with 3 weights is NP-complete by embedding on a grid the hardness construction of [14] for planar graphs of maximum degree 3 and 2 weights (the third weight is introduced to pad the construction to be a grid graph).

A qualitative difference between `AGAP` and `Robust AGAP` is that in instances that assemble robustly, finding a feasible order of assembly is easy, i.e., a simple algorithm of building any vertex (which is feasible at the time) should be able to find such an order of assembly. On the other hand, it is enough to show one maximal ordering on vertices that only partially assembles the input graph to certify a graph is a NO instance for the `Robust AGAP`.

For our analysis, we introduce the recurring notions of *inextensibility* and *forbidden structures.*

Definition 4 (Inextensibility). *Given an accretive graph assembly system $\langle G, v_s, w, \tau\rangle$, a subgraph G' of G with $V(G') \subsetneq V(G)$ is called inextensible if G' can be assembled starting from the seed vertex without building any vertex in $V(G) \setminus V(G')$, and once G' is built no other vertex can be added to the assembly. Such an order of assembly of G' is referred to as* inextensible ordering.

Remark 1. We assume that each vertex is reachable through a path of positive edges. Otherwise it is clear that the instance cannot be assembled.

Definition 5 (Forbidden Structure). *Let G be a grid graph and H a connected subgraph of G. We call $v \in V(H)$ a* boundary *vertex if v is on the grid boundary or $\exists u \in V(G) \setminus V(H)$ such that $(v,u) \in E$. We say H is a forbidden structure if the seed vertex $v_s \notin V(H)$ and, for each boundary vertex v, $\sum_{u \in (V(G) \setminus V(H)) \cap \Gamma(v)} w(v,u) < \tau$, where $\Gamma(v)$ denotes the set of vertices adjacent to v. The size of H is $|V(H)|$.*

Intuitively, the boundary vertices of a forbidden structure can be made infeasible by assembling all the outside neighbors of these vertices. The following theorem gives a *sufficient* condition when these neighbors can be assembled, and hence gives a *partial* characterization of `Robust AGAP` in terms of forbidden structures.

Theorem 7. *If G cannot robustly self-assemble, then there exists a forbidden structure. Conversely, consider a forbidden structure H in G. Let B denote the set of boundary vertices of H, and $\Gamma(B)$ the set of vertices in $V(G) \setminus V(H)$ which are adjacent to some vertex in B or are on one diagonal from a vertex of B. Then if every edge (u,v) such that $u \in \Gamma(B)$ and $v \in V(G) \setminus V(H)$ has weight at least τ, G cannot robustly self-assemble.*

Proof. The first part follows from the definition of forbidden structure. For the second part, consider a forbidden structure H with a maximum number of vertices on the grid boundary. Note that in this case the subgraph induced by $\Gamma(B)$ is connected and have positive edges only. Furthermore, every path from the seed to a vertex in $V(H)$ crosses $\Gamma(B)$. Fix an order of assembly for G and consider the first time it reaches a vertex in $\Gamma(B)$. Since no vertex in $V(H)$ is built at this point, we can build all vertices in $\Gamma(B)$ without using any vertex in $V(H)$. Since $\Gamma(B)$ includes all outside neighbors of H, this ordering makes H infeasible. □

4.1 `Robust AGAP` on Grid Graphs with 2 Weights

We now focus on `Robust AGAP` on grid graphs with two possible edge weights. The case when there is only one possible weight is trivial, i.e., the graph robustly self-assembles iff this weight is $\geq \tau$. Table 1 summarizes all possible cases when there are two possible edge weights, $w_p \geq \tau$ and $w_n < 0$. Note that when

Table 1. Cases of `Robust AGAP` on grid graphs with 2 edge weights $w_p \geq \tau$ and $w_n < 0$

Case	Results
$\tau \leq w_p + 3w_n$	Poly-time solvable
$\tau \in (w_p + 3w_n, w_p + 2w_n]$	Poly-time solvable
$\tau \in (w_p + 2w_n, w_p + w_n]$	Open problem
$\tau \in (w_p + w_n, 2w_p + 2w_n]$	Open problem
$\tau \in (2w_p + 2w_n, 2w_p + w_n]$	Open problem
$\tau \in (2w_p + w_n, 3w_p + w_n]$	Poly-time solvable (Theorem 9)
$\tau > 3w_p + w_n$	Poly-time solvable

both weights are positive the instance is trivial [11]. When there is only one positive weight, it must be at least τ, otherwise the instance is not feasible. When $\tau \leq w_p + 3w_n$, the graph G robustly self-assembles iff there is a spanning tree of positive edges (see Remark 1) since even 3 negative neighbors cannot make a vertex infeasible. If $w_p + 3w_n < \tau \leq w_p + 2w_n$, then we can show G robustly self-assembles iff there does not exists a vertex (other than the seed vertex) with 3 negative edges incident on it. Furthermore, if $3w_p + w_n < \tau$ then the graph G robustly self-assembles iff there is no negative edge in G.

The remaining cases for 2 edge weights appear nontrivial. In what follows, we make progress towards understanding the complexity of those cases by giving a poly-time algorithm that solves one of the cases. In our analysis, we will assume that the seed vertex is connected to its neighbors in G with positive weight edges.

4.2 `Robust AGAP` on Grid Graphs with $2w_p + w_n < \tau \leq 3w_p + w_n$

We now consider a nontrivial case of `Robust AGAP` on grid graphs when there are 2 edge weights, w_p and w_n, such that $2w_p + w_n < \tau \leq 3w_p + w_n$. We show that this case is poly-time solvable since in this case the existence of a forbidden structure is both sufficient and necessary condition of the fact that G cannot robustly assemble. Therefore, we first categorize forbidden structures into groups and then proceed with a theorem giving the desired characterization.

Definition 6 (Nearby Negative Edges). *A pair of disjoint edges e_1 and e_2 with* negative *weights are* nearby *iff they have adjacent nodes (see Fig. 7(a)).*

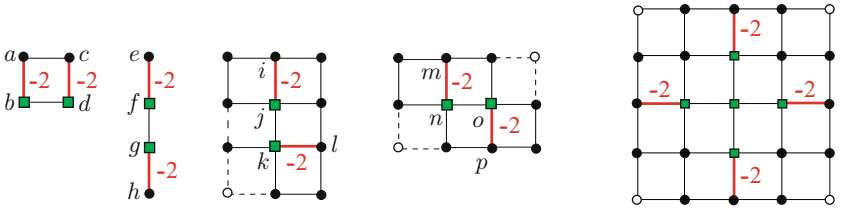
Theorem 8. *If there is a forbidden structure in G with $2w_p+w_n < \tau \leq 3w_p+w_n$ then at least one of the following conditions holds:*

(i) there is a negative path of length 2 or more, or
(ii) there is a negative edge with at least one end-point on the grid boundary, or
(iii) there is an elementary forbidden structure (shown in Fig. 7).

Furthermore, if (i), (ii), or (iii) holds, then G cannot robustly self-assemble.

Proof (of the first part of Theorem 8). If we have a negative path of length 2 or a negative edge with at least one of its end-point on the grid boundary, the statement is trivial. In fact, these are the only forbidden structures of size 1 (see Definition 5). Assume now that there are neither negative paths of length 2 nor negative edges with at least one end-point on the grid boundary.

Consider the boundary vertices of the forbidden structure. If there are two boundary vertices which are adjacent to each other, then we have nearby negative edges. Thus the only possibility is that the boundary vertices are from at distance two from each other (on the diagonal). Also note that if there are three boundary vertices on the same diagonal line, the middle vertex will be an end-point of a negative edge which has a nearby edge (one of the edges with an end-point on the diagonal). Hence, the only remaining possibility is if the boundary of the forbidden structure consists of vertices which are distance two from each other



(a) Nearby negative edges (b) Non-nearby edges

Fig. 7. Elementary forbidden structures for weights $\{-2, 1\}$ and $\tau = 1$. Solid edges without annotation have weight 1; dashed edges have weight 1 or -2. The vertices of the forbidden structures are sown with square nodes.

(only two vertices on each diagonal). The only such structure is formed by 4 vertices vertices arranged in a diamond shape (if one of the 4 vertices is missing we can always complete the structure). In this case, depending on the orientation of the incident edges, we either have nearby edges or the elementary forbidden structure in Fig. 7(b).

Now we prove the second part of the theorem in the following lemmas.

Lemma 1. *Let (u, v) be a negative edge and let p be a neighbor of u that has common neighbors with v. If π is a feasible order of assembly such that $p \prec u \prec v$ in π, then either there is an ordering (possibly inextensible) where v is built before u, or there is an ordering when u is built but v is made infeasible.*

Proof. Note that if (u, v) is a negative edge and u is built before v, then when v is built, it must have 3 neighbors connected with positive edges that have already been built. Now consider the time when p is built in π. If we cannot build the common neighbor, say q, of p and v, then by building u we make v infeasible. This is because q depends on v to be built and vice versa. Now if q can be built, we can either build v, or by building u before, we make v infeasible as it has at most two neighbors connected with positive edge.

Lemma 2. *If there is a negative edge with at least one end-point on the grid boundary or if there is a path of negative edges of length at least 2 then the grid cannot robustly self-assemble.*

Proof. Consider a negative edge (u, v) with at least one end-point on the grid boundary. If both u and v are on the boundary then there is no feasible order of assembly since by building one of the vertices, we make the other one infeasible. When only one end-points is on the boundary, say u, then it must be the case $u \prec v$ in any order of assembly since $2w_p + w_n < \tau$. However, since any neighbor of such u is as in Lemma 1, we can either built v before u or make v infeasible.

Now assume that all negative edges have end-points strictly inside the grid. Let $P = \{(v, u), (u, w)\}$ be a negative path. Consider a feasible order of assembly π. Clearly, $u \prec \{v, w\}$. Since each neighbor of u other than v and w is as in Lemma 1 with respect to either v or w, the claim follows.

Lemma 3. *If there are two nearby negative edges, then the grid cannot robustly self-assemble.*

Proof. W.l.o.g., the premise conditions of Lemma 2 do not hold. We proceed by case analysis on the types of nearby edges given in Fig. 7(a). When there are two parallel nearby negative edges (a, b) and (c, d), consider the first vertex from $\{a, b, c, d\}$ that is built, say a. Now, building c right after a completes the forbidden structure $\{b, d\}$. For what follows, we assume there are no parallel nearby negative edges.

Now consider the case of nearby edges (e, f) and (g, h) with forbidden structure $\{f, g\}$. Consider the first time in some feasible ordering of assembly, a vertex adjacent to say one of $\{e, f\}$ other than g is built. If it is a neighbor of e we can extend the ordering so far to build e. If it is a neighbor of f, we use Lemma 1 to argue that e can be built before f, otherwise e can be made infeasible. Similarly, we can argue that we can build h before g. Note that if after building e, we cannot reach a neighbor of $\{g, h\}$ then the resulting ordering is inextensible. For what follows, we assume there are no such nearby edges.

It remains to argue that if there is any of the remaining combinations of nearby edges (forbidden structures) then there is an inextensible ordering. Consider an order of assembly up to the point where we reach a vertex at distance 1 from a vertex of nearby edges for the first time. Call this vertex r.

First, consider the case of nearby edges (i, j) and (k, l). If r is the North neighbor of i, after r we can build i and follow clockwise the black nodes to build l. Similarly, if r is the East neighbor of l we build l first and proceed in counterclockwise direction to build i. For the remaining choices of r, we can follow the unique paths on the black nodes from r to i and from r to l that do not include the empty node (which might have a neighbor that is built and connected with a negative edge to it). The black nodes are such that if they cannot be built along this path (because of negative edge) it would contradict the choice of r since we reached such neighbor of nearby edges earlier. A special care is needed for the edges shown with dashed line which might be negative edges. However, the only possibility an end-point of such an edge is included in the above paths is if it coincides with r, contradicting the choice of r.

Now, let the reached nearby edges be in configuration as (m, n) and (o, p). Furthermore, *w.l.o.g.* there is no reached configuration as in the previous case (i.e., none of the horizontal dashed edges are negative). Applying the same argument as above does not work since a path from m to p always includes one of the depicted empty nodes and furthermore it can be blocked by the vertical dashed edges if negative. Suppose both vertical dashed edges are negative edges. Then, depending on where r is, we can build all vertices in the row of m (resp. p) making o (resp. n) infeasible. Now assume that at most one of the vertical dashed edges is a negative edge, say the leftmost one. We can show that even if the empty node on the row of m had a neighbor connected with negative weights that is already built we can construct m and p from r. We note that when we try to use such "disabled" vertices it is the case that they are not part of nearby edges. Our argument uses the fact that if we have a negative edge (u, v) that is

not nearby other negative edge, if we build v we can build the two horizontal (resp. vertical) neighbors of u by building the horizontal (resp. vertical) neighbors of v. For example, if the negative edge that disables one of the empty nodes in the Fig. 7(b) is horizontal and r is adjacent to m, then we can build the east vertex of o and therefore the east neighbor of p and p itself. The cases when the negative edge is vertical or r is some other vertex are slightly more involved (we need to argue for at most two non nearby edges) but use the same ideas.

Lemma 4. *If there is a forbidden structure shown in Fig. 7(b) the grid cannot robustly self-assemble (where the seed vertex is not one of the square nodes).*

Proof. W.l.o.g., we can assume that there are no nearby negative edges, otherwise the claim follows from Lemma 3. Consider the first time a round node is reached on this structure. Since there are no nearby negative edges, the round end-points of negative edges shown Fig. 7(b) can be built. Now, the square end-points of those edges cannot be built since the center vertex cannot be built.

Since the conditions of Theorem 8 are poly-time testable (and hence existence of forbidden structures is poly-time decidable), we obtain the following theorem.

Theorem 9. `Robust AGAP` on grid graphs *is poly-time solvable when* $2w_p+w_n < \tau \leq 3w_p + w_n$.

References

1. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539–544 (1998)
2. Rothemund, P.: Using lateral capillary forces to compute by self-assembly. Proc. Nat. Acad. Sci. U.S.A. 97, 984–989 (2000)
3. LaBean, T.H., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J.H., Seeman, N.C.: Construction, analysis, ligation, and self-assembly of DNA triple crossover complexes. J. Amer. Chem. Soc. 122, 1848–1860 (2000)
4. Yan, H., LaBean, T.H., Feng, L., Reif, J.H.: Directed nucleation assembly of DNA tile complexes for barcode-patterned lattices. Proc. Nat. Acad. Sci. U.S.A. 100, 8103–8108 (2003)
5. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biology 2, 2041–2053 (2004)
6. Chelyapov, N., Brun, Y., Gopalkrishnan, M., Reishus, D., Shaw, B., Adleman, L.M.: DNA triangles and self-assembled hexagonal tilings. J. Amer. Chem. Soc. 126, 13924–13925 (2004)
7. He, Y., Chen, Y., Liu, H., Ribbe, A.E., Mao, C.: Self-assembly of hexagonal DNA two-dimensional (2D) arrays. J. Amer. Chem. Soc. 127, 12202–12203 (2005)
8. Malo, J., Mitchell, J.C., Vénien-Bryan, C., Harris, J.R., Wille, H., Sherratt, D.J., Turberfield, A.J.: Engineering a 2D protein-DNA crystal. Angewandte Chemie International Edition 44, 3057–3061 (2005)
9. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: STOC, pp. 459–468 (2000)
10. Wang, H.: Proving theorems by pattern recognition II. Bell Systems Technical Journal 40, 1–41 (1961)

11. Reif, J.H., Sahu, S., Yin, P.: Complexity of graph self-assembly in accretive systems and self-destructible systems. In: Carbone, A., Pierce, N.A. (eds.) DNA 2005. LNCS, vol. 3892, pp. 257–274. Springer, Heidelberg (2006)
12. Winfree, E., Bekbolatov, R.: Proofreading tile sets: Error correction for algorithmic self-assembly. In: Chen, J., Reif, J.H. (eds.) DNA 2003. LNCS, vol. 2943, pp. 126–144. Springer, Heidelberg (2004)
13. Chen, H.-L., Goel, A.: Error free self-assembly using error prone tiles. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 62–75. Springer, Heidelberg (2005)
14. Angelov, S., Khanna, S., Visontai, M.: On the complexity of graph self-assembly in accretive systems. Natural Computing 7, 183–201 (2008)
15. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. 11, 329–343 (1982)
16. Middleton, A.A.: Computational complexity of determining the barriers to interface motion in random systems. Phys. Rev. E 59, 2571–2577 (1999)

Time Optimal Self-assembly for 2D and 3D Shapes: The Case of Squares and Cubes

Florent Becker[1], Éric Rémila[1], and Nicolas Schabanel[2]

[1] Université de Lyon – LIP, UMR 5668 ENS Lyon CNRS UCBL
{florent.becker,eric.remila}@ens-lyon.fr
[2] CNRS, Universidad de Chile — CMM

Abstract. Self-assembling tile systems are a model for assembling DNA-based nano artefacts. In the currently known constructions, most of the effort is put on garanteeing the size of the output object, whereas the geometrical efficiency of the assembling of the shape itself is left aside. We propose in this paper a framework to obtain provably time efficient self-assembling tile systems. Our approach consists in studying how the flow of information has to circulate within the desired shape to guarantee an optimal time construction. We show how this study can yield an adequate ordering of the tiling process from which one can deduced a provably time efficient tile systems for that shape. We apply our framework to squares and cubes for which we obtain time optimal self-assembling tile systems.

Keywords: Self-assembling, Tilings, Time Optimal Construction, 2D and 3D Discrete Geometry.

1 Introduction

Self-assembly is the process by which small entities combine themselves into a bigger shape by local interactions in such a way that the resulting aggregates have interesting global properties. Examples of self-assembly are found in crystal growth, coral reefs, microtubules, and so on. In [7], Winfree proposed a framework for using DNA molecules in order to manufacture nano-artefacts by self-assembly. These artefacts can even be made to encode the result of a computation, leading to a good model for DNA computing [8].

This framework consists in a refinement of *Wang tiles* where the colors of the sides are seen as different kinds of *glues*. These glues have different *strengths* which represent the strength of the affinity between the DNA sequences forming the sides of the tiles. Two equal glues will stick together, with a strength equal to the strength of the glues. The self-assembly of the tiles is controlled by the *temperature*, an integer τ such that: a tile stays attached only if the sum of the strengths of the bonds with the other tiles along its sides is at least τ, otherwise it is torn off from the rest of the crystal by thermal agitation. This model can and has indeed been implemented practically [5] with DNA.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 144–155, 2009.

Theoretical research so far on self-assembly has focused on minimising the numbers of tiles necessary to assemble a given shape. In this paper, we will focus as well on the time needed to assemble a shape. In this model, one can optimally assemble a shape with $O(K/\log K)$ tiles, where K is the Kolmogorov complexity of the shape [1,6]. This direction yields tile sets in which most of the complexity is related to the decoding of a compact representation of the shape, rather than assembling the shape efficiently. Furthermore, as far as we know, the time needed to assemble a given shape has not been studied in detail so far, and asymptotic results have been deemed sufficient. In this paper, we show how a finer understanding of the inherent parallelism of the model can lead to *time-optimal* self-assembly.

Time-constraint-based tile set design. Our approach relies on the study of the time-relationship between the tiles. More precisely, it consists in studying the flow of information within the desired shape, and deducing from it a specific order in which the tiles have to be placed, and from which we finally infer a time-optimal set of tiles. Deducing the tile set from its specifications allows to obtain directly a tile set whose behavior has already been proved formally, which is often easier than obtaining the tile set first and then proving that it behaves correctly. Interestingly enough, this method yields to a new kind of self-assembly process in which the construction is not driven by some master signals[1] but rather by a set of interdependent signals where no one takes over the other nor may progress without the others. This implies that other kinds of relationship between construction signals may have to be considered in future constructions.

On time optimality and tile set design. In order to set up a proper definition of time-optimal tile set for a shape, computer science taught us that the considered tile sets need to allow the construction of an infinite range of sizes for the object. Indeed, the optimal construction time of a *fixed size* object cannot be defined since its construction time can always be made arbitrarily small for any given tile set simply by increasing the concentrations of the tiles. Obtaining a tile set that can produce the desired shape at arbitrary size, is also a desirable design strategy: it's exactly like writing a program P that asks for n and computes $n!$ instead of rewriting a new program P_n each time a new n is requested. Decoupling the problem of decoding of the size and the problem of geometric assembling of the shape allows to focus on each problem separately and to treat each of them with more efficiency. Moreover, as we will see, our tile set can easily be extended by adding computations within the tiles that decodes the size and stops the construction at the right moment.

Our contribution. In this paper, we focus on squares and cubes and answer to the question: what is the optimal time to self-assembly these shapes? We start with the squares and design from the constraints imposed by time-optimality an order in which the tiles have to be placed, from which we deduce the tile set.

[1] For instance, in [2] the construction is driven by the master diagonal signal.

Our tile set allows to construct any square $n \times n$ in optimal parallel time $2n-2$ instead of $3n-5$ for the previously known constructions [2]. Then we extend the result to 3 dimensional self-assembly. In three dimensions, the description of the tile set could become cumbersome, and its verification even more so. We show how, by designing first the order in which the tiles should be added, and then by checking a few conditions of regularity and local determinacy on the order, we get at once the tile set and a proof of its correctness and its time optimality. One can also readily check properties such as the absence of *bubbles* in the 3D self-assembly process, which can be crucial in practice.

2 Self Assembling Tile Systems

Definition 1 (Tiles). *A* tile *is a unit square with one glue on each side. Formally, it is a quadruple of glues* $\langle \alpha_N, \alpha_S, \alpha_E, \alpha_W \rangle$ *chosen from a finite* set of glues Σ.[2] *Each glue* $\alpha \in \Sigma$ *has a* strength $g(\alpha)$*, a non-negative integer. Two tiles may be placed next to eachother only if their common sides hold the same glue* α*; in that case, we say that there is* bond *of strength* $g(\alpha)$ *between these tiles.*

Definition 2 (Tile system). *Formally, a* seeded tile system *is a quintuple* $\mathcal{T} = \langle T, t_0, \tau, \Sigma, g \rangle$*, where:*

- Σ *is a finite set of glues whose strengths are given by* g*.*
- $T \subseteq \Sigma^4$ *is a finite set of tiles.*
- $t_0 \in T$ *is a particular tile known as the* seed*.*
- τ *is a positive integer called the* temperature*.*

The "shapes" produced from a tile system are the one obtained by aggregation from the seed tile, according to the rule stating that a tile may stick to the current aggregate only if the sum of its bonds to the rest of the aggregate is at least the temperature. Formally,

Definition 3 (Productions). *A* configuration *is a partial mapping* $A : \mathbb{Z}^2 \to \mathcal{T}$*, such that all neighboring tiles hold the same glue on their common sides. The domain of* A *is called the* shape realized by A*. We say that a configuration* B extends *a configuration* A *at site* (i,j) *if: the domain of* B *is the domain of* A *plus the site* (i,j)*;* B *is identical to* A *on* A*'s domain; and the tile placed in* B *at site* (i,j) *sticks to* A*'s tiles, i.e., the sum of the strengths of the bonds around the sides of the tile* (i,j) *in* B *with the rest of the configuration is at least* τ*. We write* $A \to_{\mathcal{T}} B$ *and denote by* $\to^*_{\mathcal{T}}$ *the transitive closure.*

The productions $\mathcal{P}$ *of a tile system* $\mathcal{T}$ *are the configurations derived from the* seed configuration*,* I_0*, which consists in the single tile* t_0 *placed at* $(0,0)$*:*

[2] One can force the orientation of the tiles simply by coding the correct orientation in the glues, so we assume here w.l.o.g. that the tiles are *oriented* (no rotation of the tiles are allowed). This simplifies considerably the design of a tile system, without affecting its generality.

$\mathcal{P} = \{A : \Gamma_0 \rightarrow_{\mathcal{T}}^{*} A\}$. *We say that a production is* final *if no other production may be derived from it. The order induced by* $\rightarrow_{\mathcal{T}}^{*}$ *from the seed configuration is called the* dynamics *of* $\mathcal{T}$.

A desirable property of a tile system is that the tiles of a production have to be assembled in a fixed known (partial) order. This property turns out to be very useful in the proof of correctness of tile systems [5]. To the classic *RC condition* [5] which states that all the tiles on a given column (resp. row) have to be placed after a fixed first tile has been placed in this column (resp. row) (we say that this tile *opens* the column, resp. row), we prefer the following weaker condition which allows more geometrical freedom (for instance, assembling concave shapes) while preserving the necessary guarantee of determinism for the design and analysis purposes.

Definition 4 (The order condition). *Given an assembling sequence* $\gamma = \langle \Gamma_0 \rightarrow_{\mathcal{T}} A_1 \rightarrow_{\mathcal{T}} A_2 \rightarrow_{\mathcal{T}} \cdots \rightarrow_{\mathcal{T}} A_t = P \rangle$ *of a production* P *and two* neighboring *sites* (i, j) *and* (i', j') *in* P*, we say that* $(i, j) \prec_{P,\gamma} (i', j')$ *if the tile over* (i, j) *is placed before the tile over* (i', j') *in the construction sequence. We say that the tile system* $\mathcal{T}$ *satisfies the* order condition *(is* ordered*, for short) if for all production* P*, the relationship* $\prec_{P,\gamma}$ *between the sites of* P *is independent of the assembling sequence* γ*; in that case, the relationship, written* $\prec_P$*, defines a partial order (see for instance [3]) called the* local order *of the sites of* P.

Clearly, for any ordered tile system, the assembling sequence of a given production P is "locally deterministic": the tiles over two neighboring sites are always placed in the same order. The only uncertainty in an ordered tile system relies in the choice of the tile to place at each site; this choice will condition which production will be obtained at the end; but, every given production P is always assembled in the same order locally. Some easy facts on ordered tile systems are:

- if a production Q extends a production P, the local order $\prec_Q$ coincides with $\prec_P$ on P's sites, and moreover, since no site can be tiled before its predecessors, P is an ideal[3] $\prec_Q$ in the order theory terminology.
- since more than τ predecessors would induce uncertainty on the precedence relationship of a site with its neighbors, every site in a production P has at most τ predecessors according to $\prec_P$ and the seed t_0 is the only tile without predecessors;
- every RC tile system [5] is ordered.

3 Time Optimality and Skeleton

3.1 Time Optimal Tile Systems

Timed dynamics. The assembling of a shape is classically modeled by a Markov chain in which each tile appears at each unoccupied site at a rate proportional to its concentration [1]. We do not study here the probability that a given shape

[3] An *ideal* I of a partial order $\prec$ is a set such that for all $u \in I$, $v \prec u \Rightarrow v \in I$ (see [3]).

is produced, which depends on the concentrations of the tiles: we rather focus on minimizing the expected construction time of each production, conditioned to the event that this production is obtained. It is known from [1] that the expected construction time of a given production P is then proportional to the *length $\ell(P)$ of the longest chain of predecessors* in $\prec_P$, where the multiplicative constant depends only on the concentrations of the tiles. Since once we condition the expectation to construct one specific production, the concentrations play the same role as GHz in computers (doubling them will reduce the construction time by 2), the meaningful parameter to measure time here is $\ell(P)$. We thus aim at *minimizing the length of the longest chain of dependencies in $\prec_P$* to obtain an efficient construction time. Therefore, we say that the construction of a final production P is *time optimal* when the length $\ell(P)$ of the longest chain in $\prec_P$ is as small as possible.

Note that, for ordered tile systems, $\ell(P)$ is also exactly the time to construct P in the *parallel dynamics*, where at every time step, we place all the tiles that can be attached to the current aggregate.

Since any production is obtained by assembling tiles one after the other starting at the seed tile, we obtain the following bound on the length of the longest chain of dependencies.

Fact 1 (Trivial lower bound). *Given a 4-connected shape $S \subset \mathbb{Z}^2$, for all tile system $\mathcal{T}$ and all production P realizing S, we have: $\ell(P) \geqslant ||S||_1$, where $||S||_1$ denotes the maximum ℓ_1-distance of a site of S to the seed tile placed at the origin.*

Definition 5 (Real time assembling). *Given a family of shapes $\mathcal{S}$, we say that a tile system realizes the family $\mathcal{S}$* in real time*, if the shapes realized by its final productions are exactly $\mathcal{S}$, and for all final production P realizing a shape $S \in \mathcal{S}$, we have: $\ell(P) = ||S||_1$.*

By Fact 1, any real time tile system is by definition time optimal. The following definition extends the notion of time to every site of a production in an ordered tile system.

Definition 6 (Rank). *We define the* rank $\rho_P(u)$ *of a site u in a production P, as the length of the longest chain leading to u in $\prec_P$. We say that a site u of P is* on time *if $\rho_P(u) = ||u||_1$. Clearly, a tile system realizes a production P in real time, if the site of highest rank in P is on time. Note that in the parallel dynamics assembling a production P, each site u is tiled at time $\rho_P(u)$ exactly.*

3.2 Skeleton of a Production: Lower Bounding the Rank

Existing self-assembling constructions are driven by main signals, the rest of the production been completed by "passive" tiles filling the gaps between these signals. In fact, this has to be the case for all ordered self-assembling constructions and we formalize this fact with the following notion:

Definition 7 (Skeleton). *Let $\mathcal{T}$ be an ordered tile system. The* x-skeleton *of a production P (resp.,* y-skeleton*) is the union of the seed site and of the sites of P which have a unique predecessor in $\prec_P$, placed on their right or left (resp., above or below). The* skeleton *of P is the union of its x- and y-skeletons.*

The x-skeleton (resp. y-) of a production P is the set of sites that *open* the x-dimension (resp. y-) during the assembling of P, i.e., which are the first in each column (resp., row) to be tiled. Indeed, in an ordered tile system, each column of a production is progressively tiled by agregation from the corresponding x-skeleton site on this column. Note that even if the tiles aggregated on a column may carry as well meaningful signals, they have all to be placed by agregation from the x-skeleton tile on this column. Naturally, the same holds respectively for rows and the y-skeleton. It follows that given an ordered tile system, one can lower bound the rank $\rho_P(u)$ of each site u in a production P by the maximum of the following two quantities:

$$\sigma_X(u) = \rho_P(a) + ||u - a||_1 \quad \text{and} \quad \sigma_Y(u) = \rho_P(b) + ||u - b||_1, \tag{1}$$

where a and b are the closest x- and y-skeleton sites on the column and on the row of u in P, respectively (i.e., where a and b are the sites which open the row and the column of u, respectively, in the assembling of P).

These two lower bounds show how the "flow of information" circulates within the shape from the skeleton during the assembling. We will see next how one can use these lower bounds to understand *where* the skeleton of a shape has to be to match the real time constraints. We illustrate this approach by obtaining two real time tile systems realizing, respectively, the $n \times n$ squares S_n and the $n \times n \times n$ cubes C_n, rooted at the origin, in optimal time.

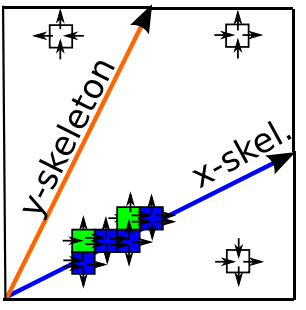
4 Assembling Squares in Real Time

4.1 A Real Time Local Order for Squares

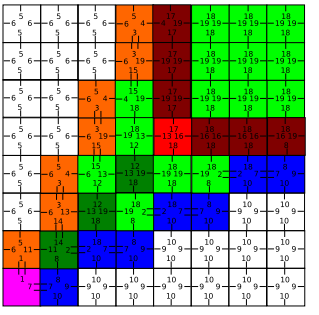
We first show how to assemble in real time the family of squares with lower left corner at the origin, $S_n = \{0, \dots, n-1\}^2$ for $n \geqslant 2$. Our approach consists in designing first a local order for the sites from the "flow of information" induced by the real time constraints within the shape; and then deduce from it an ordered tile systems matching this order.

Since $||S_{n+1}||_1 = 2n$, our goal is to obtain a local order for the square S_{n+1} such that the highest rank of the sites is (at most) $2n$. Consider an arbitrary ordered tile system $\mathcal{T}$ that realizes the square S_{n+1}. Since S_{n+1} is convex and $\mathcal{T}$ is ordered, each column (resp., row) of S_{n+1} contains exactly one site in the x-skeleton (resp., y-); otherwise, if there were two x-skeleton sites on the same column, the sites between them could be assembled from both below and above, the local order would depend on the assembling sequence and $\mathcal{T}$ would not be ordered. Let us thus denote $a_i = (i, y_i)$ and $b_j = (j, x_j)$ the sequences of sites in the x- and y-skeletons respectively. If $y_n > n/2$ (for instance in [2,4], $y_n = n-1$), then by the lower bound (1), the rank of lower right corner $(n, 0)$ would be at least $\rho(a_n) + ||(n, 0) - (n, a_n)||_1 \geqslant ||a_n||_1 + y_n = n + 2y_n > 2n$. It follows that the x-skeleton (resp. y-) of a real time ordered tile system for S_{n+1} cannot get above the site $(n, \lfloor n/2 \rfloor)$ (resp. to the left of the site $(\lfloor n/2 \rfloor, n)$).

We thus consider the following x- and y-skeletons: $a_i = (i, \lfloor i/2 \rfloor)$ and $b_j = (\lfloor j/2 \rfloor, j)$ as displayed in Figure 1(a). The lower bound (1) tells us that the



(a) The local order induced by the x- and y-skeletons within the square. In each of the three regions delimited by the x- and y-skeletons, the local order $\prec$ follows the directions of the arrows. For example, in the central region, the predecessors of (i, j) are $(i-1, j)$ and $(i, j-1)$ and its successors are $(i+1, j)$ and $(i, j+1)$. Each site $(i, \lfloor \frac{i}{2} \rfloor)$ (resp. $(\lfloor \frac{j}{2} \rfloor, j)$) of x-skeleton (resp. y-) has a unique predecessor, $(i-1, \lfloor \frac{i}{2} \rfloor)$ (resp., $(\lfloor \frac{j}{2} \rfloor, j-1)$).



(b) The tile system for even squares. The numbers on the sides of each tile are the IDs of the glues, and the number of ticks accross the boundary of the tiles stands for the strength of the glue. The temperature is 2. Strength 2 glues lie on the lines $j = 2i$ and $j = i/2$. The green tiles and the red tile on the diagonal represent the GO and STOP tiles which propagate (on time) up to the blue and orange signals carrying the x- and y-skeletons respectively either to allow them to continue their progression for the next two steps (GO) or to stop them (STOP).

Fig. 1. The skeleton and the tile systems assembling squares in real time

rank of each site $u = (i, j)$ in any local order based on this skeleton is at least: $\sigma(u) = \max(||a_i||_1 + ||u - a_i||_1, ||b_j||_1 + ||u - b_j||_1)$. More precisely,

- $\sigma(u) = i + j = ||u||_1$ if u lies between the x- and y-skeletons, i.e., if $j \geqslant \lfloor i/2 \rfloor$ and $i \geqslant \lfloor j/2 \rfloor$;
- $\sigma(u) = i + 2\lfloor i/2 \rfloor - j > ||u||_1$ if u lies below the x-skeleton, i.e., if $j < \lfloor i/2 \rfloor$;
- symmetrically, $\sigma(u) = j + 2\lfloor j/2 \rfloor - i > ||u||_1$ if u lies above the y-skeleton, i.e., if $i < \lfloor j/2 \rfloor$.

Clearly, $\sigma(u) \leqslant 2n$, for all u. Moreover, one can easily verify that σ is the rank of the following local order where the predecessors of each site $u = (i, j)$ are defined as (see Figure 1(a)):

- its east neighbor $(i-1, j)$, if $u = a_i$ belongs to the x-skeleton;
- its south neighbor $(i, j-1)$, if $u = b_j$ belongs to the y-skeleton;
- its east and north neighbors $(i-1, j)$ and $(i, j+1)$, if u lies below the x-skeleton;
- its east and south neighbors $(i-1, j)$ and $(i, j-1)$, if u lies between the x- and y-skeletons;
- its west and south neighbors $(i+1, j)$ and $(i, j-1)$, if u lies above the y-skeleton.

We are now left with implementing this real time order with an ordered tile system.

4.2 A Real Time Tile System for Squares

The main issue in implementing the real time local order above into a tile system is the *synchronization* between the two *distant* signals carrying the x- and y-skeletons. It turns out that the sites lying between the x- and y-skeletons *are all on time* (their ranks equal their ℓ_1-norms). We may then use this zone to transport just-in-time the synchronization signals. We proceed as shown in Figure 1(b). The progress of the signals carrying the x- and y-skeletons is controlled by the tile placed on the diagonal: if the GO tile is placed on the diagonal, then the GO signal propagates along the row and column and allows just-in-time the x- and y-skeletons to jump by one upward and to the right respectively and to progress for two new steps; if the STOP tile is placed on the diagonal, then the STOP signal propagates along the row and column and the progression of the x- and y-skeleton stops just-in-time forcing the production to be completed into a square. In order to obtain even square as well, we have two kinds of STOP tiles: EVEN and ODD; while EVEN stops where the x- and y-skeleton stoped, ODD allows one single extra step to extend the length of the sides by one.

The complete description of the tiles and their proof of correctness cannot be given in details due to space constraints but can be read easily in Figure 1(b). We thus conclude:

Theorem 2. *There exists an ordered self-assembling system with 24 tiles (with rotations allowed) whose final productions at temperature* 2 *realize exactly the set of all squares S_n in optimal time.*

Note that the tile system proposed in Figure 1(b) is not minimal and one can significantly reduce the number of tiles by shifting the x- and y-skeletons while preserving time optimality. The classic counter-based technique in the literature [4] which allows to control precisely the size of the assembly can also be used with this construction. It consists in adding $O(\log n / \log\log n)$ tiles to force the production of an $n \times n$ square. Instead of choosing non-deterministically the position of the STOP tile on the diagonal, one can use the on time region between the x- and y-skeletons to simulate a counter (or any other computation) and have the (ODD or EVEN) STOP tile to be put exactly at $(\lfloor n/2 \rfloor, \lfloor n/2 \rfloor)$. Moreover, using the same technique, and by launching simultaneously four real time constructions with a counter in the directions SE, NE, NW, and SW, one can also obtain a real time assembly of a square rooted on any of its sites.

5 Assembling Cubes in Real Time

5.1 The Skeleton

We will now show how our framework allows to obtain readably a tri-dimensional tile system which assemble in optimal time the family of cubes rooted at the origin, $C_n = \{0, \ldots, n-1\}^3$ for $n \geqslant 2$. We extend canonically the definitions in Sections 2 and 3 from 2D to 3D.

As observed in Section 3.2, the assembling of a convex shape by an ordered tile systems is necessarily driven by three skeletons consisting in the sequence

of sites opening each dimension. Taking inspiration from the 2D tile system designed above, we consider the following x-, y- and z-skeletons heading from the origin to the centers of the three opposite faces of the cube: $a_i = (i, \lfloor i/2 \rfloor, \lfloor i/2 \rfloor)$, $b_j = (\lfloor j/2 \rfloor, j, \lfloor j/2 \rfloor)$ and $c_k = (\lfloor k/2 \rfloor, \lfloor k/2 \rfloor, k)$.

As before, we define the *rank function σ induced by the skeleton* as follows. For each site $u = (i, j, k)$, let $\sigma_X(u) = ||a_i||_1 + ||u - a_i||_1$, $\sigma_Y(u) = ||b_j||_1 + ||u - b_j||_1$, and $\sigma_Z(u) = ||c_k||_1 + ||u - c_k||_1$. The rank of u induced by the skeleton is then: $\sigma(u) = \max(\sigma_X(u), \sigma_Y(u), \sigma_Z(u))$. Since tiles are assembled from the skeleton by definition of the skeleton, $\sigma(u)$ is clearly a lower bound on the earliest time a tile may be placed at u in the parallel dynamics.

Given two neighboring sites u and v in $\mathbb{N}^3$, we say that u is a *predecessor* of v (and v is a *successor* of u) if $\sigma(u) < \sigma(v)$. We define the *order induced by the skeleton* $\prec$ as the transitive closure of the predecessor relation: $u \prec v$ if there exists a finite sequence $(u_0 = u, u_1, \ldots, u_q = v)$ such that u_{i-1} is a predecessor of u_i for all $i = 1, \ldots, q$. Our goal is to prove that there exists a tile system whose local order is exactly $\prec$. We need first to understand the structure of this order, and thus first, the local variations of σ.

5.2 On the Rank Function Induced by the Skeleton

We denote by (e_x, e_y, e_z) the canonical basis of $\mathbb{N}^3$. A careful case study yields the following key lemma describing the variations of σ along the z-axis.

Lemma 1 (Variations of σ along the z-axis). *For all site $u = (i, j, k)$ in $\mathbb{N}^3$, if $\max(i, j) \leqslant 2k + 1$, then $\sigma(u) < \sigma(u + e_z)$; and if $\max(i, j) \geqslant 2k$, $\sigma(u) < \sigma(u - e_z)$ (provided that $k \neq 0$).*

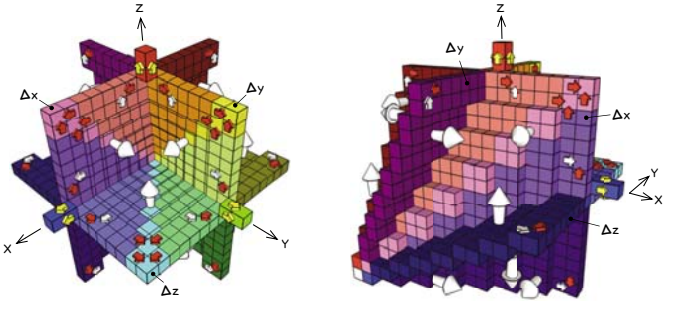


Fig. 2. The local variations of σ and the corresponding local order in $\mathbb{N}^3$ from two points of views: from (n, n, n) on the left, and from $(n, 0, n)$ on the right. Only the sites with at most 2 predecessors are represented. Arrows points in σ's growth direction along the axes. The white arrows stand for simple predecessors; the red arrow for double predecessors; and the yellow arrows for triple predecessors. One can clearly see the partition in seven regions filled with sites with 3 simple predecessors, separated by discrete planes made of sites with two predecessors including at least one double predecessor, intersecting each other on the x-, y- and z-skeletons (in pure blue, green and red respectively) made of sites with one single triple predecessor.

We thus define the following "pyramidal" discrete surfaces: $\Delta_X = \{(\lfloor \frac{\max(j,k)}{2} \rfloor, j, k) : j, k \geqslant 0\}$, $\Delta_Y = \{(i, \lfloor \frac{\max(i,k)}{2} \rfloor, k) : i, k \geqslant 0\}$ and $\Delta_Z = \{(i, j, \lfloor \frac{\max(i,j)}{2} \rfloor) : i, j \geqslant 0\}$. One can reformulate Lemma 1 as follows: σ is an increasing function of z above Δ_Z, and decreasing function of z below Δ_Z.

Figure 2 sums up this result graphically. The discrete surfaces Δ_X, Δ_Y and Δ_Z are respectively painted in blue and pink colors, green and orange colors, and blue and green colors (the meaning of the colors will be explained in the following sections). The growth directions of σ along each axis are represented by the large white arrows within each of the seven regions partitioned by these three surfaces and by small arrows within each surface.

Two important facts follows. First, σ defines a "proper" local order: σ never takes the same value on two neighboring sites and thus, for every pair of neighboring sites, one is always the predecessor one of the other. Second, the highest ranked site in each cube C_{n+1} is $3n = ||C_{n+1}||_1$, which is attained at the corner (n, n, n). It follows that the order induced by the skeleton matches the real time constraints as expected.

5.3 Classification of the Sites According to Their Relative Positions with Their Predecessors

Thanks to Lemma 1 and Figure 2, we observe the following properties of the order $\prec$ induced by σ:

Fact 3

1. *each site has at most three predecessors among its neighbors;*
2. *the origin is the unique site with no predecessor;*
3. *a site u has a unique predecessor if and only if u belongs to the skeleton; this predecessor is $u - e_z$ if u lies in the z-skeleton.*
4. *every site $u = (i, j, k)$ with exactly two predecessors has two opposite neighbors which are its successors; and these opposite successors are located in $u \pm e_z$ if and only if u belongs to Δ_Z.*

Multiplicity. We define the *multiplicity* of the predecessors of the sites as follows (the missing cases are obtained by symmetry). The multiplicity will be used later on to define the strength of the bond between the tiles attached on these sites.

- The multiplicity of the predecessors of any site with three predecesors is 1;
- If $u \neq 0$ belongs to the skeleton, the multiplicity of its unique predecessor is 3;
- If $u = (i, j, k)$ belongs to Δ_Z: if $\lfloor \frac{i}{2} \rfloor \geqslant \lfloor \frac{j}{2} \rfloor$ (u belongs to the bluish part of Δ_Z), the multiplicity of its predecessor along the y-axis is 2; if $\lfloor \frac{j}{2} \rfloor \geqslant \lfloor \frac{i}{2} \rfloor$ (u belongs to the greenish part of Δ_Z), the multiplicity of its predecessor along the x-axis is 2; if both cases apply (u belongs to the diagonal sites in cyan of Δ_Z in Figure 2), both predecessors of u get a multiplicity of 2, otherwise the other predecessor gets a multiplicity 1.

In Figure 2, the multiplicities of the predecessors with their successors are represented by white, red or yellow arrows whether the multiplicity is 1, 2, or

3 respectively. The following fact will ensure that the tile system we will derive from the order in Section 5.5, assembles the productions in the prescribed order at temperature 3.

Fact 4. *For all site $u \neq 0$, the sum of the multiplicities of its predecessors is at least* 3 *and the sum of the multiplicities of any strict subset of its predecessors is at most* 2*.*

5.4 Deducing the Successors from the Predecessors

A key property needed to obtain the tile system is that the information transmitted by the glues on the faces a site u shares with its predecessors, is enough, independently of the position of u, to completely determine the correct glues of the free faces of u. In terms of order, this property consists for now in checking that one can deduce the correct types of the successors of u from the types of its predecessors only, and a small number of criteria that can be transmitted from site to site through the glues. Consider a site $u = (i, j, k)$.

- If u has a triple predecessor, w.l.o.g., $u - e_x$, then $u = a_i$ belongs to the x-skeleton, u has four double successors $u \pm e_y$ and $u \pm e_z$, and: if i is even, then $u + e_x = a_{i+1}$ is a triple successor, and a simple successor otherwise.
- If u has two double predecessors, w.l.o.g., $u - e_x$ and $u - e_y$ (u belongs to the cyan diagonal part of Δ_Z), then $u \pm e_z$ are both simple successors, and: if i (resp. j) is even, $u + e_x$ (resp. $u + e_y$) is a double successor, and a simple successor otherwise.
- If u has two predecessors, a double and a simple, w.l.o.g., its double and simple predecessors are respectively $u - e_x$ and $u - e_y$ and u belongs to the lighter blue part of Δ_Z; then $u \pm e_z$ are simple successors and $u + e_x$ and $u + e_y$ are respectively double and simple successors as well.
- If u has three simple predecessors, then either u has three simple successors or it has one triple successor, w.l.o.g. a_{i+1}, and two simple successors; the second alternative arising only if $u = a_i + e_y + e_z$, i.e., if u is a successor of $a_i + e_y$ which is itself a successor of a_i.

Each of these cases corresponds to a different color in Figure 2.

5.5 A Real Time Tile Systems for Cubes

It follows from the study above that one can correctly guess the types of the successors of any site from the only knowledge of: 1) the multiplicity and relative position of its predecessors, 2) the parity of the corresponding coordinates, and 3) possibly the immediate proximity to a part of the skeleton. We define thus the glues as a combination of three labels as follows:

- each face holds a label chosen among $++$, $+$, $+$, $-$, $--$, $---$ according to the multiplicity of the predecessor/successor relation with the corresponding neighboring site and according to the orientation of this relation with respect to the corresponding axis.

– each pair of opposite faces holds a 0 or a 1 to propagate the parity of each coordinate.
– each even site a_{2i} of the x-skeleton propagates to $a_{2i} + e_y$ which in turn relays to $a_{2i} + e_y + e_z$ the fact that the later has to allow a triple successor on its x^+-face to attach the next element a_{2i+1} of the x-skeleton (the same holds for the y- and z-skeletons).

According to the previous subsections, we now have a set of tiles that agregate themselves from the seed tile according to the desired local order $\prec$ induced by σ. We are now left with the problem of the synchronization of the stop of the three distant signals carrying the x-, y- and z-skeletons. As for the squares in Section 4.2, we can use for that purpose the central region between the three surfaces Δ_X, Δ_Y and Δ_Z. Indeed an easy calculation demonstrates that this region consists of the union of all the $q \times q \times q$ cubes $\{q, \ldots, 2q\}^3$ and that its sites are all on time (recall definition 6) and can thus be used to propagate from the main diagonal a GO, STOP-EVEN, STOP-ODD signal to the three part of the skeleton simultaneously and without delay as we did before in the 2D case. We can thus conclude:

Theorem 5. *There exists an ordered self-assembling system with a finite number of tiles whose final productions at temperature* 3 *realize exactly the set of all cubes* C_n *in optimal time.*

References

1. Adleman, L.M., Cheng, Q., Goel, A., Huang, M.-D.A.: Running time and program size for self-assembled squares. In: STOC 2001: Proc. of the 33rd ACM symposium on Theory of computing, pp. 740–748 (2001)
2. Becker, F., Rapaport, I., Rémila, E.: Self-assemblying classes of shapes with a minimum number of tiles, and in optimal time. In: LNCS Proc. of Found. of Software Tech. and Theo. Comp. Sci., pp. 45–56 (2006)
3. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (2002)
4. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: STOC 2000: Proceedings of the 32nd symp. on Theory of computing, pp. 459–468 (2000)
5. Rothemund, P.W.K.: Theory and Experiments in Algorithmic Self-Assembly. PhD thesis, University of Southern California (2001)
6. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 344–354. Springer, Heidelberg (2005)
7. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, Caltech (1998)
8. Winfree, E.: Simulations of computing by self-assembly. In: Proceedings of the Fourth DIMACS Meeting on DNA Based Computing, pp. 213–242 (1998)

Self-assembly of Discrete Self-similar Fractals*
(Extended Abstract)

Matthew J. Patitz and Scott M. Summers**

Department of Computer Science
Iowa State University
Ames, IA 50011, USA
{mpatitz,summers}@cs.iastate.edu

Abstract. In this paper, we search for *absolute* limitations of the Tile Assembly Model (TAM), along with techniques to work around such limitations. Specifically, we investigate the self-assembly of fractal shapes in the TAM. We prove that no self-similar fractal fully weakly self-assembles at temperature 1, and that certain kinds of self-similar fractals do not strictly self-assemble at any temperature. Additionally, we extend the fiber construction from Lathrop et. al. (2007) to show that any self-similar fractal belonging to a particular class of "nice" self-similar fractals has a fibered version that strictly self-assembles in the TAM.

1 Introduction

Self-assembly is a bottom-up process by which (usually a small number of) fundamental components automatically coalesce to form a target structure. In 1998, Winfree [14] introduced the (abstract) Tile Assembly Model (TAM) - an extension of Wang tiling [12,13], and a mathematical model of the DNA self-assembly pioneered by Seeman et. al. [10]. In the TAM, the fundamental components are un-rotatable, but translatable "tile types" whose sides are labeled with glue "colors" and "strengths." Two tiles that are placed next to each other *interact* if the glue colors on their abutting sides match, and they *bind* if the strength on their abutting sides matches, and is at least a certain "temperature." Rothemund and Winfree [9,8] later refined the model, and Lathrop et. al. [6] gave a treatment of the TAM in which equal status is bestowed upon the self-assembly of infinite and finite structures. There are also several generalizations [1,7,4] of the TAM.

Despite its deliberate over-simplification, the TAM is a computationally and geometrically expressive model. For instance, Winfree [14] proved that the TAM is computationally universal, and thus can be directed algorithmically. Winfree [14] also exhibited a seven-tile-type self-assembly system, directed by a clever XOR-like algorithm, that "paints" a picture of a well-known shape, the discrete

* This research was supported in part by National Science Foundation Grants 0652569 and 0728806.

** This author's research was supported in part by NSF-IGERT Training Project in Computational Molecular Biology Grant number DGE-0504304.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 156–167, 2009.

Sierpinski triangle **S**, onto the first quadrant. Note that the underlying *shapes* of each of the previous results are infinite canvases that cover the first quadrant, onto which computationally interesting shapes are painted (i.e., full weak self-assembly). Moreover, Lathrop et. al [5] recently gave a new characterization of the computably enumerable sets in terms of weak self-assembly using a "ray construction." It is natural to ask the question: How expressive is the TAM with respect to the self-assembly of a particular, possibly infinite shape, and nothing else (i.e., strict self-assembly)?

In the case of strict self-assembly of finite shapes, the TAM certainly remains an interesting model, so long as the size (tile complexity) of the assembly system is required to be "small" relative to the shape that it ultimately produces. For instance, Rothemund and Winfree [9] proved that there are small tile sets in which large squares self-assemble. Moreover, Soloveichik and Winfree [11] established the remarkable fact that, if one is not concerned with the scale of an "algorithmically describable" finite shape, then there is always a small tile set in which the shape self-assembles. Note that if the tile complexity of an assembly system is unbounded, then every finite shape trivially (but perhaps not feasibly) self-assembles.

When the tile complexity of an assembly system is unbounded (yet finite), only infinite shapes are of interest. In the case of strict self-assembly of infinite shapes, the power of the TAM has only recently been investigated. Lathrop et al. [6] established that self-similar tree shapes do not strictly self-assemble in the TAM given any finite number of tile types. A "fiber construction" is also given in [6], which strictly self-assembles a non-trivial fractal structure.

In this paper, we search for (1) *absolute* limitations of the TAM, with respect to the strict self-assembly of shapes, and (2) techniques that allow one to "work around" such limitations. Specifically, we investigate the strict self-assembly of fractal shapes in the TAM. We prove three main results: two negative and one positive. Our first negative (i.e., impossibility) result says that no self-similar fractal fully weakly self-assembles in the TAM at temperature 1. In our second impossibility result, we exhibit a class of discrete self-similar fractals, to which the standard discrete Sierpinski triangle belongs, that do not strictly self-assemble in the TAM (at *any* temperature). Finally, in our positive result, we use simple modified counters to extend the fiber construction from Lathrop et al. [6] to a particular class of discrete self-similar fractals.

2 Preliminaries

2.1 The Tile Assembly Model

We work in the 2-dimensional discrete Euclidean space $\mathbb{Z}^2$. We write U_2 for the set of all *unit vectors*, i.e., vectors of length 1 in $\mathbb{Z}^2$. We write $[X]^2$ for the set of all 2-element subsets of a set X. All *graphs* here are undirected graphs, i.e., ordered pairs $G = (V, E)$, where V is the set of *vertices* and $E \subseteq [V]^2$ is the set of *edges*.

A *binding function* on a graph $G = (V, E)$ is a function $\beta : E \to \mathbb{N}$. (Intuitively, if $\{u, v\} \in E$, then $\beta(\{u, v\})$ is the strength with which u is bound to v by $\{u, v\}$

according to β. If β is a binding function on a graph $G = (V, E)$ and $C = (C_0, C_1)$ is a cut of G, then the *binding strength* of β on C is

$$\beta_C = \{\beta(e) \,|e \in E, e \cap C_0 \neq \emptyset, \text{ and } e \cap C_1 \neq \emptyset\}.$$

The *binding strength* of β on the graph G is then

$$\beta(G) = \min\{\beta_C \,|C \text{ is a cut of } G\}.$$

A *binding graph* is an ordered triple $G = (V, E, \beta)$, where (V, E) is a graph and β is a binding function on (V, E). If $\tau \in \mathbb{N}$, then a binding graph $G = (V, E, \beta)$ is τ*-stable* if $\beta(V, E) \geq \tau$.

A *grid graph* is a graph $G = (V, E)$ in which $V \subseteq \mathbb{Z}^2$ and every edge $\{\boldsymbol{a}, \boldsymbol{b}\} \in E$ has the property that $\boldsymbol{a} - \boldsymbol{b} \in U_2$. The *full grid graph* on a set $V \subseteq \mathbb{Z}^2$ is the graph $G_V^{\#} = (V, E)$ in which E contains *every* $\{\boldsymbol{a}, \boldsymbol{b}\} \in [V]^2$ such that $\boldsymbol{a} - \boldsymbol{b} \in U_2$.

We now give a brief sketch of the Tile Assembly Model. See [14,9,8,6] for other developments of the model.

Intuitively, a tile type t is a unit square that can be translated, but not rotated, having a well-defined "side $\boldsymbol{u}$" for each $\boldsymbol{u} \in U_2$. Each side $\boldsymbol{u}$ of t has a "glue" of "color" $\text{col}_t(\boldsymbol{u})$ - a string over some fixed alphabet Σ - and "strength" $\text{str}_t(\boldsymbol{u})$ - a natural number - specified by its type t. Two tiles t and t' that are placed at the points $\boldsymbol{a}$ and $\boldsymbol{a} + \boldsymbol{u}$ respectively, *bind* with *strength* $\text{str}_t(\boldsymbol{u})$ if and only if $(\text{col}_t(\boldsymbol{u}), \text{str}_t(\boldsymbol{u})) = (\text{col}_{t'}(-\boldsymbol{u}), \text{str}_{t'}(-\boldsymbol{u}))$.

Given a set T of tile types, an *assembly* is a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$. An assembly is *stable* if it cannot be broken up into smaller assemblies without breaking bonds of total strength at least τ. The *binding graph of* an assembly α is the binding graph $G_\alpha = (V, E, \beta)$, where (V, E) is the grid graph given by $V = \text{dom } \alpha$, and $\{\boldsymbol{m}, \boldsymbol{n}\} \in E$ if and only if (1) $\boldsymbol{m} - \boldsymbol{n} \in U_2$, (2) $\text{col}_{\alpha(\boldsymbol{m})}(\boldsymbol{n} - \boldsymbol{m}) = \text{col}_{\alpha(\boldsymbol{n})}(\boldsymbol{m} - \boldsymbol{n})$, and (3) $\text{str}_{\alpha(\boldsymbol{m})}(\boldsymbol{n} - \boldsymbol{m}) > 0$. The binding function $\beta : E \to \mathbb{Z}^+$ is given by $\beta(\{\boldsymbol{m}, \boldsymbol{n}\}) = \text{str}_{\alpha(\boldsymbol{m})}(\boldsymbol{n} - \boldsymbol{m})$ for all $\{\boldsymbol{m}, \boldsymbol{n}\} \in E$.

Self-assembly begins with a *seed assembly* σ and proceeds asynchronously and nondeterministically, with tiles adsorbing one at a time to the existing assembly in any manner that preserves stability at all times. A *tile assembly system* (*TAS*) is an ordered triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, σ is a seed assembly with finite domain, and $\tau = 2$ is the temperature. An assembly α is *terminal*, and we write $\alpha \in \mathcal{A}_\square[\mathcal{T}]$, if no tile can be stably added to it. A TAS $\mathcal{T}$ is *directed*, or *produces a unique assembly*, if it has exactly one terminal assembly.

A set $X \subseteq \mathbb{Z}^2$ *weakly self-assembles* if there exists a TAS $\mathcal{T} = (T, \sigma, \tau)$ and a set $B \subseteq T$ such that $\alpha^{-1}(B) = X$ holds for every terminal assembly α. A set X *strictly self-assembles* if there is a TAS $\mathcal{T}$ for which every terminal assembly has domain X.

An *assembly sequence* in a TAS $\mathcal{T} = (T, \sigma, \tau)$ is an infinite sequence $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \ldots)$ of assemblies in which $\alpha_0 = \sigma$ and each α_{i+1} is obtained from α_i by the "τ-stable" addition of a single tile. To prove that a particular TAS $\mathcal{T} = (T, \sigma, \tau)$ is directed, it suffices to exhibit a *locally deterministic* [11] assembly sequence.

2.2 Discrete Self-similar Fractals

In this subsection we introduce discrete self-similar fractals, and zeta-dimension [3].

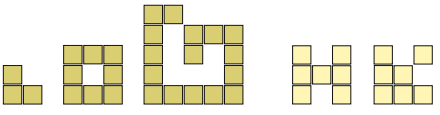
Definition 1. *Let $1 < c \in \mathbb{N}$, and $X \subsetneq \mathbb{N}^2$ (we do not consider $\mathbb{N}^2$ to be a self-similar fractal). We say that X is a c-discrete self-similar fractal, if there is a set $\{(i,i) \mid i \in \{0,\dots c-1\}\} \neq V \subseteq \{0,\dots,c-1\} \times \{0,\dots,c-1\}$ such that*

$$X = \bigcup_{i=0}^{\infty} X_i,$$

where X_i is the i^{th} stage satisfying $X_0 = \{(0,0)\}$, and $X_{i+1} = X_i \cup \left(X_i + c^i V\right)$. In this case, we say that V generates X. X is a discrete self-similar fractal if it is a c-discrete self-similar fractal for some $c \in \mathbb{N}$.

In this paper, we are concerned with the following class of self-similar fractals.

Definition 2. *A nice discrete self-similar fractal is a discrete self-similar fractal such that $(\{0,\dots,c-1\} \times \{0\}) \cup (\{0\} \times \{0,\dots,c-1\}) \subseteq V$, and $G_V^{\#}$ is connected.*



(a) Nice (b) Non-nice

Fig. 1. The first stages of discrete self-similar fractals. The fractals in (a) are nice, whereas (b) shows two non-nice fractals.

The most commonly used dimension for discrete fractals is zeta-dimension, which we use in this paper.

Definition 3. *(Doty et. al. [3]) For each set $A \subseteq \mathbb{Z}^2$, the zeta-dimension of A is*

$$Dim_\zeta(A) = \limsup_{n\to\infty} \frac{\log |A_{\leq n}|}{\log n},$$

where $A_{\leq n} = \{(k,l) \in A \mid |k| + |l| \leq n\}$.

It is clear that $0 \leq \mathrm{Dim}_\zeta(A) \leq 2$ for all $A \subseteq \mathbb{Z}^2$.

3 Impossibility Results

In this section, we explore the theoretical limitations of the Tile Assembly Model with respect to the self-assembly of fractal shapes. First, we establish that no discrete self-similar fractal weakly self-assembles at temperature $\tau = 1$ in a locally deterministic tile assembly system. Second, we exhibit a class $\mathcal{C}$ of discrete self-similar fractals, and prove that if $F \in \mathcal{C}$, then F does not strictly self-assemble in the TAM.

We are specifically interested in the method of local determinism by Soloveichik and Winfree.

Definition 4 (Soloveichik and Winfree [11]). *A TAS $\mathcal{T} = (T, \sigma, \tau)$ is locally deterministic if there exists a locally deterministic τ-T-assembly sequence $\boldsymbol{\alpha}$.*

Lemma 1 (Soloveichik and Winfree [11]). *Let $\mathcal{T} = (T, \sigma, \tau)$ be a TAS. If there exists a locally deterministic τ-T-assembly sequence, then every τ-T-assembly sequence is locally deterministic.*

The reader is encouraged to consult [11] for a detailed discussion of local determinism. To prove our first main result, we will need the following technical result.

Lemma 2. *Let $\mathcal{T} = (T, \sigma, \tau)$ be a TAS. If $\mathcal{T}$ is locally deterministic and $\alpha \in \mathcal{A}_{\square}[\mathcal{T}]$, then the binding graph G_α is a tree.*

Proof. Suppose that the binding graph G_α is not a tree. Then there is a cycle C in G_α. Let $\boldsymbol{v} \in C$, $t = \alpha(\boldsymbol{v})$ and $\boldsymbol{\alpha}$ be an assembly sequence satisfying $\alpha = \text{dom res}(\boldsymbol{\alpha})$. There must be two simple paths π_1 and π_2 in G_α from $\mathbf{0}$ to $\boldsymbol{v}$ such that $\pi_1 \neq \pi_2$. Since $\tau = 1$, there exists an assembly sequence $\boldsymbol{\alpha}'$ where $\boldsymbol{\alpha}'$ first assembles $\pi_1 \cap \pi_2$ (if such an intersection exists), then assembles $(\pi_1 - \pi_2) - \{\boldsymbol{v}\}$, and finally assembles $(\pi_2 - \pi_1) - \{\boldsymbol{v}\}$. We can use $\boldsymbol{\alpha}'$ to define the assembly sequence $\boldsymbol{\alpha}''$ that does what $\boldsymbol{\alpha}'$ does but places the tile type t at position $\boldsymbol{v}$ and then does whatever $\boldsymbol{\alpha}$ does for all $\boldsymbol{m} \notin \text{dom res}(\boldsymbol{\alpha}')$. But since $\boldsymbol{v} \in C$, the tile that $\boldsymbol{\alpha}'$ places at position $\boldsymbol{v}$ has two input sides and binds with strength 2. Thus, the assembly sequence $\boldsymbol{\alpha}'$ is not locally deterministic, and it follows by Lemma 1 that $\mathcal{T}$ is not locally deterministic.

We use the above lemma to prove the following result.

Theorem 1. *If $F \subsetneq \mathbb{N}^2$ is a discrete self-similar fractal, and F weakly self-assembles in the locally deterministic TAS $\mathcal{T}_F = (T, \sigma, \tau)$, where σ consists of a single tile placed at the origin, then $\tau > 1$.*

We omit the proof of this theorem from this version of the paper due to space constraints.

We make the assumption the that $\mathcal{T}_F$ is locally deterministic because we believe that the set of all locally deterministic tile assembly systems is representative of the set of all directed tile assembly systems at temperature 1.

We now shift our attention to the self-assembly of fractal structures at temperature 2.

Definition 5 (Lathrop et al. [6]). *Let $G = (V, E)$ be a graph, and let $D \subseteq V$. For each $r \in V$, the D-r-rooted subgraph of G is the graph $G_{D,r} = (V_{D,r}, E_{D,r})$, where*

$$V_{D,r} = \{v \in V \mid \textit{every simple path from } v \textit{ to } D \textit{ in } G \textit{ goes through } r\}$$

and $E_{D,r} = E \cap [V_{D,r}]^2$. B is a D-subgraph of G if it is a D-r-rooted subgraph of G for some $r \in V$.

Next, we exhibit a class $\mathcal{C}$ of (non-tree) "pinch-point" discrete self-similar fractals that do not strictly self-assemble. Before we do so, we establish the following lower bound.

Lemma 3. *If $X \subseteq \mathbb{Z}^2$ strictly self-assembles in the TAS $\mathcal{T} = (T, \sigma, \tau)$, where σ consists of a single tile placed at the origin, then*

$$|T| \geq \left| \left\{ B \;\middle|\; B \text{ is a unique } \operatorname{dom}\sigma\text{-subgraph of } G_X^{\#} \right\} \right|.$$

Lemma 3 is not as tight as possible, but it applies to a general class of fractals. Our second impossibility result is the following.

Theorem 2. *If $X \subsetneq \mathbb{N}^2$ is a discrete self-similar fractal satisfying (1) $\{(0,0), (0,c-1), (c-1,0)\} \subseteq V$, (2) $V \cap (\{1,\ldots c-1\} \times \{c-1\}) = \emptyset$, (3) $V \cap (\{c-1\} \times \{1,\ldots,c-1\}) = \emptyset$, and (4) $G_V^{\#}$ is connected, then X does not strictly self-assemble in the Tile Assembly Model.*

Corollary 1 (Lathrop, et al. [6]). *The standard discrete Sierpinski triangle* **S** *does not strictly self-assemble in the Tile Assembly Model.*

4 Every Nice Self-similar Fractal Has a Fibered Version

In this section, given a nice c-discrete self-similar fractal $X \subsetneq \mathbb{N}^2$ (generated by V), we define its fibered counterpart $\mathbf{X}$. Intuitively, $\mathbf{X}$ is nearly identical to X, but each successive stage of $\mathbf{X}$ is slightly thicker than the equivalent stage of X (see Figure 2 for an example). Our objective is to define sets $F_0, F_1, \ldots \subseteq \mathbb{Z}^2$, sets $T_0, T_1, \ldots \subseteq \mathbb{Z}^2$, and functions $l, f, t : \mathbb{N} \to \mathbb{N}$ with the following meanings.

1. T_i is the i^{th} stage of our construction of the fibered version of X.
2. F_i is the *fiber* associated with T_i. It is the smallest set whose union with T_i has a vertical left edge and a horizontal bottom edge, together with one additional layer added to these two now-straight edges.
3. $l(i)$ is the length (number of tiles in) the left (or bottom) edge of $T_i \cup F_i$.
4. $f(i) = |F_i|$.
5. $t(i) = |T_i|$.

These five entities are defined recursively by the equations

$$\begin{aligned}
&T_0 = X_2 \text{ (the third stage of } X),\\
&F_0 = \left(\{-1\} \times \left\{-1,\ldots,c^2\right\}\right) \cup \left(\left\{-1,\ldots,c^2\right\} \times \{-1\}\right),\\
&l(0) = c^2+1,\ f(0) = 2c^2+1,\ t(0) = (|V|+1)^2,\\
&T_{i+1} = T_i \cup \left((T_i \cup F_i) + l(i)V\right),\\
&F_{i+1} = F_i \cup (\{-i-2\} \times \{-i-2, -i-1, \cdots, l(i+1)-i-3\})\\
&\cup (\{-i-2, -i-1, \cdots, l(i+1)-i-3\} \times \{-i-2\}),\\
&l(i+1) = c \cdot l(i) + 1,\\
&f(i+1) = f(i) + c \cdot l(i+1) - 1,\\
&t(i+1) = |V| t(i) + f(i).
\end{aligned}$$

Finally, we let

$$\mathbf{X} = \bigcup_{i=0}^{\infty} T_i.$$

Note that the set $T_i \cup F_i$ is the union of an "outer framework," with an "internal structure." One can view the outer framework of $T_i \cup F_i$ as the union of a square S_i (of size $i+2$), a rectangle X_i (of height $i+2$ and width $l(i)-(i+2)$), and a rectangle Y_i (of width $i+2$ and height $l(i)-(i+2)$). Moreover, one can show that the internal structure of $T_i \cup F_i$ is simply the union of (appropriately-translated copies) of smaller and smaller X_i and Y_i-rectangles.

We have the following "similarity" between X and $\mathbf{X}$.

Lemma 4. *If $X \subsetneq \mathbb{N}^2$ is a nice self-similar fractal, then $Dim_{\zeta}(X) = Dim_{\zeta}(\mathbf{X})$.*

In the next section we sketch a proof that the fibered version of every nice self-similar fractal strictly self-assembles.

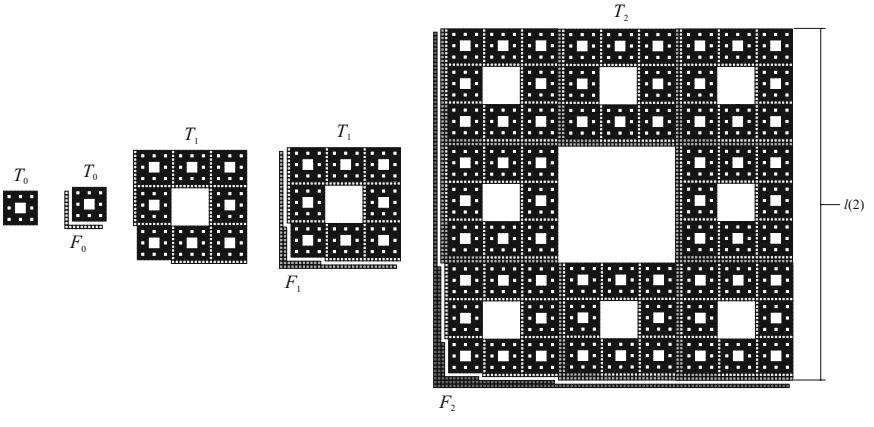


Fig. 2. Construction of the fibered Sierpinski carpet. The progressively darker bands of grey tiles represent (possibly translated copies of) F_0, F_1, and F_2, respectively.

5 Sketch of Main Construction

Our second main theorem says that the fibered version of every nice self-similar fractal strictly self-assembles in the Tile Assembly Model (regardless of whether the latter strictly self-assembles).

Theorem 3. *For every nice self-similar fractal $X \subset \mathbb{N}^2$, there is a directed TAS in which $\mathbf{X}$ strictly self-assembles.*

We now give a brief sketch of our construction of the singly-seeded TAS $\mathcal{T}_{\mathbf{X}} = (X_{\mathbf{X}}, \sigma, 2)$ in which $\mathbf{X}$ strictly self-assembles. The full construction is implemented in C++, and is available at the following URL: `http://www.cs.iastate.edu/~lnsa`.

Throughout our discussion, $S_{\boldsymbol{u}}$, $Y_{\boldsymbol{u}}$, and $X_{\boldsymbol{u}}$ refer to the square, the vertical rectangle and the horizontal rectangle, respectively, that form the "outer framework" of the set $((T_i \cup F_i) + l(i) \cdot \boldsymbol{u})$ (See the right-most image in Figure 4).

5.1 Construction Phase 1

Here, directed graphs are considered. Let X be a nice (c-discrete) self-similar fractal generated by V. We first compute a directed spanning tree $B = (V, E)$ of $G_V^{\#}$ using a breadth-first search, and then compute the graph $B^{\mathrm{R}} = \left(V, E^{\mathrm{R}}\right)$, where

$$E^{\mathrm{R}} = \{(\boldsymbol{v}, \boldsymbol{u}) \mid (\boldsymbol{u}, \boldsymbol{v}) \in E \text{ and } \boldsymbol{u} \neq (0,0)\} \cup \{((0,1),(0,c-1)),((1,0),(c-1),0)\}.$$

Figure 3 depicts phase 1 of our construction for a particular nice self-similar fractal.

Notation 1. *For all* $\mathbf{0} \neq \boldsymbol{u} \in V$, $\boldsymbol{u}_{in}$ *is the unique location* $\boldsymbol{v}$ *satisfying* $(\boldsymbol{u}, \boldsymbol{v}) \in E^{R}$.

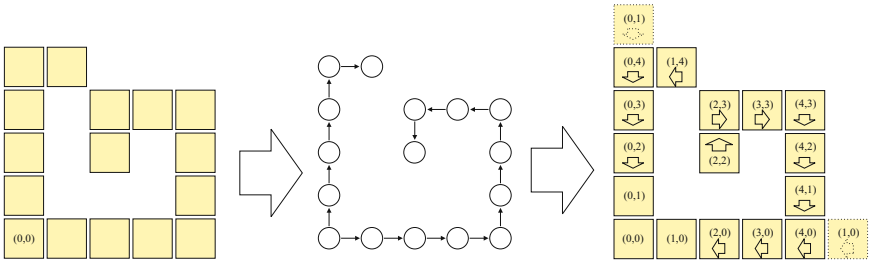


Fig. 3. Phase 1 of our construction. Notice the two special cases (right-most image) in which we define $(0,1)_{\mathrm{in}}$ and $(1,0)_{\mathrm{in}}$.

5.2 Construction Phase 2

In the second phase we construct, for each $(0,0) \neq \boldsymbol{u} \in V$, a finite set of tile types $T_{\boldsymbol{u}}$ that self-assemble a particular subset of $\mathbf{X}$. There are two cases to consider.

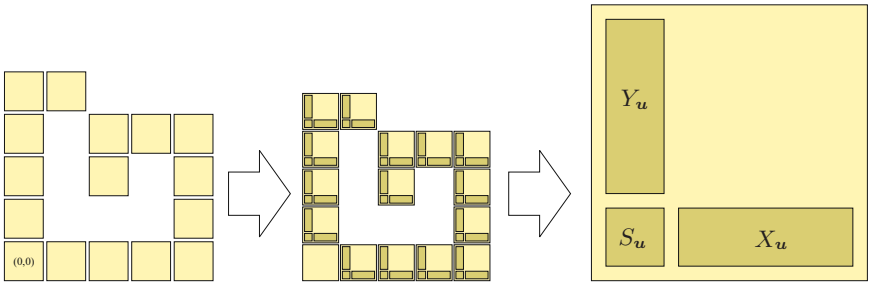


Fig. 4. Let V be the left-most image. The first arrow represents phase 2 of the construction. The second arrow shows a magnified view of a particular point in V. Each point $(0,0) \neq \boldsymbol{u} \in V$ can be viewed conceptually as three components: the tile sets $T_{S_{\boldsymbol{u}}}$, $T_{X_{\boldsymbol{u}}}$ and $T_{Y_{\boldsymbol{u}}}$ that ultimately self-assemble the square $S_{\boldsymbol{u}}$, and the horizontal and vertical rectangles $X_{\boldsymbol{u}}$ and $Y_{\boldsymbol{u}}$ respectively.

Case 1. In the first case, we generate, for each $\boldsymbol{u} \in V - \{(0,0),(0,1),(1,0)\}$, three sets of tile types $T_{S_{\boldsymbol{u}}}$, $T_{X_{\boldsymbol{u}}}$, and $T_{Y_{\boldsymbol{u}}}$ that, when combined together, and assuming the presence of $((T_i \cup F_i) + l(i) \cdot \boldsymbol{u}_{\text{in}})$, self-assemble the set $((T_i \cup F_i) + l(i) \cdot \boldsymbol{u})$, for any $i \in \mathbb{N}$.

Case 2. In the second case, we generate, for each $\boldsymbol{u} \in \{(0,1),(1,0)\}$, the same three sets of tile types ($T_{S_{\boldsymbol{u}}}$, $T_{X_{\boldsymbol{u}}}$, and $T_{Y_{\boldsymbol{u}}}$) that self-assemble the set $((T_i \cup F_i) + l(i) \cdot \boldsymbol{u})$ "on top of" the set $((T_{i-1} \cup F_{i-1}) + l(i-1) \cdot \boldsymbol{u}_{\text{in}})$, for any $i \in \mathbb{N}$.

Finally, we let $T_{\mathbf{X}} = \bigcup_{(0,0) \neq \boldsymbol{u} \in V} T_{\boldsymbol{u}}$, where $T_{\boldsymbol{u}} = T_{S_{\boldsymbol{u}}} \cup T_{X_{\boldsymbol{u}}} \cup T_{Y_{\boldsymbol{u}}}$. Figure 4 gives a visual interpretation of the second phase of our construction. Our TAS is $\mathcal{T}_{\mathbf{X}} = (T_{\mathbf{X}}, \sigma, 2)$, where σ consists of a single "seed" tile type placed at the origin. Our full construction yields a tile set of 5983 tile types for the fractal generated by the points in the left-most image in Figure 4.

5.3 Details of Construction

Note that in our construction, the self-assembly of the sub-structures $S_{\boldsymbol{u}}$, $Y_{\boldsymbol{u}}$, and $X_{\boldsymbol{u}}$ can proceed either *forward* (away from the axes) or *backward* (toward the axes).

Forward Growth. We now discuss the self-assembly of the set $((T_i \cup F_i) + \boldsymbol{u} \cdot l(i))$ for $\boldsymbol{u} \in V$ satisfying $\boldsymbol{u}_{\text{in}} \in (\boldsymbol{u} + \{(-1,0),(0,-1)\})$.

If $\boldsymbol{u} \notin \{(0,0),(0,1),(1,0)\}$ (i.e., case 1 of phase 2), then the tile set $T_{S_{\boldsymbol{u}}}$ self-assembles the square $S_{\boldsymbol{u}}$ directly on top (or to the right) of, and having the same width (height) as, the rectangle $Y_{\boldsymbol{u}_{\text{in}}}$ ($X_{\boldsymbol{u}_{\text{in}}}$). If $\boldsymbol{u} \in \{(0,1),(1,0)\}$ (i.e., case 2 of phase 2), then the tile set $T_{S_{\boldsymbol{u}}}$ self-assembles the square $S_{\boldsymbol{u}}$ on top (or to the right) of the set $Y_{\boldsymbol{u}_{\text{in}}}$ such that right (top) edge of the former is flush with that of the latter. Note that in case 2, the width of $Y_{\boldsymbol{u}_{\text{in}}}$ is always one less than that of $S_{\boldsymbol{u}}$. In either case, it is straightforward to construct such a tile set $T_{S_{\boldsymbol{u}}}$.

The tile set of $T_{Y_{\boldsymbol{u}}}$ self-assembles a fixed-width base-c counter (based on the "optimal" binary counter presented in [2]) that, assuming a width of $i \in \mathbb{N}$, implements the following counting scheme: Count each positive integer j, satisfying $1 \leq j \leq c^i - 1$, in order but count each number exactly

$$[\![c \text{ divides } j]\!] \cdot \rho(j) + [\![c \text{ does not divide } j]\!] \cdot 1$$

times, where $\rho(j)$ is the largest number of consecutive least-significant 0's in the base-c representation of j, and $[\![\phi]\!]$ is the *Boolean* value of the statement ϕ. The *value* of a row is the number that it represents. We refer to any row whose value is a multiple of c as a *spacing row*. All other rows are *count* rows. The *type* of the counter that self-assembles $Y_{\boldsymbol{u}}$ is $\boldsymbol{u}$.

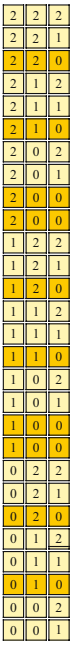


Fig. 5. Example of a base-3 modified binary counter. The darker shaded rows are the spacing rows.

Each counter self-assembles on top (or to the right) of the square $S_{\boldsymbol{u}}$, with the width of the counter being determined by that of the square. It is easy to verify that if the width of $S_{\boldsymbol{u}}$ is $i+2$, then $T_{\boldsymbol{Y_u}}$ self-assembles a rectangle having a width of $i+2$ and a height of

$$\left(c^2+1\right)c^i + \frac{c^i-1}{c-1} = l(i) - (i+2),$$

which is exactly $Y_{\boldsymbol{u}}$. Figure 5 shows the counting scheme of a base-3 counter of width 3. We construct the set $T_{X_{\boldsymbol{u}}}$ by simply reflecting the tile types in $T_{Y_{\boldsymbol{u}}}$ about the line $y = x$, whence the three sets of tile types $T_{S_{\boldsymbol{u}}}$, $T_{X_{\boldsymbol{u}}}$, and $T_{Y_{\boldsymbol{u}}}$ self-assemble the "outer framework" of the set $((T_i \cup F_i) + \boldsymbol{u} \cdot l(i))$.

The "internal structure" of the set $((T_i \cup F_i) + \boldsymbol{u} \cdot l(i))$ self-assembles as follows. Oppositely oriented counters attach to the right side of each contiguous group of spacing rows in the counter (of type $\boldsymbol{u}$) that self-assembles $Y_{\boldsymbol{u}}$. The number of such spacing rows determines the height of the horizontal counter, and its type is $(0, j/c \mod c)$, where j is the value of the spacing rows to which it attaches. We also hard code the glues along the right side of each non-spacing row to self-assemble the internal structure of the points in the set T_0.

The situation for $X_{\boldsymbol{u}}$ is similar (i.e., a reflection of its vertical counterpart), with the exception that the glues along the top of each non-spacing row are configured differently than they were for $Y_{\boldsymbol{u}}$. This is because nice self-similar fractals need not be symmetric.

One can prove that, by recursively attaching smaller oppositely-oriented counters (of the appropriate type) to larger counters in the above manner, the internal structure of $((T_i \cup F_i) + \boldsymbol{u} \cdot l(i))$ self-assembles.

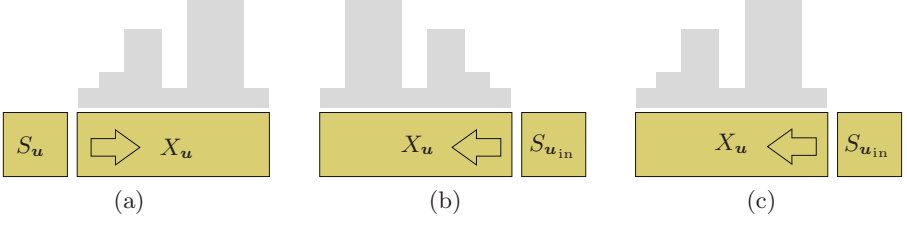


Fig. 6. (a) Depicts forward growth, (b) Shows what happens if the tile set $T_{X_{\boldsymbol{u}}}$ were to simply "count in reverse," and (c) is the desired result

Reverse Growth. We now discuss the self-assembly of the set $((T_i \cup F_i) + \boldsymbol{u} \cdot l(i))$, for all $\boldsymbol{u} \in V$ satisfying $\boldsymbol{u}_{\text{in}} \in (\boldsymbol{u} + \{(1,0),(0,1)\})$.

In this case, the tile set $T_{Y_{\boldsymbol{u}}}$ ($T_{X_{\boldsymbol{u}}}$) self-assembles the set $Y_{\boldsymbol{u}}$ ($X_{\boldsymbol{u}}$) directly below (or to the left of) the square $S_{\boldsymbol{u}_{\text{in}}}$, and grows toward the x-axis (or y-axis) according to the base-c counting scheme outlined above. We also configure $T_{Y_{\boldsymbol{u}}}$ ($T_{X_{\boldsymbol{u}}}$) so that the right (or top)-most edge of $Y_{\boldsymbol{u}}$ ($X_{\boldsymbol{u}}$) is essentially the "mirror" image of its forward growing counterpart (See Figure 6). This last step ensures that the internal structure of $((T_i \cup F_i) + \boldsymbol{u} \cdot l(i))$ self-assembles correctly. Next,

the square $S_{\boldsymbol{u}}$ attaches to the bottom (or left)-most edge of $Y_{\boldsymbol{u}}$ ($X_{\boldsymbol{u}}$). Finally, the set $X_{\boldsymbol{u}}$ ($Y_{\boldsymbol{u}}$) self-assembles via forward growth from the left (or top) of the square $S_{\boldsymbol{u}}$.

Proof of Correctness. To prove the correctness of our construction, we use a local determinism argument. The details of the proof are technical, and therefore omitted from this version of the paper.

6 Conclusion

In this paper, we (1) established two new absolute limitations of the TAM, and (2) showed that fibered versions of "nice" self-similar fractals strictly self-assemble. Our impossibility results motivate the following question: Is there a discrete self-similar fractal $X \subsetneq \mathbb{N}^2$ that strictly self-assembles in the TAM? Moreover, our positive result leads us to ask: If $X \subsetneq \mathbb{N}^2$ is a discrete self-similar fractal, then is it always the case that X has a "fibered" version $\mathbf{X}$ that strictly self-assembles, and that is similar to X in some reasonable sense?

Acknowledgment. We thank Dave Doty, Jim Lathrop, Jack Lutz, and Aaron Sterling for useful discussions.

References

1. Aggarwal, G., Goldwasser, M.H., Kau, M.-Y., Schweller, R.T.: Complexities for generalized models of self-assembly. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms (2004)
2. Cheng, Q., Goel, A., de Espanés, P.M.: Optimal self-assembly of counters at temperature two. In: Proceedings of the First Conference on Foundations of Nanoscience: Self-assembled Architectures and Devices (2004)
3. Doty, D., Gu, X., Lutz, J.H., Mayordomo, E., Moser, P.: Zeta-dimension. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 283–294. Springer, Heidelberg (2005)
4. 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, Florida, pp. 571–580 (January 2006) (2007)
5. Lathrop, J.I., Lutz, J.H., Patitz, M.J., Summers, S.M.: Computability and complexity in self-assembly. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 349–358. Springer, Heidelberg (2008)
6. Lathrop, J.I., Lutz, J.H., Summers, S.M.: Strict self-assembly of discrete sierpinski triangles. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) CiE 2007. LNCS, vol. 4497, pp. 455–464. Springer, Heidelberg (2007)
7. Majumder, U., LaBean, T.H., Reif, J.H.: Activatable tiles: Compact, robust programmable assembly and other applications. In: Garzon, M.H., Yan, H. (eds.) DNA 2007. LNCS, vol. 4848, pp. 15–25. Springer, Heidelberg (2008)
8. Rothemund, P.W.K.: Theory and experiments in algorithmic self-assembly, Ph.D. thesis, University of Southern California (December 2001)

9. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pp. 459–468 (2000)
10. Seeman, N.C.: Nucleic-acid junctions and lattices. Journal of Theoretical Biology 99, 237–247 (1982)
11. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM Journal on Computing 36, 1544–1569 (2007)
12. Wang, H.: Proving theorems by pattern recognition – II. The Bell System Technical Journal XL(1), 1–41 (1961)
13. Wang, H.: Dominoes and the AEA case of the decision problem. In: Proceedings of the Symposium on Mathematical Theory of Automata (New York, 1962), pp. 23–55. Polytechnic Press of Polytechnic Inst. of Brooklyn, Brooklyn (1963)
14. Winfree, E.: Algorithmic self-assembly of DNA, Ph.D. thesis, California Institute of Technology (June 1998)

Speeding Up Local-Search Type Algorithms for Designing DNA Sequences under Thermodynamical Constraints*

Suguru Kawashimo, Yen Kaow Ng, Hirotaka Ono,
Kunihiko Sadakane, and Masafumi Yamashita

Dept. of Computer Science and Communication Engineering, Kyushu University,
744 Motooka, Nishi-ku, Fukuoka, Fukuoka 819-0395, Japan
{kawa,kalngyk}@tcslab.csce.kyushu-u.ac.jp,
{ono,sada,mak}@csce.kyushu-u.ac.jp

Abstract. We present general techniques to speed up local search type algorithms for designing DNA sequences which satisfy thermodynamical constraints based on the minimum free energy (MFE) criteria. MFE based constraints are generally difficult to handle in local search type algorithms, since these algorithms typically require a large number of time-consuming calculations of MFE to find an improved solution. In this paper, we introduce general techniques to reduce such calculations of MFE. The ideas are based on the reuse of MFE computations and fast approximation of MFE, both of which fit the nature of local search type algorithms. In computational experiments, our techniques succeeded in speeding up typical local search type algorithms without degenerating the original performance of the algorithms.

Keywords: DNA Sequence Design, Local Search, Statistical Thermodynamical Constraints.

1 Introduction

Designing DNA sequences is an important task for a number of techniques in nanotechnology and nanocomputing, e.g. Adleman's DNA solution for the Hamiltonian path [1], and DNA tiling with its self-assemble [23]. In these techniques, it is important to control the DNA molecules' reactions so that they react only in expected ways, since unexpected reactions of DNA sequences may cause errors. Sequence design is an approach to avoid these unexpected reactions by using DNA sequences that fulfill certain constraints which prohibit the unexpected reactions [4,7].

Such constraints can be roughly classified into combinatorial and thermodynamical types. Early studies of sequence design have treated combinatorial constraints due to their simplicity [5,6,13,14,20,21]. However, there is a growing interest in thermodynamical constraints due to a need for sophistication.

* This research partly received financial support from Scientific Research Fund of Ministry of Education, Culture, Sports, Science and Technology (KAKENHI).

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 168–178, 2009.

Recently, several studies have been made on designing sequences with thermodynamical constraints [10,15,18,19,22] based on *Minimum Free Energy* (MFE).

Sequence design can be considered as a problem of combinatorial optimization. In such problems, optimal solutions may be difficult to find (i.e. NP-hard). Since in sequence design a good but not necessarily optimal solution often suffices, local search methods have been considered [13,14,15,20,21,22]. Using local search methods for sequence design also allows one to adapt to different constraints easily.

In local search, one starts with an arbitrary solution S, and keep replacing S with a "better" solution found from within the neighborhood of S. In the case of thermodynamical constraints, solutions are evaluated on a given criteria that is based on the Gibbs standard free energy, for instance the MFE. The currently known method for computing MFE requires a costly $O(n^3)$ time, where n is the input sequence's length [2,16,24]. Since local search type algorithms need the MFE value of each solution for evaluation, MFE computation time frequently becomes a bottleneck on their efficiency.

This paper presents two general techniques for overcoming this difficulty: (1) reuse values calculated for one MFE computation in later MFE computations, (2) use a fast approximation of MFE values to determine if actual MFE computation would be needed. To evaluate the effects of these techniques on the computation time of MFE based local search methods for sequence design, we apply the techniques to the two methods: Dynamic Neighborhood Search [14,15] and Stochastic Local Search [20,21,22]. In both cases we observe significant speed-up in our results.

Related Work. Asahiro showed that DNA sequence design problem under a certain condition is NP-hard [6]. Tanaka *et al.* proposed a random-generation based method for thermodynamically constrained sequence design [19]. The method requires many evaluations, as in local search type methods. To reduce the calculation time of MFEs they used a fast approximation method. In an earlier paper, we proposed a Dynamic Neighborhood Search method which attempts to circumvent the heavy calculations through a few search strategies, and obtained some success [15]. Tulpan *et al.* succeeded in designing sequence sets under very complicated thermodynamical constraints by the Stochastic Local Search method [22]. However, the total running time of their method is not clear because they excluded the time for calculating energy values in their evaluation of the search time. The stochastic local search method has also been applied to the Secondary Structure Design (Reverse Folding) problem in a few studies [3,11], and the heavy computation needed for calculating MFEs were also remarked in these studies.

2 Preliminaries

A DNA sequences s is a string over the nucleotides {`A`,`T`,`C`,`G`}. A DNA sequence (or sequences) form a *secondary structure* (called a *conformation*) through Watson-Crick hydrogen bonds, which can occur between the nucleotides `A` and `T`, or between `G` and `C`. Each conformation of a sequence (or sequences) has a Gibbs

standard *free energy*, the value of which can be measured through experiments. The Gibbs standard free energy of a conformation can be computed in time linear to the total length of the conformation's sequences. Conformations with smaller Gibbs standard free energies tend to be more stable. The *Minimum Free Energy* (MFE) of a sequence (resp., sequences) is the minimum value among free energies of all possible conformations of a sequence (resp., sequences).

Let $s = s_1 s_2 \cdots s_n$ and $s' = s'_1 s'_2 \cdots s'_n \in \{\mathtt{A}, \mathtt{T}, \mathtt{G}, \mathtt{C}\}^n$ be DNA sequences of length n. Each DNA sequence has a direction. The left end s_1 (resp., s'_1) of sequence s (resp., s') corresponds to the $5'$ end of the DNA sequence. Let $wcc(s)$ denote the Watson-Crick complement of the DNA sequence s, that is, $wcc(s) = \bar{s}_n \bar{s}_{n-1} \cdots \bar{s}_1$, where $\bar{s}_i = \mathtt{A}$ (resp., $\mathtt{T}$) if $s_i = \mathtt{T}$ (resp., $\mathtt{A}$) and $\bar{s}_i = \mathtt{C}$ (resp., $\mathtt{G}$) if $s_i = \mathtt{G}$ (resp., $\mathtt{C}$). Let S be a set of sequences. In the context of sequence design problems, we call a phenomenon that a sequence in S forms completely hydrogen bonds with its complement sequence, a *hybridization*, and call a conformation that is not a hybridization, a *miss-hybridization*. The constraints described below are introduced in order to avoid any miss-hybridization. The MFE between s and s' is represented by $\Delta G(s, s')$, and can be calculated in $O(n^3)$ time through dynamic programming [2,16,24]. Let $wcc(S) = \{wcc(s) \mid s \in S\}$. Given threshold parameters t_{ww}, t_{wc}, and t_{cc}, we define the following constraints:

***word-word*:** $\Delta G_{ww}(S) \stackrel{\text{def}}{=} \min_{s,s' \in S} \{\Delta G(s, s')\} \geq t_{ww}$.
***word-complement*:** $\Delta G_{wc}(S) \stackrel{\text{def}}{=} \min_{s \in S, s' \in wcc(S), s \neq wcc(s')} \{\Delta G(s, s')\} \geq t_{wc}$.
***complement-complement*:** $\Delta G_{cc}(S) \stackrel{\text{def}}{=} \min_{s,s' \in wcc(S)} \{\Delta G(s, s')\} \geq t_{cc}$.

Our problem is to "*find S such that $\Delta G_{ww}(S) \geq t_{ww}$, $\Delta G_{wc}(S) \geq t_{wc}$, and $\Delta G_{cc}(S) \geq t_{cc}$*".

Note that the constraints apply as well to self reactions of sequences. Hence we can allow sequences that conform to hairpins, bulge-loops, internal-loops and multi-loops.

While we use only the above constraints in the present study, the method we propose should be applicable to other MFE-based criteria (e.g. energy gap [21]), and with some careful adjustments, applicable to other criteria, such as melting temperature and DNA error rate [17], as well.

2.1 Local Search

Combinatorial optimization problems are often stated as to find, given a scoring function f over the solution space, a solution x such that $f(x)$ is minimized or maximized. Let $N(x)$ denote the *neighborhood* of x (i.e., a set of solutions which are obtained by slight perturbations to x). Operations which perturb a solution x to obtain $N(x)$ are called *neighborhood operations*. When x satisfies $f(x) < f(x')$ (or $f(x) > f(x')$) for $\forall x' \in N(x)$, x is called the *local optima*. In general, there are a lot of local optima in a given solution space.

The basic idea behind local search methods can be stated as follows:

(1) Select an initial solution x.
(2) Search in $N(x)$.

(3) If an improved solution x' is found in (2), the solution x is replaced by x', and go to (2). Otherwise current x is a local optima. Return x.

A point to consider in implementing local searches is in deciding which improved solution to use in step (3) when there is more than one improved solution in the neighborhood $N(x)$. There are two basic strategies for this: (i) *first admissible move strategy*, which moves immediately when an improved solution is found in $N(x)$, and (ii) *best admissible move strategy*, which moves to the best solution after checking all solutions in $N(x)$.

3 Techniques to Reduce MFE Evaluations in Local-Search

We first define our neighborhood operations. For a sequence s, we call the change of a base at a certain position in s to another base a *flip*. For a set of sequences S,

$$N(S) \stackrel{\text{def}}{=} \{S' \mid S' \text{is a sequence set obtained by flipping a sequence in } S\}. \quad (1)$$

This definition is adopted in Kawashimo *et al.* [14,15]. Tulpan *et al.* [20,21] use a similar definition for neighborhood.

To determine if a solution in $N(S)$ is an improved solution, we use a score which depends on the constraints given earlier. For each constraint, each calculation of $\Delta G(s, s')$ takes $O(n^3)$ time, and this calculation is computed for every pair of sequences in $S \cup wcc(S)$, i.e., for a total of $O(m^2)$ pairs where $m = |S|$. Thus, it takes at most $O(m^2 n^3)$ time to evaluate one solution. This runtime can be easily improved by noticing that there are overlaps between the pairs of sequences in S and the pairs of sequences in any $S' \in N(S)$. By reusing the values, the calculation for each $S' \in N(S)$ can be done in $O(mn^3)$ time.

While tractable, this $O(mn^3)$ computation presents a heavy bottleneck to the efficiency of the method, because the computation needs to be repeated for every S' from a very numerous $N(S)$. In this section, we suggest two techniques to reduce this computation. The techniques utilize a characteristic of local search type algorithms, that is, the difference between a current solution and a new solution is small.

3.1 Reuse of DP Tables for Calculating MFE

MFE between two sequences can be calculated using a dynamic programming (DP) approach, which corresponds to the computation of a table as in Figure 1 [2,24]. Here, the vertical and horizontal axes correspond to the concatenation of the two sequences. In this DP, an MFE value is calculated by filling in all the cells in the table, where the value of a cell is determined by referring to the upper right region of the cell.

In the following, we consider how to obtain the DP table for a flipped sequence s' of original sequence s, from the DP table for s. To do so, we examine which parts of the DP table need to be recalculated. If a base is flipped in a neighborhood operation, only a column and a row of the new DP table that correspond

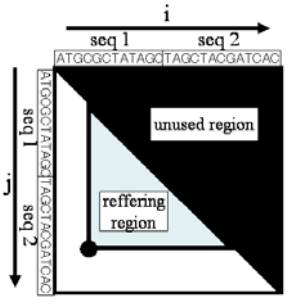


Fig. 1. Table of dynamic programming

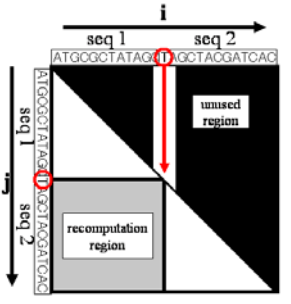


Fig. 2. Difference after a neighborhood operation *(Encircled base (*T*) is flipped)*

to that base will take on different values. Except for a rectangular region, all the cells in the table will have the same values as before the neighborhood operation (see Figure 2), since their values are decided from their upper right region. Therefore, by preserving the table before a neighborhood operation, we can skip the computations for the unchanged regions and reduce the time in calculating MFEs. In this paper we refer to this technique as "reuse table".

3.2 Approximate Calculation of MFE

In this subsection, we show a technique to quickly approximate MFE values. The main incentive in obtaining such approximations is that we may be able to tell, from the approximation, if a solution is not likely to be an improvement over the current one, and may hence skip its MFE computation.

An MFE-structure (between two sequences s, s') before a neighborhood operation (on s' say) tends to be similar to an MFE-structure after the operation, because their sequences differ by only a single base in s' (see Figure 3). It is hence conceivable for us to obtain an approximation of the MFE-structure after a neighborhood operation from the MFE-structure prior to the operation with only slight alterations. There are a few possible alterations, for which we consider only the following (see Figure 4):

(1) disassociate base-pair including flipped base.
(2) disassociate all base-pairs taking continuous stuck including base-pair selected in (1).
(3) form new base-pair including flipped base.

The reason why we consider alteration (2) is the following: since continuous stucks are known to be stable, we consider that deleting all base-pairs in the continuous stucks make more stable conformation than dividing the continuous stuck.

We list out the structures obtained by the above alterations, calculate the free energies of each new structure (after flipping the base), and use the minimum

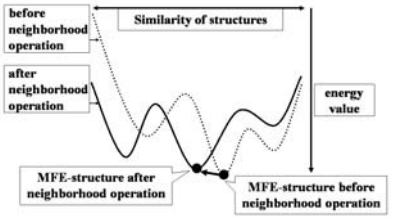


Fig. 3. Concept of the similarity of MFE-structure

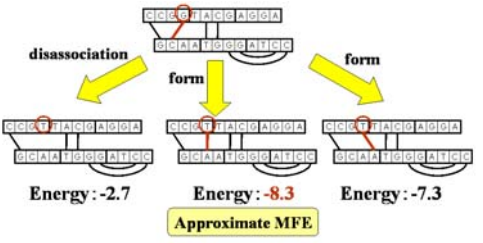


Fig. 4. Concept of ΔG^*

of these free energies as an approximation of the MFE after the neighborhood operation. We denote this approximate MFE by $\Delta G^*(s, s')$. Clearly, $\Delta G^*(s, s') \geq \Delta G(s, s')$.

3.3 Preliminary Evaluation

We perform some preliminary computational experiments to evaluate the effectiveness of our techniques, i.e. "reuse table" and "$\Delta G^*(s, s')$". We use the `PairFold` package [2] for calculating MFEs. The setting temperature is 37°C. In the experiments, we randomly generate 100 pairs of sequences, and calculate MFEs by two ways: one by ordinary dynamic programming and the other by using "reuse table". We measure the times taken for both cases. We compute $\Delta G^*(s, s')$ as described in Subsection 3.2, and measure the times taken for their computation. We evaluate how frequently the approximation $\Delta G^*(s, s')$ matches $\Delta G(s, s')$ exactly. We also try to see how well $\Delta G^*(s, s')$ approximates $\Delta G(s, s')$ by looking at $\Delta G^*(s, s') - \Delta G(s, s')$. Table 1 shows these values.

Table 1. Preliminary test

length	normal t	"reuse table" t	$\Delta G^*(s, s')$ t	% match	difference	$\Delta G(s, s')$
15	9.889	7.296	0.852	61.2%	0.517	-3.550
20	38.230	26.778	1.332	70.3%	0.629	-4.866
25	109.395	75.073	1.968	41.1%	0.855	-6.559
30	257.872	171.895	2.604	34.4%	1.025	-7.941
35	518.924	336.433	4.016	35.7%	1.045	-9.324
40	949.095	573.524	5.308	34.5%	1.094	-10.526
45	1481.909	901.084	6.700	38.9%	1.032	-12.469
50	2205.982	1294.197	9.445	44.1%	0.902	-14.275

normal t : time (sec) used to calculate MFEs w/o using proposed techniques.
"reuse table" t : time (sec) used to calculate MFE with "reuse table" technique.
$\Delta G^*(s, s')$ time : time (sec) used to calculate $\Delta G^*(s, s')$.
% match : percentage of instances where $\Delta G^*(s, s') = \Delta G(s, s')$.
difference : average of $\Delta G^*(s, s') - \Delta G(s, s')$ (kcal/mol).
$\Delta G(s, s')$: average of $\Delta G(s, s')$ (kcal/mol).

As shown in the table, the times used to calculate MFEs when using the "reuse table" technique are only 75% to 60% of the times when "reuse table" is not used. The times needed to calculate "$\Delta G^*(s, s')$" are about 10% to 0.5% those for calculating $\Delta G(s, s')$ exactly. On the other hand, the averages of $\Delta G^*(s, s') - \Delta G(s, s')$ are at most 1.1 kcal/mol, and the average error ratios are at most 15%. These results show that: (1) "reuse table" has effectively reduced calculation time, (2) $\Delta G^*(s, s')$ can be computed relatively quickly, and (3) $\Delta G^*(s, s')$ gave good approximations of $\Delta G(s, s')$. These results also suggest that our techniques would be more effective for large n, since the improvements in running time for larger n are greater.

4 Application for Local Search Type Algorithms

In this section, we try to apply the ideas proposed in section 3 to local search type sequence design methods. Our target methods are: the Dynamic Neighborhood Search [14,15] and the Stochastic Local Search [20,21,22].

The Dynamic Neighborhood Search method dynamically redefines neighborhood so as to form a tree structure which is obtained by chains of the neighborhood (Equation 1), and searches on the tree. This method is performed based on the first admissible move strategy and the objective function in this method is defined as follows [15]:

$$Value(S) \stackrel{\text{def}}{=} \min\{\Delta G_{ww}(S) - t_{ww}, 0\} + \min\{\Delta G_{wc}(S) - t_{wc}, 0\} + \min\{\Delta G_{cc}(S) - t_{cc}, 0\} \quad (2)$$

It is clear that $Value(S) \leq 0$, and $Value(S) = 0$ if and only if our MFE constraints are satisfied. In the search, we consider the greater the value of $Value(S)$, the better S is.

The Stochastic Local Search method moves at random with a probability parameter, which enables the method to find various solutions. This method is performed based on the best admissible move strategy and the objective function in this method is defined as the number of pairs that violate the given constraints [20].

In spite of the differences in move strategy and objective function, these two methods share characteristics of local search methods which makes our methods applicable.

4.1 Case: Dynamic Neighborhood Search

The Dynamic Neighborhood Search method is based on the first admissible move strategy, i.e. the search moves to a better solution, if and when such a solution is found. However, in the neighborhood structure of the method, a solution typically has only a few better neighboring solutions. Hence, the total evaluation time is dominated by evaluations of neighborhood solutions that are not better, and we can expect the total evaluation time to be greatly reduced if these evaluations can be avoided. So, we introduce a preprocessing phase using

$\Delta G^*(s, s')$ to compute an approximate MFE value, which utilizes the relation $\Delta G^*(s, s') \geq \Delta G(s, s')$.

We let $Value^*(S)$ be an approximation of $Value(S)$ (Equation 2) in which $\Delta G^*(s, s')$ is used instead of $\Delta G(s, s')$. Since $\Delta G^*(s, s') \geq \Delta G(s, s')$, $Value^*(S) \geq Value(S)$ holds.

Now to check if a neighbor solution S_{new} is an improvement over S_{old}, we can perform the following:

(1) (Preprocessing phase) Calculate $Value^*(S_{new})$. If $Value^*(S_{new}) \leq Value(S_{old})$, then S_{new} is no better than S_{old}. Otherwise, go to (2).

(2) Calculate $Value(S_{new})$ with "reuse table". S_{new} is an improvement of S_{old} if and only if $Value(S_{new}) > Value(S_{old})$.

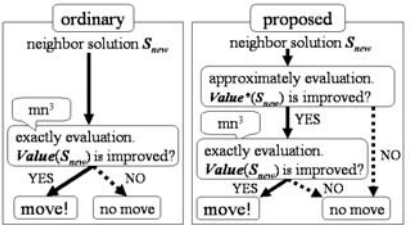


Fig. 5. Utilization of $\Delta G^*(s, s')$

Figure 5 summarizes the method. The use of the approximation $\Delta G^*(s, s')$ will not change the result of the search. However, when the approximation is insufficient to identify bad solutions, the preprocessing becomes an extra cost to bear.

To see how the technique helps in speeding up the search, we implement and perform computational experiments. For each experiment, we fix a length n, a size m, and thresholds t_{ww}, t_{wc}, t_{cc}. We randomly generate an initial set of m sequences, and apply the Dynamic Neighborhood Search method until it satisfies $\Delta G_{ww}(S) \geq t_{ww}$, $\Delta G_{wc}(S) \geq t_{wc}$, and $\Delta G_{cc}(S) \geq t_{cc}$. Each experiment is repeated for five trials. The values for each experiment are averages taken over its five trials. We perform experiments for three cases: without the proposed techniques, with "reuse table" and with both "reuse table" and "$\Delta G^*(s, s')$". Table 2 shows the results obtained.

As shown in this table, the running times when "reuse table" is used are about 95% to 85% of the running times when it is not used; while the running times when both "reuse table" and "$\Delta G^*(s, s')$" are used are about 95% to 75% of the running times when they are not used. Since the results produced are the same, the techniques have lost nothing in speeding the algorithm. For the same m and n, the cases of larger t_{ww}, t_{wc}, t_{cc} values tend to show better improvements from the use of "$\Delta G^*(s, s')$", i.e. compared to when only "reuse table" is used. One possible explanation is that the weaker constraints due to the smaller t_{ww}, t_{wc}, t_{cc} values resulted in an abundance of improved solutions in the neighborhood. Since it is easier to find an improved solution, the extra cost in preprocessing becomes unwarranted. In fact, there are two cases where the use of preprocessing resulted in a little extra running time. Intuitively, we may say that the benefits from preprocessing will be more significant in the harder instances (or constraints).

Table 2. Average time (sec) for sequence design using Dynamic Neighborhood Search

n	m	t_{ww}, t_{wc}, t_{cc}	w/o proposed tech.	"reuse table"	both techniques
10	50	-4.0	27.14	**25.01**	25.27
10	50	-3.5	153.60	134.59	**129.27**
10	100	-5.0	81.18	**77.08**	78.00
10	100	-4.0	505.59	452.68	**433.83**
15	50	-6.0	50.67	46.41	**46.40**
15	50	-5.0	187.71	163.59	**158.04**
15	100	-7.0	186.20	174.73	**174.16**
15	100	-6.0	407.63	373.57	**370.60**
20	50	-6.5	233.53	206.45	**203.76**
20	50	-5.5	999.39	830.07	**741.99**
20	100	-7.5	846.52	770.79	**768.51**
20	100	-6.5	2031.15	1781.03	**1714.57**

w/o proposed tech.: without using the proposed techniques.
"reuse table" : with "reuse table" technique only.
both techniques : with both "reuse table" and "$\Delta G^*(s, s')$" techniques.

4.2 Case: Stochastic Local Search

As mentioned, the Stochastic Local Search method is based on the best admissible move strategy. The strategy evaluates all neighboring solutions and moves to the best improved solution. For the present study, we consider the following two possible alternative implementations of the best admissible move strategy using the our proposed techniques:

(A) Evaluate all neighboring solutions with "reuse table" and move to the best improved solution.

(B) Approximately evaluate all neighboring solutions with "$\Delta G^*(s, s')$" and move to the best improved solution according to the approximations. After this, exactly evaluate the solution (to get a value for comparisons with neighboring solutions).

(A) is a speed-up version of the original Stochastic Local Search using the "reuse table" technique, in which the implementation follows the one described in Subsection 4.1. In (B), the improved solution to move to is decided according to approximate evaluations. Hence it is possible that the move may be different from that selected with exact evaluations.

To see how these methods compare, we implement and perform computational experiments[1] in the same way as in Section 4.1. We show in Table 3) results from experiments for the three cases: without proposed techniques, with "reuse table" (i.e. A) and with "$\Delta G^*(s, s')$" (i.e. B).

As shown in the table, both the running times when "reuse table" is used and when "$\Delta G^*(s, s')$" is used are about 80% to 70% those when none of them are

[1] We implement the simplest Stochastic Local Search method [20]. Our implementation improves the set until it satisfies constraints without random multi-start method. We set $\theta = 0.2$, $N\_STEP = 50$ and $fr = 10$.

Table 3. Average time (sec) for sequence design using Stochastic Local Search

n	m	t_{ww}, t_{wc}, t_{cc}	w/o proposed tech.	"reuse table"	both techniques
10	50	-6.5	29.75	**29.75**	29.88
10	50	-5.5	93.18	**71.98**	72.36
10	100	-7.5	119.62	**94.97**	95.20
10	100	-6.5	251.22	**199.47**	200.56
15	50	-9.5	96.47	69.25	**69.19**
15	50	-8.5	184.14	132.91	**132.48**
15	100	-10.5	362.80	267.83	**267.13**
15	100	-9.5	654.20	483.00	**482.07**
20	50	-13.0	211.99	146.87	**145.70**
20	50	-12.0	299.50	207.59	**206.06**
20	100	-14.0	443.63	**308.08**	310.52
20	100	-13.0	934.59	**650.05**	654.08

w/o proposed tech.: without using the proposed techniques.
"reuse table" : with "reuse table" technique only.
both techniques : with both "reuse table" and "$\Delta G^*(s, s')$" techniques.

used. There are not much difference between the running times of "reuse table" and "$\Delta G^*(s, s')$". However, "reuse table" requires significantly more memory to store all the DP tables prior to the neighborhood operation, while "$\Delta G^*(s, s')$" requires only the MFE-structures. Our results also show the speed-up to be more significant for larger n (see Subsection 3.3).

5 Concluding Remarks

As future work, application of the techniques we proposed to the Secondary Structure Design problem [12] could be interesting. The Secondary Structure Design problem can be considered as a problem of combinatorial optimization [8], and there have been some studies on solving it using Stochastic Local Search [3,11].

References

1. Adleman, L.: Molecular Computation of Solutions to Combinatorial Problems. Science 266(5187), 1021–1024 (1994)
2. Andronescu, M., Zhang, Z., Condon, A.: Secondary Structure Prediction of Interacting RNA Molecules. J. Mol. Biol. 345(5), 987–1001 (2005)
3. Andronescu, M., Fejes, A., Hutter, F., Condon, A., Hoos, H.: A New Algorithm for RNA Secondary Structure Design. J. Mol. Biol. 336(3), 607–624 (2004)
4. Arita, M., Nishikawa, A., Hagiya, M., Komiya, K., Gouzu, H., Sakamoto, K.: Improving Sequence Design for DNA Computing. In: Proc. of 5th Genetic Evol. Comput. Conf. (GECCO), pp. 875–882 (2000)
5. Arita, M., Kobayashi, S.: DNA Sequence Design Using Templates. New Generation Computing 20(3), 263–273 (2002)
6. Asahiro, Y.: Simple Greedy Methods for DNA Word Design. In: Proc. of 9th World Multi-Conference on Systemics, Cybernetics and Informatics, vol. 3, pp. 186–191 (2005)

7. Deaton, R., Kim, J., Chen, J.: Design and test of noncrosshybridizing oligonucleotide building blocks for DNA computers and nanostructures. Appl. Phys. Lett. 82(8), 1305–1307 (2003)
8. Flamm, C., Hofacker, I., Maurer-Stroh, S., Stadler, P., Zehl, M.: Design of multistable RNA molecules. RNA 7(2), 254–265 (2001)
9. Garzon, M., Deaton, R., Neathery, P., Franceschetti, D., Murphy, R.: A new metric for DNA computing. In: Proc. of the 2nd Genetic Programming Conf., pp. 472–478 (1997)
10. Garzon, M., Phan, V., Roy, S., Neel, A.: In search of optimal codes for DNA computing. In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 143–156. Springer, Heidelberg (2006)
11. Hernandez, R., Hoos, H., Condon, A.: Computational RNA secondary structure design: empirical complexity and improved methods. BMC Bioinformatics 8(1), 34 (2007)
12. Hofacker, I., Fontana, W., Stadler, P., Bonhoeffer, S., Tacker, M., Schuster, P.: Fast Folding and Comparison of RNA Secondary Structures. Monatsh. Chem. 125, 167–188 (1994)
13. Kashiwamura, S., Kameda, A., Yamamoto, M., Ouchi, A.: Two-Step Search for DNA Sequence Design. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 87(6), 1446–1453 (2004)
14. Kawashimo, S., Ono, H., Sadakane, K., Yamashita, M.: DNA sequence design by dynamic neighborhood searches. In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 157–171. Springer, Heidelberg (2006)
15. Kawashimo, S., Ono, H., Sadakane, K., Yamashita, M.: Dynamic neighborhood searches for thermodynamically designing DNA sequence. In: Garzon, M.H., Yan, H. (eds.) DNA 2007. LNCS, vol. 4848, pp. 130–139. Springer, Heidelberg (2008)
16. Lyngsø, R., Zuker, M., Pedersen, C.: Fast evaluation of internal loops in RNA secondary structure prediction. Bioinfomatics 15, 440–445 (1999)
17. Rose, J., Deaton, R., Suyama, A.: Statistical thermodynamic analysis and design of DNA-based computers. Natural Computing 3, 443–459 (2004)
18. Shorteed, M., Chang, S., Hong, D., Phillips, M., Campion, B., Tulpan, D., Andronescu, M., Condon, A., Hoos, H., Smith, L.: A thermodynamic approach to designing struct-free combinatorial DNA word set. Nucl. Acids Res. 33(15), 4965–4977 (2005)
19. Tanaka, F., Kameda, A., Yamamoto, M., Ohuchi, A.: Design of nucleic acid sequences for DNA computing based on a thermodynamic approach. Nucl. Acids Res. 33(3), 903–911 (2005)
20. Tulpan, D., Hoos, H., Condon, A.: Stochastic Local Search Algorithms for DNA Word Design. In: Hagiya, M., Ohuchi, A. (eds.) DNA 2002. LNCS, vol. 2568, pp. 229–241. Springer, Heidelberg (2003)
21. Tulpan, D., Hoos, H.: Hybrid Randomized Neighborhoods Improve Stochastic Local Search for DNA Code Design. In: Proc. Advances in Artificial Intelligence, 16th Conference of the Canadian Society for Computational Studies of Intelligence. LNCS, vol. 671, pp. 418–433. Springer, Heidelberg (2003)
22. Tulpan, D., Andronescu, M., Changf, S., Shortreed, M., Condon, A., Hoos, H., Smith, L.: Thermodynamically based DNA strand design. Nucl. Acids Res. 33(15), 4951–4964 (2005)
23. Winfree, E., Liu, F., Wenzler, L., Seeman, N.: Design and self-assembly of DNA crystals. Nature 394, 539–544 (1998)
24. Zuker, M., Stiegler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucl. Acids Res. 9, 133–148 (1981)

Sequentiality Induced by Spike Number in SNP Systems

Oscar H. Ibarra[1], Andrei Păun[2,3,4,⋆], and Alfonso Rodríguez-Patón[3]

[1] Department of Computer Science University of California, Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu
[2] Department of Computer Science, Louisiana Tech University, Ruston PO Box 10348, Louisiana, LA-71272 USA
apaun@latech.edu
[3] Departamento de Inteligencia Artificial, Faculdad de Informática, Universidad Politécnica de Madrid - UPM, Campus de Montegancedo s/n, Boadilla del Monte 28660 Madrid, Spain
arpaton@fi.upm.es
[4] Bioinformatics Department, National Institute of Research and Development for Biological Sciences, Splaiul Independenţei, Nr. 296, Sector 6, Bucharest, Romania

Abstract. The spiking neural P systems are a class of computing devices recently introduced as a bridge between spiking neural nets and membrane computing. In this paper we consider sequential SNP systems where the sequentiality of the system is induced by a simple choice: the neuron with the maximum number of spikes out of the neurons that can spike at one step will fire. This corresponds to a global view of the whole network that makes the system sequential. We study the properties of this restriction.

1 Introduction

In this paper we consider a new restriction on the rule application (or neuron firing) in SNP systems. Several authors have recently noticed that the maximal parallelism way of rule application (which is widely used in membrane systems) is rather non-realistic in some cases. This fact motivated the consideration of various "strategies" for rule application in membrane systems (or neuron firing in SNPs), for details we refer the interested reader to [3,8,9].

Here we consider the spiking restriction on neurons in the following way: if at any step there are more than one neuron that can spike (according to their pre-defined rules) then only the neuron(s) containing the maximum number of spikes (among the currently "active" neurons) will fire. This is contrasting with the maximal parallel application of the rules case, in which case all the "active" neurons will fire at that step. To exemplify the firing mechanism of the new

⋆ Corresponding author.

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 179–190, 2009.

strategy, let us consider four neurons: n_1, n_2, n_3, n_4 that are the only neurons that can fire at this step (according to their internal rules and the contents of spikes for each of them). In such a case we would find the maximum number of spikes stored in n_1 through n_4, say we have the values 5, 3, 7, 1 thus the neuron n_3 holds the maximum number of spikes, thus n_3 will fire at the next step. After n_3 fires, we update the number of the spikes in the whole system according to this neuron's spiking, and at the next step the neurons n_1, n_2, n_4 together with n_3 will be checked if they can fire (in the new conditions as they may be rendered inactive by an increment in their number of spikes stored). If there is a tie for the maximum number of spikes stored in the active neurons, then all the neurons containing the maximum will fire.

The main motivation behind this spiking strategy is the observation that in a population of cells of the same type (neurons in this case) which have similar types of rules (the spiking rules in our case) one can notice that the cells containing larger numbers of a specific molecule species are more active (spike faster/more frequently) than the cells containing less numbers of the same molecule. Another observation is the fact that the neurons that receive a large number of spikes are more probable to spike than the neurons that do not receive many spikes. The same modeling path was taken also when the *integrate-and-fire* models were defined for neurons, which leads to the neurons that receive more spikes to fire faster than the neurons that receive lower numbers of spikes.

The restriction proposed above makes the spiking of the neurons in the system almost sequential (more than one neuron can spike only in the special case when there is a tie in the number of spikes contained, the two or more active neurons that contain the maximum number of spikes over all the active neurons at that step will spike). Because of this, we will call this application strategy *pseudo-sequential* with respect to maximum. One can also consider the sequential strategy which resolves the ties by choosing for the next spiking neuron nondeterministically one of the neurons containing the maximum number of spikes at that moment (out of the active neurons). This second strategy will be called in the following *sequential* with respect to maximum.

We will consider also the difference between these devices from the point of view of generators versus acceptors, specifically, we notice a major difference between systems with deterministic neurons working as generators as opposed to acceptors. We see that the acceptors are universal whereas the generators are only able to generate one single value (thus are non-universal).

2 Basic Description and Definitions

The spiking neural P systems (in short, SNP) were recently introduced in [4], and then investigated in [11] and [12], thus incorporating in membrane computing [10] ideas from spiking neurons, see, e.g., [1], [5], [6].

We now give a more detailed description of the SNP; such a system is represented as a directed graph consisting of a set of neurons (nodes of a graph) connected by synapses (directed edges of the graph). The neurons send signals

(spikes) along these synapses by means of firing rules, which are of the form $E/a^c \to a;t$, where E is a regular expression, c is the number of spikes consumed by the rule that spikes a single a, and t is the delay between firing the rule and emitting the spike. A rule can only be used if the number of spikes in the neuron are "covered" by expression E, in the sense that the current number of spikes in the neuron, n, is such that a^n is contained in the set $L(E)$. In the time interval between firing a rule and emitting the spike, the neuron is closed/blocked – it does not receive other spikes and cannot fire. After the time interval, the neuron is again open and can again fire and receive other spikes. There are also rules for forgetting spikes, of the form $a^s \to \lambda$ (s spikes are just removed from the neuron). *In this paper, for convenience, we will also refer to the forgetting rules as firing rules.* Starting from a fixed initial distribution of spikes in the neurons (initial configuration) and using the rules in a synchronized manner (a global clock is assumed), the system evolves. A computation is a sequence of transitions starting from the initial configuration. A transition is maximally parallel in the sense that all neurons that are fireable must fire. However, in any neuron, at most one rule is allowed to fire. Details can be found in [4].

An SNP can be used as a computing device in various ways. Here, as in previous papers, we will use them as generators of numbers. We will only consider SNPs with three types of neurons:

1. A neuron is *bounded* if every rule in the neuron is of the form $a^i/a^j \to a;t$, where $j \leq i$, or of the form $a^k \to \lambda$, provided there is no rule of the form $a^k/a^j \to a;t$ in the neuron. Note that there can be several such rules in the neuron. These rules are called *bounded rules*. (For notational convenience, we will write $a^i/a^i \to a;t$ simply as $a^i \to a;t$.)
2. A neuron is *unbounded* if every rule in the neuron is of the form $E/a^k \to a;t$ where the language associated with E is infinite (i.e. we have at least one * or + in the regular expression E). (Again, there can be several such rules in the neuron.) These rules are called *unbounded rules*. As an example, the neuron having the following three rules is unbounded: $a^{2k+3}/a^5 \to a;1$ and $a^{2k+3}/a^6 \to a;2$ and $a^{2k}/a^2 \to a;1$.
3. A neuron is *general* if it can have *general rules*, i.e., bounded as well as unbounded rules. As an example, the neuron having the following three rules is general: $a^{2k+3}/a^5 \to a;1$ and $a^{15}/a^6 \to a;2$ and $a^{2k}/a^2 \to a;1$.

An SNP is bounded if all the neurons in the system are bounded. If, in addition, there are unbounded neurons then the SNP is said to be unbounded. A general SNP has general neurons.

It was recently shown in [2] that a set $Q(\Pi) \subseteq N^1$ is recursively enumerable if and only if it can be generated by a 1-output unbounded SNP Π all of whose unbounded neurons have only one rule – either $a(a)^*/a \to a;0$ or $a(a)^*/a \to a;2$.

We can generalize the SNP by allowing it to produce k outputs. A *k-output SNP Π* has k output neurons, $O_1, ..., O_k$. We say that Π generates a k-tuple $(n_1, ..., n_k) \in N^k$ if, starting from the initial configuration, there is a sequence of steps such that each output neuron O_i generates exactly two spikes a a (the

times the pair a a are generated may be different for different output neurons) and the time interval between the first a and the second a is n_i. Moreover, after all the output neurons have generated their pair of spikes, the system eventually *halts*, in the following sense:

> *Π halts* if it reaches a configuration where all neurons are open but no neurons are fireable. In fact, for the constructions in this paper, this will correspond to the configuration in which all neurons, except for a specified subset R of neurons, have zero spikes, and those in R have exactly two spikes.

The set of all k-tuples generated is denoted by $Q(\Pi)$.

In this paper, we study SNPs operating in sequential and pseudo-sequential mode as described above. Informally, this means that at every step of the computation, if there is at least one neuron with at least one rule that is fireable, we only allow to fire the neuron(s) that is(are) fireable and contain the maximum number of spikes; and for each neuron firing only one spiking rule (nondeterministically chosen) is to be fired.

As defined before in [3], we can consider the effect of the notion of strong sequentiality (or pseudo-sequentiality) and the weak one on the power of such devices.

1. *Case 1*: At every step, there is at least one neuron with a fireable rule (the strong case). We show that:
   (a) We obtain universality for unbounded SNP systems with delays.
   (b) We also get universality even for the case of systems without delays, but in this case the type of rules needs to be extended (a neuron can send more than one spike at a time).
2. *Case 2*: Not every step has at least one neuron with a fireable rule (the weak case). (Thus, the system might be dormant until a rule becomes fireable. However, the clock will keep on ticking.) We will consider this second case in the future studies of such systems. Once should note that even for the restrictive previous case we obtain universality, thus we need to investigate systems with even lower power than in that case.

For the basic definitions and prerequisites we refer the interested reader to [13], [10], and [14]. We will use in the following universality proofs the fact that register machines are universal, but due to the space limitations we will not provide the prerequisite description of the register machines, the reader is referred to [14] as this is a common proof technique.

3 Spiking Neural P Systems

The original definition of spiking P systems was given in [4]; the interested reader can find in the reference above the motivation, basic results etc. Let us recall the basic definition in the following.

A *spiking neural membrane system* (abbreviated as SNP), of degree $m \geq 1$, is a construct of the form

$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, i_0)$, where:

1. $O = \{a\}$ is the singleton alphabet (a is called *spike*);
2. $\sigma_1, \ldots, \sigma_m$ are *neurons*, of the form
   $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:
   a) $n_i \geq 0$ is the *initial number of spikes* contained in σ_i;
   b) R_i is a finite set of *rules* of the following two forms:
      (1) $E/a^c \rightarrow a; d$, where E is a regular expression over a, $c \geq 1$, and $d \geq 0$;
      (2) $a^s \rightarrow \lambda$, for some $s \geq 1$, with the restriction that for each rule $E/a^c \rightarrow a; d$ of type (1) from R_i, we have $a^s \notin L(E)$;
3. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$ (*synapses* between neurons);
4. $i_0 \in \{1, 2, \ldots, m\}$ indicates the *output neuron* (i.e., σ_{i_0} is the output neuron).

The rules of type (1) are *firing* (we also say *spiking*) *rules*, and they are applied as follows. If the neuron σ_i contains k spikes, and $a^k \in L(E), k \geq c$, then the rule $E/a^c \rightarrow a; d$ can be applied. The application of this rule means consuming (removing) c spikes (thus only $k - c$ remain in σ_i), the neuron is fired, and it produces a spike after d time units (as usual in membrane computing, a global clock is assumed, marking the time for the whole system, hence the functioning of the system is synchronized). If $d = 0$, then the spike is emitted immediately, if $d = 1$, then the spike is emitted in the next step, etc. If the rule is used in the step t of the computation and $d \geq 1$, then we have the following setting: in steps $t, t + 1, t + 2, \ldots, t + d - 1$ the neuron is *closed* (this corresponds to the refractory period from neurobiology), so that it cannot receive new spikes (if a neuron has a synapse to a closed neuron and tries to send a spike along it, then that particular spike is lost). In the step $t + d$, the neuron spikes and becomes again open, so that it can receive spikes (which can be used starting with the step $t + d + 1$).

The rules of type (2) are the *forgetting* rules; they are applied as follows: if the neuron σ_i contains exactly s spikes, then the rule $a^s \rightarrow \lambda$ from R_i can be used, meaning that all s spikes are removed from σ_i.

If a rule $E/a^c \rightarrow a; d$ of type (1) has $E = a^c$, then we will write it in the following simplified form: $a^c \rightarrow a; d$.

In each time unit, if a neuron σ_i can use one of its rules, then a rule from R_i *must* be used. Since two firing rules, $E_1/a^{c_1} \rightarrow a; d_1$ and $E_2/a^{c_2} \rightarrow a; d_2$, can have $L(E_1) \cap L(E_2) \neq \emptyset$, it is possible that two or more rules can be applied in a neuron, and in that case, only one of them is chosen non-deterministically. Note however that, by definition, if a firing rule is applicable, then no forgetting rule is applicable, and vice versa.

Thus, the rules are used in the sequential manner in each neuron, but neurons were previously considered to function in parallel with each other. It is important to notice that the applicability of a rule is established based on the *total* number of spikes contained in the neuron. Thus, e.g., if a neuron σ_i contains 5 spikes, and R_i contains the rules $(aa)^*/a \rightarrow a; 0$, $a^3 \rightarrow a; 0$, $a^2 \rightarrow \lambda$, then none of these rules can be used: a^5 is not in $L((aa)^*)$ and not equal to a^3 or a^2. However, if the

rule $a^5/a^2 \to a; 0$ is in R_i, then it can be used: two spikes are consumed (thus three remain in σ_i), and one spike is produced and sent immediately ($d = 0$) to all neurons linked by a synapse to σ_i, and the process continues.

One can associate a set of numbers with Π in several ways. We follow here the idea of [4] and we consider the intervals between the very first two consecutive spikes of the output neuron as numbers computed by a computation. Furthermore, we will consider only halting computations.

Let us consider in the following the sequentiality based on maximum for SNP systems:

Definition 1

1. SNP systems defined as above are working in the max sequentiality manner if (by definition) the system is choosing as the spiking neuron at each step only one of the neurons that can fire (thus the system works in a sequential way), and furthermore, the spiking neuron chosen at each time-step has the maximum number of spikes stored among all the other active neurons in that step.
2. Systems can work in max pseudo-sequentiality manner if (by definition) at each time-step fire all the neurons that store the maximum number of spikes among all the active neurons at that step.

Of course *max sequentiality* is forcing the system to work in a sequential manner as at each step at most one neuron can fire, whereas the *max pseudo-sequentiality* allows two or more neurons to fire at the same time if all those neurons hold the exactly same number of spikes and that number is the highest value of spikes that is stored among all the active neurons at that moment.

We pass now to give the results of the paper.

4 Max Sequentiality Result

We will start the description of results of these systems by giving the first theorem about systems based on strongly max sequentiality with delays. We will sow the universality of such systems as opposed to the result in [3] where the strongly sequential sequential were shown to be not universal.

Theorem 1. *Max sequentiality with delays for unbounded SNP systems even in the strongly sequential mode is universal.*

Proof. We will show that any register machine can be simulated by a system working in a Max sequentiality manner, with rules using delays. As we will see in the following, we will need the delays only for the ADD rules in the register machine.

Let us give a brief description of the construction: In the system we will have neurons associated with each label in the program code of the register machine, also the registers will be modeled by neurons holding $2n$ spikes for the value n being stored in the register. Thus the ADD module will increase by 2 the number of spikes stored in the neuron associated with the register

r (effectively incrementing the register r in the simulation) and then choose nondeterministically a new instruction to execute out of the two possibilities given by the ADD instruction.

In the following we give the neurons necessary to simulate an $l_1 : (ADD(r)l_2, l_3)$ rule:

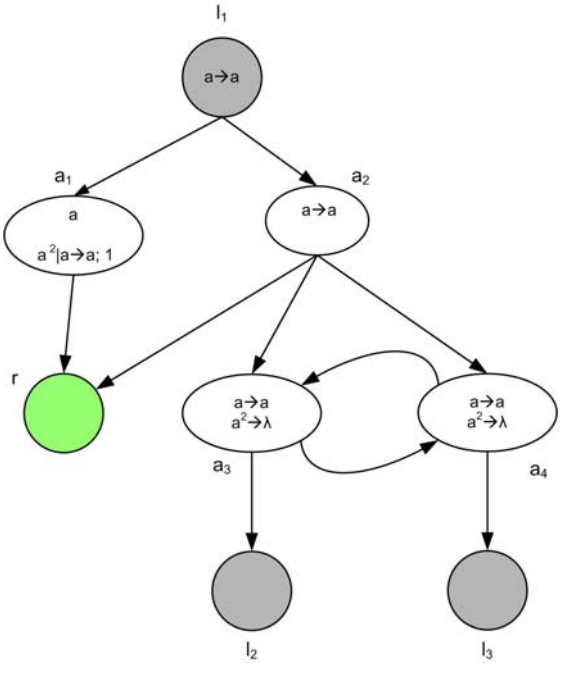


Fig. 1. The addition module for $l_1 : (\text{ADD } r,\ l_2,\ l_3)$

The module works as follows: the neuron l_1 spikes, signalling that the instruction l_1 is being executed, then the neurons a_1 and a_2 are activated, since a_1 has at this moment two spikes and a_2 only one, a_1 fires first, but has a delay of size one associated with its rule, at the next step a_2 fires (since at that moment is the only neuron fireable), making the spikes from a_1 and a_2 to arrive at the same time in the neuron r. We will see later that the neurons of type r can only fire when they hold an odd value, thus receiving two spikes keeps the neuron r inactive. At the same time a_2 sends a spike towards also neurons a_3 and a_4. The job of the neurons a_3 and a_4 is to choose nondeterministically which register rule to activate next: l_2 or l_3. This is achieved by the fact that when receiving the spike from a_2, both a_3 and a_4 are activated, both have exactly one spike at this moment, so we need to choose nondeterministically one to fire. Depending of the choice, the corresponding instruction l_2 (for a_3) and l_3 (for a_4) is activated. This is done in two steps: a_3 (or a_4) fires, then it sends to the other neuron a_4

(or a_3) another spike, making the forgetting rule applicable, and another spike to the neuron simulating the label of the next instruction. Since the a_4 (or a_3) neuron holds two spikes versus one spike for the label neuron, we first apply the forgetting rule, and then we continue to simulate the work of the register machine.

We will now give the module simulating the SUB instruction from the register machine. We show how are we simulating all the ADD and SUB instructions in general.

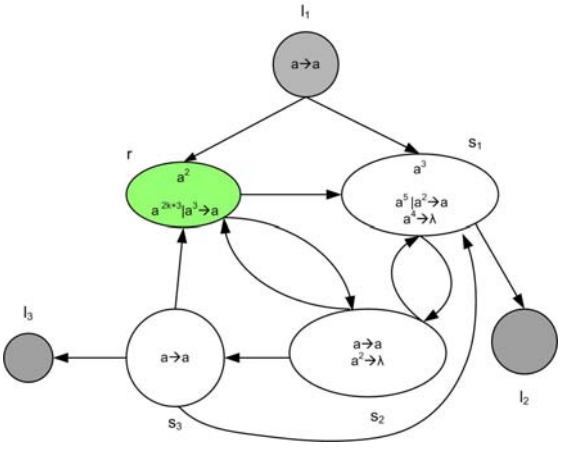


Fig. 2. The subtract module for l_1 : (SUB r, l_2, l_3)

When the neuron l_1 fires, it sends two spikes, one in the neuron r (modeling the register that is decremented or checked for zero) and one in the neuron s_1. We note that we will start with two spikes in the neurons modeling the registers (we will take care of the correct counting in the finalizing module), we also start with three spikes in the neuron s_1 thus at the next step the neuron s_1 contains exactly 4 spikes, whereas the neuron r contains exactly $2n + 3$ spikes, where n is the contents of the register r in the counter automaton.

We have now two possible cases:

Case I: if the register r is empty, it means that the neuron r holds exactly 3 spikes. Since these are the only two neurons that can fire at this moment (r and s_1), then s_1 will execute first since it has four spikes (one more than r). This means that all four spikes in neuron s_1 are deleted through the forgetting rule, then at the next step r spikes sending one spike back to s_1 and activating s_2. At the next step s_2 fires sending one more spike in s_1 and sending another one back to r which was empty. At the next step s_3 fires also replenishing the two initial spikes in r and the third spike in s_1. This means that we reached a configuration similar to the original configuration when l_1 spiked, and now l_3 is activated (since the register was empty).

Let us consider the case when the register r would be non-empty:

Case II: then r would hold $2n + 3$ spikes, with $n \geq 1$, thus r will hold at least 5 spikes, more than the four held by s_1. Thus r spikes sending one spike to s_1 and another spike to s_2. At the next step s_1 will have 5 spikes as opposed to s_2 that holds only one, thus s_1 spikes removing two spikes. That means that at the next step we will have s_1 holding 3 spikes and being inactive, s_2 holding 2 and l_2 holding one. Thus next s_2 will forget its two spikes, making the configuration as before and then the simulation can continue with l_2.

It is clear that the rules from the register machine are correctly simulated by the modules presented above. What remains is the finishing stage in which the output register is read and processed in our setting:

Without loss of generality we can assume that the output register is never decrementing (the register machine can be easily changed by adding another register and a couple of rules that would copy the contents of the output to the new register that would never decrement).

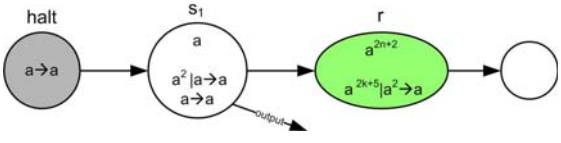


Fig. 3. The halting module

When we activate the halting label in the register machine we send a spike in the output neuron (the neuron s_1 in the picture above). Thus at the next step s_1 spikes (being the only active neuron), then both r and s_1 are active. Thus the one holding the maximum number of spikes will fire. One can notice that in r we are deleting exactly 2 spikes each time, thus s_1 will let r spike as long as the register r (in the register machine) is non-empty, and at each time step two more spikes are removed (thus the register r is decremented by one each clock cycle). Thus the second time that s_1 spikes would have been exactly n clock cycles after the first spike, making the whole system to correctly simulate the work of the starting register machine. This completes the proof. □

If we consider the case of extended systems (where the neurons can send more than one spike through the synapse in one clock cycle), then we can easily remove the delay that appears in the *ADD* module.

Theorem 2. *Extended systems with max sequentiality: unbounded and no delays (thus strongly sequential) are universal.*

Proof. We change the *ADD* module in the following way: change the rule in a_1 to: $a^2|a \to a^2$ and remove the synapse between a_2 and r. Everything else remains the same. □

In the following section we will consider an even more realistic way of spiking for the neurons in the system: if there are ties for the maximum number of spikes stored in active neurons, then all the active neurons holding the maximum number of spikes will fire.

5 Max Pseudo-Sequentiality

We start by noticing that there is no nondeterminism at the level of the system: from each configuration to the next, we know for sure which neurons fire (this was not the case with the max sequentiality discussed in the previous section, for example one can refer to the work of neurons a_3 and a_4 in figure 1). Since the SUB and $HALT$ modules given for the Theorem 1 have always a single neuron holding the maximum, it only remains to describe the ADD module for this case. We will see that we can give a system that does not use delays as the previous ADD module was the only one using this feature.

Theorem 3. *For systems working in the Max pseudo-sequentiality mode, we have universality of such systems without delays for unbounded SNP systems.*

Proof. As mentioned above, we will describe the ADD module without delays:

We start with the neuron l_1 firing, then the neurons a_1 and a_2 are activated as they both receive a spike from l_1 since there is a tie of the maximum spikes contained in the active neurons, both a_1 and a_2 fire, sending exactly two spikes in r and another two spikes in a_3. We have incremented the register 1, so what remains now is the nondeterministic jump to either l_2 or l_3. This is done with

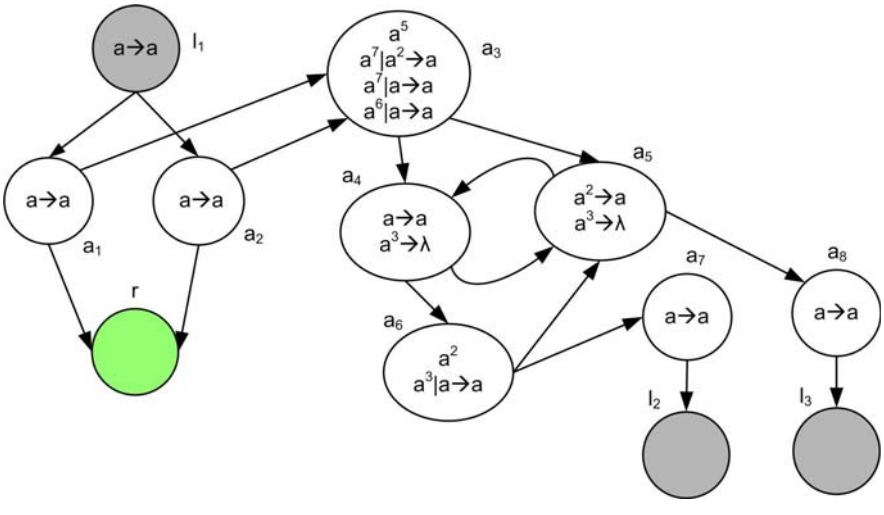


Fig. 4. The addition module for l_1 : (ADD r, l_2, l_3)

the help of neurons a_3 through a_8. Because the contents of r will have an even number of spikes, it is inactive at the next step, thus only a_3 can fire. At this moment we will have a nondeterministic choice between firing using the rule $a^7|a^2 \to a$ or $a^7|a \to a$. The choice is whether the neuron will fire once (with $a^7|a^2 \to a$) or twice (through $a^7|a \to a$, and then $a^6|a \to a$).

Case I: let us assume that a_3 fires the rule $a^7|a^2 \to a$, then at the next step only a_4 is active, and after it fires, both a_5 and a_6 are active, but a_5 holds 2 spikes and a_6 holds 3, so at the next step a_5 (with 3 spikes) and a_7 (with one spike) are active, so a_5 fires, erasing all its spikes and then a_7 activates the new instruction to be executed, l_2. Let us consider the other case.

Case II: a_3 fires the rule $a^7|a \to a$, then at the next step both a_3 (with 6 spikes) and a_4 (with 1 spike) are active, then a_3 spikes once more, making a_5 the only active neuron, at the next step a_4 with 3 spikes is activated, together with a_8. We first forget the spikes from a_5, and then a_8 activates the label l_3.

Thus we correctly simulated the increment instruction on register r. The SUB and $HALT$ modules remain the same. This completes the proof. □

An interesting observation is the fact that if one considers deterministic neurons (neurons in which the regular languages associated with each rule are disjoint), then such a system cannot produce nondeterminism. Thus we have the following result:

Theorem 4. *A system of deterministic neurons working in a maximal pseudo-sequential manner (as a generator) is non-universal.*

Proof. We notice that we cannot have nondeterminism at the level of neurons: they are deterministic. Since the determinism at the level of the system is also removed by the pseudo-sequentiality, then each such system will generate at most one value for each starting configuration. □

This previous result contrasts the fact that if such devices are used in an acceptor mode, then they are universal:

Theorem 5. *A system of deterministic neurons working in a maximal pseudo-sequential manner (as an acceptor) is universal.*

Proof. We start with the register r containing exactly $2n+2$ spikes (for the value n to be accepted or rejected from the set). Then using the ADD and SUB given above one simulates correctly the instructions in the register machine. Thus if we reach the neuron with the label $HALT$, we should accept the value n, whereas if we do not reach the neuron $HALT$, then we should reject the value n. □

6 Final Remarks

We plan to continue the investigation of this special type of sequentiality; we have already obtained results about the converse, min-sequentiality that will be included in a journal version of the article. Another direction would be to consider

SNP systems that are stochastic with respect to the next spiking neuron in the following sense: each active neuron is having a probability of spiking that increases with the number of spikes stored in the respective neuron. We will pursue more avenues of research in this direction as we believe that this model can be more relevant to an experimental implementation of such a stochastic system.

Acknowledgements

We gratefully acknowledge support in part from NSF Grants CCF-0430945, CCF-0523572 and CCF-0524136, support from LA BoR RSC grant LEQSF (2004-07)-RD-A-23, support from INBRE Program of the NCRR (a division of NIH), support from CNCSIS grant RP-13, support from CNMP grant 11-56 /2007, support from Spanish Ministry of Science and Education (MEC) under project TIN2006-15595, and support from the Comunidad de Madrid (grant No. CCG07-UPM/TIC-0386 to the LIA research group).

References

1. Gerstner, W., Kistler, W.: Spiking Neuron Models. Single Neurons, Populations, Plasticity. Cambridge Univ. Press, Cambridge (2002)
2. Ibarra, O.H., Păun, A., Păun, G., Rodriguez-Paton, A., Sosik, P., Woodworth, S.: Normal forms for spiking neural P systems. Theor. Comput. Sci. 372(2-3), 196–217 (2007)
3. Ibarra, O.H., Woodworth, S., Yu, F., Păun, A.: On spiking neural P systems and partially blind counter machines. In: Calude, C.S., Dinneen, M.J., Păun, G., Rozenberg, G., Stepney, S. (eds.) UC 2006. LNCS, vol. 4135, pp. 113–129. Springer, Heidelberg (2006)
4. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae 71(2-3), 279–308 (2006)
5. Maass, W.: Computing with spikes. Special Issue on Foundations of Information Processing of TELEMATIK 8(1), 32–36 (2002)
6. Maass, W., Bishop, C. (eds.): Pulsed Neural Networks. MIT Press, Cambridge (1999)
7. Minsky, M.: Computation – Finite and Infinite Machines. Prentice Hall, Englewood Cliffs (1967)
8. Păun, A., Popa, B.: P Systems with Proteins on Membranes. Fundamenta Informaticae 72(4), 467–483 (2006)
9. Păun, A., Popa, B.: P systems with proteins on membranes and membrane division. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 292–303. Springer, Heidelberg (2006)
10. Păun, G. (ed.): Membrane Computing – An Introduction. Springer, Berlin (2002)
11. Păun, G., Pérez-Jiménez, M.J., Rozenberg, G.: Spike trains in spiking neural P systems. International Journal of Foundations of Computer Science 17(4), 975–1002 (2006)
12. Păun, G., Pérez-Jiménez, M.J., Rozenberg, G.: Infinite spike trains in spiking neural P systems (submitted, 2006)
13. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, vol. 3. Springer, Berlin (1997)
14. The P Systems Web Page, http://psystems.disco.unimib.it

Author Index