Towards Domain-based Sequence Design for DNA Strand Displacement Reactions

David Yu Zhang

California Institute of Technology, Pasadena, CA, USA
dzhang@dna.caltech.edu

Abstract. DNA strand displacement has been used to construct a variety of components, devices, and circuits. The sequences of involved nucleic acid molecules can greatly influence the kinetics and function of strand displacement reactions. To facilitate consideration of spurious reactions during the design process, one common strategy is to subdivide DNA strands into domains, continuous nucleic acid bases that can be abstracted to act as a unit in hybridization and dissociation. Here, considerations for domain-based sequence design are discussed, and heuristics are presented for the sequence design of domains. Based on these heuristics, a randomized algorithm is implemented for sequence design.

Introduction

DNA strand displacement is a process through which a single-stranded DNA molecule (*strand*) reacts with a multi-stranded DNA *complex* to release another DNA strand (Fig. 1AB). Typically, strand displacement is facilitated by *toeholds*, short complementary single-stranded domains that act to colocalize the invading strand with the complex. The thermodynamics of the toehold region can be calculated based on sequence [1] [2] [3], and largely determine the kinetics of the strand displacement reaction if the invading strand does not possess significant secondary structure [4].

DNA strand displacement has been used to construct a number of dynamic DNA devices, including logic gates and circuits [5] [7] [6] [8] [9], catalytic reactions and networks [11] [10] [12] [13] [14] [15] [16], and nanoscale motors and walkers [17] [18] [19] [20]. These devices' mechanisms are based on Watson-Crick complementarity, and are expected to function for a wide variety of DNA sequences. Nevertheless, the kinetics of these devices' function are slowed if the sequences possess significant secondary structure (Fig. 1C) [21] [22]. As strand displacement-based DNA devices become more reliable, many different such devices will be integrated to construct complex reaction networks with more advanced functions. Many different DNA molecules must thus perform their function simultaneously without interfering with each other. Thus, an automated method for DNA sequence design is needed for strand displacement-based DNA devices.

Although excellent algorithmic methods for DNA sequence design have been presented [23] [24] [25] [26] [27] [28], these methods generally maximize the probability of DNA strands forming desired structures and complexes at equilibrium, such as in the case for many DNA self assembly applications. For strand displacement-based devices and networks, thermodynamics-based methods of sequence design are not guaranteed to provide satisfactory sequences, because they neglect the consideration of kinetic pathways between DNA strand and complex states. In Fig. 2, for example, the strand shown in Fig. 2A may proceed through strand displacement slower than the strand in Fig. 2B, despite being the former possessing a minimum free energy (mfe) structure with standard free energy (ΔG°) closer to 0.

This paper presents a domain-based approach to sequence design of strand displacement-based reaction, networks, and devices. Domains are consecutive bases that serve as functional units in binding and dissociation (Fig. 1A), providing a useful abstraction for the design DNA devices. The sequences of involved DNA strands are obtained by concatenating the sequences of the strand's constituent domains. This domain-based approach to sequence design is simple and generalizes to

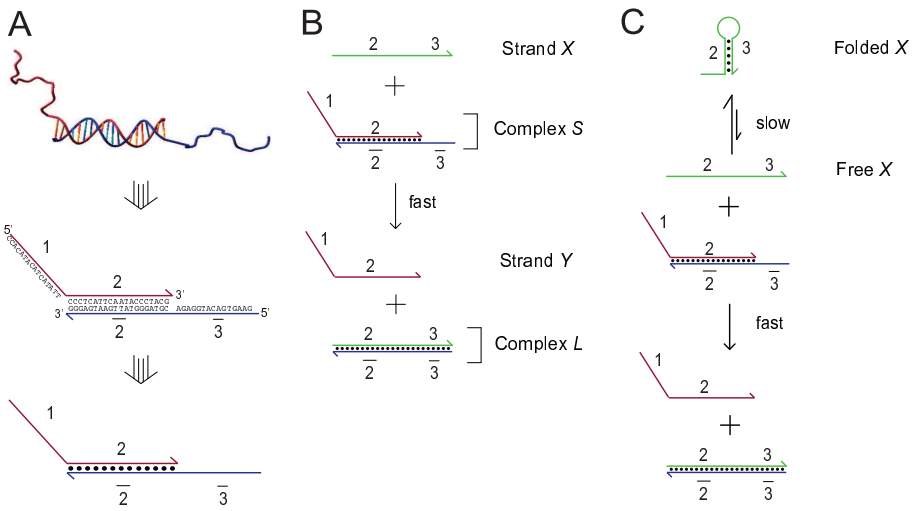

Fig. 1: DNA strand displacement **(A)** DNA abstraction. DNA strands and complexes are represented by directional lines, with the hook denoting the 3' end. DNA strands can be functionally abstracted into domains, consecutive bases of DNA that act as a unit in binding and dissociation. Domains are represented by numbers; a barred domain denotes a domain complementary in sequence to the unbarred domain (e.g. domain $\bar{2}$ is complementary to domain 2). Domain 2 is referred to in this paper as a "branch migration domain" and domain $\bar{2}$ is referred to as a "complement domain." Domains 3 and $\bar{3}$ are referred to as "toehold domains." **(B)** DNA strand displacement. **(C)** Secondary structure could hinder the kinetics of strand displacement. Strand X needs to unfold into a free state before it can branch migrate.

many different possible DNA complexes and reactions, but is limited in its ability to eliminate undesirable spurious hybridization between DNA strands, particularly at the interface between different domains.

The other extreme, sequence design based on thermodynamic and kinetic analysis of all possible intermediates in strand displacement reactions, would potentially allow a rigorous method for generating optimal sequences for any reaction network. However, such a sequence designer would require significantly more computational resources, and is not available at present time. Additionally, although the thermodynamics of most DNA structural motifs have been carefully characterized over the past 20 years [1] [2] [3], two particular ones relevant to the analysis of multi-stranded intermediate complexes remain elusive: the energetics of pseudo-knotted complexes [29] [30] [31] and coaxial stacks [32] [33] [34] [35]. This incompleteness of DNA thermodynamic data suggest that even these more complex methods may not generate truly "optimal" sequences.

Considerations

In this section, considerations for domain-based sequence design are presented. These considerations are considered generally relevant to sequence design for DNA constructions involving strand displacement.

Avoidance of long continuous regions of spurious hybridization. Long continuous regions of spurious binding in the branch migration domain should be avoided with priority over several shorter regions of spurious binding, even if the latter result in a more negative minimum free energy structure for the domain or strand. The reason for this is because branch migration is a sequential

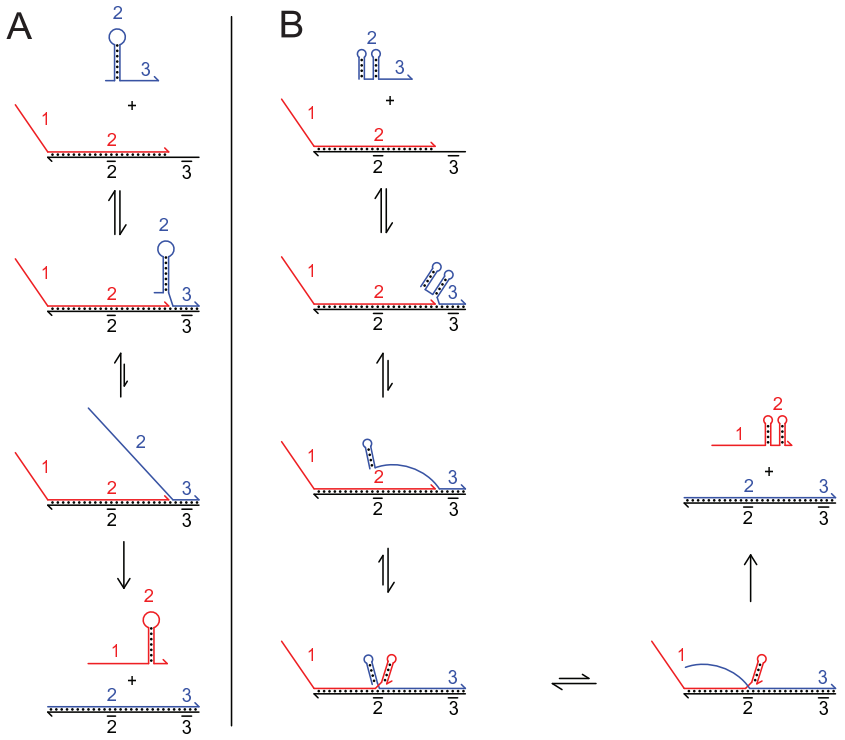

Fig. 2: Failure of thermodynamics-based sequence design. The ΔG° (standard free energy) of the folded domain 2 is more negative in **(B)** than in **(A)**, but the sequential nature of branch migration means that the strand displacement reaction involving **(B)** could be faster than that of **(A)**.

process: the binding energy of the strongest continuous region of hybridization likely determines the activation energy of the branch migration process.

When multiple different helices are present as in Fig. 2B, only the first of them needs to spontaneously open in order for branch migration to initiate. In contrast, when long continuous regions of spurious hybridization exist, branch migration over each of these structured bases is energetically unfavorable, and the kinetics of the overall branch migration process will be slowed exponentially in the energy of the binding region. To take a numerical example, if the hairpin shown in Fig. 2A has energy of -10 kcal/mol, while each of the hairpins in Fig. 2B has energy -7 kcal/mol, the kinetics of the strand displacement reaction may be a factor of 100 faster for reaction in Fig. 2B at room temperature, despite having an strand mfe of -14 kcal/mol.

Furthermore, because the displaced domain is the same sequence as the branch migration domain, the displaced portion of the domain will start forming the structures that spontaneously opened in branch migration domain. For example, in the bottom-left panel of Fig. 2B, the red strand forms the hairpin on its 3' end, analogous to the 3' hairpin of the blue strand. The formation of secondary structure by the displaced strand thus can facilitate branch migration.

There are other possible mechanisms for branch migration through highly structured domains of DNA, such as through 4-way branch migration. However, the author expects that most sequences generated with intent to avoid spurious hybridization will exhibit low enough amounts of spurious hybridization that the dominant pathway for branch migration will be through spontaneous dissociation of spurious hybridization.

Interaction and crosstalk: domain concentration effects. We define a "branch migration domain" to be a domain which competes to binding to a "complement domain" via branch migration. In Fig. 1, the 2 domain is a branch migration domain, while the $\bar{2}$ domain is a complement domain. For consistency, branch migration domains are all represented by unbarred numbers.

In strand displacement reactions, there is a necessary excess of branch migration domains over complement domains. For example, there are two copies of domain 2 in Fig. 2, but only one copy of domain $\bar{2}$. This excess manifests as a very large difference in the concentrations of single-stranded branch migration domains and complement domains. Almost all of the complement domains will be double-stranded at all times, while the branch migration domains will be single-stranded in significant concentrations.

Spurious hybridization occurs only between two single-stranded domains; this causes potential spurious hybridization involving a complement domain to be significantly less likely, and that between two complement domains to be nearly non-existent. Consequently, sequence design should primarily seek to minimize spurious binding between different branch migration domains (here refered to as *interactions*, Fig. 3B). Spurious hybridization between branch migration domains and complement domains (here refered to as *crosstalk*, Fig. 3B) is also undesirable insofar as two different domains may non-specifically displace each other in binding to their respective complements. However, mismatches destabilize the thermodynamics and also significantly impede the kinetics of branch migration [13] [36]. Because most domain sequences designed will differ from each other by at least a few bases, crosstalk is usually not a problem in practice.

Of interactions, self-interactions (wherein a domain significantly hybridizes to an identical copy of itself) can be considered the most problematic for two reasons: First, intra-domain and intramolecular hybridization are entropically favored because of high local concentration. Second, assuming the thermodynamics of all interactions to be equal, dimerization is likely to be more prevalent than other interactions, because the concentration of a single-stranded domain correlates perfectly with itself (while different domains may not be single-stranded in high concentration at the same time in dynamic circuits).

Thus, sequence design for branch migration domains should avoid domain sequences with self-interactions with highest priority, other interactions with secondary priority, and minimize crosstalk only insofar as it does not hurt the previous two criteria. For sequence design of toehold domains, crosstalk is a larger consideration on the par of interactions.

Minimizing guanine frequency in branch migration domains. Of the four canonical DNA nucleotides, guanine (G) stands out in being the most problematic with regards to sequence design for synthetic biology purposes, both because it is promiscuous (binding to thymine (T) nearly as strongly as adenine (A) [1]) and because it can form guanine quartets [37], a quadruplex structure based on non-Watson Crick binding, usually requiring at least 4 consecutive G's.

For these reasons, it is desirable to minimize the total concentration of guanine nucleotides used in the system. As pointed out previously, branch migration domains are necessarily in higher concentrations than complement domains–this leads to a natural strategy of minimizing the frequency of G's in branch migration domains. This strategy is further desirable because both the branch migration domains and the complement domains are now constructed with only 3 of the 4 nucleotides (C/A/T for the former, G/A/T for the latter), which drastically reduces the chance of self-interactions.

The idea of using only some of the four nucleotide bases in designing DNA sequences was first proposed by Mir [39], who suggested that restricted alphabets may practically avoid nonspecific hybridization, based on experimentally observed hybridization behavior of various sequences [38]. Experimental use of DNA strands with restricted alphabets for DNA nanotechnology purposes is relatively recent [10] [4] [40].

Avoidance of long A/T and long G/C regions. For certain applications, it may be desirable to avoid long uninterrupted stretches of weak (A/T) bases or strong (G/C) bases. Long continuous

regions of weak A/T bindings will melt at significantly lower temperatures and breathe significantly more than those with more mixed base distributions. Additionally, the hybridization rate constant of domains with only A/T bases have been reported to be an order of magnitude lower than that of one with a uniform distribution of G/C/A/T bases [4].

On the other hand, long continuous regions of strong G/C bindings are more likely to be spuriously hybridized to other strands and domains, because the strong G/C binding will counteract the destabilizing influences of DNA bulges and mismatches [1]. Additionally, long continuous regions of C/G sequence are more likely to adopt Z-DNA configurations [41]. Finally, branch migrate are likely to proceed more slowly in G/C rich regions due to their stronger base stacking thermodynamics, which may in turn necessitate stronger toehold domains for fast strand displacement [4].

Breathing near the end of helices One frequent concern in the design and construction of strand displacement reactions and networks is blunt end strand exchange, strand displacement in the absence of single-stranded toehold domains. The rate constant of blunt end strand exchange has been reported to be on the order of 1 M^{-1} s^{-1} for multiple sequences [42] [43] [4]. This rate constant could be significantly higher if the branch migration domain is terminated with A or T.

The mechanism for blunt end strand exchange is postulated as the following: The base pairs at the end of a double-stranded branch migration domain "breathes," temporarily unbinding despite the bound state being more favorable. Any strands possessing the branch migration domain that happens to be in the local vicinity effectively has a few bases of toehold until the ends of the complex re-hybridizes. Terminating branch migration domains with A/T base pairs thus is likely to increase the rate constant of blunt end strand exchange because A/T bases are weaker than G/C ones. Furthermore, measured thermodynamic parameters show that DNA helices are destabilized when closed by an A-T base pair (by 0.15 kcal/mol at room temperature) [1], in addition to the weaker binding thermodynamics.

Consequently, it is desirable for all branch migration and complement domains to start and end with G or C nucleotides. For reasons given previously, it is recommended that branch migration domains start and end with C's.

Interface between domains. One problem specific to domain-based sequence design is the interface between domains. Typically, a stretch of binding typically requires 3-4 consecutive complementary bases in order to be stable, and such binding will be not be visible to the sequence design software if the bases span multiple domains. Rather than abandoning domain-based sequence design altogether, it should be possible to weight spurious hybridization near the ends of the domains appropriately so that concatenated domains are unlikely to form long spurious hybridization regions at their interface.

Consideration of the potential interface problems through calcuating the interactions and crosstalks for all pairwise concatenations of domains is not recommended for two reasons: First, this causes a runtime slowdown quartic in the number of domains (N^4 pair-wise folds needed for N^2 pair-wise concatenations, where N is the original number of domains), negating the speed advantage of low-level domain-based design software. Second, many of the possible pairwise concatenations will not actually be present in solution, and optimizing the interaction and crosstalk potential of these non-existent domain combinations will actually worsen the crosstalk and interaction problems of the strands that do exist.

Implementation

The Domain Design (DD) software presented here employs heuristics to quantify the considerations discussed in the previous section; these heuristics are used to generate an overall "score" for each domain, the worst (maximum) of which is the global score for a set of domains. Simply put, the score is a metric that roughly correlates with the likelihood that the kinetics of strand displacement reactions deviate from predictable models [4]. The overall score for a set of domains is simply the

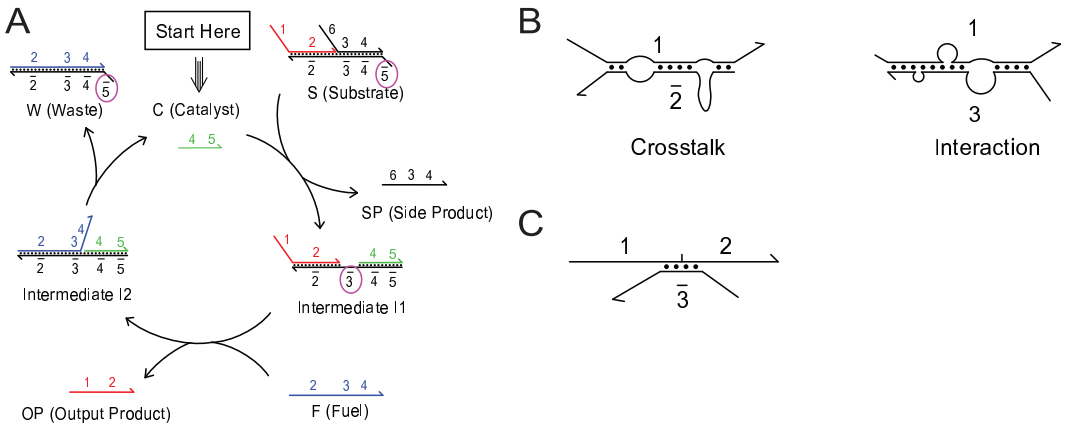

Fig. 3: Spurious partial hybridization between unrelated domains. **(A)** Asymmetry of single-stranded prevalence. In the catalytic reaction cycle (adapted from Zhang et al. [10]), the only barred domains that appear in single-stranded forms are $\bar{3}$ and $\bar{5}$, the toeholds. **(B)** Accordingly, we draw a distinction between "crosstalk" and "interaction," with the former denoting spurious partial hybridization between an unbarred domain and a barred domain, while the latter refers to the hybridization between two unbarred domains. Interactions are likely to have greater effects on kinetics than crosstalk. **(C)** One shortcoming of domain-based sequence design are the possibility of crosstalk and/or interactions at the interface between two domains.

maximum (worst) of the domain scores. DD's algorithm performs sequence design by starting with a random set of sequences, and continually attempting to mutate these sequences to achieve improved (lower) scores.

DD assumes that the domains designed are branch migration domains or the unbarred toehold domains, and evaluates crosstalk potential by automatically generating the complements to all designed domains. One good design strategy may be to design all the toehold domains first, and subsequently design the branch migration domains (with the toehold domain sequences locked–see User Interactivity: Base locking).

The algorithm uses a number of scoring parameters, some of which are hard-coded and can be turned on or off at the user's discretion. The values of the hard-coded parameters are by no means guaranteed to be optimal or even close to optimal–these represent only the author's best guess at good parameter values for designing 10 or fewer different domains, each of length between 5 and 30 nucleotides. The pseudocode for the Domain Design software is given in Fig. 4.

Score calculation. The score of a domain is computed as the sum of the domain intrinsic score and the worst of its crosstalks and interactions.

The domain intrinsic score accounts for score penalties based only on the sequence of the domain, rather than potential spurious hybridization. Domains with four consecutive G's or C's receive a +50 to score, so as to strongly discourage the formation of G quartets. Domains with six consecutive G/C nucleotides or six consecutive A/T nucleotides receive a +20 to score, if the user elects to turn on the option for avoiding long regions of G/C binding and long regions of A/T binding. Finally, the user-defined "importance" of the domain is also added to the score (see User Interactivity: Domain importance).

The crosstalk and interaction scores are evaluated similarly, by evaluating every potential way that the two domains can spurious hybridize and identifying the way that yields the highest score. The score of each way of binding is calculated as thus: each G-C base pair contribute +2 score and each A-T pair contribute +1 score. Base pairing of x consecutive nucleotides without intervening bulges or mismatches gain a further score of $+2^{x-4}$ for $x \geq 5$, thus strongly discouraging the

formation of long unbroken stretches for spurious hybridization. Mismatch and bulge structures between stretches of hybridization each contribute -3 score, with an additional -0.5 score for every base in the bulge or mismatch in excess of 1. The scores of hybridization segments starting at the first base of a domain or ending with the last base of a domain are increased by +3, to reduce the potential for interface problems when concatenating domains.

The difference in DD's treatment of crosstalk, interactions, and self-interactions is that each's score is modified linearly. Self-interactions receive a +5 to score, interactions default to no adjustment, and crosstalk scores are divided by 2 and then a -10 to score is applied. Thus, DD places higher priority on self-interactions and lower priority on crosstalk.

In practice, DD will optimize the sequences of a set of 10 domains, each 20 nt long, in about a minute to the point where the overall score of the system is between +10 and +15.

Algorithm. DD calculates interaction and crosstalk score of a pair of domains by using a dynamic programming algorithm similar to those used mFold, DINAMelt, and PairFold [44] [3] [45]. See source code for details.

In every run of the main loop, DD attempts to improve the overall score by mutating one of the domains. If the overall score was improved through the mutation, DD keeps the mutation. If the overall score was unchanged through the mutation, DD keeps the mutation with 0.2 probability. If the overall score was worsened through the mutation, DD discards the mutations. At the user's option, DD will either randomly select one of the domains for attempted mutation with uniform probability, or will target the domain currently with the worst score with probability $\frac{1}{3} + \frac{2}{3N}$, where N is the number of domains designed. Targeting the domain with worst score for mutations is expected to improve the speed of the software.

The number of bases attempted to be mutated is roughly exponentially distributed, with $\frac{1}{2}$ probability of mutating 1 base, $\frac{1}{4}$ of mutating 2 bases, etc., up to 10 bases or M (the number of bases in the domain), which ever is smaller. The simultaneous mutation of multiple bases is thought to prevent the domain sequences from entering local score optima.

Each base attempted to be mutated is replaced by a random base drawn from the bases allowed to occur within the domain (see User Interactivity: Base Composition). This means that there is some probability that an attempted mutation does not actually change the base, because the new base happens to be the same as the old. At the user's option, the mutation process can be biased against the incorporation of G, so that only 4% of attempted mutations result in a G (assuming G is an allowed base in the first place).

Time and space complexity. The time complexity of the algorithm can be estimated easily. Define the problem to be the design of N different domains, each of length M. After every mutation attempt, the software must update the interaction and crosstalk scores of all domains with the mutated domain, which involves checking $O(N)$ interactions and crosstalks. Calculating an interaction or crosstalk score using dynamic programming requires $O(M^2)$ time. Thus, the algorithm has time complexity $O(NM^2)$ per mutation attempt.

The number of total mutation attempts needed to in order to grant each domain a fixed expected number of mutations attempts will scale linearly with N. Thus, DD requires $O(N^2M^2)$ time to reduce the overall score to decent levels.

In addition to the $O(NM)$ space needed to store the sequences of the domains, there are two major contributions to the space complexity of the algorithm. First, during the evaluation of a crosstalk or interaction score, $O(M^2)$ space is needed for the dynamic programming algorithm. Second, there are $O(N^2)$ pairwise crosstalk and interaction scores between the N domains; these scores need to be stored in order to allow fast score updating (not storing the scores worsens the time complexity by a factor of N). Thus, the overall space complexity of the algorithm is $O(N^2 + NM + M^2) = O(N^2 + M^2)$.

```
FOR i = 1:N
  FOR j = 1:N
    Score(i,j) = Domain_intrinsic_score(i) + Max(Crosstalk(i,j), Interaction(i,j))
  END FOR
  Domain_Score(i) = Max_over_j(Score(i,j))
END FOR
Global_score = Max_over_i(Domain_Score(i))

WHILE (TRUE)
  k = Random(1,N)
  New_domain(k) = Mutate(Domain(k))

  FOR i = 1:M
    New_Score(i, k) = Domain_intrinsic_score(i) + Max(Crosstalk(i,k), Interaction(i,k))
    New_Score(k, i) = New_Score(i,k)
    IF (i != k)
      Domain_Score(i) = Max_over_j(Score(i,j))
    END IF
  END FOR
  Domain_Score(k) = Max_over_j(Score(k, j)
  New_global_score = Max_over_i(Domain_Score(i))

  IF (New_global_score < Global_Score)
    Domain(k) = New_domain(k)
    FOR i = 1:M
      Score(i,k) = New_score(i,k)
      Score(k,i) = New_score(k,i)
    END FOR
    Global_Score = New_global_score
  END IF

  IF (Keyboard_input())
    Prompt_user_menu()
  END IF
END WHILE
```

Fig. 4: Pseudo-code for the Domain Design software.

User Interactivity

The Domain Design software (DD) presented here was written with the intention of being a tool for helping the informed DNA engineer design sequences, rather than being an arbitrator of sequences. Accordingly, Domain Design strives to maximize the ease with which the user can modify and adjust the sequence design process: The user may pause the sequence optimization process at any time to tweak sequences as well as a number of design parameters, including the ones listed below. User interactive sequence design is considered advantageous because the user may be able to identify sequence problems that the software is not able to detect.

Base locking. DD acknowledges that the user may possess certain constraints or preferences in the design of sequences. For example, the user may require the promoter sequence for the T7 RNA polymerase [46] [47] or a deoxyribozyme sequence [48] [49] [50] to be present at a given location. To this end, the user can manually change the sequences of any domain, and can furthermore lock part or all of a domain sequence from being mutated by the software. Locked bases are visually displayed in red (Fig. 5A).

Domain importance. Different domains serve different purposes in a reaction network, and the frequency of a domain being single-stranded also varies among domains. For example, toehold domains are required to colocalize the reactants of the strand displacement reaction, and interactions/crosstalk involving toeholds may slow the kinetics of strand displacement much more than similar interactions/crosstalk between branch migration domains. Consequently, sequence-based interactions and crosstalk in the different domains need to be weighed differently. A rigorous and

justified method for weighing the domains must take into account the the context of the overall reaction network, and is beyond the scope of a low-level domain-based approach to sequence design.

Instead, DD implements a user-defined "importance" parameter, which acts as an additive term to the score of the domain (Fig. 5B). In DD, the overall score of the set of domain sequences is determined by the domain with the worst (highest) score, and in a typical run the scores of all of the domains will be similar. The additive importance term to score means that high importance domains must have lower values for the other score terms in order to have similar score to domains with low importance.

Base composition. DD allows the user to specify the base compositions of each of the domains. For reasons outlined previously, it may be desirable to design certain domains using 3-letter C/A/T alphabets. Furthermore, the user also may wish to restrict certain domains to only A/T bases or G/C bases, so that the domain is weak- or strong-binding. This feature also facilitates the sequence design of non-standard DNA structures, such as Z-DNA [41] (C/G), or Hoogsteen triplex bindings (C/T) [51].

Designing sequences to function with an existing system. Finite research funding usually implies that DNA nanotechnologists will tend to use and reuse the same DNA oligonucleotide strands for many different purposes. It is not only economical but also practical, because using previously tested strands eliminates the chance for unexpected sequence-specific problems. Given this tendency, it is therefore important for sequence design software to be able to design optimal sequences given the constraints of the sequences of the existing strands.

DD allows the user to load sequences from files before attempting to design new sequences. At the user's option, the domains loaded in this manner can be all be locked for the new design process. The user may also save the sequences generated by DD at any time, and the save file will include information such as the importance, composition, and lock status of all bases and domains.

Discussion

This paper discusses the author's considerations for sequence design of DNA strand displacement-based reactions and cascades, and makes a tentative relative weighing of the considerations in the form of various score penalties. A randomized algorithm was presented that attempts to minimize the overall score of the domains to be designed by mutating a few bases at a time, and accepting mutations that lead to improved (lower) scores. The Domain Design (DD) software runs the randomized algorithm for sequence design while allowing the user to dynamically intervene.

At its heart, DD is a low-level domain-based approach to sequence design, and does not consider the ways in which the domains are concatenated to form DNA strands, nor the overall architecture of the reaction network using those strands. As a result, the sequences generated by DD will in general not be as good as those generated by software intended specifically for particular applications. The comparative advantage of DD is its speed, simplicity, and generalizability–by ignoring the details of particular strands and systems, it seeks to generate sequences that are "good enough" for a wide variety of DNA nanotechnology applications based on strand displacement. Empirical evidence of the success of DD lie in experimental works published by the author, who used DD to design the sequences for a variety of strand displacement-based reactions and networks exhibiting catalysis [12] [4] [13] [8].

Currently, most experimental work on dynamic DNA nanotechnology have been very limited in scale, with numbers of different DNA strands concurrently in solution being less than 100. At these scales, it is relatively easy to design sequences that do not significantly interact or crosstalk with one another. As a result, many different sequence design software (both published and unpublished) will likely generate sequences that work decently well for their intended applications, and it is not easy to critically evaluate their relative performances. As researchers develop and experimentally test larger

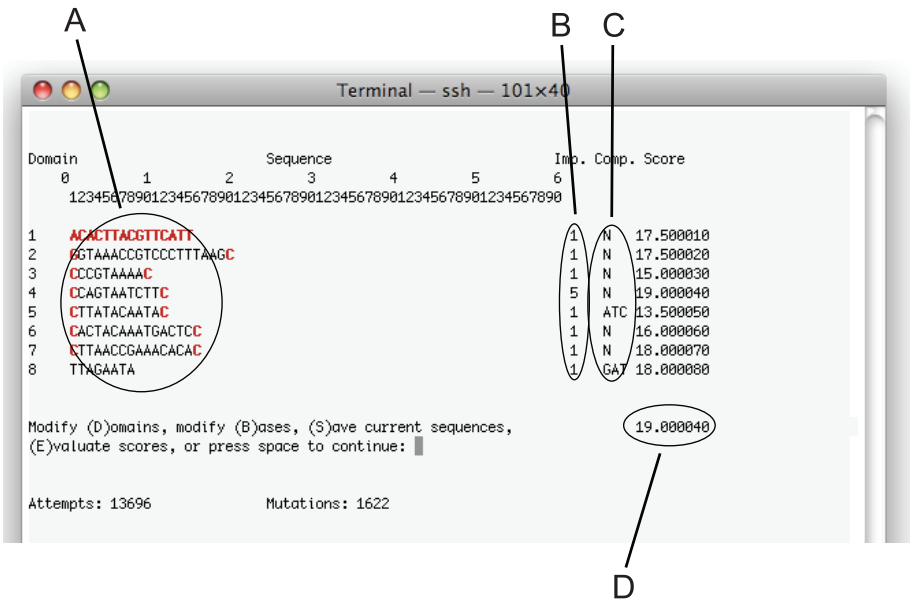

Fig. 5: Sample run of Domain Design (DD) software. The sequences of the domains are shown in **(A)**, with the bases in red denoting bases that are "locked" and prevented from mutation. The user can lock or unlock bases and domains through the course of the design process, as well as manually changing the sequences of the domains. The numbers in the **(B)** column list the "importance" of each domain, which is implemented as an additive adjustment to the score of the domain. **(C)** shows the nucleotide compositions of each of the domains, with "N" denoting no constraints (all four nucleotides may be used). The overall score of the set of domains is shown in **(D)**. The tiny decimal portion in the overall score shows the number of the domain currently possessing the worst score, to facilitate user intervention. In the figure, Domain 4 current has the worst score.

reaction networks [40] [52] [53], the demand for large numbers of DNA sequences that must exist stably simultaneously in solution will drive the development of ever better and easy-to-use DNA sequence design software.

Source code for Domain Design can be downloaded at:
http://www.dna.caltech.edu/∼dzhang/softsource/DD.c

Acknowledgements: This research is supported in part by NSF grants 0832824 and 0728703. DYZ is supported by the Fannie and John Hertz Foundation.

The author thanks Erik Winfree and David Soloveichik for testing the DD code. The author thanks Erik Winfree for suggestions regarding the manuscript.

References

1. J. SantaLucia, D. Hicks, *Annu. Rev. Biophys. Biomol. Struct.* **33,** 415 (2004).
2. R. M. Dirks, J. S. Bois, J. M. Schaeffer, E. Winfree, N. A. Pierce, *SIAM Rev.* **49,** 65 (2007).
3. Zuker, M. *Nucleic Acids Res.* **31,** 3406 (2003).
4. D. Y. Zhang, E. Winfree, *J. Am. Chem. Soc.* **131,** 17303 (2009).
5. G. Seelig, D. Soloveichik, D. Y. Zhang, E. Winfree, *Science* **314**, 1585 (2006).
6. M. Hagiya, S. Yaegashi, K. Takahashi, *Nanotechnology: Science and Computation*, 293-308 (2006).
7. B. M. Frezza, S. L. Cockroft, M. R. Ghadiri, *J. Am. Chem. Soc*, **129**, 14875 (2007)
8. D. Y. Zhang, "Cooperative DNA strand displacement for DNA quantitation, detection, and logic." (submitted, 2010).
9. Z. Xie, S. J. Liu, L. Bleris, Y. Benenson, *Nuc. Acids Res.* (2010, DOI: 10.1093/nar/gkq117).
10. D. Y. Zhang, A. J. Turberfield, B. Yurke, E. Winfree, *Science* **318**, 1121 (2007).
11. A. J. Turberfield, J. C. Mitchell, B. Yurke, A. P. Mills, M. I. Blakey, F. C. Simmel, *Phys. Rev. Lett.* **90**, 118102 (2003).
12. D. Y. Zhang, E. Winfree, *J. Am. Chem. Soc.* **130,** 13921 (2008).

13. D. Y. Zhang, E. Winfree, *Nuc. Acid Res.* (2010, pre-published online DOI:10.1093/nar/gkq088).
14. G. Seelig, B. Yurke, E. Winfree, *J. Am. Chem. Soc.* **128,** 12211 (2006).
15. J. S. Bois, S. Venkataraman, H. M. T. Choi, A. J. Spakowitz, Z. G. Wang, N. A. Pierce, *Nuc. Acid Res.* **33,** 4090 (2005).
16. S. J. Green, D. Lubrich, A. J. Turberfield, *Biophysical Journal* **91,** 2966 (2006).
17. B. Yurke, A. J. Turberfield, A. P. Mills, F. C. Simmel, J. L. Neumann, *Nature* **406**, 605 (2000).
18. R. M. Dirks, N. A. Pierce, *Proc. Nat. Acad. Sci.* **101**, 15275 (2004).
19. P. Yin, H. M. T. Choi, C. R. Calvert, N. A. Pierce, *Nature* **451**, 318 (2008).
20. T. Omabegho, R. Sha, N. C. Seeman, *Science* **324,** 67 (2009).
21. Y. Gao, L. K. Wolf, R. M. Georgiadis, *Nuc. Acids Res.* **34,** 3370 (2006).
22. W. Sun, C. Mao, F. Liu, N. C. Seeman, *J. Mol. Biol.* **282,** 59 (1998).
23. R. M. Dirks, M. Lin, E. Winfree, N. A. Pierce, *Nucleic Acids Res.* **32,** 1392 (2004).
24. J. Seifferf, A. Huhle, *J. Biomol. Struct. Dyn.* **25,** 453 (2008).
25. F. Tanaka, A. Kameda, M. Yamamoto, A. Ohuchi, *Nuc. Acids Res.* **33,** 903 (2005).
26. D. Tulpan, M. Andronescu, S. B. Chang, M. R. Shortreed, A. Condon, H. H. Hoos, L. M. Smith, *Nuc. Acids Res.* **33,** 4951 (2005).
27. J. Sager, D. Stefanovic, *Lecture Notes of Computer Science* **3892,** 275-289 (2006).
28. N. C. Seeman, *J. Biomol. Struct. Dyn.* **8,** 573-581 (1990).
29. S. Cao, S. Chen, *Nuc. Acids Res.* **34,** 2634 (2006).
30. A. Xayaphoummine, T. Bucher, H. Isambert, *Nuc. Acids Res.* **33,** W605 (2005).
31. R. M. Dirks, N. A. Pierce, *J. Comput. Chem.* **25,** 1295 (2004).
32. E. Protozanova, P. Yakovchuk, M. D. Frank-Kamenetskii, *J. Mol. Biol.* **342**, 775 (2004).
33. D. V. Pyshnyi, E. M. Ivanova, *Russian Chemical Bulletin* **51**, 1145 (2002).
34. V. A. Vasiliskov, D. V. Prokopenko, A. D. Mirzabekov, *Nuc. Acid Res.* **29**, 2303 (2001).
35. D. V. Pyshnyi, E. M. Ivanova, *Nucleosides, Nucleotides, and Nucleic Acids* **23**, 1057 (2004).
36. I. G. Panyutin, P. Hsieh, *J. Mol. Biol.* **230,** 413 (1993).
37. D. Sen, W. Gilbert, *Methods in enzymology* **211,** 191 (1992).
38. E. M. Southern, S. C. Casegreen, J. K. Elder, M. Johnson, K. U. Mir, L. Wang, J. C. Williams, *Nuc. Acids Res.* **22,** 1368 (1994).
39. Mir, K. U. A restricted genetic alphabet for DNA computing. *DNA based computers II. DIMACS* **44,** 243-246 (1998).
40. L. Qian, E. Winfree, *Lecture Notes of Computer Science* **5347,** 70 (2009).
41. C. Mao, W. Sun, Z. Shen, N. C. Seeman, *Nature* **397,** 144 (1999).
42. L. P. Reynaldo, A. V. Vologodskii, B. P. Neri, V. I. Lyamichev, *J. Mol. Bio.* **297**, 511 (2000).
43. B. Yurke, A. P. Mills, *Genet. Prog. Evol. Mach.* **4**, 111 (2003).
44. M. Zuker, D. H. Mathews, D. H. Turner, "Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A Practical Guide In RNA Biochemistry and Biotechnology," in J. Barciszewski and B.F.C. Clark, eds., NATO ASI Series, Kluwer Academic Publishers, (1999).
45. R. A. Dimitrov, M. Zuker, *Biophys. J.* **87,** 215 (2004).
46. J. Kim, K. S. White, E. Winfree, *Mol. Syst. Biol.* **2,** 68 (2006).
47. W. U. Dittmer, F. C. Simmel *Nano Lett.* **4,** 689 (2004).
48. Stojanovic, M. N., Semova, S., Kolpashchikov, D., Macdonald, J., Morgan, C. & Stefanovic, D. *J. Am. Chem. Soc.* **127,** 6914-6915 (2005).
49. R. Pei, S. K. Taylor, D. Stefanovic, S. Rudchenko, T. E. Mitchell, M. N. Stojanovic, *J. Am. Chem. Soc.* **128,** 12693 (2006).
50. K. Lund, A. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, S. Taylor, R. Pei, M. N. Stojanovic, N. Walter, E. Winfree, H. Yan, *Nature* (in press, 2010).
51. M. D. Frank-Kamenetskii, S. M. Mirkin, *Annu. Rev. Biochem.* **64,** 65 (1995).
52. D. Soloveichik, G. Seelig, E. Winfree, *Proc. Nat. Acad. Sci.* (2010, pre-published online DOI:10.1073/pnas.0909380107).
53. A. Phillips, L. Cardelli, *Journal of the Royal Society Interface* **6,** S419 (2009).