Automated Constraint-Based Nucleotide Sequence Selection for DNA Computation

Alexander J. Hartemink[†‡], David K. Gifford[†], and Julia Khodor[†]

Abstract. We present techniques for automating the design of computational systems built using DNA, given a set of high-level constraints on the desired behavior and performance of the system. We have developed a program called SCAN that exploits a previously implemented computational melting temperature primitive to search a "nucleotide space" for sequences satisfying a pre-specified set of constraints, including hybridization discrimination, primer $5'$ end and $3'$ end stability, secondary structure reduction, and prevention of oligonucleotide dimer formation. The first version of SCAN utilized 24 hours of compute time to search a space of over 7.5 billion unary counter designs and found only 9 designs satisfying all of the pre-specified constraints. One of SCAN's designs has been implemented in the laboratory and has shown a marked performance improvement over the products of previous attempts at manual design. We conclude with some novel ideas for improving the overall speed of the program that offer the promise of an efficient method for selecting optimal nucleotide sequences in an automated fashion.

Keywords: SCAN, automated, constraint, nucleotide, sequence selection, DNA computation

1. Introduction

We have conducted a series of experiments implementing various DNA computational systems in the laboratory and based on this experience, we have gained insights into fundamental constraints imposed on DNA computational models as they are put into practice (Khodor and Gifford 1998).

In particular, given the vast space of nucleotide sequences capable of implementing a particular model of computation, and given the diversity of constraints impinging upon the model's successful implementation, manually selecting nucleotide sequences capable of satisfying each of the necessary constraints simultaneously is a difficult task.

1991 *Mathematics Subject Classification.* Primary 92-04, 92C40, 80-04; Secondary 68U30.

This paper has been prepared for the Proceedings of the Fourth Annual DIMACS Workshop on DNA Based Computers.

[†]Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139, amink@mit.edu, gifford@mit.edu, jkhodor@mit.edu.

[‡]Partially funded by a National Science Foundation Graduate Research Fellowship.

To simplify the design of our computational systems, we have developed a tool for constraint-based selection of nucleotide sequences. This tool incorporates domain knowledge that has proven to be important in our experimental process. However, we have also formulated a framework for systematically solving a general nucleotide selection problem and have produced the program SCAN to assist in the selection process.

Constraint-based selection is quite different from some previous work on automated sequence selection (Deaton *et al.* 1996). Other nucleotide selection efforts have been directed at ensuring that sequences do not contain any significant amount of overlap in order to increase the specificity of large sets of DNA "words". These programs, however, do not seek to address general design problems where a wide variety of constraints need to be satisfied simultaneously. For our purposes in particular, it was necessary to include diverse constraints such as hybridization discrimination, primer $5'$ end and $3'$ end stability, secondary structure elimination, and reduction of oligonucleotide dimer formation.

In previous work (Hartemink and Gifford 1997), we implemented a computational primitive for the calculation of hybridization melting temperature. Using this primitive, we developed a program named BIND to analyze the thermal hybridization properties of extant nucleotide sequences. This initial version of BIND proved to be extremely useful in analyzing designs for a programmed mutagenic unary counter. However, analysis of the thermal hybridization properties of extant nucleotide sequences is not sufficient; the more difficult inverse problem needs to be addressed, *viz.*, given a set of desired thermal hybridization properties, is it possible to select, in an automated fashion, nucleotide sequences satisfying these constraints?

Fortunately, the computational primitive at the kernel of BIND also provides the leverage for addressing the inverse problem. SCAN exploits the computational melting temperature primitive contained in BIND to search a "nucleotide space" for sequences satisfying a set of pre-specified design constraints.

The first version of the program utilized 24 hours of compute time to search a space of over 7.5 billion unary counter designs and found only 9 designs satisfying all of the pre-specified constraints. One of SCAN's designs has been implemented in the laboratory and has shown a marked performance improvement over the products of previous attempts at manual design. These successful laboratory results have provided the impetus for seeking to improve the overall speed of the program. With this goal in mind, we are examining new approaches that offer the hope of an efficient method for the automated selection of constraint-satisfying nucleotide sequences.

Overview of this Paper. In the following section, we provide a short, descriptive review of programmed mutagenesis because some of the later material is based upon this technique. Then, in section 3, we discuss how a simple unary counter can be implemented using programmed mutagenesis. This background allows us to consider the example of the unary counter when we turn in the next section to a discussion of the process of designing a general DNA computer. In section 5, we present a description of how the SCAN program was used to winnow a large space of unary counter design candidates down to a small set of constraint-satisfying designs.

2. A Description of Programmed Mutagenesis

Programmed mutagenesis is a technique for programmatically rewriting DNA sequences by incorporating sequence-specific oligonucleotides into newly manufactured strands of DNA. Three significant advantages to using programmed mutagenesis for DNA computation are:

i. The pool of oligonucleotide rewrite rules can be designed to cause sequence-specific programmed changes to occur, including the propagation of programmed changes up and down a DNA molecule and the evolution of a programmed sequence of changes over the course of future replication events. Thus, sequential computations with programmatically evolving state can be carried out, resulting in *constructive computation*, as contrasted with *selective computation* which requires all possible solutions to a problem to be present *ab initio*.

ii. The sequence specificity of the oligonucleotide rewrite rules allows multiple rules to be present at each step of the reaction, with only a fraction of them being active during each cycle. This reduces human effort since it permits the computation to be carried forward by thermocycling the reactants in the presence of thermostable polymerase and ligase. Ideally, there is no need for human (or robotic) intervention between computational cycles.

iii. All the components necessary to implement programmed mutagenesis are present *in vivo*. Therefore it may eventually be possible to harness the internal workings of the cell for computation, thereby capitalizing on the cell's homeostatic capabilities to ensure that the computation takes place in a stable and well-regulated environment.

The salient point regarding programmed mutagenesis is that it relies on the binding specificity of its rewrite rules to ensure that the template strand of DNA is being rewritten in a systematic way. For example, if rewrite rule ρ_i is meant to be applied to a strand of DNA representing state σ_i, producing a strand representing state σ_{i+1}, and rewrite rule ρ_{i+1} is subsequently meant to be applied to the strand of DNA representing state σ_{i+1} to produce a strand representing σ_{i+2}, it should be the case that ρ_{i+1} cannot be applied to σ_i and ρ_i cannot be applied to σ_{i+1}. If this condition is satisfiable, then both of the rewrite rules can be present in the reaction and yet the system can only evolve from the state representing σ_i to the state representing σ_{i+2} by first passing through σ_{i+1}, with each rewrite rule being applied in sequence, thereby capturing the notion of programmatic computation.

Limited forms of programmed mutagenic unary counters have been built in the laboratory and the technique is believed to be generally extensible. In the next section, we present the design of a simple unary counter to describe how the technique of programmed mutagenesis can be applied to eventually realize the goal of DNA computation.

3. A Programmed Mutagenic Unary Counter

To demonstrate the function of programmed mutagenesis, we implemented a unary counter using DNA. The value of the counter is encoded in a template molecule as the number of **X** or **Y** symbols it contains,

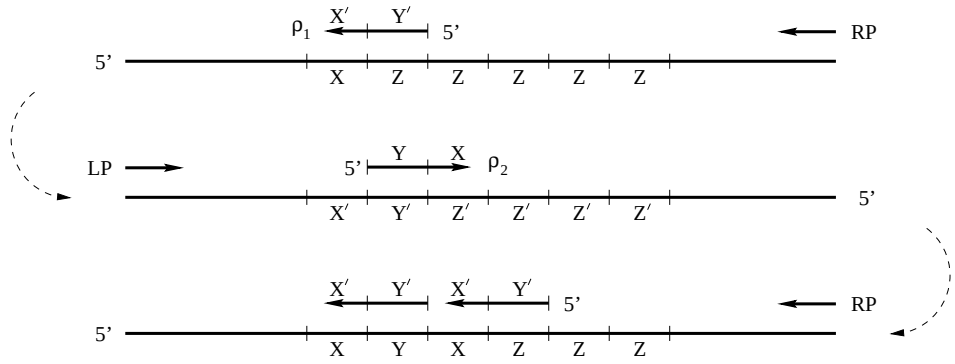

FIGURE 1. *A programmed mutagenic unary counter: In each cycle, one mutagenic primer hybridizes with the current template. Thermostable polymerase extends primers and the extended products are joined by thermostable ligase into a full-length strand. This strand becomes the new template in the following cycle.*

where **X**, **Y**, and **Z** symbols are shorthand representations of specific 12-nucleotide sequences that differ from one another in a few base positions.

As shown in figure 1, the initial template contains a sequence of six 12-mers, designated by the symbol sequence **XZZZZZ**, and encodes the number one. During each counter cycle, the first **Z** in the sequence is replaced by either an **X** or a **Y**, thereby increasing the value stored in the counter.

In order to transform **Z**'s into **X**'s and **Y**'s, mutagenic oligonucleotide primers are employed to act as rewrite rules ρ_1 and ρ_2. Each mutagenic primer binds to the template at a temperature permissive for non-specific binding, thereby allowing some nucleotides to be altered and an **X** or **Y** to be written in place of a **Z**.

In each cycle, the counter uses as its input template the product of the immediately preceding cycle, and the product of cycle N encodes the number $N+1$. The counter is implemented by thermocycling a reaction that begins with the initial template strand, ρ_1, ρ_2, thermostable polymerase and ligase, and outside primers LP and RP (needed to produce the full-length product on each cycle). Each thermal cycle consists of a high temperature step, that denatures the double-stranded DNA and prepares it for the polymerization-ligation step; and a low temperature step, that is permissive for primer annealing, polymerization, and ligation. Ideal values for these temperatures were chosen on the basis of data from laboratory experiments and from calculations performed by the BIND simulator (Hartemink and Gifford 1997).

Rewrite rule 1 consists of the 12-mer $\mathbf{Y}'$ followed by the first nine bases of $\mathbf{X}'$ while rewrite rule 2 consists of the 12-mer **Y** followed by the first nine bases of **X**. The lengths of the oligonucleotide primers were reduced from 24 bases to 21 bases by simply truncating each primer after 21 bases. If a primer hybridizes with the template, the last three bases will be replaced by the action of polymerase during the replication step, but shortening the rules significantly reduces the likelihood of cross-rule $3'$ dimers forming between primers in

solution, which is a critical concern in the design of the counter. This constraint is one of SCAN's selection criteria.

In the first cycle, rule 1 is designed to hybridize with the template so that the nine bases of $\mathbf{X}'$ bind to the $\mathbf{X}$ in the template and the $\mathbf{Y}'$ binds to the $\mathbf{Z}$ following the $\mathbf{X}$ (as shown in the figure). If this hybridized mutagenic oligonucleotide is successfully incorporated into full-length product after the action of thermostable polymerase and ligase, then the newly produced template strand will contain the six 12-mers represented by the symbol sequence $\mathbf{Z}'\mathbf{Z}'\mathbf{Z}'\mathbf{Z}'\mathbf{Y}'\mathbf{X}'$ [1].

In the second cycle, rule 2 is designed to hybridize with the new template so that $\mathbf{Y}$ binds to the $\mathbf{Y}'$ in the template and the nine bases of $\mathbf{X}$ bind to the $\mathbf{Z}'$ preceding the $\mathbf{Y}'$. If this hybridized mutagenic oligonucleotide is successfully incorporated into full-length product after the action of thermostable polymerase and ligase, then the newly produced template strand will contain the six 12-mers represented by the symbol sequence **XYXZZZ** and the counter will thus have counted to three.

This counter construction presumes that primers annealing to the template with a small number of mismatches can be extended and successfully ligated to other polynucleotides, while primers with a greater number of mismatches cannot be effectively incorporated into a new full-length strand. Selecting the nucleotide sequences to ensure that this is true is part of the challenge of designing a successful unary counter. Once again, SCAN uses these constraints as part of its selection criteria.

4. Designing a Unary Counter: A Case Study

To demonstrate how SCAN can assist in transforming DNA computation from theory into practice, it is helpful at this stage to discuss the ways in which it was employed in the unary counter design process. Though presented in the context of a specific application of programmed mutagenesis, the overarching framework of the process should be representative of any general DNA computer design process. A flowchart illustrating the process is included in figure 2, shown on page 6.

After determining the computational goal of the DNA computer, a high-level representation is required to design the necessary template and primers. In this example, the specific 12-nucleotide sequences represented by the symbols $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ were used. These symbols formed the basis of both the template and the primers that act as rewrite rules, as described in the previous section.

In the context of programmed mutagenesis, next the positions of mismatches between the template and primers need to be determined. For the unary counter, selecting mismatch geometries describing the positions of mismatched nucleotides in the $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ symbols induces a skeletal structure on the template and primers.

Since the precise locations of the various mismatches have a large effect on the operation of the thermostable polymerase and ligase, the mismatch geometries of the primers are configured to allow successful polymerization and ligation of correctly bound rewrite rules, as well as strict enzyme specificity for incorrectly bound rewrite rules.

[1] Recall that the orientation of the newly formed complementary strand is opposite that of the template.

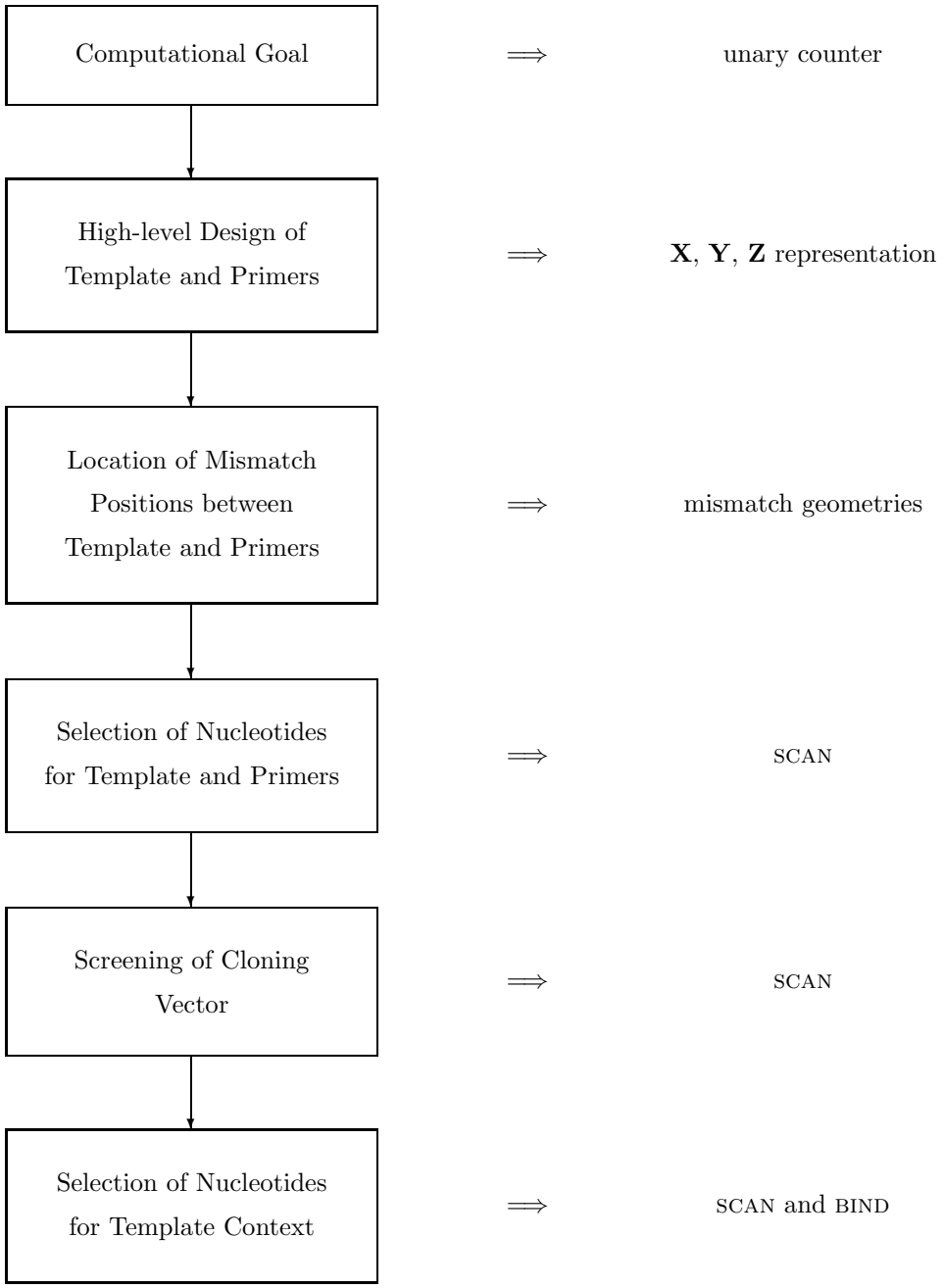

FIGURE 2. *Designing a DNA computer: Each box on the left represents a step in the general design process, while the labels on the right indicate the specific instance of that step in the case of a programmed mutagenic unary counter.*

The mutagenic primers are able to modify the native template sequence by being incorporated into a complementary strand even though they are not perfectly matched with the template. Because the primers need to possess mismatches relative to the desired binding site in the template, they are inherently less stable than perfectly matched primers. In order to be successfully incorporated, they need to be ligated and extended with thermostable ligase and polymerase, respectively. However, these enzymes require that the primer bind to the template with sufficient stability for the appropriate enzyme complex to be formed. This means that the mismatches within the mutagenic primers that are intended to be incorporated should not be too close to either the $5'$ ligation end of the primer or the $3'$ polymerization end of the primer. These mismatches do provide some specificity leverage in that it is possible to design the counter such that mutagenic primers that are *not* intended to be incorporated have mismatches at either their $5'$ or $3'$ ends to prevent the formation of full-length product.

To test various mismatch geometries, a number of laboratory experiments were performed in which mutagenic primers with different mismatch geometries were placed in test tubes with a template sequence and thermostable polymerase and ligase. Each geometry was evaluated in terms of its ability to be successfully polymerized and ligated. Based on this series of experiments, five mismatch geometries were selected for consideration at the next stage by SCAN. The five geometries are shown in figure 3. It should be noted that geometry 5 describes a design involving rewrite rules of length 22; in this scenario, the first ten bases of **X** are present rather than the first nine.

4.1. Nucleotide Selection. The mismatch geometries are merely skeletons describing the positions where Watson-Crick pairings should occur and where mismatches should occur. Once the mismatch relationships between positions in the template and primers are determined, specific nucleotides need to be selected in order to flesh out the skeletal structures, yielding a specific sequence for the template and primers. This is where SCAN was employed. In the unary counter example, given a fixed mismatch geometry for the **X**, **Y**, and **Z** symbols, only a finite number of possible nucleotide sequences conform to that geometry. It is straightforward enough to have SCAN evaluate the performance criteria associated with each possible sequence in the space. Admittedly, checking every possible sequence is a naïve approach, but for a first pass, it provides a simple way of determining whether automated selection of sequences is even worthwhile.

Because the template containing the computational core is embedded in a context sequence and then cloned into a vector (see figure 4 on page 9), once the primer sequences have been determined, it remains to carefully screen the cloning vector and then select the nucleotides for the context sequence. These steps are critical to ensure that the vector and context do not contain subsequences where the primers can potentially hybridize. Minimizing the degree of possible interference with the primers is crucial.

5. The SCAN Program

We used SCAN to produce a design with superior discrimination properties capable of demonstrating that the unary counter is operating as hypothesized. SCAN generated candidate DNA sequences by selecting specific nucleotides to occupy each position in the skeleton strands, and then simulated the unary counter's

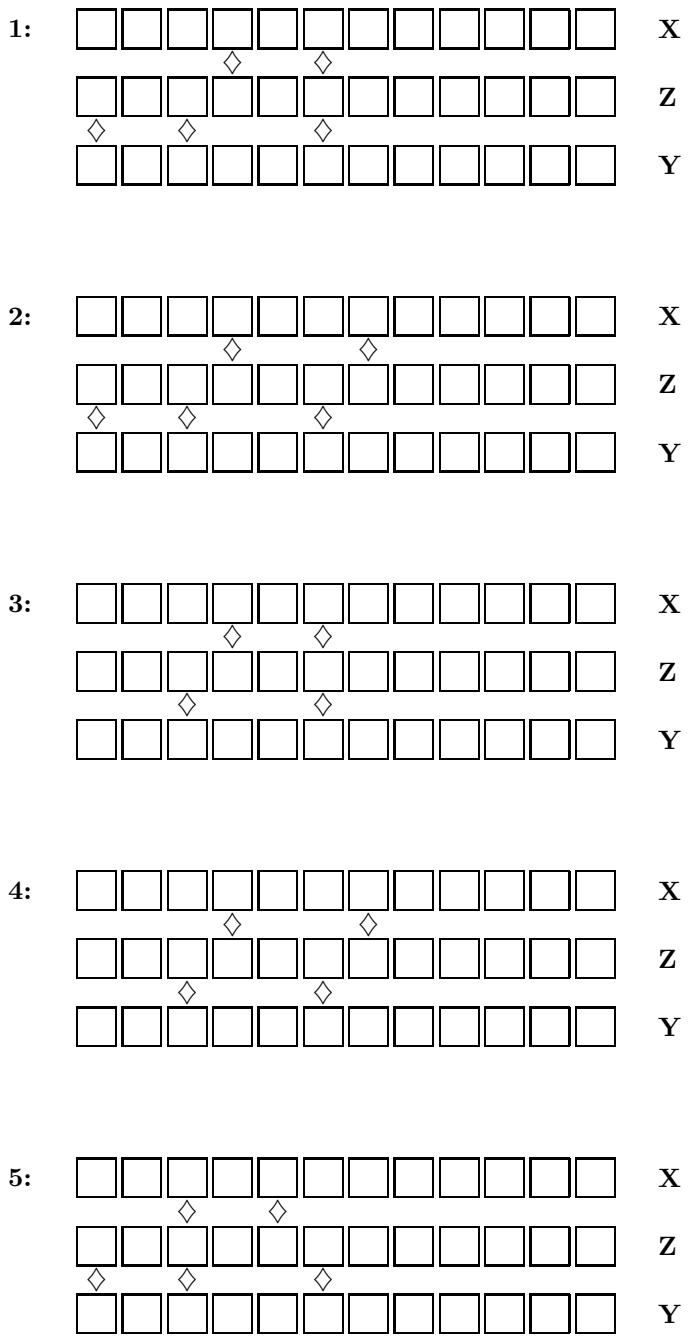

FIGURE 3. *Mismatch geometries: Each mismatch geometry provides a skeletal picture of the nucleotide composition for the three* ***X****,* ***Y****, and* ***Z*** *symbols. Mismatched nucleotides between adjacent symbols are indicated by the* ♢ *character. All five geometries were used by* SCAN *as skeletons for nucleotide selection.*

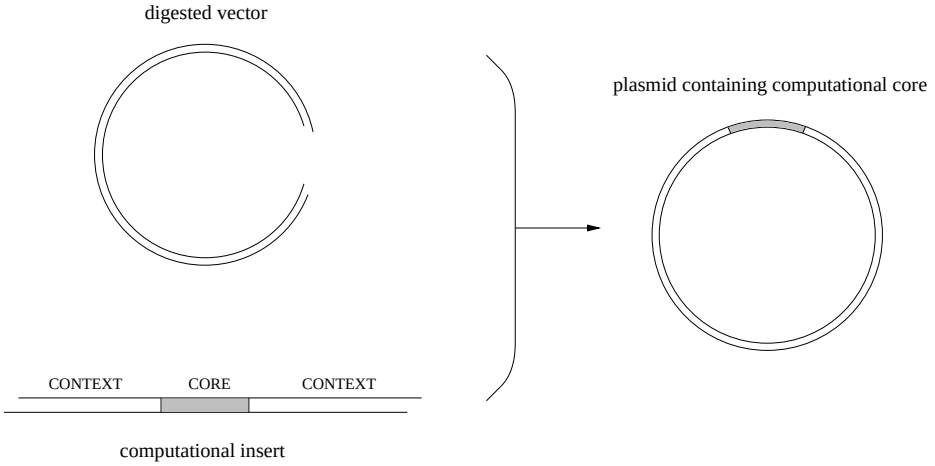

FIGURE 4. *Cloning a computational insert into a vector: The computational core of the insert is manufactured inside a larger context. The context provides some spacing between the core and the eventual plasmid and allows the insert to be cloned into a vector by standard restriction digestion and ligation. The resulting plasmid can then be maintained* in vivo.

discrimination properties with those nucleotides. Five different mismatch geometries were considered, as outlined in figure 3; over 2 billion nucleotide combinations were scanned for the first two geometries, while over 1 billion nucleotide combinations were scanned for the last three. In total, over 7.5 billion different design candidates were screened for suitability.

Unary counter design candidates were filtered on the basis of a number of distinct criteria, as presented in table 1. First, candidates were screened on the basis of their discrimination and non-interference properties. Rules that are "active" in any particular cycle must be incorporated into product strands at the correct site along the template. If it is possible for the rewrite rule to be incorporated in the wrong location, the computation will not be reliable; the unary counter must be discriminating in its rewrite rule incorporation.

In addition to verifying that "active" rules are being incorporated in the correct positions, we must also verify that the "inactive" rules are not interfering with the incorporation process. As described in section 2, it is an important feature of programmed mutagenesis that various rewrite rules are able to be present in the system simultaneously. Therefore, it becomes a critical factor in the design process that rules that are not applicable in a particular cycle should not be incorporated into product strands. In other words, it should be the case that "inactive" rules have a low binding affinity over the entire length of the template. Consequently, in any successful design, rules must be carefully selected so as to prevent them from being incorporated at the wrong time.

Second, candidate designs were screened to ensure that the constituent strands had minimal secondary structure. Since all the strands are present in the system at once, it is quite possible that undesirable side reactions could be taking place unintentionally. For example, a strand may possess some secondary structure

TABLE 1. *Specifications for* SCAN *filter criteria*

Filter Criterion	Specification
Strong Discrimination	
Min Correct Binding T_m	45° C
Min Difference between Correct and Incorrect Binding T_m	20° C
Max Inactive Rule Binding T_m	-5° C
Max Difference between Correct Binding T_m *across Cycles*	6° C
Minimal Secondary Structure	
Max 3′ Hairpin T_m	40° C
Max 5′ Hairpin T_m	50° C
Max 3′ Self-Dimer T_m	0° C
Max 3′ Cross-Dimer T_m	0° C
Low Plasmid Interference (pUC19)	
Max Binding T_m *along Entire Length of Plasmid*	20° C

TABLE 2. *Number of candidates passing through successive* SCAN *filters*

Filter	Geometry 1	Geometry 2	Geometry 3	Geometry 4	Geometry 5
None (Initial Pool)	2147483648	2147483648	1073741824	1073741824	1073741824
Discrimination	21326	30728	727	498	1086
Secondary Structure	39	784	0	29	0
Plasmid Interference	0	8	0	1	0

that causes it to fold back onto itself and hairpin, thereby preventing it from interacting with other strands as desired. Alternatively, a primer could hybridize with a copy of itself or with another primer, forming unwanted dimers and thereby allowing undesired side reactions to proceed or preventing necessary reactants from interacting as planned. Therefore, it is critical that the constituent strands of any unary counter design be screened for possible secondary structure and undesirable side reactions.

Third, since the unary counter is embedded in a larger plasmid, it is important that the chosen primer sequences be compatible with the plasmid in which they will later be inserted. For this reason, the plasmid should be scanned for possible alternate primer binding locations and if any are found, either another plasmid needs to be selected or a different design needs to be considered. Because the plasmid is used mainly as a repository for the counter, we relax the specification to allow alternate binding sites with melting temperatures as high as 20° C (see table 1).

After the three filtering stages, only nine designs remained: eight designs conforming to geometry 2 and one design conforming to geometry 4. A complete breakdown is provided in table 2.

TABLE 3. *Performance characteristics of unary counter designs*

Characteristic	Manual	SCAN.L	SCAN.2	SCAN.4
Cycle 1 Discrimination				
Correct Binding T_m	55.1	46.9	46.0	46.3
Incorrect Binding T_m	45.4	26.5	25.8	25.2
Inactive Rule Binding T_m	20.2	<-20.0	-17.1	-11.7
Cycle 2 Discrimination				
Correct Binding T_m	52.6	45.1	50.7	48.9
Incorrect Binding T_m	38.0	24.9	26.6	28.6
Inactive Rule Binding T_m	13.7	-15.4	-15.6	<-20.0
Rule 1 Secondary Structure				
3′ Hairpin T_m	40.9	37.8	21.5	37.4
5′ Hairpin T_m	46.3	31.6	43.4	39.3
3′ Self-Dimer T_m	-10.3	<-20.0	<-20.0	<-20.0
Rule 2 Secondary Structure				
3′ Hairpin T_m	59.6	40.7	24.2	36.7
5′ Hairpin T_m	45.3	28.7	27.6	34.3
3′ Self-Dimer T_m	4.5	<-20.0	<-20.0	-8.3
Cross-Rule Interference				
3′ Cross-Dimer T_m	44.1	6.2	-0.4	-4.2

Two of these designs have been singled out for inspection: SCAN.2, which is based upon mismatch geometry 2, and SCAN.4, which is based upon mismatch geometry 4. Table 3 contains a full description of the performance characteristics for both designs, along with a summary of the data for a design entitled SCAN.L, representing an early SCAN design that has already been inserted into a plasmid and studied experimentally in the laboratory. For the purposes of comparison, we also provide performance characteristics of our last manually designed counter.

The overall operating temperature is higher for the manual design because it uses rewrite rules of length 24. When we examine the difference between the correct and incorrect binding T_m, while the manual design's level of discrimination is only 9.7° C in the first cycle and 14.6° C in the second, the new designs are all above 20° C in each cycle. Moreover, the likelihood of secondary structure interference in the automated designs has been reduced, and in the case of 3′ cross-dimer formation, dramatically reduced. Preliminary experiments using SCAN.L reveal that designs produced by SCAN exhibit superior discrimination properties and perform markedly better than earlier versions of the unary counter that were designed manually; some of these experimental results are reported in Khodor and Gifford (1998).

6. Conclusion

The excellent performance of the unary counter designs produced by SCAN has been encouraging. In addition to stimulating continued laboratory research, these results have provided the impetus for considering how the automated selection process might be made more efficient. We continue to consider and test algorithmic improvements for improving the program's overall running time.

Acknowledgements. We would like to thank the anonymous referees for their helpful suggestions. Hartemink also gratefully acknowledges the support of the National Science Foundation.

References

Deaton, R., Murphy, R., Garzon, M., Franceschetti, D., and Stevens, Jr., S., 1996, Good encodings for DNA-based solutions to combinatorial problems. Proceedings of the Second Annual DIMACS Workshop on DNA Based Computers, June.

Hartemink, A., and Gifford, D., 1997, Thermodynamic simulation of deoxyoligonucleotide hybridization for DNA computation. Proceedings of the Third Annual DIMACS Workshop on DNA Based Computers, June.

Khodor, J., and Gifford, D., 1998, Design and implementation of computational systems based on programmed mutagenesis. Proceedings of the Fourth Annual DIMACS Workshop on DNA Based Computers, June.