On the Hardness of Counting and Sampling Center Strings

Christina Boucher and Mohamed Omar

Abstract—Given a set S of n strings, each of length ℓ, and a nonnegative value d, we define a *center string* as a string of length ℓ that has Hamming distance at most d from each string in S. The #CLOSEST STRING problem aims to determine the number of center strings for a given set of strings S and input parameters n, ℓ, and d. We show #CLOSEST STRING is impossible to solve exactly or even approximately in polynomial time, and that restricting #CLOSEST STRING so that any one of the parameters n, ℓ, or d is fixed leads to a fully polynomial-time randomized approximation scheme (FPRAS). We show equivalent results for the problem of efficiently sampling center strings uniformly at random (u.a.r.).

Index Terms—Biological sequence analysis, motif recognition, fully polynomial-time randomized approximation scheme (FPRAS), journal, fully polynomial almost uniform sampler (FPAUS), computational complexity

1 Introduction

FINDING similar regions in multiple DNA, RNA, or protein sequences plays an important role in many applications, including universal PCR primer design [4], [16], [18], [26], genetic probe design [16], antisense drug design [16], [3], finding transcription factor binding sites in genomic data [27], determining an unbiased consensus of a protein family [1], and motif recognition [16], [24], [25]. The CLOSEST STRING problem formalizes these tasks and can be defined as follows: given a set of n strings S of length ℓ over the alphabet Σ and parameter d, the aim is to determine if there exists a string s that has Hamming distance at most d from each string in S. We refer to s as the *center string* and let $d(x, y)$ be the Hamming distance between strings x and y.

The CLOSEST STRING was first introduced and studied in the context bioinformatics by Lanctot et al. [16]. Frances and Litman [11] showed the problem to be NP-complete, even in the special case when the alphabet is binary, implying there is unlikely to be a polynomial-time algorithm for this problem unless $\mathrm{P} = \mathrm{NP}$. Since its introduction, the investigation of efficient approximation algorithms and exact heuristics for the CLOSEST STRING problem has been thoroughly considered [9], [10], [12], [16], [17], [19], [20].

S is *pairwise bounded* if the Hamming distance for each pair strings in S is at most $2d$. The CLOSEST STRING problem reduces to separating pairwise bounded sets with a center string, and if so, finding one such string, from those that do not. A set of strings S with at least one center string is a *motif set*; S is a *decoy set* if it is pairwise bounded but does not have a center string. We note that a center string for a given set S is not necessarily unique.

A related, uninvestigated problem is determining the computational difficultly in finding the number of center strings for a set of strings. In many biological applications, including the ones listed above, it is useful to identify the all possible center strings, rather than only determining whether one exists. Further, an important relationship between the number of center strings for a given set of strings S and the computational difficulty of solving the decision version of #CLOSEST STRING for an instance S has been shown. Specifically, empirical analysis demonstrates that for sufficiently large n, all motif sets are clustered together and are characterized as having one center string, which is a string of length ℓ containing the symbol that occurs most frequently at each position [2]. Imperative to the analytical explanation of this conjecture is the development of an algorithm to efficiently count the number of center strings for a given set of strings.

We give the formal description of this counting problem as follows:

#CLOSEST STRING

INSTANCE: parameters n, ℓ, and d and a set S of n length-ℓ strings from the alphabet Σ.

OUTPUT: the number of distinct strings taken from the alphabet Σ that have distance at most d from each string in S.

Countless sampling and counting problems have been studied, including the sampling and counting versions of the following problems: matchings in a graph [14], the graph-coloring [6], [13], [21], Hamiltonian path [7], independent set [8], and knapsack [5], [22].

This paper focuses on the computational difficulty of counting and sampling center strings exactly or approximately. To the best of our knowledge this is the first consideration of this problem but is motivated by problems addressing the use of biological data. We show #CLOSEST STRING is #P-complete, implying it is in the complexity class of hard counting problems.

Given that this problem cannot be solved efficiently, we investigate if it can be reasonably approximated efficiently. Many #P-complete problems have an *fully polynomial-time randomized approximation scheme (FPRAS)* that produces an approximation of arbitrarily small error with high probability, and whose running time is polynomial in the size of the problem and in its accuracy. Jerrum et al. [15] showed that every #P-complete problem either has an FPRAS or is impossible to approximate.

For sampling problems, the aim is to obtain a *fully polynomial almost uniform sampler (FPAUS)*, which outputs solutions that achieve an approximation to a given distribution of solutions. To be precise, suppose π is a distribution on a sample space Ω. For a distribution μ, the *total variation* distance $d_{TV}(\mu, \pi)$ is given by $d_{TV}(\mu, \pi) = \frac{1}{2}\sum_{x\in\Omega} |\mu(x) - \pi(x)|$. An FPAUS approximating π is an algorithm whose output fits a distribution μ satisfying $d_{TV}(\mu, \pi) \leq \delta$, and that has running in polynomial time in the size of the input and $\log(1/\delta)$.

In absence of the existence of an FPRAS or an FPAUS for a general counting or sampling problem, interest remains in showing an FPRAS or an FPAUS exists for a restricted version of the problem, i.e., when one of the problem parameters is fixed. We prove that, although there does not exist an FPRAS for the general #CLOSEST STRING problem, there does exist an FPRAS for the most natural restricted versions of #CLOSEST STRING. Similarly, there does not exist an FPAUS for sampling center strings uniformly at random (u.a.r.) but restricting interest to the sampling problem where one of the parameters is fixed leads to an FPAUS.

2 Hardness Results

We show the #CLOSEST STRING problem is #P-complete, and that there does not exist an FPRAS for #CLOSEST STRING under reasonable complexity assumptions.

2.1 Reduction from 3-SAT

We note that a problem is #P-complete if and only if it is in #P and every problem in #P can be reduced to it by a polynomial-time counting reduction. Therefore, to prove that #CLOSEST STRING is #P-hard, it is sufficient to show that #3-SAT, a #P-complete problem, can be reduced to #CLOSEST STRING. #3-SAT aims to determine for

- *C. Boucher is with the Department of Computer Science, Colorado State University, 1873 Campus Delivery, Fort Collins, CO 80523-1873. E-mail: cboucher@cs.colostate.edu.*
- *M. Omar is with the Department of Mathematics, California Institute of Technology, Pasadena, CA 91125. E-mail: momar@caltech.edu.*

Manuscript received 12 May 2011; revised 7 Jan. 2012; accepted 29 Apr. 2012; published online 23 May 2012.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2011-05-0118. Digital Object Identifier no. 10.1109/TCBB.2012.84.

given a 3-CNF formula F, how many satisfying assignments exist for F. Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A *literal* is either x_i or $\neg x_i$ for some i. We refer to a *3-clause* as a disjunction of three distinct literals, made of three different variables.

In order to correctly reduce #3-SAT to #CLOSEST STRING, the reduction will produce two sets of strings from each #3-SAT instance: *transformation strings* and *constraint strings*. There will be a transformation string for each clause in the #3-SAT instance that encodes that clause. The purpose of the *constraint strings* is to ensure that each solution s to the #CLOSEST STRING instance is of the form $s \in \{00, 11\}^n \cdot 00$. We first describe the transformation strings and then describe the *constraint strings*.

We present the reduction of a single 3-clause and then extend it to a general 3-CNF formula. Given a 3-clause ω over the variables X, we define the length-$2(n+1)$ string $s = s(1) \ldots s(2n) \cdot 00$ as follows:

$$s(2i-1)s(2i) = \begin{cases} 00, & \text{if } \omega \text{ contains the literal} \neg x_i, \\ 11, & \text{if } \omega \text{ contains the literal } x_i, \\ 01, & \text{otherwise.} \end{cases}$$

A *block* of a string s is a length-2 substring $s(2i-1)s(2i)$, and a block is a *repetition* if $s(2i-1) = 2(i)$. Therefore, s can be defined in terms of $n+1$ blocks, where exactly three of the blocks are repetitions, the last block of s is equal to 00, and the remaining blocks are equal to 01. Let Ω_X denote the set of all 3-clauses on X, and $\phi : \Omega_X \to \{0,1\}^{2(n+1)}$ be the one-to-one mapping from Ω_X onto the set of all binary strings of length $2(n+1)$ as defined above. For any 3-CNF formula $\Omega = \omega^1 \wedge \cdots \wedge \omega^t$ over the variables $x_1, \ldots, x_n$, we define the set of strings transformation strings as the set $\{\phi(\omega^1), \ldots, \phi(\omega^t)\}$. This ends our description of the transformation strings.

As previously stated, the constraint strings ensure that each solution to the #CLOSEST STRING is such that the first $2n$ correspond to n blocks that are repetitions. Let γ_r be the strings that correspond to the r-times concatenation of 01, and similarly, γ_r^c be the string that corresponds to the r-times to the concatenation of 10. We define α_n^1 to be the set of four strings of length $2n$ that are equal to $\{\gamma_1, \gamma_1^c\} \cdot \{\gamma_{n-1}, \gamma_{n-1}^c\}$, i.e., this set is $\gamma_1 \cdot \gamma_{n-1}$, $\gamma_1 \cdot \gamma_{n-1}^c$, $\gamma_1^c \cdot \gamma_{n-1}$, $\gamma_1^c \gamma_{n-1}^c$. Similarly, α_n^2 is equal to $\{\gamma_2, \gamma_2^c\} \cdot \{\gamma_{n-2}, \gamma_{n-2}^c\}$. Generalizing this definition we have that α_n^i is equal to $\{\gamma_i, \gamma_i^c\} \cdot \{\gamma_{n-i}, \gamma_{n-i}^c\}$. Finally, let α_n be equal to $\{\alpha_n^1 \cup \alpha_n^2 \cup \cdots \cup \alpha_n^n\}$, and $S_n = \{s_\alpha \cdot 00 \mid \forall s_\alpha \in \alpha_n\}$. The size of S_n is $4n$. This set S_n defines the set of constraint strings. The following lemma demonstrates that the addition of the constraint strings to the construction implies that any center string is such that the first n blocks are repetitions.

Lemma 1. *Any string $s \in \{0,1\}^{2n}$ has Hamming distance at most n from each string α_n, where $n > 2$, if and only if s can be defined in terms of n blocks, where each block is a repetition.*

Proof. If s is a set of n blocks and each block is a repetition then it follows that $d(s, s_\alpha) \leq n$ for all s_α in α_n. Therefore, it follows that we need only to prove the reverse direction. Assume to the contrary that there exists a string $s \in \{0,1\}^{2n}$ that has Hamming distance at most n from each string in S_n and there is a position i, where $1 \leq i \leq n$ and $s(2i-1) \neq s(2i)$. We assume without loss of generality that $s(2i-1) = 0$ and $s(2i) = 1$. We have that the positions $2i-1$ and $2i$ in s match to 01 and mismatch to 10. Now consider the string x in α_n^i that is equal to $\gamma_i^c \cdot \gamma_{n-i}$. This string x has Hamming distance $d+2$ from s, where the Hamming distance between γ_{i-1}^c and first $2(i-1)$ positions of s, and the Hamming distance between γ_{n-i-1} and the last $2(n-i-1)$ positions of s is equal to $n-1$. Note that all the mismatches counted in d come from positions other than $2i-1$ and $2i$. Therefore, it follows that $n-1 \leq d$. Since s can have Hamming distance at most n also from x, it follows that $n-2 \geq d$, contradicting that $n-1 \leq d$. □

For any 3-CNF formula $\Omega = \omega^1 \wedge \cdots \wedge \omega^t$ over the variables $x_1, \ldots, x_n$, we generate a #CLOSEST STRING instance with the set of strings $S = \{\phi(\omega^1), \ldots, \phi(\omega^t)\} \cup S_n$. The distance parameter for the constructed instance is equal to $n+1$. This reduction can be computed in polynomial time from Ω. We assume that $n > 2$, otherwise the problem can be solved in polynomial time.

2.2 Correctness of the Reduction

The following two lemmas establish the correctness of the reduction, where the first follows directly from the construction. We note that the reduction has to be shown to be parsimonious, meaning for each unique satisfying assignment to the clauses in Ω there is a center string, and for each center string there exists an unique satisfying assignment to the clauses in Ω.

Lemma 2. *For each satisfying assignment to the clauses in Ω, there is a center string s that has Hamming distance $n+1$ from the strings in S.*

Proof. Let $A(x_1, \ldots, x_n) \in \{0,1\}^n$ be an assignment satisfying the clauses of Ω, and define the string $s_A = s_A(1) \ldots s_A(2n)$ as follows:

$$s_A(2i-1)s_A(2i) = \begin{cases} 00, & \text{if } \mathrm{A(x_i)} = 0, \\ 11, & \text{if } \mathrm{A(x_i)} = 1, \\ 01, & \text{otherwise.} \end{cases}$$

We note that the string s_A has Hamming distance equal to $n+1$ from each of the strings in S. Let s_A and $s_{A'}$ be such that $s_A(j) = s_{A'}(j)$, for all $j = 1, \ldots, 2(n+1)$ then it follows from our construction that $A(x_i) = A'(x_i)$ for all $i = 1, \ldots, n$. Therefore, for each satisfying assignment there exists a center string for the constructed #CLOSEST STRING instance. □

Lemma 3. *For each center string for the constructed #CLOSEST STRING instance, there exists a corresponding (unique) satisfying assignment to clauses in Ω.*

Proof. Let s be a string that has Hamming distance equal to $n+1$ from each of the strings in S. Since s has Hamming distance at most $n+1$ from all the $4n$ strings in S_n, it follows from Lemma 1 that $s \in \{00, 11\}^n \cdot 00$. Consider the assignment A_s to $x_1, x_2, \ldots, x_n$ corresponding to the string s, defined as follows: $A_s(x_i) = 0$ if $s(i)s(i+1) = 00$; otherwise, $A_s(x_i) = 1$. Observe that if s has Hamming distance at most $n+1$ from a string in the set $\phi(\omega^j)$ then A_s satisfies clause ω^j. Given two center strings s and s', where $s(j) = s'(j)$ for all $j = 1, \ldots, 2n$ it follows that $s(i)s(i+1) = s'(i)s'(i+1)$ and therefore, $A_s(x_i) = A_{s'}(x_i)$ for all i. □

Our main theorem follows directly from Lemmas 2 and 3, which show the number of satisfying formulae to Ω is parsimonious to the number of center strings to the set S.

Proposition 1. *#CLOSEST STRING is #P-complete.*

2.3 Further Hardness Results

Randomization can be quite powerful in achieving an approximation scheme for several #P-complete problems. Jerrum et al. showed that the problem of counting the number of simple cycles in a directed graph is not approximable by proving that the existence of an almost uniform generator for this problem implies the existence of a randomized polynomial time algorithm for determining the existence of a Hamiltonian cycle in a directed graph [15]. FPRAS is the subclass of #P counting problems whose answer, y, is approximable in the following sense: there exists a randomized algorithm that, with probability at least $1-\delta$, approximates y to within an ϵ multiplicative factor in time polynomial in n (the input size), $1/\epsilon$, and $\log(1/\delta)$.

The results of Jerrum et al. [15] imply that every #P-complete problem exhibits an FPRAS or is not approximable. Given a #P-complete problem an important question to answer is if there exists

an FPRAS for the problem—since the existence implies the approximability of the problem. Unfortunately, the existence of an FPRAS for #CLOSEST STRING is unlikely.

Observation 1. There is no FPRAS for #CLOSEST STRING, unless $NP = RP$.

Consider a problem π and I be an instance of the problem π, and let $\#(I)$ denote the number of solutions for I. An FPRAS can be used to distinguish between the case where $\#(I)$ is equal to zero and when $\#(I)$ is greater than zero; hence, providing a randomized polynomial time algorithm for the decision version of the problem π. Therefore, π must be contained in the class BPP. Since it is unlikely that BPP equal to NP, all NP-complete problems are believed not to contain an FPRAS [23, p. 309]. Since CLOSEST STRING problem is NP-complete [11], there exists no FPRAS for #CLOSEST STRING, unless BPP = NP.

The notions of counting and sampling are closely related. Jerrum et al. established the equivalence between the existence of an FPRAS and an FPAUS; namely for self-reducible problems there exists an FPRAS if and only if there exists an FPAUS [15]. It follows that we have the following negative result concerning the approximability of sampling center strings

Observation 2. There is no FPAUS for sampling center string uniformly at random, unless $NP = RP$.

3 Counting and Sampling with Fixed Parameters

Observation 2 is evidence that determining the number of center strings is computationally difficult to approximate and therefore, to achieve progress on the existence of an FPRAS we consider the problem where one of the parameters is fixed. We first determine if the decision problem corresponding to the restricted version of #CLOSEST STRING can be solved in polynomial time, since otherwise we could show there does not exist an FPRAS by using similar argument to that for Observation 2. In order for the existence of an FPRAS to be possible for some restricted version of the #CLOSEST STRING problem, the corresponding decision problem has to be *fixed parameter tractable (FPT)*, meaning there exists an algorithm that is exponential only in the size of a fixed parameter while polynomial in the remaining, unfixed parameters.

CLOSEST STRING is trivially FPT when the parameter ℓ is fixed since the enumeration algorithm that tries all possible length ℓ strings is polynomial-time when ℓ is fixed. Gramm et al. prove CLOSEST STRING is FPT when n or d is fixed [12]. These results imply that restricting #CLOSEST STRING such that at least one of ℓ, d, or n is fixed leads to a problem that may have an FPRAS. The $O(d(|\Sigma|\ell e)^d)$ algorithm that determines the set of strings that have distance at most d from s_1 is an FPRAS when d is fixed. These enumeration algorithms prove the existence of an FPAUS for the problem of sampling center strings u.a.r. when ℓ or d is fixed. Next, we prove there exists both an FPRAS and an FPAUS for the respective problems of counting and sampling center strings when the number of strings is fixed by showing there exists an exact polynomial-time algorithm for counting and sampling center strings when n is fixed. We note that the ILP construction used in the proof of Propositions 2 and 3 was first considered by Gramm et al. [12] in order to demonstrate CLOSEST STRING is FPT when parameterized by n.

Proposition 2. *When the number of strings is fixed and Σ is the binary alphabet there exists a polynomial-time algorithm for #CLOSEST STRING and for sampling center strings u.a.r.*

Proof. The goal is to give an integer linear programming (ILP) formulation such that the number of variables depends only on the value of n. Let $S = \{s_1, \ldots, s_n\}$ denote a set of n binary strings, each of length ℓ, and denote each string s_j as $s_j(1) \cdots s_j(\ell)$. Let $\mathcal{C}_S$ denote the set of center strings for the set S. Given a set of n strings of length ℓ, we associate an $n \times \ell$ matrix whose columns are the n strings. We refer to the *columns* of an instance of #CLOSEST STRING as the columns of this corresponding matrix. Each column can be one of 2^n possible vectors. By considering column types, i.e., all possible 2^n vectors that can compose a column, we show how #CLOSEST STRING restricted to the binary alphabet can be formulated as an ILP with 2^n variables.

Let $\vec{b} = [b_1, \ldots, b_n]^T$ correspond to one particular column type, and let $\mathcal{P}(\vec{b}) = \{i \mid [s_1(i), s_2(i), \ldots, s_n(i)]^T = \vec{b}\}$. Therefore, $|\mathcal{P}(\vec{b})|$ is equal to the number of columns in our $n \times \ell$ matrix that equal $\vec{b}$. For a string $u = u(1) \ldots u(\ell)$, let $\rho_u(\vec{b})$ be the number of elements $j \in \mathcal{P}(\vec{b})$ for which $u(j) = 0$. Hence, $|\mathcal{P}(\vec{b})| - \rho_u(\vec{b})$ is the number of elements $j \in \mathcal{P}(\vec{b})$ for which $u(j) = 1$. Therefore, u has distance at most d from s_i if and only if

$$\sum_{\vec{b} \in \{0,1\}^n} (b_i \rho_u(\vec{b}) + (1 - b_i)(|\mathcal{P}(\vec{b})| - \rho_u(\vec{b}))) \leq d.$$

Thus, $\mathcal{C}_S$ is nonempty if and only if there is a feasible integer solution to

$$\sum_{\vec{b} \in \{0,1\}^n} (2b_i - 1)\rho(\vec{b}) + (1 - b_i)|\mathcal{P}(\vec{b})| \leq d \qquad (1 \leq i \leq n), \tag{1}$$

$$0 \leq \rho(\vec{b}) \leq |\mathcal{P}(\vec{b})| \qquad (\vec{b} \in \{0,1\}^n), \tag{2}$$

for variables $\rho(\vec{b}) \in \mathbb{Z}$.

Assuming there is a solution to this integer program, we know the number of strings u corresponding to each such solution. It is exactly

$$\prod_{\vec{b} \in \{0,1\}^n} \binom{|\mathcal{P}(\vec{b})|}{\rho(\vec{b})}. \tag{3}$$

To sample the strings in $\mathcal{C}_S$.

1. Generate random values of each of the numbers $\rho(\vec{b})$, for each $\vec{b}$, with the correct probabilities—proportional to the formula in (3).
2. Sample exactly uniformly from the vectors corresponding to the given $\rho(\vec{b})$ by choosing subsets of given size uniformly at random.

This immediately shows that for fixed n there is a polynomial-time algorithm for perfect sampling, since one can run through all possible values of the set of variables $\rho(\vec{b})$ (there are at most ℓ values for each of these, hence at most ℓ^{2^n} values in total) and for each one, compute the value of (3). Then choose between one of these polynomially many values with the required probability, and then perform step (2). □

A slight modification of the proof for the previous proposition leads to the following stronger result that eliminates the requirement that the alphabet is binary.

Proposition 3. *When the number of strings is fixed there exists a polynomial-time algorithm for #CLOSEST STRING and for sampling center strings u.a.r.*

Proof. The goal is to give an ILP formulation such that the number of variables depends only on the value of n. Let $S = \{s_1, \ldots, s_n\}$ denote a set of n strings from the alphabet Σ, each of length ℓ, and denote each string s_j as $s_j(1) \cdots s_j(\ell)$. Let $\mathcal{C}_S$ denote the set of center strings for the set S. The nth Bell number is the number of partitions of a set of size n. Using the same terminology defined in the proof of Proposition 2, there exists at most $B_n \leq n!$ unique column types, where B_n is nth Bell number.

Let $\vec{b} = [b_1, \ldots, b_n]^T$ correspond to one particular column type, and let $\mathcal{P}(\vec{b}) = \{i \mid [s_i(1), s_i(2), \ldots, s_i(n)]^T = \vec{b}\}$. For a string $u = u(1), \ldots, u(\ell)$ and bit $\nu \in \Sigma$, let $\rho_u(\vec{b}, \nu)$ be the number of positions $j \in P(\vec{b})$ for which $u(j) = \nu$. Then, by the same arguments as in Proposition 2, the set $\mathcal{C}_S$ is nonempty if and only if there is a feasible integer solution to

$$\sum_{\vec{b} \in \{0,1\}^n} \sum_{\nu \in \left(\Sigma - \mu(\vec{b}, i)\right)} \rho(\vec{b}, \nu) \leq d \qquad (1 \leq i \leq n), \tag{4}$$

$$0 \leq \rho(\vec{b}, \nu) \leq |\mathcal{P}(\vec{b})|, \tag{5}$$

for the variables $\rho(\vec{b}, \nu)$, where $\mu(\vec{b}, i)$ is the ith bit of $\vec{b}$. Sampling and counting the solutions to this ILP can be done equivalently to sampling and counting the solutions to the ILP given in the proof of Proposition 2. □

4 Conclusion

Counting and sampling from a specific distribution is a well-studied area in discrete mathematics and theoretical computer science that has been useful for the study of combinatorial problems. The problem of counting and sampling center strings has a natural application to several bioinformatic problems, including motif recognition. We prove #CLOSEST STRING is #P-complete and does not exist an FPRAS, the problem of sampling center strings u.a.r. does not have an FPAUS, and any natural restriction of these counting and sampling problems yields an FPRAS and an FPAUS, respectively. This work suggests some open areas of study, including developing a more efficient sampling and counting algorithms, investigating the existence of a rapidly mixing chain for more restricted sampling problems, or proving hardness results that show the non-existence of a rapidly mixing chain when a single parameter is fixed.

Acknowledgments

The authors would like to thank Nick Wormald for his discussions and insights concerning the results presented in this paper. They gratefully acknowledge research support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] A. Ben-Dor, G. Lancia, J. Perone, and R. Ravi, "Banishing Bias from Consensus Strings," *Proc. Eighth Ann. Symp. Combinatorial Pattern Matching (CPM)*, pp 247-261, 1997.

[2] C. Boucher and D.G. Brown, "Detecting Motifs in a Large Data Set: Applying Probabilistic Insights to Motif Finding," *Proc. First Conf. Bioinformatics and Computational Biology (BICoB)*, pp. 139-150, 2008.

[3] X. Deng, G. Li, Z. Li, B. Ma, and L. Wang, "Genetic Design of Drugs without Side-Effects," *SIAM J. Computing*, vol. 32, no. 4, pp. 1073-1090, 2003.

[4] J. Dopazo, A. Rodríguez, J.C. Sáiz, and F. Sobrino, "Design of Primers for PCR Amplification of Highly Variable Genomes," *Computer Applications in the Biosciences*, vol. 9, pp. 123-125, 1993.

[5] M. Dyer, "Approximate Counting by Dynamic Programming," *Proc. 35th Ann. ACM Symp. Theory of Computing (STOC)*, pp. 693-699, 2003.

[6] M. Dyer and A. Frieze, "Randomly Colouring Graphs with Lower Bounds on Girth and Maximum Degree," *Proc. IEEE 42nd Symp. Foundations of Computer Science (FOCS)*, pp. 579-587, 2001.

[7] M. Dyer, A. Frieze, and M. Jerrum, "Approximately Counting Hamilton Paths and Cycles in Dense Graphs," *SIAM J. Computing*, vol. 27, no. 5, 1262-1272, 1998.

[8] M. Dyer, A. Frieze, and M. Jerrum, "On Counting Independent Sets in Sparse Graphs," *SIAM J. Computing*, vol. 31, no. 5, pp. 1527-1541, 2002.

[9] M.R. Fellows, J. Gramm, and R. Neidermeier, "On the Parameterized Intractability of Closest Substring and Related Problems," *Proc. 19th Ann. Symp. Theoretical Aspects of Computer Science (STACS)*, pp. 262-273, 2002.

[10] M.R. Fellows, J. Gramm, and R. Niedermeier, "On the Parameterized Intractability of Motif Search Problems," *Combinatorica*, vol. 26, pp. 141-167, 2006.

[11] M. Frances and A. Litman, On Covering Problems of Codes," *Theoretical Computer Science*, vol. 30, no. 2, pp. 113-119, 1997.

[12] J. Gramm, R. Niedermeier, and P. Rossmanith, "Fixed-Parameter Algorithms for Closest String and Related Problems," *Algorithmica*, vol. 37, no. 1, pp. 25-42, 2003.

[13] T.P. Hayes and E. Vigoda, "A Non-Markovian Coupling for Randomly Sampling Colorings," *Proc. IEEE 44th Ann. Symp. Foundations of Computer Science (FOCS)*, pp. 618-627, 2003.

[14] M.R. Jerrum and A. Sinclair, "Approximating the Permanent," *SIAM J. Computing*, vol. 18, no. 6, pp. 1149-1178, 1989.

[15] M.R. Jerrum, L.G. Valiant, and V. Vazirani, "Random Generation of Combinatorial Structures from a Uniform Distribution," *Theoretical Computer Science*, vol. 43, pp. 169-188, 1986.

[16] J.K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang, "Distinguishing String Selection Problems," *Information and Computation*, vol. 185, pp. 41-55, 2003.

[17] M. Li, B. Ma, and L. Wang, "Finding Similar Regions in Many Strings," *J. Computer and System Sciences*, vol. 65, no. 1, pp. 73-96, 2002.

[18] K. Lucas, M. Busch, S. Össinger, and J.A. Thompson, "An Improved Microcomputer Program for Finding Gene- and Gene Family-Specific Oligonucleotides Suitable as Primers for Polymerase Chain Reactions or as Probes," *Computer Applications in the Biosciences*, vol. 7, pp. 525-529, 1991.

[19] B. Ma, "A Polynomial Time Approximation Scheme for the Closest Substring Problem," *Proc. 11th Ann. Symp. Combinatorial Pattern Matching (CPM)*, pp. 99-107, 2000.

[20] B. Ma and X. Sun, "More Efficient Algorithms for Closest String and Substring Problems," *Proc. 12th Ann. Int'l Conf. Research in Computational Molecular Biology (RECOMB)*, pp. 396-409, 2008.

[21] M. Molloy, "The Glauber Dynamics on Colorings of a Graph with High Girth and Maximum Degree," *Proc. 36th Ann. ACM Symp. Theory of Computing (STOC)*, pp. 91-98, 2002.

[22] B. Morris and A. Sinclair, "Random Walks on Truncated Cubes and Sampling 0-1 Knapsack Solutions," *Proc. 40th Ann. Symp. Foundations of Computer Science (FOCS)*, pp. 230-240, 1999.

[23] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge Univ. Press, 1995.

[24] G. Pavesi, G. Mauri, and G. Pesole, "An Algorithm for Finding Signals of Unknown Length in DNA Sequences," *Bioinformatics*, vol. 17, pp. S207-S214, 2001.

[25] P. Pevzner and S. Sze, "Combinatorial Approaches to Finding Subtle Signals in DNA Strings," *Proc. Eighth Int'l Conf. Intelligent Systems for Molecular Biology (ISMB)*, pp. 269-278, 2000.

[26] V. Proutski and E.C. Holme, "Primer Master: A New Program for the Design and Analysis of PCR Primers," *Computer Applications in the Biosciences*, vol. 12, pp. 253-255, 1996.

[27] M. Tompa et al., "Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites," *Nature Biotechnology*, vol. 23, no. 1, pp. 137-144, 2005.