On the Hardness of Counting and Sampling Center Strings

Christina Boucher[1] and Mohamed Omar[2]

[1] David R. Cheriton School of Computer Science,
University of Waterloo
cabouche@cs.uwaterloo.ca
[2] Department of Mathematics,
University of California, Davis
momar@math.ucdavis.edu

Abstract. Given a set S of n strings, each of length ℓ, and a non-negative 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 unique 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 an FPRAS. We show equivalent results for the problem of efficiently sampling center strings uniformly at random.

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 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 *et al.* [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 P = 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

E. Chavez and S. Lonardi (Eds.): SPIRE 2010, LNCS 6393, pp. 127–134, 2010.

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 unique 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 unique 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-colouring [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 our knowledge this is the first consideration of this problem but is motivated by problems addressing the analysis and 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 a *fully-polynomial-time randomized approximation scheme (FPRAS)* which produces with high probability an approximation of arbitrarily small error in time that is polynomial with respect to both the size of the problem and 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. In absence of the existence of an FPRAS or FPAUS for a general counting or sampling problem, interest remains in showing an FPRAS or 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, the most natural restricted versions of #Closest String lead to the existence of an FPRAS. Similarly, we show that although there does not exist an FPAUS for sampling center strings uniformly at random (u.a.r.), restricting interest to this 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. 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. 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 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.

Proposition 1. *#Closest String is #P-complete.*

Proof. First, we present the reduction of a single 3-clause, and then extend it to a general 3-CNF formula. For a 3-clause ω over the variables in X we define the string $s = s(1), \ldots, s(2n)$ by:

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

Note that s is defined via its blocks and exactly three of them are repetitions. Let $\phi : X \to \{0,1\}^{2n}$ be the one-to-one mapping from X onto the set of all binary strings of length $2n$ as defined above (*i.e.* the transformation from ω to s). It is shown in [11] show that for any 3-clause over the variable $x_1, \ldots, x_n$, denoted as ω, and assignment $v \in \{0,1\}^n$, v satisfies ω if and only if $\phi(v)$ has distance at most n from s.

Let $F = \omega_1 \wedge \ldots \wedge \omega_t$ be a 3-CNF formula over the variables $x_1, \ldots, x_n$ then $v \in \{0,1\}^n$ is a satisfying assignment to F if and only if $\phi(v)$ has distance less than n from each of the strings in the set $\{s_1, \ldots, s_t\}$, where $s_i = \phi(\omega_i)$. Hence, the number of satisfying formulae to F is parsimonious to the number of center strings to the set $\{s_1, \ldots, s_t\}$. □

Randomization can be quite powerful in achieving an approximation scheme for several #P-complete problems. Jerrum *et al.* show 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 a FPRAS for the problem – since the existence implies the approximability of the problem. Unfortunately, the existence of a 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, page 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.

When ℓ is fixed the enumeration algorithm that attempts all $|\Sigma|^\ell$ strings is an FPRAS. The $O(d(|\Sigma|\ell e)^d)$ algorithm that determines which strings from the set of strings that have distance at most d from s_1 proves the existence of an FPRAS when d is fixed. These enumeration algorithms prove the existence of a FPAUS for the problem of sampling center strings u.a.r. when ℓ or d is fixed. Next we show 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. In fact, we show a stronger result: that there exists an exact polynomial-time algorithm for counting and sampling center strings when n is fixed.

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 can think of these strings as a $n \times \ell$ matrix. We refer to the *columns* of an instance of #Closest String as the the columns of a matrix. There are 2^n possible number of unique columns. Using the column types, we show how #Closest String restricted to the binary alphabet can be formulated as an ILP with 2^n variables.

Let $\boldsymbol{b} = [b_1, \ldots, b_n]^T$ correspond to one particular column type, and let $\mathcal{P}(\boldsymbol{b}) = \{i \mid (s_i(1), s_i(2), \ldots, s_i(n)) = \boldsymbol{b}\}$ (*i.e.* $\mathcal{P}(\boldsymbol{b})$ is the set of positions in S which are equal to $\boldsymbol{b}$). Therefore, $|\mathcal{P}(\boldsymbol{b})|$ is equal to the number of positions in S that are equal to $\boldsymbol{b}$.

For a string $u = u(1), \ldots, u(\ell)$, let $\rho(\boldsymbol{b})$ be the number of positions that are equal to $\boldsymbol{b}$ where u is equal to 0 (*i.e.* $j \in \mathcal{P}(\boldsymbol{b})$ and $u(j) = 0$). Hence, $|\mathcal{P}(\boldsymbol{b})| - \rho(\boldsymbol{b})$ is the number of positions that are equal to $\boldsymbol{b}$

Therefore, u has distance at most d from s_i if and only if

$$\sum_{\boldsymbol{b}} b_i \rho(\boldsymbol{b}) + (1 - b_i)(|\mathcal{P}(\boldsymbol{b})| - \rho(\boldsymbol{b})) \leq d.$$

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

$$\sum_{\boldsymbol{b}} (2b_i - 1)\rho(\boldsymbol{b}) + (1 - b_i)|\mathcal{P}(\boldsymbol{b})|) \leq d \qquad (1 \leq i \leq n) \tag{1}$$

$$0 \leq \rho(\boldsymbol{b}) \leq \mathcal{P}(\boldsymbol{b}) \tag{2}$$

for the variables $\rho(\boldsymbol{b})$.

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

$$\prod_{\boldsymbol{b}} \binom{\mathcal{P}(\boldsymbol{b})}{\rho(\boldsymbol{b})} \tag{3}$$

where the product is over all feasible values of the variables $\rho(\boldsymbol{b})$.

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

(i) generate random values of each of the numbers $\rho(\boldsymbol{b})$, for each $\boldsymbol{b}$, with the correct probabilities — proportional to the formula in (3)

(ii) sample exactly uniformly from the vectors corresponding to the given $\rho(\boldsymbol{b})$ 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 $\rho(\boldsymbol{b})$ (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 (ii). □

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. See the Appendix for the details of the proof.

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.*

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 a FPRAS, the problem of sampling center strings u.a.r. does not have a FPAUS, and any natural restriction of these counting and sampling problems yields an FPRAS and 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.

Acknowledgements

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

References

1. Ben-Dor, A., Lancia, G., Perone, J., Ravi, R.: Banishing bias from consensus strings. In: Hein, J., Apostolico, A. (eds.) CPM 1997. LNCS, vol. 1264, pp. 247–261. Springer, Heidelberg (1997)
2. Boucher, C., Brown, D.G.: Detecting motifs in a large data set: applying probabilistic insights to motif finding. In: Rajasekaran, S. (ed.) BICoB 2009. LNCS, vol. 5462, pp. 139–150. Springer, Heidelberg (2009)
3. Deng, X., Li, G., Li, Z., Ma, B., Wang, L.: Genetic design of drugs without side-effects. SIAM Journal on Computing 32(4), 1073–1090 (2003)
4. Dopazo, J., Rodríguez, A., Sáiz, J.C., Sobrino, F.: Design of primers for PCR amplification of highly variable genomes. Computer Applications in the Biosciences 9, 123–125 (1993)
5. Dyer, M.: Approximate counting by dynamic programming. In: Proc. of STOC, pp. 693–699 (2003)
6. Dyer, M., Frieze, A.: Randomly colouring graphs with lower bounds on girth and maximum degree. In: Proc. of FOCS, pp. 579–587 (2001)
7. Dyer, M., Frieze, A., Jerrum, M.: Approximately counting hamilton paths and cycles in dense graphs. SIAM Journal on Computing 27(5), 1262–1272 (1998)
8. Dyer, M., Frieze, A., Jerrum, M.: On counting independent sets in sparse graphs. SIAM Journal on Computing 31(5), 1527–1541 (2002)
9. Fellows, M.R., Gramm, J., Neidermeier, R.: On the parameterized intractability of CLOSEST SUBSTRING and related problems. In: Alt, H., Ferreira, A. (eds.) STACS 2002. LNCS, vol. 2285, pp. 262–273. Springer, Heidelberg (2002)
10. Fellows, M.R., Gramm, J., Niedermeier, R.: On the parameterized intractability of motif search problems. Combinatorica 26, 141–167 (2006)
11. Frances, M., Litman, A.: On covering problems of codes. Theoretical Computer Science 30(2), 113–119 (1997)
12. Gramm, J., Niedermeier, R., Rossmanith, P.: Fixed-parameter algorithms for closest string and related problems. Algorithmica 37(1), 25–42 (2003)
13. Hayes, T.P., Vigoda, E.: A non-markovian coupling for randomly sampling colorings. In: Proc. of FOCS, pp. 618–627 (2003)
14. Jerrum, M.R., Sinclair, A.: Approximating the permanent. SIAM Journal on Computing 18(6), 1149–1178 (1989)
15. Jerrum, M.R., Valiant, L.G., Vazirani, V.: Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science 43 (1986)
16. Lanctot, J.K., Li, M., Ma, B., Wang, S., Zhang, L.: Distinguishing string selection problems. Information and Computation, 41–55 (2003)
17. Li, M., Ma, B., Wang, L.: Finding similar regions in many strings. Journal of Computer and System Sciences 65(1), 73–96 (2002)
18. Lucas, K., Busch, M., Össinger, S., Thompson, J.A.: 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 7, 525–529 (1991)
19. Ma, B.: A polynomial time approximation scheme for the closest substring problem. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 99–107. Springer, Heidelberg (2000)
20. Ma, B., Sun, X.: More efficient algorithms for closest string and substring problems. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 396–409. Springer, Heidelberg (2008)

21. Molloy, M.: The glauber dynamics on colorings of a graph with high girth and maximum degree. In: Proc. of STOC, pp. 91–98 (2002)
22. Morris, B., Sinclair, A.: Random walks on truncated cubes and sampling 0-1 knapsack solutions. In: Proc. of FOCS, pp. 230–240 (1999)
23. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
24. Pavesi, G., Mauri, G., Pesole, G.: An algorithm for finding signals of unknown length in DNA sequences. Bioinformatics 17, S207–S214 (2001)
25. Pevzner, P., Sze, S.: Combinatorial approaches to finding subtle signals in DNA strings. In: Proc. of 8th ISMB, pp. 269–278 (2000)
26. Proutski, V., Holme, E.C.: Primer master: A new program for the design and analyiss of PCR primers. Computer Applications in the Biosciences 12, 253–255 (1996)
27. Tompa, M., Li, N., Bailey, T.L., Church, G.M., De Moor, B., Eskin, E., Favorov, A.V., Frith, M.C., Fu, Y., Kent, W.J., et al.: Assessing computational tools for the discovery of transcription factor binding sites. Nature Biotechnology 23(1), 137–144 (2005)

Appendix

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. Let α be a letter in Σ. The nth Bell number is the number of partitions of a set of size n. Without loss of generality, we assume that the first string is equal to α^ℓ, since any set of strings can be trivially converted to an equivalent set where this is true. 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 $\boldsymbol{b} = [b_1, \ldots, b_n]^T$ correspond to one particular column type, and let $\mathcal{P}(\boldsymbol{b}) = \{i \mid (s_i(1), s_i(2), \ldots, s_i(n)) = \boldsymbol{b}\}$. Let $|\mathcal{P}(\boldsymbol{b})|$ be equal to the number of positions in S that are equal to $\boldsymbol{b}$. For a string $u = u(1), \ldots, u(\ell)$, let $\rho(\boldsymbol{b}, \nu)$ be the number of positions that are equal to $\boldsymbol{b}$ where u is equal to ν and $\nu \in \Sigma$.

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

$$\sum_{\boldsymbol{b}} \sum_{\nu \in (\Sigma - \nu(\boldsymbol{b}, i))} \rho(\boldsymbol{b}, \nu) \leq d \qquad (1 \leq i \leq n) \tag{4}$$

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

for the variables $\rho(\boldsymbol{b}, \nu)$, where $\nu(\boldsymbol{b}, i)$ is symbol of string i at column $\boldsymbol{b}$. 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. □