An $O(n \log n)$ Algorithm for finding Dissimilar Strings

Sarmad Abbasi [1]

Department of Computer Science, Rutgers University, Piscataway NJ 08855, U.S.A.

Anirvan Sengupta

Theory Group, Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974 U.S.A.

Let Σ be a finite alphabet and $x \in \Sigma^n$. A string $y \in \Sigma^m$ is said to be k-dissimilar to x, if no k length substring of x is equal to any k length substring of y. We present an $O(n \log n)$ algorithm which on input $x \in \Sigma^n$ and an integer $m \leq n$ outputs an integer k and $y \in \Sigma^m$ such that:

- y is k-dissimilar to x.
- There does not exist a string z of length m which is $k-1$ dissimilar to x.

Key words: algorithms, analysis of algorithms, combinatorial problems, computational molecular biology, probabilistic method, Lovász local lemma.

1 Introduction

Let Σ be a finite alphabet and $x \in \Sigma^n$. A string $y \in \Sigma^m$ is said to be k-dissimilar to x, if no length k substring of x is equal to any k length substring of y; that is,

$$x_i x_{i+1} \cdots x_{i+k} \neq y_j y_{j+1} \cdots y_{j+k}$$

for all $1 \leq i \leq n-k$ and $1 \leq j \leq m-k$.

We let $D(x, m)$ denote the minimum k such that there exists a string $y \in \Sigma^m$ which is k-dissimilar to x. Motivated by a problem in molecular biology we consider the following question.

[1] The author was partially supported by an NSF grant.

Dissimilarity Problem: Given $x \in \Sigma^n$ and an integer m, compute $k = D(x, m)$ and a $y \in \Sigma^m$ such that, y is k-dissimilar to x.

We present an $O(n \log n)$ algorithm for the Dissimilarity Problem. The analysis of our algorithm depends on upper bounding the following combinatorial function.

$$\max_{x \in \Sigma^n} D(x, m).$$

This function seems to be of independent interest. We describe a $\log_C n + \log_C \log_C n + O(1)$ upper bound on this function using the Probabilistic Method [1] (where C denotes the size of the alphabet Σ). This bound is fairly close to the $\log_C n + O(1)$ lower bound. However, closing the gap between these bounds seems to an intriguing and challenging combinatorial problem.

The remaining of the paper is organized as follows: In the next section we describe the motivation of our problem. In section 3 we present an algorithm for the dissimilarity problem. In section 4 we give the analysis of our algorithm by upper bounding the function $D(x, m)$. The last section discusses some further aspects of this problem and gives some concluding remarks.

2 Motivation

Computational Molecular Biology is an extremely fast growing field. This area offers many interesting and challenging problems in computer science [6]. The motivation of our problem also comes from molecular biology.

A DNA molecule is a polymer consisting of two interwound helical strands [5]. Each strand is a sequence of *nucleotides* drawn from the set $\{A, T, G, C\}$. The two strands are complementary in the sense that each A on one strand is bonded to its complementary nucleotide T on the other, and each C is bonded to a G. In certain experiments performed by Norris and Nagaraj [7], a strand of DNA is given: One is asked to design an oligonucleotide (a smaller strand of DNA) of a certain length, which will not easily bind to the given DNA. Therefore, it is desirable that no long stretch on the oligonucleotide should be complementary to a long stretch on the DNA. A stretch of length k on the oligonucleotide, which is complementary to a stretch of length k on the DNA is called a k-homology in the molecular biology literature. Thus if we want to design an oligonucleotide which does not bind to a given DNA strand we must avoid any k-homologies between the DNA and the oligonucleotide (for large k). This problem reduces to finding dissimilar strings over the alphabet $\{A, T, G, C\}$ and complementing the obtained solution; that is, replacing each

occurrence of A with T and vice versa and treating G and C similarly. Hence the problem of finding non-binding oligonucleotides abstracted as described above reduces to our Dissimilarity Problem.

3 The Algorithm

First we present an algorithm, which takes a string $x \in \Sigma^n$ and two positive integers k and m as input and decides if $D(x,m) \leq k$. Furthermore, if $D(x,m) \leq k$ it provides us with an encoding of all the strings of length m which are k-dissimilar to x. The algorithm runs in time $O(C^k + n)$, where $C = |\Sigma|$.

For any string $v = v_1 v_2 \cdots v_{k-1}$ of length $k-1$ and $a \in \Sigma$, we define $\mathrm{succ}(v,a) = v_2 \ldots v_{k-1} a$. Let $G_k = (V,E)$ be a directed graph with the vertex set $V = \Sigma^{k-1}$. For every $v \in V$ and $a \in \Sigma$ there is a directed edge from vertex v to $succ(v,a)$. The graph G_k has C^{k-1} vertices with each vertex having indegree and outdegree equal to C. For any $y \in \Sigma^m$ we identify y with a walk $W(y)$ on G_k given by,

$$W(y) = y_1 y_2 \cdots y_{k-1}, y_2 y_3 \cdots y_k, \ldots, y_{m-k+1} y_{m-k+2} \cdots y_m.$$

$W(y)$ is a valid walk on G_k, since $(y_i \cdots y_{i+k-1}, y_{i+1} \cdots y_{i+k})$ is an edge of G_k.

A string $u = u_1 \cdots u_k$ of length k corresponds to an edge $(u_1 \cdots u_{k-1}, u_2 \cdots u_k)$ of G_k. Furthermore, u is a substring of y if and only if $(u_1 \cdots u_{k-1}, u_2 \cdots u_k)$ is an edge on the walk $W(y)$. This simple observation leads us to the following lemma.

Lemma 1 *A string y is k-dissimilar to x if and only if $W(x)$ and $W(y)$ do not share a common edge on G_k.*

PROOF. If y is not k-dissimilar to x then there exist an i and j such that

$$x_i \cdots x_{i+k} = y_j \cdots y_{j+k}.$$

where $1 \leq i \leq n-k$ and $1 \leq j \leq m-k$. Hence $(x_i \cdots x_{i+k-1}, x_{i+1} \cdots x_k)$ is an edge on both $W(x)$ and $W(y)$. On the other hand if $W(y)$ and $W(x)$ share an edge $(u_1 \cdots u_{k-1}, u_2 \cdots u_k)$ then $u = u_1 \cdots u_k$ is a substring of both x and y. Hence y is not k-dissimilar to x. □

The above lemma immediately suggests the following algorithm.

Algorithm A

Input: $x \in \Sigma^n$ and two positive integers k and m.

Output: "Yes" If $D(x,m) \leq k$, otherwise "No".

Step 1: Construct the graph G_k.

Step 2: Delete all the edges corresponding to $W(x)$ to construct $G_k(x)$.

Step 3: If the remaining graph; $G_k(x)$, contains a walk of length $m-k+1$ Output "Yes", otherwise Output "No".

Note that since G_k has C^{k-1} vertices and C^k edges, therefore it can be constructed in time $O(C^k)$. Step 2 of the algorithm takes $O(n)$ steps. While the last step takes $O(C^k)$ steps. Since, if $m-k+1 > C^{k-1}$ we only need to check if $G_k(x)$ contains a cycle. Otherwise, the problem of finding a walk of length l in a graph (or reporting that none exists) can be solved in time linear the number of edges in the graph. Hence the entire procedure takes time $O(n + C^{k-1})$.

We solve the Dissimilarity Problem by using Algorithm A for $k = 1, \cdots$, repeatedly until the algorithm returns "Yes". More precisely, $D(x,m)$ is the first value k_f, for which the Algorithm A outputs "Yes". Once a the value of $D(x,m)$ is known, we can output a string corresponding to any walk of length $m-k+1$ as y. This solves the Dissimilarity problem completely and hence we can proceed with the analysis of this algorithm.

4 Analysis

The total time taken by the algorithm described above is bounded by,

$$T(n) \leq \sum_{k=1}^{k_f} O(C^k + n)$$

The hidden constants in the O-notation do not depend on the index k. Furthermore, since $k_f = D(x,m)$ we have,

$$T(n) = O(nD(x,m)) + \sum_{k=1}^{D(x,m)} O(C^k) = O(nD(x,m) + C^{D(x,m)}).$$

To complete the analysis we need an upper bound for $D(x,m)$. We prove two results bounding $D(x,m)$. To obtain these bounds we make use of the

probabilistic method. Our first bound follows from a very simply probabilistic argument.

Theorem 2 *For any* $x \in \Sigma^n$ *with* $|\Sigma| = C$

$$D(x, m) \leq \lceil \log_C n + \log_C m \rceil.$$

PROOF. We choose $y \in \Sigma^m$ by choosing each y_i in Σ uniformly and independently. For $1 \leq i \leq m - k$ and $1 \leq j \leq n - k$ let $A_{i,j}$ the event that

$$y_j y_{j+1} \cdots y_{j+k} = x_i x_{i+1} \cdots x_{i+k}.$$

Then

$$\Pr[A_{i,j}] = C^{-k}.$$

Hence the

$$\Pr[\bigvee_{i,j} A_{i,j}] \leq \sum_{i,j} \Pr[A_{i,j}] = (n - k)(m - k)C^{-k}$$

If $k = \lceil \log_C n + \log_C m \rceil$ then the above probability is less than 1. Hence there exists $y \in \Sigma^m$ such that,

$$y_j y_{j+1} \cdots y_{j+k} \neq x_i x_{i+1} \cdots x_{i+k};$$

that is, y is k-dissimilar to x. $\square$

The above theorem gives us an *a priori* upper bound on $D(x, m)$. Which proves an $O(nm)$ time bound on the complexity of our algorithm. This bound is quite good if m is much smaller compared to n. However, if we take advantage of the fact that each event $A_{i,j}$ is independent of most of the other events, the above bound can be improved significantly: in fact the bound can be made independent of m. To this end we make use of the *Lovász local lemma* [4], which deals with events which are mostly independent of each other. In a typical application the Local Lemma proves the existence of certain structures without supplying any efficient way of finding these structure. Beck [2] has converted some these applications into polynomial time algorithms. Our algorithm is a an example of such an application. However, in our case proving the existence dissimilar strings actually proves the efficiency of our algorithm.

For our purpose we only state the symmetric form of the Local Lemma. For a more general setting see [1].

Lemma 3 (Lovász Local Lemma) *Let $S = \{B_1, \ldots, B_N\}$ be a set events in an arbitrary probability space with $\Pr[B_i] \leq p$. Suppose for each B_i there exists a set $S_i \subseteq S - \{B_i\}$ of size at most d, such that B_i is independent of all the events in $S - (S_i \cup \{B_i\})$. Furthermore, if*

$$ep(d+1) < 1$$

then $\Pr[\bigwedge_{i=1}^{N} \overline{B}_i] > 0$.

PROOF. See [1]. □

Theorem 4 *For every $x \in \Sigma^n$ and for every $m \geq 1$*

$$D(x,m) \leq \log_C n + \log_C \log_C n + O(1).$$

PROOF. As in Theorem 1 we choose $y \in \Sigma^m$ by choosing each y_i in Σ uniformly and independently. We let and let $A_{i,j}$ be the event

$$x_i \cdots x_{i+k} = y_j \cdots y_{j+k}.$$

Note that each event $A_{i,j}$ is independent of the set of events

$$\{A_{r,s} : |s - j| \geq k\}.$$

This is because the events $A_{r,s}$ do not depend on $y_j, \ldots, y_{j+k}$, whereas $A_{i,j}$ depends only on the these values of y. Therefore for each $A_{i,j}$ we can define

$$S_{i,j} = \{A_{r,s} : |s - j| < k, 1 \leq s \leq n\}$$

to satisfy the hypothesis of the local lemma. The size of each $S_{i,j}$ is at most $(k-1)n$. Whereas $\Pr[A_{i,j}] = C^{-k}$. We now observe that if $k = \log_C n + \log_C \log_C n + 4$ then for n large enough $e((k-1)n + 1)C^{-k} < 1$. Therefore, by the local lemma the $\Pr[\bigwedge_{i,j} \overline{A}_{i,j}] > 0$. Hence there exists a y which is k dissimilar to x. □

This theorem completes the analysis of our algorithm. Since

$$T(n) = O(nD(x,m) + C^{D(x,m)})$$

we have $T(n) = O(n \log n)$ as promised.

A natural question to ask is the following: How close is the bound of Theorem 2 to the truth? A simple observation lets us give a $\log_C n + O(1)$ lower bound on the function $D(n, m)$.

Lemma 5 *There exists a string $x \in \Sigma^n$ such that for all $m > \log_C n$,*

$$D(x, m) \geq \log_C n + O(1)$$

PROOF. Assume that $n = k - 1 + C^{k-1}$ and consider the graph G_k defined in section 3. G_k is connected and the indegree of each vertex is equal to the outdegree of each vertex. Therefore by Euler's theorem [3] G_k contains an Euler tour. Let x be a the string corresponding to the Euler tour. Recall that an Euler tour is a walk on G_k which contains all the edges. Hence x contains all the strings of length k. Hence for every $m \geq k$, there is no $y \in \Sigma^m$ which is k-dissimilar to x. Therefore we have,

$$D(x, m) > k \geq \log_C n + O(1).$$

If n is not of the form $k - 1 + C^{k-1}$ then let k_0 be the largest integer such that $k_0 + C^{k_0 - 1} < n$. Let z be any string of length $k_0 + C^{k_0 - 1}$ such that $D(z, m) \geq k_0$. Then for any string y of length $n - (k_0 - 1 + C^{k_0})$ we have,

$$D(zy, m) \geq k_0 = \log_C n + O(1).$$

□

5 Conclusion

As stated the graph $G_k(x)$ is actually an encoding of all the strings which are k-dissimilar to x. This actually proved to be useful [7], since we can apply further heuristics to output k-dissimilar strings which satisfy some other constraints or have some desired properties. Two extensions of our algorithm are immediate. We can find a string y which is k dissimilar to a set of strings $\{x_1, \ldots, x_k\}$. Also, in some applications in molecular biology one wants to avoid say a 90%-homology between substrings of length k. Our algorithm can be modified to search for such strings.

The most intriguing combinatorial problem is to close the gap between the upper and lower bounds on the function $\max_{x \in \Sigma^n} D(x, m)$. Improving the upper bound would improve on the time complexity of the given algorithm. However, this requires a slight modification. Recall that for each iteration (with

input parameter k) of Algorithm A requires time $O(n+C^k)$ instead of $O(C^k)$. Hence the first $\log_C n$ iterations take time $O(n \log n)$. This time can be reduced to $O(n \log \log n)$ by incorporating a binary search. With this strategy improvement on the stated upper bound will significantly improve the analysis of this algorithm. However, to device a linear time algorithm seems to call for a totally different approach.

Acknowledgement

We would like to thank D. Norris and V. H. Nagaraj for bringing this problem to our attention. We would also like to thank Endre Szemerédi for his encouragement and support.

References

[1] N. Alon and J. Spencer, *The Probabilistic Method*, (1992) John Wiley, New York.

[2] J. Beck, *An algorithmic approach to the Lovász Local Lemma I.*, Random Structures and Algorithms **2**, 343-365.

[3] B. Bollobás, *Graph Theory*, (1979) Springer-Verlag, New York.

[4] P. Erdős and L. Lovász, Problems and results on 3-chromatic hypergraphs and some related questions, in A. Hajnal et. al., eds, *Infinite and Finite Sets*, North-Holland, Amsterdam, (1975) 609-628.

[5] D. M. Freifelder, *Molecular Biology*, (1987) Jones and Bartlett, Boston.

[6] R. Karp, *Mapping the Genome: Some Combinatorial Problems Arising in Molecular Biology*, STOC 1993, 278-284.

[7] D. Norris and V. H. Nagaraj, *Personal Communication* 1996.