Algorithms for testing that sets of DNA words concatenate without secondary structure [⋆]

Mirela Andronescu[1], Danielle Dees[2], Laura Slaybaugh[3], Yinglei Zhao[1], Anne Condon[1], Barry Cohen, and Steven Skiena[4]

[1] University of British Columbia, The Department of Computer Science
andrones@cs.ubc.ca, szhao@cs.ubc.ca, condon@cs.ubc.ca

[2] Georgia Institute of Technology, College of Computing
danielle@cc.gatech.edu

[3] Rose-Hulman Institute of Technology, Computer Science Department
slaybalj@rose-hulman.edu

[4] State University of New York at Stony Brook, Computer Science Department
bcohen@cs.sunysb.edu, skiena@cs.sunysb.edu

Abstract. We present an efficient algorithm for determining whether *all* molecules in a combinatorial set of DNA or RNA strands are structure free, and thus available for bonding to their Watson-Crick complements. This work is motivated by the goal of testing whether strands used in DNA computations or as molecular bar-codes are structure free, where the strands are concatenations of short words. We also present an algorithm for determining whether all words in S^*, for some finite set S of equi-length words, are structure-free.

1 Introduction

In search-and-prune DNA computations, many long, DNA strands are created from a small number of short strands, called words. For example, Braich et al. [1] use the words pairs

(TATTCTCACCCATAA, CTATTTATATCCACC), (ACACTATCAACATCA, ACACCTAACTAAACT),
(CCTTTACCTCAATAA, CTACCCTATTCTACT), (CTCCCAAATAACATT, ATCTTTAAATACCCC),

(AACTTCACCCCTATA, TCCATTTCTCCATAT), (TCATATCAACTCCAC, TTTCTTCCATCACAT)

to construct 2^6 strands, namely those obtainable by taking one of the words from each pair, and concatenating these in pair order. (Since there are two strands per pair and 6 pairs in total, the number of strands obtainable in this way is 2^6.)

Word sets are carefully designed using computational or information-theoretic methods, so that the resulting long strands behave well in the computation. One

⋆ This material is based upon work supported by the U.S. National Science Foundation under Grant No. 0130108, by the National Sciences and the Engineering Research Council of Canada, and by the by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-0555.

property of well-designed words is that the long strands formed from these words do not form secondary structures. For example, the design of Braich et al. uses a 3-letter alphabet, ensures that no sequence of length 8 appears more than once in any strand, and more. The success of the design rests (in part) on the hypothesis that words satisfying these design criteria form strands with no secondary structure. While this hypothesis seems plausible, to our knowledge there is currently no rigorous argument that supports the hypothesis, either for the design of Braich et al. or for any other word design currently employed in DNA computing. For this reason, an efficient algorithm that can test whether a given word set produces strands with no secondary structure would be valuable.

With this motivation, we consider the following problem, which we call the **structure freeness problem for combinatorial sets**. We are given a list of t pre-designed pairs of words $\{w(i), \bar{w}(i), 1 \leq i \leq t\}$. (Here the bar notation is not intended to represent Watson-Crick complementation). All words have the same length l, and are strings over $\{A, C, G, T\}$ representing DNA strands with the start of the string corresponding to the 3' end of the strand. Determine whether all of the 2^t strands in the set S denoted by the regular expression $(w(1) + \bar{w}(1))(w(2) + \bar{w}(2)) \ldots (w(t) + \bar{w}(t))$ are structure free, that is, are predicted to have no secondary structure, according to the standard, pseudoknot-free, thermodynamic models.

Zuker and Steigler [14] developed an efficient algorithm, based on dynamic programming, for determining the optimal (lowest energy) secondary structure of an RNA molecule. Their method can also be applied to DNA molecules, given suitable thermodynamic parameters [13]). The running time of their algorithm is $O(n^4)$ on a strand of length n, and an improved algorithm of Lyngso et al. [10] runs in time $O(n^3)$. If the algorithm of Lyngso et al. were applied to each of the words in S independently, the running time would be $O(2^t n^3)$, where $n = tl$ is the length of the strands in S.

In Section 3 we describe an algorithm for the structure freeness problem for combinatorial sets that runs in time $O(n^3)$. The algorithm is a simple generalization of the algorithms of Zuker and Steigler and Lyngso et al. [14, 10]. We also present experimental results that compare the performance of our algorithm with an exhaustive search approach to the structure freeness problem for combinatorial sets, and describe cases where we have found structures in previously reported word designs.

Our algorithm easily generalizes to solve the following problem. Let S_i be a set of strands, all having the same length l_i, for $1 \leq i \leq t$, where the l_i are not restricted to be of the same length. Are all strands in the set $S = S_1 \times S_2 \ldots S_t$ structure free? The running time of our algorithm for this extended problem is $O(max_i |S_i|^2 n^3)$, where here $n = l_1 + l_2 + \ldots + l_t$ is the length of strands in S. The generalized algorithm can be used to verify structure freeness of strands in the combinatorial sets of Faulhammer et al. [5], and others.

Our algorithm can also be used to verify that sets of molecular tags, or barcodes, such as those of Brenner et al. [3], are structure free. In this design, tags are constructed from the set of 8 4mer words

{TTAC, AATC, TACT, ATCA, ACAT, TCTA, CTTT, CAAA}.

This set of words is constructed over a 3-letter alphabet (G's omitted), each word has exactly one "C", and each word differs from the others in three of the four bases. Call this set S. The tags constructed by Brenner et al. are all of the 8^8 strands in the set S^8.

This example motivates the second problem studied in this paper. Suppose that in the future, Brenner et al. need more tags, and use strands from the set S^9 or S^{10}. Even if all strands in the set S^8 are structure free, it might be possible that some strand in S^9 or S^{10} has structure. Ideally, one would like to know that all words in S^* are structure free. Here, S^* is the set of strands obtained by concatenating zero or more copies of strands in S together. Note that S^* contains an infinite number of strands if S is not empty. The set S^* is often called the Kleene closure of S.

Thus, we define the **structure freeness problem for Kleene sets** as follows. Given a set S of words, are all words in S^* structure free? In Section 4 we show that there is a constant m (depending on S) such that if some word of S^* has structure, then some word in S^m has structure. This reduces the structure freeness problem for Kleene sets to the structure freeness problem for combinatorial sets. However, our bound on m is exponential in the size of S and the length of the words in S. Whether or not a better bound on m can be obtained is an interesting open question.

Before getting to our algorithms in Sections 3 and 4, we provide background on thermodynamic models for calculating the free energy of RNA and DNA secondary structures in Section 2, along with an overview of the secondary structure prediction algorithm of Zuker and Steiglitz [14].

1.1 Related Work

Our algorithm for determining structure freeness of combinatorial sets was already developed by Cohen and Skiena [4], but applied to a different problem in that work. They were interested in determining, among the RNA sequences coding for a given protein P, which has the most stable secondary structure?

Other important related work concerns algorithms for predicting the secondary structure of a RNA or DNA strand. In contrast with the Zuker-Steigler algorithm which is the basis for our work and returns the free energy of the most stable structure accessible to a given strand, the partition function approach of McCaskill [11] measures the propensity of a strand to fold in terms of the sum of the Gibb's factors ($Z = \exp(-\Delta G/RT)$) of all folded structures. The quantity Z can be calculated in $O(n^3)$ time, using a dynamic programming algorithm. An extension of the approach used in our paper to use the partition function of McCaskill would provide more insight on the propensity of a strand to fold than does the "all or nothing" approximation of our current algorithm. We believe that the techniques described in our paper can be extended to the partition function approach, and view this as an important next step for this work.

Other thermodynamic models for DNA and RNA structure folding, that could be used as a basis for approaching the structure freeness problem for combinatorial sets, have been proposed by Hartemink et al. [7] and by Rose et al. [12]). In addition, Rose et al. [?] describe a thermodynamic model (based on a statistical zipper model), for estimating folding propensity of a mixture of DNA strands.

2 Background and Notation

Under the appropriate chemical conditions, an RNA or DNA strand may fold upon itself by forming intramolecular bonds between pairs of its bases, as illustrated in Figure 1 below. In what follows, if $s = s_1 s_2 \ldots s_n$ is an RNA sequence and $1 \leq i < j \leq n$, then $i.j$ denotes the base-pairing of s_i with s_j. A secondary structure of s is a set of base pairs such that each base is paired at most once. More precisely, for all $i.j$ and $i'.j'$ in the set, $i = i'$ if and only if $j = j'$. A *pseudoknot* in a secondary structure is a pair of base pairs $i.j, i'.j'$ in the structure with $i < i' < j < j'$. Throughout this work, we restrict our attention to secondary structures that have no pseudoknots.

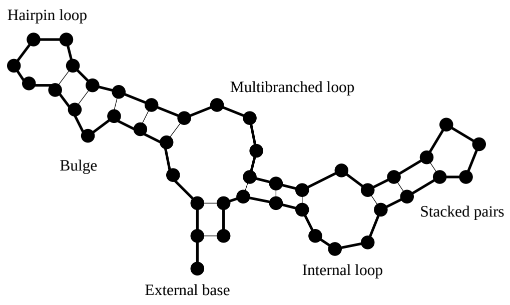

Fig. 1. Secondary structure of an RNA strand. The thick black line indicates the backbone and the thin lines indicate paired bases.

We can classify loops formed by the bonding of base pairs in a structure according to the number of base pairs that they contain. A hairpin loop contains exactly one base pair. An internal loop contains exactly two base pairs. A bulge is an internal loop with one base from each of its two base pairs adjacent. A stacked pair is a loop formed by two consecutive base pairs (i, j) and $(i + 1, j - 1)$. A multibranched loop is a loop that contains more than two base pairs. An external base is a base not contained in any loop. One base pair in any given loop is closest to the ends of RNA strand. This is known as the *exterior* or *closing* pair. All other pairs are *interior*. More precisely, the exterior pair is the one that maximizes $j-i$ over all pairs $i.j$ in the loop.

Speaking qualitatively, bases that are bonded tend to stabilize the RNA, where as unpaired bases form destabilizing loops. The so-called free energy of a secondary structure, measures (in *kcal/mole*) the stability of a secondary structure at fixed temperature - the lower the free energy, the more stable the structure. The free energy of a folded RNA or DNA strand can be estimated as the sum of the free energies of its component loops, if the secondary structure contains no pseudoknots. Through thermodynamics experiments, it has been possible to estimate the free energy of common types of loops (see for example [13] for thermodynamic data for DNA).

2.1 Standard Free Energy Model and Notation

Computational methods for predicting the secondary structure of an RNA or DNA molecule are based on models of the free energy of loops. The parameters of these models are driven in part by current understanding of experimentally determined free energies, and in part by what can be incorporated into an efficient algorithm. The free energy of a loop depends on temperature; throughout we assume that the temperature is fixed. We next summarize the notation used to refer to the free energy of loops, along with some standard assumptions that are incorporated into loop free energy models. We refer to a model that satisfies all of our assumptions as a **standard free energy model**.

- $eS_s(i,j)$. This function gives the free energy of a stacked pair that consists of $i.j$ and $(i+1).(j-1)$. $eS_s(i,j)$ depends on the bases involved in the stack. We use the notation e-Stack(a,b,c,d), where $a,b,c,d \in \{A,C,G,T\}$, to denote the free energy of a stacked pair involving ab (with the 3' end at a) and cd (with the 3' end at c, so that the pairs are (a,d) and (b,c)). Thus, for a strand $s = s_1 s_2 \ldots s_n$, $eS_s(i,j) =$ e-Stack$(s_i, s_{i+1}, s_{j-1}, s_j)$. Because stacked complementary base pairs are stabilizing, eS_s values are negative if s_i is the Watson-Crick complement of s_j and s_{i+1} is the Watson-Crick complement of s_{j-1}. (The values may be negative in other cases, such as when one of the base pairs is G-U, sometimes called a "wobble pair".)
- $eH_s(i,j)$. This function gives the free energy of a hairpin loop closed by $i.j$. We assume that for all but a finite number of small cases, $eH_s(i,j)$ depends only on the length of the loop, s_i and s_j, and on the unpaired bases adjacent to s_i and s_j on the loop. In particular, $eH(i,j) =$ e-Stack$(s_i, s_{i+1}, s_{j-1}, s_j)+$ e-Length$(j-i-1)$, for some function e-Length. Moreover, we assume that the function e-Length(j) grows at most linearly in j.
- $eL_s(i,j,i',j')$. This function gives the free energy of an internal loop or bulge with exterior pair i,j and interior pair i',j'. We assume that $eL_s(i,j,i',j')$ is given by e-Stack$(s_i, s_{i+1}, s_{j-1}, s_j)$ +e-Stack$(s_{j'}, s_{j'+1}, s_{i'-1}, s_i')$ +e-Length$(i'-i-1, j-j'-1)$ where we abuse notation slightly by using e-Length as a function with two parameters here, different from the function e-Length$(\cdot)$ used for hairpins, in this case growing at most linearly in $i+j$.

We omit the free energy of multiloops for brevity in this abstract. The **free energy of a strand s with respect to a fixed secondary structure F** is the sum of the free energies of the loops of F. Sometimes when the strand s is understood, it is convenient to refer simply to the free energy of the structure F. In this paper, we define the **free energy of a strand** s to be the minimum free energy of the strand, with respect to all structures F. (We note that this is a simplification; a better approach would be to define the free energy in terms of the sum of Gibb's factors [11].) We say that a strand s is **structure free** (at a given fixed temperature) if its free energy is greater than 0. If the free energy of s is less than 0, we say that the strand s **has structure**. We note that our algorithm can trivially be modified to test whether any strand in a combinatorial set has free energy below any other threshold, too.

2.2 An Algorithm for Secondary Structure Prediction

Let $W_s(j)$ be the minimum free energy taken over all structures of the strand $s_1 s_2 \ldots s_j$. Our goal is to compute $W_s(n)$, which is the free energy of strand s. Zuker and Steigler [14] described a dynamic programming algorithm for computing $W_s(j)$; their algorithm can easily be extended to produce a structure whose free energy is $W_s(n)$. The algorithm is based on the following recurrences:

$$W_s(j) = \begin{cases} 0, \text{ for } j = 0 \\ \min(W_s(j-1), \ \min\limits_{1 \leq i < j} (V_s(i,j) + W_s(i-1)), \text{ for } j > 0. \end{cases}$$

Here, $V_s(i,j)$ is the free energy of the optimal structure for $s_i \ldots s_j$, assuming $i.j$ forms a base pair in that structure, also expressible as a recurrence:

$$V_s(i,j) = \begin{cases} +\infty \ \text{ for } i \geq \ j \\ \min(eH_s(i,j), eS_s(i,j) + V_s(i+1, j-1), VBI_s(i,j), VM_s(i,j)); \text{ for } i < j \end{cases}$$

In turn, $VBI_s(i,j)$ is the free energy of the optimal structure for $s_i \ldots s_j$, assuming $i.j$ closes a bulge or internal loop:

$$VBI_s(i,j) = \begin{cases} +\infty \ \text{ for } j - i \leq 1 \\ \min\limits_{\substack{i'.j' \\ i<i'<j'<j}} \ (eL_s(i,j,i',j') + V_s(i',j')) \end{cases}$$

$VM_s(i,j)$ is the free energy of the optimal structure for $s_i \ldots s_j$, assuming $i.j$ closes a multibranched loop (details omitted).

The running time of the resulting for the above algorithm is $O(n^4)$. It can be improved to $O(n^3)$ using the method of Lyngso et al. [10].

3 Structure Freeness for Combinatorial Sets

In this section, we describe an algorithm for testing that all strands in a combinatorial set are structure free. The input to our algorithm is a list of words $w(1), \overline{w}(1), w(2), \overline{w}(2), \ldots w(t), \overline{w}(t)$, each word has length l. Let

$$S = \{z_1 z_2 \ldots z_t | z_i \in \{w(i) \text{ or } \overline{w}(i)\}\}.$$

Intuitively, the set S is the set of strands of length $n = lt$ obtained by following a path in the graph of Figure 2 (first suggested by Lipton [9]) from the left end to the right end and concatenating the edge labels along the path.

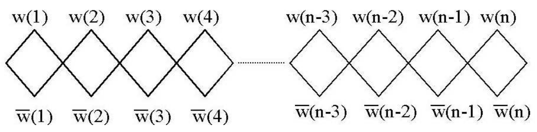
w(1) w(2) w(3) w(4)
w(n-3) w(n-2) w(n-1) w(n)
$\overline{w}(1)$ $\overline{w}(2)$ $\overline{w}(3)$ $\overline{w}(4)$
$\overline{w}(n-3)$ $\overline{w}(n-2)$ $\overline{w}(n-1)$ $\overline{w}(n)$

Fig. 2. "Diamond Graph": paths in the graph correspond to strands of S.

While conceptually the algorithm is quite simple, the description uses a lot of notation that blurs the intuition. In Section 3.1 we first describe an algorithm for a very simple model of secondary structure formation, which captures the intuition. The details of our algorithm for the standard free energy model are presented in Section 3.2.

3.1 Algorithm for the No Repeated k-Strings Model

A simple model of RNA folding is the *no repeated k-strings model* (simplified from the staggered zipper model), which defines a sequence to have non-empty secondary structure if there is a string of length k that repeats without overlaps anywhere in the string. This captures the notion that long runs of stacked pairs are a primary source of secondary structure, and correspond to occurrences of a given substring and its reverse complement in sequence S. Avoiding the reverse complement instead of a repetition does not introduce any important algorithmic complexity.

Cohen and Skiena [4] prove, under the no repeated k-strings model, that it is NP-complete to verify that there is at least one structure-free sequence in a combinatorial set generated in a similar fashion to the diamond graph above. Here, we show that verifying *all* are structure-free can be done in polynomial time. The distinction is exactly that of proving a given CNF-formula is satisfiable versus proving that it is a tautology.

We can verify that all 2^t sequences of length $n = lt$ contain no repeated k-string in $O(kn^2)$ time. For each of the n starting positions in the sequence, there are at most two such positions in the "diamond graph" which generate the sequence starting from this position (either $w(i)$ or $\bar{w}(i)$). Given any two word/position pairs, we now walk forward a total of k characters, comparing the equality of the symbol on each path at every character. At most two such paths are active in each walk, because the paths meet at in-degree-2 vertices of the diamond graph. The existence of a length-k shared walk represents a repeated k-string in at least one sequence, and terminates the algorithm.

We next enhance this basic approach for the more general class of secondary structures of the standard models.

3.2 Algorithm for the Free Energy Model

Let $W'_S(n)$ be the minimum of $W_s(n)$ (as defined in Section 2.2), taken over all $s \in S$. We present in this section an algorithm to compute $W'_S(n)$. The algorithm runs in time $O(n^4)$. Using techniques of Lyngso et al. [10] the running time can be improved to $O(n^3)$. In describing our algorithm, we follow the approach of Section 2.2. Let S_j be the set of prefixes of length j of strands in S and let $W'_S(j)$ be the minimum of $W_s(j)$ taken over all $s \in S_j$. That is, $W'_S(j)$ is the optimal energy of the strand in S_j, with the lowest energy secondary structure:

$$W'_S(j) = \begin{cases} 0, \text{ for } j = 0, \\ \min_{s \in S_j} W_s(j), \text{ for } 0 < j \leq n. \end{cases}$$

Roughly speaking, we would like to be able to express $W'_S(j)$ in terms of $W'_S(j-1)$. It turns out to be more convenient to introduce some new functions to work with in our recurrences. We use $word\#(j)$ to refer to the index of the word to which base j of a string in S belongs. Thus, $word\#(j) = \lceil j/l \rceil$, where l is the word length.

Let $S(T,j)$ be the subset of S in which the word containing the jth base is $w(word\#(j))$. Intuitively, $S(T,j)$ corresponds to the set of paths that go through the **top** of the diamond containing the jth base in Figure 2. Then, $S(T,j)$ is

$$\{z_1 z_2 \ldots z_t | z_i \in \{w(i),\ \overline{w}(i)\} \text{ for } i \neq word\#(j) \text{ and } z_{word\#(j)} = w(word\#(j))\}.$$

Similarly, let $S(B,j)$ be the subset of S in which the word containing the jth base is $\overline{w}(word\#(j))$. Intuitively, $S(B,j)$ corresponds to the set of paths that go through the **bottom** of the diamond containing the jth base in Figure 2.

Let $W'_S(b_j,j)$ be the minimum of $W_s(j)$ for all $s \in S(b_j,j)$. Formally:

$$W'_S(b_j,j) = \min_{s \in S(b_j,j)} W_s(j), \text{ for } b_j \in \{T,B\}.$$

Also, $W'_S(j) = \min_{b_j \in \{T,B\}} W'_S(b_j,j)$. So if we can calculate the $W'_S(b_j,j)$, we are done. It turns out to be easier to obtain recurrence for $W'_S(b_j,j)$ than for $W'_S(j)$.

Similarly, we use generalizations of the functions V_s, VBI_s, in our recurrences. For example, we use $V'_S(b_i,b_j,i,j)$ to denote $\min_{s \in S(b_i,b_j,i,j)} V_s(i,j)$, where $S(T,T,i,j)$ is the subset of S corresponding to the set of paths that go through the top of the diamond containing the ith base and the diamond containing the jth base. $S(b_i,b_j,i,j)$ is defined similarly for other values of b_i and b_j. The base case is $W'_S(b_0,0) = 0$, for $b_0 \in \{T,B\}$. For $j > 0$ we have

$$W'_S(b_j,j) = \min \begin{cases} \min_{b_{j-1} \in X(j-1)} W'_S(b_{j-1}, j-1) \\ \min_{(b_i,b_{i-1}) \in X(i,j)} (V'_S(b_i,b_j,i,j) + W'_S(b_{i-1}, i-1)) \end{cases}$$

where $X(j-1)$ and $X(i,j)$ are given by the tables of Figure 3. Due to space limitations, we do not describe further recurrences here.

$word\#(j)$ == $word\#(j-1)$	$X(j-1)$
yes	$\{b_j\}$
no	$\{*\}$

$word\#(i)$ == $word\#(i-1)$	$word\#(i)$ == $word\#(j)$	$X(i,j)$
yes	yes	$\{(b_j, b_j)\}$
yes	no	$\{(T,T),(B,B)\}$
no	yes	$\{(b_j, *)\}$
no	no	$\{(*, *)\}$

Fig. 3. Tables for $X(j-1)$ and $X(i,j)$. In the entries of the right hand column of each table, each $*$ may be replaced by a T or a B.

3.3 Experimental Results

We have implemented our algorithm, which we refer to as *comfold* in what follows. Our implementation uses Turner's free energy parameters, as incorporated in the mfold algorithm [15]. At this version (1.0 alpha), there are still some small differences in the calculated energy values, compared with Zuker's mfold or Vienna package's RNAfold. Another difference is that comfold does not return suboptimal energies, as mfold does.

We compared the running time performance of *comfold* with that of a simple (exponential time) exhaustive search algorithm, which we call *exhaust_v*. The *exhaust_v* algorithm uses the Vienna package's library to fold each of the possible combinations of words, computes the energy for each, and then returns the combination that folds into the smallest energy. Figure 4 shows the running time of *comfold* compared to *exhaust_v*, plotted on a logarithmic scale. Our results were obtained on a dual Pentium III 1GHz processor machine, with 1GB of RAM We measured the CPU time in seconds (y) when running *comfold* and *exhaust_v*, for different number of groups (x) of randomly generated strands. For the left graph, we used 2 words of length 10 in each group and for the right graph we used 3 words of length 10 in each group. The graph indicates that the running time of *exhaust_v* grows exponentially, while that of *comfold* grows polynomially. At this version of *comfold*, *exhaust_v* is faster for a small number of groups. However, it is outperformed by *comfold* when the number of groups is 13 and 9 for the 2 words in a group set and the 3 words in a group set, respectively. Our results also show that for a set with 4 strands in a group, *comfold* is faster than *exhaust_v* starting with 8 groups.

We have also tested several carefully designed combinatorial sets for structure freeness, using our algorithms. The free energy results reported here in Table 1 are for both *comfold* and *exhaust_v*. We note that the results reported here use the RNA free energy values at 37 degrees Celsius, rather than the DNA values. Every time we found that a set had structure, we computed the minimum free energy of the combination having the smallest minimum free energy. We calculated this strand's DNA free energy by using Michael Zuker's folder for DNA. The results are reported in Table 1, together with the CPU time in seconds for both *comfold* and *exhaust_v*.

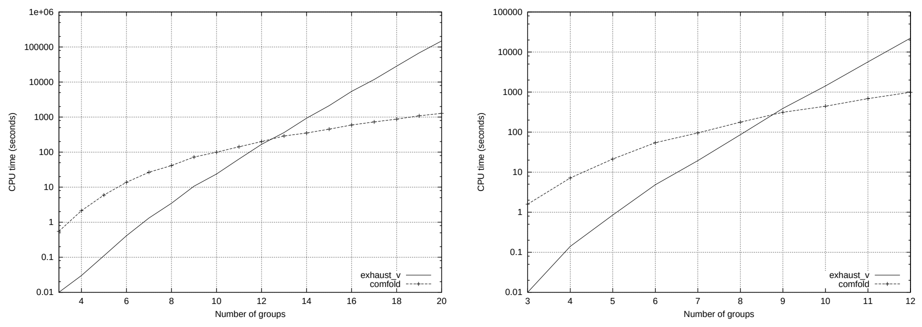

Fig. 4. Performance of *comfold* compared to *exhaust_v*. We measured the CPU time (seconds) for both *comfold* and *exhaust_v*, for a variable number of groups, of 2 words each (left) and 3 words each (right).

Combinatorial set (source)	RNA mfe	DNA mfe	time comfold	time exhaust_v
Braich et al.[1]	0	0	36.50	0.53
Braich et al.[2]	0	0	2,826.63	165,317.00
Brenner et al.[3]	-0.10	0	1,027.25	6,851.59
Faulhammer et al.[5]	-2.90	-2.20	444.01	63.95
Frutos et al.[6]	-18.40	-11.80	1,062.31	7.31

Table 1. Results from testing some cited combinatorial sets for structure freeness.

Our results show that the combinatorial set of Braich et al.[1] described in the introduction is indeed structure free, as well as the set for solving the 20-variable 3-SAT problem with a DNA computer [2].

In contrast, one strand in the set of Brenner et al. did have a very slight negative energy (-0.10kcal/mol), namely, the strand TCTAATCATACTAATC-CTTTTACTATCATTAC with corresponding structure ..((((.(((.(((.....))).))).)))) (here parentheses denote matching pairs, and dots denote unpaired bases). However, we obtained a positive DNA energy for this sequence.

We also found a strand with negative free energy in the reported sets of Faulhammer et al.[5]. This is CTCTTACTCAATTCTTCTACCATATCAACATCT TAATAACATCCTCCACTTCACACTTAATTAAAATCTTCCCTCTTTACA CCTTACTTTCCATATACAAGTACATTCTCCCTACTCCTTCATAATCTT ATATTCTCAATATAATCACATACTTCTCCAACATTCCTTATCCCACAC ACATTTTAAATTTCACAA, that has an energy of -2.90kcal/mol for RNA, and an energy of -2.20kcal/mol for DNA.

The sets of Frutos et al. [6] with two pairs of word labels contain the strand AACGGCATGAAGGCAATTCGCCTTCATGCGTT, with an RNA free energy of -18.40kcal/mol and a DNA free energy of -11.80kcal/mol.

We note the tests reported here were done at a folding temperature of 37 degrees Celsius. The tests can be run at other folding temperatures, depending on the design requirements of the user.

4 Structure Freeness for Kleene Sets

In this section we provide an algorithm for the following problem: given a finite set S of words of equal length, are all words in S^* structure free? We call this the structure freeness problem for Kleene sets. We show that there is a constant m, which depends on S, such that if there is a word with structure in S^*, then there is a word with structure in S^m. The constant m is bounded by $\exp(poly(|S|, l))$, where l is the maximum length of the words in S. With the constant m in hand, the structure-freeness problem for Kleene sets is solvable using the method of Section 3. We prove the existence of m as follows.

1. *There is a strand with structure in S^* if and only if there is a strand in S^* with a so-called* **energy bounded** *structure.*
   We define what is an energy bounded structure in detail in Section 4.1, but the intuition is as follows. If there is a strand with structure in S^* that has a very long hairpin loop, we can remove whole words from the unpaired bases of the loop to obtain a shorter strand in S^* which also has structure, and the energy of the shorter hairpin is bounded. Other parts of the structure that contribute a high positive free energy to the structure can similarly be replaced by a hairpin with bounded energy.
2. *The set of strands of S^* that have energy bounded structures is context free.* Moreover, the size of the context free grammar that generates this set is bounded by a polynomial in $|S|$ and the maximum length l of the words in S. We note that for a given strand with an energy bounded structure, the optimal structure for this strand may not be an energy bounded structure.

From this, in Theorem 1 we apply standard results on context free languages that if there is a strand in S^* that has structure, then there is such a strand of length at most $\exp(poly(|S|, l)$.

4.1 Energy Bounded Structures

A secondary structure F is E-**energy bounded** for a strand s if, with respect to s, no substructure of F has negative energy, and no substructure of F is **expensive**, i.e. has energy above some threshold E. Here, a **substructure** of F is any proper subset of F that contains all base pairs (i', j') of F with $i \leq i' \leq j' \leq j$, for some (i, j) in F, and contains no other base pairs. We call this the substructure of F that is **bounded by** (i, j). We say that **a strand** s **has** E-**energy bounded structure** if the free energy of some E-energy bounded structure is less than 0.

Lemma 1. *Let S be a set of strands. There exists a constant E that grows at most linearly in the length l of words in S such that the following is true: There is a strand with structure in S^* if and only if there is a strand with E-energy bounded structure in S^*.*

One direction of the proof is easy: regardless of the value of E, if there is a strand with E-energy bounded structure in S^*, then trivially this strand is a strand with structure in S^*. The proof of the other direction is based on the fact that for suitable E, expensive substructures can be "cut" out of a structure, and replaced by E-energy bounded structures.

4.2 The Language of Strands with Energy Bounded Structures is Context Free

There is a context free grammar that generates the language of strands with energy bounded structures.

Lemma 2. *The language of strands with E-energy bounded structures, when the energy is measured according to the standard energy assumptions, is context free. There is a context free grammar that generates this language which has a number of nonterminals linear in E, in which each production has bounded length (independent of E).*

Proof. The proof provides a context free grammar for the language of strands with energy bounded structures. The terminals of the grammar are $\{A, C, G, T\}$, and the nonterminals generate structures with fixed energy values. First, we consider a simple case, namely how to generate energy bounded hairpins. Specifically, we focus on developing a context free grammar that generates the set of all possible strings that can form a hairpin loop (with no dangling ends) that is closed by the base pair $(t, \bar{t})$ and has energy e. Call this set $H(t, e)$. We will use the nonterminal $H_{t,e}$ to generate the set $H(t, e)$. Since $H(t, e)$ is finite, we could simply have one production from $H_{t,e}$ that directly generates each string in $H(t, e)$. However, according to the standard energy model, other than a few special cases (such as tri-loops or tetra-loops), the free energy of a hairpin formed by pairing the outside bases in the string $tau_1u_2 \ldots u_jb\bar{t}$ (where t, a, b and $u_i, 1 \leq i \leq j$ are in $\{A, C, G, T\}$) can be expressed as the sum of two terms, namely a stacking energy term, e-Stack$(t, a, b, \bar{t})$, and a length term e-Length(j), where $j + 2$ is the number of unpaired bases in the hairpin. By taking advantage of this, fewer rules are needed. Specifically, from $H_{t,e}$, we add rules of the form:

$$H_{t,e} \rightarrow taU_{j-2}b\bar{t}$$

for each $a, b \in \{A, C, G, T\}$ for which there exists a j for which $e-$e-Stack$(t, a, b, \bar{t}) =$ e-Length(j). Here, U_{j-2} is a nonterminal that generates all sequences of $j - 2$ bases: for each $t \in \{A, C, G, T\}$ and $1 < i \leq j$, we add the rules

$$U_i \rightarrow tU_{i-1} \text{ and } U_1 \rightarrow t.$$

The total number of rules generated from $H_{t,e}$ is a constant independent of e. In addition, the number of nonterminals U_i is approximately e-Length$^{-1}(e)$. Generalizing from the case of hairpins, we need other new nonterminals in our grammar that generate different types of structures. Mirroring the notation used in the recurrence relations of Section 2.2, we use the following nonterminals (details omitted):

- W_e**:** generates the set of strings that can form an E-energy-bounded structure with energy e.
- $V_{t,e}$**:** generates the set of strings that can form an E-energy bounded structure that is closed by the base pair $(t, \bar{t})$ and has energy e;
- $I_{t,e}$**:** generates the set of strings that can form an E-energy-bounded structure with energy e, the outermost loop of which is an interior loop (other than stacked pair) that is closed by the base pair $(t, \bar{t})$.
- $M_{t,e}$**:** generates the set of strings that can form an E-energy bounded structure with energy e, the outermost loop of which is a multi-loop that is closed by the base pair $(t, \bar{t})$.

All of the productions above have length that is bounded (there is a maximum of nine symbols on the right hand side of any rule).

Lemma 3. *Let S be a set of strands, each of length l. Let L be the language of strands in S^* with E-energy bounded structures. Then L is context free, and there is a context free grammar that generates L, for which the number of nonterminals is $O(E(l|S|)^3)$ and each rule has bounded length (independent of E, l, and S).*

The proof of this lemma follows from classical results of automata theory, that the intersection of a context free language and a regular language is context free (details omitted).

The main theorem of this section follows from the previous lemmas, together with classical results from automata theory.

Theorem 1. *There is a constant m, which depends on S, such that if there is a word with structure in S^*, then there is a word with structure in S^m. The constant m is bounded by $\exp(poly(|S|, l))$, where l is the length of words in S.*

5 Conclusions

We present an efficient algorithm for determining structure freeness of combinatorial sets. The algorithm should prove useful in verifying the quality of word sets designed for DNA and RNA computations.

An important direction for future work is to use partition functions to cover the range of alternative structures. Another direction for future work is to study whether our algorithm can be generalized to determine structure freeness of more general sets of strands, such as those used in Adleman's Hamiltonian graph experiment.

References

1. Braich, R. S., C. Johnson, P. W.K. Rothemund, D. Hwang, N. Chelyapov and L. M. Adleman. *Solution of a satisfiability problem on a gel-based DNA computer.* Proceedings of the 6th International Conference on DNA Computation Springer-Verlag LNCS, vol. 2054, 2000, pages 27-41.
2. Braich, R. S., Chelyapov, N., Johnson, C., Rothemund, P. W.K. and Adleman, L. *Solution of a 20-variable 3-SAT Problem on a DNA computer.* Science 296, 2002, 499-502.
3. Brenner, S., Williams, R. S., Vermaas, E. H., Storck, T., Moon, K., McCollum, C., Mao, J-I., Luo, S., Kirchner, J. J., Eletr S., DuBridge, R. B., Burcham, T., and Albrecht. G. *In vitro cloning of complex mixtures of DNA on microbeads: Physical separation of deferentially expressed cDNAs.* Proceedings of the National Academy of Sciences of the United States of America 2000 February 15; 97(4): 1665-1670.
4. Cohen, B. and S. Skiena. *Designing RNA Sequences: Natural and Artificial Selection*, Proc. 6th Int. Conf. Computational Molecular Biology (RECOMB '02).
5. Faulhammer, D., Cukras, A. R., Lipton, R.J., and Landweber, L.F. *Molecular computation: RNA solutions to chess problems.* Proc. Natl. Acad. Sci. USA, 97: 1385-1389.
6. Frutos, A. G., Liu, Q., Thiel, A. J., Sanner, A. M. W., Condon, A. E., Smith, L. M., and Corn, R. M. *Demonstration of a Word Design Strategy for DNA Computing on Surfaces.* Nucleic Acids Research, 25 4748 (1997).
7. A.J. Hartemink and D. K. Gifford, *Thermodynamic simulation of deoxyoligonucleotide hybridization.* Prel. Proc. 3rd DIMACS Workshop on DNA Based Computers, June 23-27, 1997, U. Penn., pages 15-25.
8. I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster, *Fast Folding and Comparison of RNA Secondary Structures.* Monatsh.Chem. 125, 1994, 167-188.
9. R. Lipton, *DNA solution of hard computational problems* Science 268 (1995), 542-545.
10. Lyngso, R. B., Zuker, M., and Pedersen, C. N. S. *Internal Loops in RNA Secondary Structure Prediction.* Proc. Third International Conference in Computational Molecular Biology, pages 260-267, April 1999.
11. J.S. McCaskill, *The equilibrium partition function and base pair binding probabilities for RNA secondary structure.* Biopolymers, Vol 29, pages 1105-1119, 1990.
12. J. A. Rose, R. Deaton, D. R. Franceschetti, M. Garzon, and S. E. Stevens, Jr., *A statistical mechanical treatment of error in the annealing biostep of DNA computation.* Special program in DNA and Molecular Computing at the Genetic and Evolutionary Computation Conference (GECCO-99), Orlando, FL., July 13-17, 1999, Morgan Kaufmann.
13. SantaLucia, Jr., J. *A Unified View of Polymer, Dumbell, and Oligonucleotide DNA Nearest-Neighbor Thermodynamics.* Proc. Natl. Acad. Sci. 1998, 95, 1460.
14. Zuker, M. and Steigler, P. *Optimal Computer Folding of Large RNA Sequences using Thermodynamics and Auxiliary Information.* Nucleic Acids Research, 9, (1981), pages 133-148.
15. Zuker, M., D.H. Mathews, and D.H. Turner, *Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A Practical Guide.* In RNA Biochemistry and Biotechnology, J. Barciszewski and B.F.C. Clark, eds., NATO ASI Series, Kluwer Academic Publishers, Dordrecht, NL, 1999, pages 11-43.