

Composition Alignment

Gary Benson*

Department of Biomathematical Sciences
The Mount Sinai School of Medicine
New York, NY 10029-6574
benson@ecology.biomath.mssm.edu

Abstract. In this paper, we develop a new approach for analyzing DNA sequences in order to detect regions with similar nucleotide *composition*. Our algorithm, which we call **composition alignment** or, more whimsically, **scrambled alignment**, employs the mechanisms of string matching and string comparison yet avoids the overdependence of those methods on position-by-position matching. In composition alignment, we extend the matching concept to *composition matching*. Two strings have a composition match if their lengths are equal and they have the same nucleotide content.

We define the composition alignment problem and give a dynamic programming solution. We explore several composition match weighting functions and show that composition alignment with one class of these can be computed in $O(nm)$ time, the same as for standard alignment. We discuss statistical properties of composition alignment scores and demonstrate the ability of the algorithm to detect regions of similar composition in eukaryotic promoter sequences in the absence of detectable similarity through standard alignment.

1 Introduction

Most algorithms which characterize DNA functional sites concentrate on identifying *position-specific patterns*, such as consensus patterns or weight-matrix-based profiles. Each is a string $P = p_1p_2 \cdots p_k$ in which p_i is specified as either a single character or a choice (weighted or unweighted) of characters. The data for a pattern are typically collected from multiple known occurrences of the functional site. Well known examples include the TATA box, a frequent component of the eukaryotic gene promoter, where transcription from DNA to RNA starts [9] and the weight matrices used to describe transcription binding sites in the TRANSFAC database [16].

Often, position-specific patterns have low selectivity. That is, when used to search for unknown occurrences, they yield many non-functional sites. This, in part, is due to the nature of the patterns which are frequently short and degenerate (more than one letter choice at some positions) making random matching

* Partially supported by NSF grants CCR-0073081 and DBI-0090789.

a relatively common event. A less obvious reason is that functional sites contain multiple features working synergistically and accurate recognition requires identifying several of those features. What hampers current detection methods is the likelihood that some sequence properties which contribute to function *are not reducible to position-specific patterns* and thus are not easily incorporated into current search techniques.

An example is double helical strand separation, a necessary antecedent for several important DNA functions. Rather than exhibiting position specific properties, the propensity for strand separation appears to be spread, in a complex way, over thousands of nucleotides. Programs have been developed to calculate regions of conjectured low energy requirement for strand separation [5, 6, 30, 31] and these programs have had some success in identifying sites which are functionally dependent on this process.

In this paper, we develop a new approach for DNA sequence analysis which employs the mechanisms of string matching and string comparison yet avoids the overdependence of these methods on position-by-position matching. We initiate the study of what we call the *composition property*, which manifests as similarity in *composition*, rather than in position-specific patterns.

Definition: *Composition* is a vector quantity describing the frequency of occurrence of each alphabet letter in a particular string. Let S be a string over Σ . Then, $C(S) = (f_{\sigma_1}, f_{\sigma_2}, \dots, f_{\sigma_{|\Sigma|}})$ is the composition of S , where for $\sigma_i \in \Sigma$, f_{σ_i} is the fraction of the characters in S that are σ_i . Note that the *order* of letters in S is irrelevant as it has no effect on the composition of S . Two strings S and T have a *composition match* if their lengths are equal and $C(S) = C(T)$.

There is an accumulating body of evidence that variation in composition along the DNA strands contributes to function. There are a number of important DNA features, at both large and small scales, whose unifying characteristic is composition bias including:

- *Isochores*. These multi-megabase regions of genomic sequence are specifically GC-rich or GC-poor. GC-rich isochores exhibit greater gene density. Human ALU and L1 retrotransposons appear preferentially in isochores with composition that approaches their own [7, 8, 25].
- *CpG islands*. These regions of several hundred nucleotides are rich in the dinucleotide CpG which is generally underrepresented (relative to overall GC content) in eukaryotic genomes. The level of methylation of the cystine (C) in these dinucleotide clusters has been associated with gene expression in nearby genes [12, 11, 13].
- *Protein binding regions*. Within these domains, tens of nucleotides long, dinucleotide, or base-step composition, can contribute to DNA flexibility, allowing the helix to change physical conformation, a common property of protein-DNA interactions [24, 19, 14, 18].

The springboard for our study of the composition property is a new alignment algorithm which we call **composition alignment** or, more whimsically,

scrambled alignment. Standard sequence alignment is based on single character matching. In composition alignment, we extend the matching concept to substrings which have the same composition *i.e.*, composition matching. This allows us to identify subsequences that share regions of similar composition. More specifically, composition alignment is a pairing of substrings of *exactly matching* composition separated by insertions, deletions or mismatches.

As an example, let

$$\begin{aligned} X &= AACGTCTTTGAGCTC \\ Y &= AGCCTGACTGCCTA \end{aligned}$$

One possible composition alignment for X and Y is

$$\begin{array}{c} AACGTCTTTGAGCTC \\ | |<-> | <---> \\ AGCCTGACTGCCTA \end{array}$$

where symbols between the letters are used to indicate single character matches ($|$) and substring composition matches ($< - - - >$). Note that composition matches (either single or multicharacter) can occur consecutively in an alignment.

An idea related to composition alignment is the “swap” operation in string comparison, first discussed by Lowrance and Wagner [21, 28]. A swap or transposition is the exchange of two letters that are side-by-side. Matching two letters and their transposition is equivalent to a composition match for substrings of length two. Lowrance and Wagner gave a $O(nm)$ time algorithm for alignment including the swap operation. More recently, Amir et al. [1] gave an algorithm for finding all swapped matchings of a pattern of length m in a text of length n with time complexity $O(nm^{1/3} \log m \log \sigma)$. This was later improved to $O(n \log m \log \sigma)$ [2] and followed by an algorithm to count the number of swaps in each swapped match within the same time complexity [3].

The remainder of the paper is organized as follows. In section 2 we present a formal description of the composition alignment problem. In section 3 we present our composition alignment algorithm and discuss scoring functions for composition matches. In section 4 we discuss the match length *limit* parameter and its effect on the statistical relevance of scores. Finally, in section 5, we demonstrate the use of composition alignment on real biological sequences.

2 Problem Description

The problem we address is the following:

Composition Alignment

Given: Two sequences, S of length m , and T of length n , over an alphabet Σ , and a scoring function $cm(s, t)$ for the score of a composition match between substrings s and t .

Find: The best scoring alignment (global or local) of S with T such that the allowed scoring options include composition match between substrings of S and T as well as the standard options of single character match, single character mismatch, insertion and deletion.

For the remainder of this paper, we will assume 1) that the alphabet Σ is fixed and 2) that similarity scoring is used. The latter means that matches are given positive weight; mismatches, insertions and deletions are given negative weight; and the best alignment has the highest score.

3 Composition Alignment Algorithm

Our composition alignment algorithm is similar to standard alignment algorithms [22, 26, 15] and is computed using dynamic programming. A single step in filling the dynamic programming array, W , can be analyzed in the following way. Given two sequences $X = x_1 \cdots x_n$ and $Y = y_1 \cdots y_m$, the best composition alignment of the two prefix strings $X[1, i] = x_1 \cdots x_i$ and $Y[1, j] = y_1 \cdots y_j$ ends in one of the following four ways:

New:

1. A *composition match* between suffixes of length l , $1 \leq l \leq \min(i, j, limit)$, *i.e.*, $x_{i-l+1} \cdots x_i$ and $y_{j-l+1} \cdots y_j$, where *limit* is an upper bound on the length of a substring that can participate in a composition match. The necessity of *limit* will be explored further in section 4.

Standard:

1. A *mismatch* between x_i and y_j .
2. A *deletion* of x_i .
3. A *deletion* of y_j .

The score in cell (i, j) is the maximum obtained by these four possibilities (global alignment) or the maximum among these four and a score of zero (local alignment). Note that the score of alternative 1 is

$$W[i-l, j-l] + cm(x_{i-l+1} \cdots x_i, y_{j-l+1} \cdots y_j),$$

where W is the alignment score matrix and $cm()$ is the score function for composition matches. The time complexity of the algorithm is $O(nmZ)$ where Z is the time required, per (i, j) pair, to find the best length l in alternative 1 and compute its score. In the next section, we show how to precompute the *length* of the *shortest* composition match for every (i, j) pair in constant time per pair.

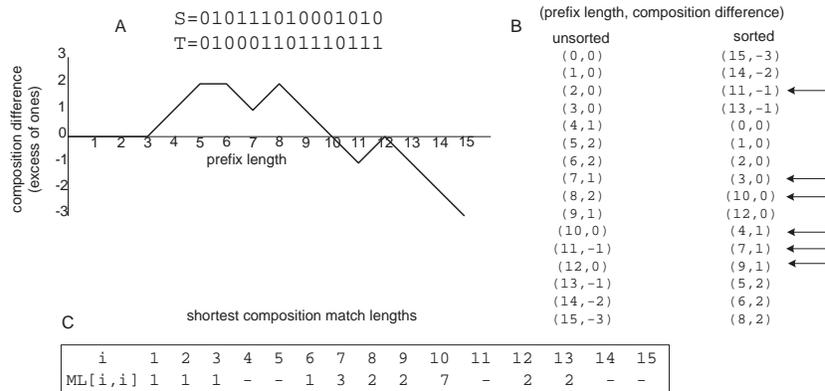


Fig. 1. A) Graph of composition differences (excess of 1's in prefixes of S relative to T). B) Ordered pairs (prefix length, composition difference) unsorted and sorted using composition difference as the key. Arrows mark prefixes mentioned in text. C) The $ML[i, j]$ array for diagonal zero ($i = j$).

3.1 Finding Composition Matches

Our goal here is to find the length, l , of the shortest suffixes of the strings $X[1, i]$ and $Y[1, j]$ which have a composition match, and to do this for every (i, j) pair, in constant time per pair. For example, if $X = AACGTCTTTGAGCTC$ and $Y = AGCCTGACTGCCTA$, then for the pair $(4, 8)$, the shortest suffixes that have a composition match are $X[2, 4] = ACG$ and $Y[6, 8] = GAC$ each with a length of 3. To find these matches, we use composition difference.

Definition: *Composition difference* is a vector quantity for two strings. Let x and y be strings over an alphabet Σ . Then $CD(x, y) = (c_{\sigma_1}, \dots, c_{\sigma_{|\Sigma|}})$ is the composition difference of x and y , where $c_{\sigma_r} = c_{\sigma_r}^x - c_{\sigma_r}^y$ is the difference between the number of times σ_r occurs in x ($c_{\sigma_r}^x$) and the number of times it occurs in y ($c_{\sigma_r}^y$). For example, let $\Sigma = \{0, 1\}$, $x = 010111010001000$ and $y = 010001101110111$. Then $CD(x, y) = (3, -3)$ because x has three more zeros and three fewer ones than y .

We compute composition match lengths for one diagonal of the alignment matrix at a time. Each diagonal is defined by a value d , $-n \leq d \leq m$ and contains the index pairs (i, j) such that $j - i = d$. The substrings processed in a diagonal for $d \geq 0$ are $X[1, k]$ and $Y[1 + d, k + d]$ and for $d \leq 0$ are $X[1 - d, k - d]$ and $Y[i, k]$ for all values of $k = 1, \dots, \min(m, n)$ such that both strings are non-null.

To illustrate, let $S = 010111010001000$ and $T = 010001101110111$. We show how to process diagonal zero. Figure 1A is a plot of the composition differences for successive prefixes of S and T . Since the components of any composition difference vector must sum to zero, we plot only the excess of ones in S relative to T (the second number in the composition difference vector).

The *key observation* is that two identical composition differences at prefix lengths g and h with $g < h$, indicate a composition match between the substrings of S and T from position $g + 1$ to position h , *i.e.*, of length $h - g$. As shown in the plot of our example, prefix lengths 3 and 10 both have composition difference $(0, 0)$ so there is a composition match between the substrings $S[4, 10]$ and $T[4, 10]$. Similarly, prefix lengths 4, 7 and 9 each have composition difference $(-1, 1)$, so there is a composition match for the substring pairs $(S[5, 7], T[5, 7])$, $(S[5, 9], T[5, 9])$, and $(S[8, 9], T[8, 9])$. Note that prefix length 11 has composition difference $(1, -1)$ which is not shared by any prefix length shorter than 11. Therefore, there is *no* composition match for the prefixes ending at position 11 in these two strings.

To identify composition matches, we compute the ordered pairs (*prefix length, composition difference*) and then sort these using composition difference as the key (Figure 1B). This is done with *radix sort* ([10] p. 178) across the σ values and *counting sort* ([10] p. 175) within each σ value. Notice that both sorts are stable, *i.e.* they do not rearrange elements with the same key. Shortest composition matches are determined by scanning the sorted list to find *adjacent* elements with the same key (composition difference). The difference between prefix lengths is then stored in an array $ML[i, j]$. In our example, since prefix lengths 10 and 3 are adjacent with the same composition difference, we store $7 (= 10 - 3)$ as the shortest composition match for prefixes of S and T of length 10.

Time complexity. The sorting is linear in the number of elements when the alphabet is fixed. For a single diagonal of the alignment matrix, the number of elements is the diagonal length. Over all diagonals, the number of elements is $(n + 1)(m + 1)$. Therefore the time complexity to preprocess the strings to find all shortest composition match lengths is $O(nm)$.

3.2 Scoring functions

The overall complexity of our composition alignment algorithm depends on the complexity of computing the best length l for a composition match. This in turn depends on the scoring function, $cm()$, for composition matches. Below, we discuss several scoring functions we have tested.

Functions based on match length. In this group, the score of a composition match depends on the *length, k* , of the match. We have tested

- Function 1: $cm(k) = ck$
- Function 2: $cm(k) = c\sqrt{k}$
- Function 3: $cm(k) = c \log(k + 1)$

where c is a constant. These functions are additive or subadditive, meaning that $cm(i + j) \leq cm(i) + cm(j)$. Function 1 (additive) treats matches of different lengths equally. Functions 2 and 3 (subadditive) give less weight, per character, to long composition matches, than to short composition matches.

A convenient property of additive or subadditive functions is that, when computing alternative 1 of the alignment score, for any (i, j) pair, it is sufficient to find the length of the *shortest* suffixes of $X[1, i]$ and $Y[1, j]$ which have a composition match.

Lemma 1. *For an index pair (i, j) , let $l = l_1 < l_2 < \dots < l_k$, $1 \leq l \leq \min(i, j)$, be the lengths for which there is a composition match between the suffixes $X[i - l + 1, i]$ and $Y[j - l + 1, j]$. Then, the score for the best alignment which ends in a composition match between suffixes of $X[1, i]$ and $Y[1, j]$ is equal to the score when the suffixes have length l_1 . That is, $\forall h, 2 \leq h \leq k, W(i - l_1, j - l_1) + cm(l_1) \geq W(i - l_h, j - l_h) + cm(l_h)$.*

Proof. Assume by way of contradiction that there is an $l_h > l_1$ such that

$$W(i - l_h, j - l_h) + cm(l_h) > W(i - l_1, j - l_1) + cm(l_1).$$

Let $l_h = l' + l_1$. Then

$$\begin{aligned} W(i - l_h, j - l_h) + cm(l' + l_1) &> W(i - l_1, j - l_1) + cm(l_1) \\ W(i - l_h, j - l_h) + cm(l' + l_1) - cm(l_1) &> W(i - l_1, j - l_1) \end{aligned}$$

but by additivity or subadditivity,

$$cm(l') \geq cm(l' + l_1) - cm(l_1)$$

so

$$W(i - l_h, j - l_h) + cm(l') > W(i - l_1, j - l_1)$$

which is a contradiction because $W(i - l_1, j - l_1)$ is assumed to be optimal including the possibility that the alignment which yields this score ends with the composition match of length l' . \square

This means that breaking up a long composition match into shorter matches (if possible) will leave the score the same (function 1) or increase the score (functions 2 and 3). The alignment shown in the introduction contains a 4 character composition match which is broken into a single character match and a 3 character match.

Theorem 2. *Composition alignment with an additive or subadditive composition match scoring function has time complexity $O(nm)$.*

Proof: Follows from the discussion in section 3.1. \square

Functions based on substring composition. Here, the score of a composition match depends not just on length, but on the composition of the matching substrings. We have tested:

– Function 4: $cm(x, y, k) = ck \cdot H(C, B)$

where x and y are substrings with common composition C , k is their length, and c is a constant. $H(C, B)$ is the relative entropy of composition C given a background composition B . *Relative entropy* is defined as

$$H(C, B) = - \sum_{\sigma \in \Sigma} f_{\sigma} \log(f_{\sigma}/b_{\sigma})$$

where f_{σ} is a frequency in the composition vector C and b_{σ} is the corresponding frequency in the background composition B . This function can only be used if for every non-zero f_{σ} in C there is a corresponding non-zero b_{σ} in B , else there will be a divide-by-zero problem. In our studies this has not arisen because we use the overall frequency of letters in the sequences to be aligned as the background. If divide-by-zero is possible, H can be replaced with the unweighted Jensen-Shannon divergence [20]. We have not yet tested this function extensively.

Function 4 favors composition matches where the substrings differ significantly from the background. This could, for example, be a long repetition of a single letter, assuming the background is relatively balanced. This function is not additive or subadditive, so finding the shortest composition match for any (i, j) pair does not always yield the optimal match length. Since a longer match may yield a higher alignment score, we must test all substring match lengths per (i, j) pair. This can easily be done in time linear in the number of match lengths by stepping through the ML array which stores the shortest match lengths, but requires at most $\min(m, n)$ tests per (i, j) pair. In practice though, the *limit* parameter explained below restricts the number of tests to at most *limit* per (i, j) pair.

Theorem 3. *Composition alignment using function 4 for composition match scoring and the limit parameter has time complexity $O(nm \cdot \text{limit})$.*

Proof. Follows from discussion above. \square

Retrieving the alignment. After the alignment score array W has been computed, the optimal alignment is retrieved by tracing back as in standard alignment. When a score $W[i, j]$ is derived from a composition match, we need to reference the length of the matching substring. For Functions 1, 2 and 3, this is done by querying the $ML[i, j]$ value. For Function 4, we must store the optimal match lengths separately as the scores are being computed and then refer to these values when tracing back. In either case, retrieving the alignment requires $O(n + m)$ time.

4 Alphabet Size and the Limit Parameter

The study of alignment using similarity scoring has shown that for ungapped local alignments of randomly generated sequences, the parameter space for match and mismatch weights is divided into logarithmic and linear regions. In the logarithmic region, the parameters produce alignment scores proportional to the logarithm of sequence lengths whereas in the linear region, the scores are directly proportional to the sequence lengths [29]. It is generally accepted that

weight combinations which fall within the logarithmic region are useful for detecting *biologically related* sequences, whereas those in the linear region do not distinguish between related and unrelated sequences. The same general features have been observed in gapped local alignments where gap weight is an additional parameter [27, 4]. The rubric for determining if parameters fall within the logarithmic or linear regions is to look at the expected score per aligned letter pair (ungapped alignments [17]) or the expected global alignment score (gapped alignments [4]). In either case, if the expected score is negative, and assuming that positive scores are possible, then the parameters fall within the logarithmic region.

Here we are primarily interested in how the *limit* parameter, the length of the longest allowed composition match, fits into this framework. When $limit = 1$, composition alignment is equivalent to standard alignment. When $limit = 2$, any pair of adjacent letters in one sequence is allowed to match its transposition in the other sequence. This corresponds to the swap operation mentioned earlier [21]. For $limit = 3$, both scrambled triplets and transposed doublets are allowed to match, etc. Intuitively, allowing scrambled letters to match should increase the amount of matching. If too much matching occurs, then the average score will be positive and the alignments will not be meaningful.

4.1 Expected fraction of matching characters in alignments

We have examined both ungapped and gapped composition alignments to determine the expected fraction of aligned letter pairs that are counted as matches.

Ungapped alignments. Suppose we have a binary alphabet and we examine ungapped aligned strings of length 2 where the characters are generated iid with probability 0.5. Under single character matching, the expected fraction of characters counted as matching is 0.5, *i.e.* on average half the characters will be counted as matches. When we allow composition matches with $limit = 2$, the expected number of characters counted as matches increases to 0.625. For aligned strings of length 3, the results are similar. For single character matching the expected fraction is still 0.5, but for composition matching with $limit = 3$ (substring pairs of length 2 or 3 can match as long as they have the same composition), the expected fraction of matches is 0.6875. As the sequence length grows, calculating the fraction of matches becomes complicated, so we turn to simulation. When the string length reaches 10, and we allow composition matching with $limit = 10$, the fraction of matches is above 0.82 (table 1).

For the four letter DNA alphabet, when the letters are generated iid with probability 0.25, the fraction of characters matching grows similarly but more slowly. For the sixteen letter dinucleotide alphabet, the probability grows until reaching an apparent asymptotic upper bound around 0.075. For dinucleotides, we first generate an iid DNA sequence and then convert it to a dinucleotide sequence. Notice that consecutive letters in the dinucleotide sequence are not

Sequence length	1	2	3	4	5	6	7	8	9	10
Binary (%)	50.0	62.5	68.75	72.7	75.6	77.3	78.9	80.3	81.3	82.4
DNA (%)	25.0	30.0	32.3	35.3	37.5	39.7	40.7	42.4	43.3	44.2
Dinucleotide (%)	6.2	6.5	6.7	6.9	7.1	7.3	7.3	7.3	7.4	7.5

Table 1. Fraction of characters counted as matching in randomly generated ungapped alignments where *limit* equals alignment length, for three alphabets. In each case, all letters in an alphabet have equal probability. Results are derived from simulations except for sequence length 1 in all alphabets and lengths 2 and 3 in the binary alphabet.

independent (i.e. if the first dinucleotide is AC, then the next must start with a C).

To investigate the fraction of matches in longer sequences where the *limit* is smaller than the sequence length, we use global composition alignment to count the matches. Here, insertions and deletions are not allowed, all matches are weighted 1 and all mismatches weighted zero. Results for DNA sequences of length 100 and dinucleotide sequences of length 400 are shown in Table 2. As can be seen in the table, the fraction matching in DNA sequences is nearly 45% with *limit* = 5 and reaches 50% when *limit* = 9. For dinucleotides, the fraction levels off at 7.78% for *limit* \geq 20.

For *local, ungapped* composition alignments with DNA sequences using alignment parameters (composition match constant, single character match, mismatch) = (1, 1, -1), alignment scores grow in proportion to the log of the sequence length for *limit* \leq 5 (data not shown). For *limit* between 6 and 10, growth is proportional to the square root of the sequence length. Note that with these alignment parameters, the expected score of an aligned pair is negative until *limit* = 9 (table 2). Thus negative expected score per aligned letter pair is an inaccurate predictor of logarithmic score growth for ungapped composition alignments.

Gapped alignments. For gapped alignments, we use simulations with actual parameter values and composition match scoring functions because the interac-

DNA: sequence lengths = 100; iid; $p = 0.25$										
<i>limit</i>	1	2	3	4	5	6	7	8	9	10
fraction matching (%)	25.0	33.7	38.6	42	44.4	46.3	47.8	49.0	50.0	51.0
dinucleotide: sequence lengths = 400; iid; $p = 0.25$										
<i>limit</i>	1	2	5	10	20	30	40	50		
fraction matching (%)	6.25	6.81	7.66	7.76	7.78	7.78	7.78	7.78	7.78	

Table 2. Fraction of characters counted as matching in longer randomly generated DNA and dinucleotide sequences, after composition alignment without gaps, for various *limit* values.

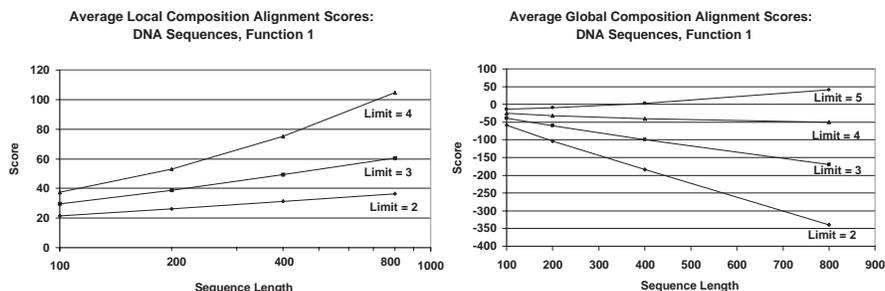


Fig. 2. Function 1 results from alignment of randomly generated DNA sequences (iid, $p = 0.25$) when *limit* is allowed to vary. Alignment parameters: (2,2,-3,-5). Left) Average local scores proportional to log of sequence lengths (note log scale) for $limit \leq 3$. Right) Average global scores become positive at $limit = 5$.

tion of these determines the number of gapped positions. The results are useful for several purposes:

1. To define values for the limit parameter that fall within the logarithmic region.
2. To test for concurrence between 1) the change from negative to positive average global alignment score and 2) the change from logarithmic to linear behavior in the local alignment score. This is not obvious because the assumptions that underly the theory for alignment scores do not include scrambled alignment.
3. To estimate local alignment score distributions so that the composition alignment algorithm can be used to search for statistically significant alignments in real biological sequences.

We used DNA sequences generated iid with probability 0.25 for each letter and two sets of alignment parameters (composition match constant, single character match, mismatch, indel), (2, 2, -3, -5) and (2, 2, -7, -7). Here we summarize some of the more important results.

Function 1. Local alignment scores grow in proportion to the logarithm of sequence length for $limit \leq 3$. At $limit = 4$, scores are proportional to the square root of sequence length. Note though, that the average global alignment score does not become positive until $limit = 5$. See figure 2. These results indicate that function 1 should be used with *limit* set to 3 and not higher. Also, positive or negative expected global score is inaccurate as a predictor of the parameter values that yield logarithmic growth in local alignment scores.

Functions 2 and 3. Local alignment scores for function 2 are logarithmically related to sequence length below $limit = 10$. Average global scores do not become positive with *limit* as high as 50. See figure 3. Function 3 behaves similarly.

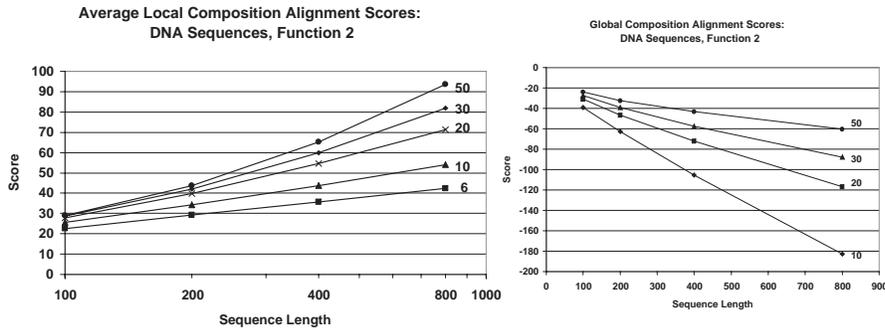


Fig. 3. Function 2 results from alignment of randomly generated DNA sequences. Same conditions as in figure 2. Limit values shown to right of curves. Left) Average local scores proportional to log of sequence lengths (note log scale) for $limit < 10$. Right) Average global scores do not become positive at $limit$ as high as 50.

Again, average global score is an inaccurate predictor of local alignment score behavior.

Function 4. Local alignment scores are proportional to the log of the sequence lengths up to $limit = 50$.

5 Examples

We tested our composition alignment algorithm on a set of 1796 human promoter sequences from the Eukaryotic Promoter Database (EPD) [23] maintained by the Bioinformatics Group of the Swiss Institute for Experimental Cancer Research. The database contains a collection of roughly 3000 annotated non-redundant eukaryotic RNA polymerase II promoters for which the transcription start site has been experimentally determined. Each sequence is 600 bases long and consists

```

Composition Alignment:
GCCCGCCCGCCGCGCTCCCGCCCGCCGCTCTCCGTGGCCC-CGCCG-CGCTGCCGCGCCGCGCTGC
<->| | | | <> | <> | | <> | | | | <> | | <> | <-> | | | | | | <> | <> | | | | <> <> | <> | | | <-> | |
CCCGCCCGCCGCGCTCCCGCCCGCCCG-CCCT-TGGCCCAGCCGCTCGCTCGGCTCCGCTCCCTGGC

Standard Alignment:
CGCCGCCGCG
CGCCGCCGCG

```

Fig. 4. Composition alignment and standard alignment of promoters EP27006 and EP73975, positions 474-539 and 430-495 respectively. Composition of aligned subsequences (top alignment) is (0.01, 0.59, 0.30, 0.11). Background composition of these promoters is (0.11, 0.44, 0.34, 0.11). Standard alignment is not statistically significant.



Fig. 5. Composition alignment of promoters EP73298 and EP11149, positions 323-409 and 444-534 respectively. Composition of left two thirds, (0.01, 0.61, 0.30, 0.08), is dramatically different from composition of right third, (0.19, 0.16, 0.56, 0.09).

of 500 bases upstream and 100 bases downstream of the transcription initiation point. The sequences are non-redundant as selected from the database, which means that no two share greater than 50% sequence identity.

The sequences were aligned pairwise using composition match scoring function 1 with alignment parameters (composition match constant, single character match, mismatch, indel) of (2, 2, -7, -7). This produced a score W . Each pair was also aligned with a standard alignment algorithm using the same parameter values, producing a score S . Those pairs for which 1) W was above the statistical significance cutoff score for composition alignment for a set of sequences this large (as determined by simulation) and 2) $W \geq 3 \cdot S$ were retained. The second criterion was used to exclude composition alignments that scored highly because they were redetecting good standard alignments. Two high scoring alignments are shown here.

The first example was obtained with the promoter pair EP27006 and EP73975 (Figure 4). The standard local alignment which is not statistically significant is shown for comparison. The composition alignment is characterized by high GC content (89%), an enrichment over the background frequency of these sequences (78%). The number of CpG dinucleotides found in the aligned regions is more than expected given either the background composition of these subsequences or the entire sequences. This suggests that the aligned regions are part of CpG islands which are defined [13] as being 200 bp subsequences with a C+G content exceeding 50% and a ratio of observed CpG to expected CpG in excess of 0.6. CpG islands are known to occur in the 5' region of many genes. This alignment is typical of many obtained with the promoter set.

A second example involves promoters EP73298 and EP11149 (Figure 5). Again, the standard local alignment score for this pair is not statistically significant. An interesting feature of the composition alignment is the change in composition of the subsequences from left to right. The left two thirds is GC rich with C dominant, and a single A: (0.01, 0.61, 0.30, 0.08). The situation changes at the right which is G dominant with the fraction of As, and Cs equivalent: (0.19, 0.16, 0.56, 0.09). The fraction of Ts is roughly the same throughout. Notice that the background composition for these sequences is typical, GC rich with the complementary nucleotides balanced: (0.15, 0.36, 0.34, 0.15).

6 Conclusion

We define a new type of alignment problem, *composition alignment* which extends the matching concept to substrings of equal length and the same nucleotide composition. We give an algorithm for composition alignment which has time complexity $O(nm)$ for a fixed alphabet when the composition match scoring function is additive or subadditive. The time complexity is $O(nm \cdot limit)$ for a relative entropy scoring function where *limit* is an upper bound on the length of the substrings that can match. We explore how *limit* fits into the framework of the logarithmic and linear regions of alignment parameter space. When computing gapped alignments, using our function 1, *limit* should be set to 3. For functions 2 and 3 *limit* should be under 10 and for function 4, *limit* can be as high as 50. We give two examples of composition alignments for human RNA polymerase II promoters where the composition alignment scores are statistically significant even though there is no detectable similarity with standard alignment.

References

1. A. Amir, Y. Aumann, G. Landau, M. Lewenstein, and N. Lewenstein. Pattern matching with swaps. *J. Algorithms*, 37:247–266, 2000.
2. A. Amir, R. Cole, R. Hariharan, M. Lewenstein, and E. Porat. Overlap matching. In *Proc. 12th ACM-SIAM Sym. on Discrete Algorithms*, pages 279–288, 2001.
3. A. Amir, M. Lewenstein, and E. Porat. Approximate swapped matching. *Information Processing Letters*, 83:33–39, 2002.
4. R. Arratia and M. Waterman. A phase transition for the score in matching random sequences allowing deletions. *Ann. Appl. Prob.*, 4:200–225, 1994.
5. C.J. Benham. Duplex destabilization in superhelical DNA is predicted to occur at specific transcriptional regulatory regions. *J. Mol. Biol.*, 255:425–434, 1996.
6. C.J. Benham. The topologically driven strand separation transition in DNA—methods of analysis and biological significance. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 47:173–198, 1999.
7. G. Bernardi. The isochore organization of the human genome. *Annu. Rev. Genet.*, 23:637–661, 1989.
8. G. Bernardi. The human genome: Organization and evolutionary history. *Annu. Rev. Genet.*, 29:445–476, 1995.
9. P. Bucher. Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *J. Mol. Biol.*, 212:563–578, 1990.
10. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
11. W. Doerfler. DNA methylation and gene activity. *Ann. Rev. Biochem.*, 52:93–124, 1983.
12. G. Felsenfeld and J. McGhee. Methylation and gene activity, 1982.
13. M.G. Garden and M. Frommer. CpG islands in vertebrate genomes. *J. Mol. Biol.*, 196:261–282, 1987.
14. D.S. Goodsell and R.E. Dickerson. Bending and curvature calculations in B-DNA. *Nucleic Acids Research*, 22:5497–5503, 1994.

15. O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.
16. T. Heinemeyer, X. Chen, H. Karas, A. Kel, O. Kel, I. Liebich, T. Meinhardt, I. Reuter, F. Schacherer, and E. Wingender. Expanding the TRANSFAC database towards an expert system of regulatory molecular mechanisms. *Nucleic Acids Res.*, 27:318–322, 1999.
17. S. Karlin and S. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA*, 87:2264–2268, 1990.
18. H-S. Koo, H-M. Wu, and D.M. Crothers. DNA bending at adenine - thymine tracts. *Nature*, 320:501–506, 1986.
19. M. Lewis, G.Chang, N.C. Horton, M.A. Kercher, H.C. Pace, M.A. Schumacher, R.G. Brennan, and P. Lu. Crystal structure of the lactose operon repressor and its complexes with DNA and inducer. *Science*, 271:1247–1254, 1996.
20. J. Lin. Divergence measures based on the Shannon entropy. *IEEE Trans. Inf. Theor.*, 37:145–151, 1991.
21. R. Lowrance and R.A. Wagner. An extension of the string-to-string correction problem. *JACM*, 22:177–183, 1975.
22. S. Needleman and C. Wunch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.
23. R. P erier, V. Praz, T. Junier, C. Bonnard, and P. Bucher. The Eukaryotic Promoter Database (EPD). *Nucleic Acids Research*, 28:302–303, 2000.
24. S.C. Schultz, G.C. Shields, and T.A. Steitz. Crystal structure of a CAP-DNA complex: The DNA is bent by 90 degrees. *Science*, 253:1001–1007, 1991.
25. A. Smit. The origin of interspersed repeats in the human genome. *Curr. Opin. Genet. Dev.*, 6:743–748, 1996.
26. T. Smith and M. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.
27. M. Vingron and M. Waterman. Sequence alignment and penalty choice: review of concepts, case studies and implications. *J. Mol. Biol.*, 235:1–12, 1994.
28. R.A. Wagner. On the complexity of the extended string-to-string correction problem. In *Proceedings 7th ACM STOC*, pages 218–223, 1975.
29. M. Waterman, L. Gordon, and R. Arratia. Phase transitions in sequence matches and nucleic acid structure. *Proc. Natl. Acad. Sci. USA*, 84:1239–1243, 1987.
30. E. Yeramian. Genes and the physics of the DNA double-helix. *Gene*, 255:139–50, 2000.
31. E. Yeramian, S. Bonnefoy, and G. Langsley. Physics-based gene identification: proof of concept for *Plasmodium falciparum*. *Bioinformatics*, 18:190–193, 2002.