in Proceedings of the 16th International Conference on Pattern Recognition, Quebec, Canada, Aug. 2002, Volume 3, pp. 1019–1022

Fast Semi-local Alignment for DNA Sequence Database Search

Yong-Sheng Chen† Yi-Ping Hung‡∗ Chiou-Shann Fuh∗

†Integrated Brain Research Lab., Dept. of Medical Research & Education,
Taipei Veterans General Hospital, Taipei, Taiwan
‡Institute of Information Science, Academia Sinica, Taipei, Taiwan
∗Dept. of Computer Science and Information Engineering, National Taiwan University, Taiwan

Abstract

Given a query DNA sequence, our goal is to find in the DNA sequence database all the sequence segments that are similar to the query. In this paper we present a string-to-signal transform technique that can transform a DNA sequence into a four-channel signal. Without considering gaps, the edit distance between two DNA sequences can be calculated as the sum of absolute difference (SAD) between their corresponding four-channel signals. The algorithm proposed in this paper can then be applied to speed up the process of searching for the desired sequence segments that yield small SADs. In addition to efficiency, this algorithm guarantees the optimal search. That is, all the sequence segments that are similar enough to the query can be found without any miss.

1. Introduction

Recently, DNA molecules of many organisms, including the human species, have been sequenced and the amount of sequence information has been on a rapid increase. Confronted with a wealth of genomic sequences, scientists can not analyze biological information and interpret genetic messages without using efficient sequence analysis tools. Among these tools is the sequence database search tool. When a new DNA molecule is sequenced, the next step that a biologist will be eager to take is to find in databases the DNA sequences that are similar to the newly obtained sequence [4]. This routine is very important because sequence homology implies the evolution and gene function clues[3].

A DNA sequence can be represented as a string over an alphabet of four characters {A, T, G, C}. The similarity score of sequence alignment between two DNA sequences can be obtained by calculating the edit distance between their corresponding strings. The problem of finding the minimum edit distance (or the maximum similarity score) is called pairwise *global alignment* of sequences. This problem can be solved by using dynamic programming techniques [5, 6] and the computational cost is proportional to the length product of the two sequences.

Dynamic programming is not suitable for searching large-scale sequence databases due to its high computational cost. Instead of adopting the global alignment, Altschul et al. proposed a basic *local alignment* search tool, BLAST [1]. Their goal is to find in the database the sequence segments that are similar enough to a segment in the query sequence according to a local similarity score, the maximal segment pair (MSP) score. Here, a sequence segment is a contiguous stretch of nucleotides in a sequence. BLAST gains its efficiency mainly by neglecting gaps for insertions and deletions of nucleotides when calculating the local similarity scores. Moreover, BLAST can only find an approximation of the desired set of alignments by using a heuristic method to skip unlikely sequence segments. Nevertheless, BLAST becomes the most widely-used search engine for sequence databases due to its efficiency and versatility.

In this work, we consider another alignment strategy, the *semi-local alignment*. Given a query sequence, our goal is to search the sequence databases for the sequence segments that are similar enough to the *whole* query sequence. Similar to BLAST [1], gaps are not considered here. By using the proposed string-to-signal transform, we can transform the sequences in the database and the query sequence into four-channel signals. The problem of finding the sequence segments having high enough similarity score can be transformed into the problem of finding the signal segments having small enough sum of absolute difference (SAD). By utilizing the distance lower bound [2], the proposed algorithm can efficiently finds the desired segments. Another advantage of the proposed algorithm is that it can guarantee to find in the database all the desired sequence segments without any miss.

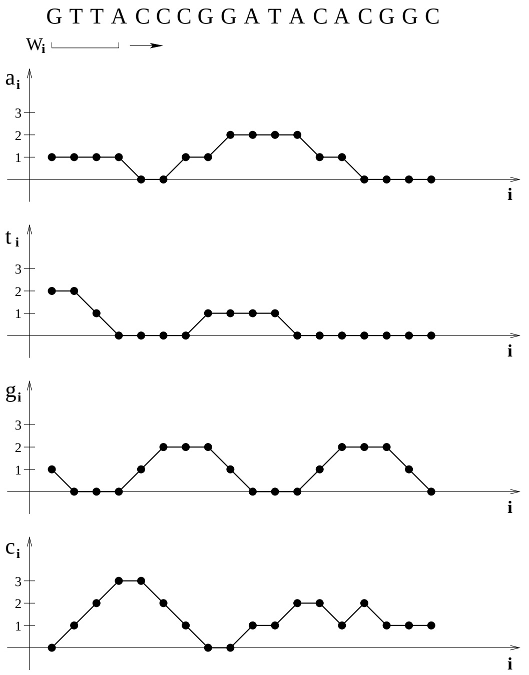

Figure 1. String-to-signal transform for a DNA sequence of 18 nucleotides, GTTACCCGGATACACGGC. The size of the sliding window is set to be 4.

2. String-to-Signal Transform

Consider a DNA sequence S that consists of s nucleotides, $S = n_1 n_2 \cdots n_s$. We use a sliding window of length w to scan through the DNA sequence S. Let W_i denote the sliding window located at position i in S, $i = 1, \ldots, s$. To prevent the boundary condition, we extend S to be of length $s + w - 1$ by concatenating a sequence of $w - 1$ null characters at the tail of S. Within the sliding window W_i is the segment of DNA sequence containing the nucleotides $n_i n_{i+1} \cdots n_{i+w-1}$. Let $\mathbf{a}$ denote the channel-A signal of length s. Each element a_i of $\mathbf{a}$ is assigned to be the number of occurrences of the nucleotide ‘A’ within the sliding window W_i. That is, a_i can be calculated by:

$$\begin{aligned} a_i &= \sum_{j=i}^{i+w-1} F_A(n_j), \\ F_A(n) &= \begin{cases} 1 & \text{if } n = \text{‘A’} \\ 0 & \text{otherwise} \end{cases}, \end{aligned}$$

where $i = 1, \ldots, s$. By using similar equations, we can obtain another three channels, $\mathbf{t}$, $\mathbf{g}$, and $\mathbf{c}$, for nucleotides ‘T’, ‘G’, and ‘C’, respectively. These four vectors, $\mathbf{a}$, $\mathbf{t}$, $\mathbf{g}$, and $\mathbf{c}$, are the transformed four-channel signal for the DNA sequence S. Fig. 1 shows an example of this string-to-signal transform. Notice that extra memory storage is required to store this four-channel signal.

3. Rapid DNA Sequence Database Search

3.1. Similarity Score Calculation

Let P denote a DNA sequence of s nucleotides in the databases. P can be represented as a string, $P = p_1 p_2 \cdots p_s$, where $p_i \in \{\mathrm{A}, \mathrm{T}, \mathrm{G}, \mathrm{C}\}$. The sequence segment P_i of length d starting at position i can be represented as $P_i = p_i p_{i+1} \cdots p_{i+d-1}$, $i = 1, \ldots, s - d + 1$. Given a query sequence Q of length d, $Q = q_1 q_2 \cdots q_d$, the similarity score, $S(i)$, between Q and P_i is defined as:

$$S(i) = \sum_{j=1}^{d} F(q_j, p_{i+j-1}),$$

$$F(p, q) = \begin{cases} S_I & \text{if } p = q \text{ (identity)} \\ -S_M & \text{if } p \neq q \text{ (mismatch)} \end{cases},$$

where $S_I, S_M \geq 0$ are the scores for identity and mismatch, respectively. Usually, S_I is larger than S_M (in BLAST, $S_I = 5$ and $S_M = 4$). Notice that gaps are not considered here when calculating the similarity score, $S(i)$.

Suppose there are $M(i)$ mismatches (or replacements of nucleotides) between Q and P_i. The similarity score $S(i)$ can also be calculated by:

$$S(i) = (d - M(i)) \cdot S_I - M(i) \cdot S_M. \tag{1}$$

Scanning through the sequence P, all the similarity scores $S(i)$, $i = 1, \ldots, s - d + 1$, between the query sequence Q and the sequence segment P_i at positions i can be calculated by using Eq. (1). Whether the sequence segment P_i is similar enough to the query sequence Q can be determined by comparing the similarity score $S(i)$ with a threshold T_S. That is, P_i is reported if $S(i) \geq T_S$, which can be rewritten as:

$$M(i) \leq \frac{d \cdot S_I - T_S}{S_I + S_M}.$$

Let T_M denote the threshold number of mismatches:

$$T_M = \frac{d \cdot S_I - T_S}{S_I + S_M}.$$

We can then determine whether the sequence segment P_i is similar enough to the query sequence Q by calculating the number of mismatches, $M(i)$, and comparing with the threshold T_M. Only there exists small enough number of mismatches does the sequence segment be reported.

3.2. Occurrence Difference and Its Lower Bound

The number of mismatches, $M(i)$, between the sequence segment P_i and the query sequence Q can be calculated by using the occurrence vectors of P_i and Q. For the DNA sequence P, its four occurrence vectors $\mathbf{p}_a$, $\mathbf{p}_t$, $\mathbf{p}_g$, and $\mathbf{p}_c$ record the occurrence of nucleotides A, T, G, and C at each position. For example, the value of p_{a_i} in $\mathbf{p}_a$ is set to be 1 if p_i is A; otherwise, it is set to be 0. At each position i, only one among p_{a_i}, p_{t_i}, p_{g_i}, and p_{c_i} will be 1 and others are all 0s. Notice that these four occurrence vectors, $\mathbf{p}_a$, $\mathbf{p}_t$, $\mathbf{p}_g$, and $\mathbf{p}_c$, can be obtained by using the proposed string-to-signal transform while setting the window size to be 1. By using the same method, we can obtain the four occurrence vectors, $\mathbf{q}_a$, $\mathbf{q}_t$, $\mathbf{q}_g$, and $\mathbf{q}_c$, for the query sequence Q.

The total number of occurrence differences for nucleotide A between Q and P_i can be calculated as the sum of absolute difference SAD_a between $\mathbf{q}_a$ and $\mathbf{p}_a$:

$$\mathrm{SAD}_a(i) = \sum_{j=1}^{d} |q_{a_j} - p_{a_{i+j-1}}|,$$

where $i = 1, \ldots, s-d+1$. Similarly, we can define SAD_t, SAD_g, and SAD_c which account for the total number of occurrence differences for other three nucleotides, T, G, and C, respectively. The total number of occurrence differences for all the four nucleotides, SAD, can then be calculated as:

$$\mathrm{SAD}(i) = \mathrm{SAD}_a(i) + \mathrm{SAD}_t(i) + \mathrm{SAD}_g(i) + \mathrm{SAD}_c(i),$$

where $i = 1, \ldots, s-d+1$. Since a mismatch (replacement) of a pair of nucleotides consists of a deletion of one nucleotide followed by an insertion of another nucleotide, the total number of occurrence differences for all the four nucleotides is equal to twice the total number of mismatches, i.e., $\mathrm{SAD}(i) = 2M(i)$, where $i = 1, \ldots, s-d+1$. Consequently, the threshold can be set to be $2T_M$ when determining whether the number of mismatches for the corresponding pair of sequences is small enough. To sum up, we can find the sequence segment P_i such that the similarity score between P_i and the query sequence Q is greater than or equal to a threshold T_S by calculating each $\mathrm{SAD}(i)$, $i = 1 \ldots, s-d+1$, and looking for the position i such that $\mathrm{SAD}(i) \leq 2T_M$.

Remember that the four occurrence vectors can be obtained by using the string-to-signal transform, where the size of the sliding window is set to be 1. When a larger window size $w > 1$ is chosen, we can obtain another four-channel signal, $\{\mathbf{p}_a{}^w, \mathbf{p}_t{}^w, \mathbf{p}_g{}^w, \mathbf{p}_c{}^w\}$, by using the string-to-signal transform. Each channel is of the same length, s, as the original DNA sequence. The query sequence Q can also be transformed by using the same window size w. For simplicity, we assume that d is a multiple of w. Instead of sliding through every positions of Q, the numbers of occurrences are only calculated and recorded for non-overlapping windows located at positions $1, w+1, 2w+1, \ldots, d-w+1$. Each channel of the obtained four-channel signal, $\{\mathbf{q}_a{}^w, \mathbf{q}_t{}^w, \mathbf{q}_g{}^w, \mathbf{q}_c{}^w\}$, is of length d/w.

The total number of occurrence differences, $\mathrm{SAD}_a{}^w$, for $\mathbf{p}_a{}^w$ and $\mathbf{q}_a{}^w$ can be calculated by

$$\mathrm{SAD}_a{}^w(i) = \sum_{j=0}^{d/w-1} |q_a{}^w{}_{j+1} - p_a{}^w{}_{i+j\cdot w}|,$$

where $i = 1, \ldots, s-d+1$. We can calculate $\mathrm{SAD}_t{}^w$, $\mathrm{SAD}_g{}^w$, and $\mathrm{SAD}_c{}^w$ by using similar equations. The total number of occurrence differences, SAD^w, between these two four-channel signals is then defined as:

$$\mathrm{SAD}^w(i) = \mathrm{SAD}_a{}^w(i) + \mathrm{SAD}_t{}^w(i) + \mathrm{SAD}_g{}^w(i) + \mathrm{SAD}_c{}^w(i).$$

Following a similar derivation given in [2], we can obtain:

$$\mathrm{SAD}^w(i) \leq \mathrm{SAD}(i), \qquad (2)$$

where $i = 1, \ldots, s-d+1$. That is, $\mathrm{SAD}^w(i)$ is a lower bound of $\mathrm{SAD}(i)$.

3.3. Proposed Algorithm

If $\mathrm{SAD}^w(i)$ is already larger than the threshold $2T_M$, the calculation of $\mathrm{SAD}(i)$ can be saved. The reason is that $\mathrm{SAD}(i)$ has no chance to be smaller than the threshold according to the lower bound inequality described in Eq. (2). Otherwise, $\mathrm{SAD}(i)$ still has to be calculated to determine whether the sequence segment P_i is similar enough to the query sequence Q. Notice that the computational cost of $\mathrm{SAD}^w(i)$ is $O(d/w)$ while that of $\mathrm{SAD}(i)$ is $O(d)$. We can gain computational efficiency when $\mathrm{SAD}^w(i)$ is larger than the threshold $2T_M$ and the calculation of $\mathrm{SAD}(i)$ can be saved for many positions. The proposed algorithm for rapid DNA sequence database search is summarized below.

```
/* Preprocessing Stage */
for each DNA sequence $P$ of length $s_P$ in the database
    Perform the string-to-signal transform to obtain two
        four-channel signals:
        (1) $\{\mathbf{p}_a, \mathbf{p}_t, \mathbf{p}_g, \mathbf{p}_c\}$, sliding window size is 1
        (2) $\{\mathbf{p}_a{}^w, \mathbf{p}_t{}^w, \mathbf{p}_g{}^w, \mathbf{p}_c{}^w\}$, sliding window size is $w$

/* DNA Sequence Search Stage */
Given a query DNA sequence $Q$ of length $d$
    Perform the string-to-signal transform to obtain two
        four-channel signals:
        (1) $\{\mathbf{q}_a, \mathbf{q}_t, \mathbf{q}_g, \mathbf{q}_c\}$, sliding window size is 1
        (2) $\{\mathbf{q}_a{}^w, \mathbf{q}_t{}^w, \mathbf{q}_g{}^w, \mathbf{q}_c{}^w\}$, sliding window size is $w$
    for each DNA sequence $P$ of length $s_P$ in the database
        for $i = 1$ to $s_P - d + 1$
            Calculate $\mathrm{SAD}^w(i)$
            if $\mathrm{SAD}^w(i) \leq 2T_M$
                Calculate $\mathrm{SAD}(i)$
                if $\mathrm{SAD}(i) \leq 2T_M$
                    Output $P_i$
                endif
            endif
        next i
    endfor
```

For the query sequence and the sequence segment under examination, if their occurrence numbers of the four

Table 1. Mean search time, t, for different length, d, of the query sequence (mutation rate is set to be $1/128$).

d	256	512	1024	2048	4096
t (ms)	273.7	282.4	330.6	420.6	523.9

nucleotides accumulated in a window differ a lot, the resulted number of mismatches will be even larger. If their occurrence numbers are close to each other, on the other hand, we still cannot determine whether the number of mismatches is small enough because the order of nucleotides in the window is unknown. To determine the similarity, we have to calculate the number of mismatches by using the occurrence vectors obtained when the window size is 1.

4. Experiments

Two experiments of the proposed algorithm for DNA sequence database search were conducted on a PC with a Pentium III 700 MHz CPU. We show the mean execution time of various inquiries when searching the DNA sequence database of E. coli. There are totally 4,662,239 nucleotides constituting 400 sequences in the database. For each of the 400 sequences, we obtained two sets of four-channel signals by using the string-to-signal transform while the window size was set to be 1 and 128. We also generated a query set consisting of 4000 sequences. Each query sequence is of length d and is randomly extracted from a contiguous interval of a sequence in the database. After the extraction, nucleotides of the query sequence are randomly replaced by others at a mutation rate $1/m$. The threshold T_M is set to be $2 \cdot d/m$. All the 400 sequences in the database are scanned through and the sequence segments having small enough number of mismatches (i.e., $\mathrm{SAD} \leq 2T_M$) are reported.

In the first experiment, we perform the DNA sequence database search for different length, d, of the query sequence. The mutation rate, $1/m$, is set to be $1/128$. As the length d increases from 256 to 4096, the threshold T_M increases from 4 to 64 and the mean search time also increases from 273.7 ms to 523.9 ms, as shown in Table 1.

The second experiment was conducted to analyze the computational efficiency of the proposed algorithm for different mutation rate, $1/m$. When the mutation rate is lower (i.e., when m is larger), the query sequence will be more similar to the obtained matched sequence segment. We can assign a smaller value for the threshold T_M. Hence, calculations of SAD for more sequence segments can be skipped because their lower bounds, SAD^w, have more chance to be larger than T_M. Thus improves the efficiency. As m increases from 128 to 2048, T_M reduces from 64 to 4 and the mean search time also reduces from 523.9 ms to 181.3 ms, as shown in Table 2.

Table 2. Mean search time, t, for different mutation rate, $1/m$ (the length, d, of the query sequence is 4096).

m	128	256	512	1024	2048
t (ms)	523.9	329.9	234.6	191.8	181.3

For reference, we performed the sequence database search on the same machine by using BLAST, whose efficiency is roughly independent of m. The mean search time is 205.7 ms. Notice that it is not very fair to compare the computational efficiency of the proposed algorithm with that of BLAST because of the following two reasons: (1) the proposed algorithm can find, without any miss, all the sequence segments which are similar enough to the query sequence while BLAST cannot guarantee this optimal search; and (2) the proposed algorithm performs the semi-local alignment while BLAST performs the local alignment.

5. Conclusions

A fast semi-local alignment algorithm for DNA sequence database search is proposed. The DNA sequences are transformed into four-channel signals by using the string-to-signal transform. The sequence segments in the database that are similar enough to the query sequence can be found efficiently, at the expense of extra memory storage, by calculating and comparing the SAD distances between the obtained four-channel signals. Another advantage of the proposed algorithm is that it guarantees the optimal search. That is, all the sequence segments that are similar enough to the query sequence can be found without any miss.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Blasic local alignment search tool. *J. of Molecular Biology*, 215:403–410, 1990.

[2] Y.-S. Chen, Y.-P. Hung, and C.-S. Fuh. Winner-update algorithm for nearest neighbor search. In *Proceedings of the International Conference on Pattern Recognition*, vol. 2, pp. 708–711, Barcelona, Spain, Sept. 2000.

[3] J. P. Fitch and B. Sokhansanj. Genomic engineering: Moving beyond DNA sequence to function. *Proceedings of the IEEE*, 88(12):1949–1971, Dec. 2000.

[4] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, New York, 1997.

[5] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. of Molecular Biology*, 48:443–453, 1970.

[6] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. of the ACM*, 21(1):168–173, 1974.