Space-conserving Optimal DNA-Protein Alignment*

Pang Ko Mahesh Narayanan Anantharaman Kalyanaraman Srinivas Aluru
Department of Electrical and Computer Engineering
Iowa State University
{kopang, maheshn, ananthk, aluru}@iastate.edu

Abstract

DNA-protein alignment algorithms can be used to discover coding sequences in a genomic sequence, if the corresponding protein derivatives are known. They can also be used to identify potential coding sequences of a newly sequenced genome, by using proteins from related species. Previously known algorithms either solve a simplified formulation, or sacrifice optimality to achieve practical implementation. In this paper, we present a comprehensive formulation of the DNA-protein alignment problem, and an algorithm to compute the optimal alignment in $O(mn)$ time using only four tables of size $(m+1) \times (n+1)$, where m and n are the lengths of the DNA and protein sequences, respectively. We also developed a Protein and DNA Alignment program PanDA that implements the proposed solution. Experimental results indicate that our algorithm produces high quality alignments.

1. Introduction

Over the past few years, complete genomes of several model eukaryotic organisms, both plant and animal, and an assortment of bacteria have been sequenced. Sequencing of many more complete, and in some cases partial genomes (e.g. maize) is currently underway. Processing these raw genomic sequences is essential to discover the genes within these sequences, assign putative functions, and computationally predict the intronic and exonic boundaries of these genes. As protein sequences indirectly reflect the splicing behavior of their corresponding genomic sources, methods that align protein sequences with their corresponding genes can be used for both annotation and intron-exon predictions. Moreover, since protein sequences tend to be better conserved than DNA sequences over time, DNA-protein alignment methods allow using proteins from distant-related organisms for predicting genes and their coding sequences in newly sequenced organisms. It is also of great interest − especially in light of recent "gene-enrichment" strategies [8, 9] for economically important crops − in low coverage sequencing projects that may be subject to a higher error rate within single-coverage areas than traditional BAC-based approaches. Our DNA-protein alignment method can overcome potential bottlenecks caused by these errors and mutation events, and provides a comprehensive solution for locating potential coding sequences within genomes.

The problem of aligning a DNA sequence with a protein sequence poses more intriguing challenges than just locating the appropriate codon to align to each amino acid. Both biological events (such as presence of introns and mutation events) and sequencing artifacts introduced by errors in experimental processes complicate the problem. Computationally, introns are typically detected as sufficiently long nucleotide insertions (based on a threshold length) that may or may not contain the appropriate start and end splice signals. In addition, introns between two consecutive exons may occur either between two successive codons (*in-frame introns*) or within a codon (*out-of-frame introns*). Special care is required while detecting out-of-frame introns because the nucleotides that form a codon may no longer be consecutive. Henceforth, we will refer to this event as *codon-splitting*. Another independent complication arises from mutation events that insert, delete or substitute nucleotides in the DNA sequence. Insertions and deletions in coding sequences may cause frameshift errors depending on their lengths, and substitutions may cause mismatches in the alignment. Methods that compute alignments based on DNA-protein sequence homology [1, 3, 4, 6, 10] typically vary in the extent to which the aforementioned cases are incorporated

* Research supported by NSF under ACI-0203782.

IEEE COMPUTER SOCIETY

into their problem definitions, and more so on how these cases are handled to affect the optimality of the solution [7].

Let m be the length of the DNA sequence and n be the length of the protein sequence to be aligned. Gelfand *et al.* [3] introduced the first combinatorial approach to detect exon-intron structure based on DNA-protein alignment. The run-time of the algorithm is $O(mnc + nb)$, where b is the number of blocks in the genomic sequence and c is the genomic coverage by the blocks. In the worst case b could be $\Theta(m^2)$, and c could be $\Theta(m)$. Huang *et al.* [6] introduced a method that addresses both introns and frameshift errors, but a direct implementation of their algrithm is not efficient [6]. Of the 12 tables that the algorithm uses, 4 tables have a two-dimensional structure (corresponding to the two sequences being aligned). The remaining 8 tables are three-dimensional, where the third dimension (of size 20 - one for each amino acid) is introduced specifically to handle out-of-frame introns. Deeming the resulting 164 2-dimensional tables as impractical, Huang *et al.* developed *NAP* [6] that uses only 10 tables at the risk of generating sub-optimal alignments. The run-time of this algorithm is $O(mn)$. The implementation also uses Hirschberg's method [5] to further reduce space usage at the cost of roughly doubling the runtime.

Compared to the algorithm by Huang *et al.*, Gotoh's algorithm [4] uses only seven tables, involving a novel "tron"-based approach that takes advantage of an observed pattern in the genetic code. This algorithm assumes that if a codon is split, it is always split by an intronic length insertion. Such an assumption may not hold especially while aligning genes with proteins obtained from their orthologs. Another issue lies in the treatment of consecutive gap characters in the nucleotide sequence. An alignment of a codon that starts with a gap character to an amino acid is given a gap opening penalty even if it follows another gap character. This behavior may have the impact of reporting an undesirable "optimal" alignment, because, we would expect a continuing gap to receive a gap continuation penalty only. In the context of applying DNA-protein alignment methods to proteins and genomic sequences derived from different species, a thorough treatment of all such significant cases is essential. The program *aln* is an implementation of this algorithm.

In this paper, we present a simple, space-economical, optimal algorithm for DNA-protein alignment. Let m and n be the lengths of the DNA and protein sequences, respectively. We make the following contributions:

- A comprehensive formulation of the DNA-protein alignment problem as an optimization problem taking into account indels, substitutions, frameshift errors, and intronic insertions between and within codons.
- An algorithm to compute an optimal alignment in $O(mn)$ time using only four tables of size $(m + 1) \times (n + 1)$. This is significant because three tables are typically used even for the simplest of the alignment problems with affine gap penalties − global alignment of two DNA sequences.
- Development of a Protein and DNA Alignment program *PanDA* that produces an optimal alignment for global, end-gap free or local DNA-protein alignments.

2. Problem Formulation

Let $A = a_1a_2 \ldots a_m$ be a DNA sequence where each a_i is a nucleotide, and $B = b_1b_2 \ldots b_n$ be a protein sequence where each b_j is an amino acid residue. Let $A' = a'_1a'_2 \ldots a'_{m'}$ be a sequence obtained from A by inserting one or more gap characters (denoted by '-'). An alignment between A' and B is valid if it satisfies the following two conditions: (i) Each b_j is aligned with three characters a'_x, a'_y, a'_z in A' such that $1 \leq x < y < z \leq m'$, and (ii) For any two amino acids b_j and b_k, $1 \leq j < k \leq n$, that are aligned with a'_x, a'_y, a'_z and $a'_{x'}, a'_{y'}, a'_{z'}$ respectively, $z < x'$. Conditions (i) and (ii) imply that $|A| + 3|B| \geq |A'| \geq 3|B|$. Let $S_{A,B} = \{A'|A' \text{ is valid}\}$. Note that there could be multiple valid alignments between each A' and B. We denote the set of valid alignments for a specific A' as $F_{A',B}$. Following the convention in [6], let $\sigma(a'_xa'_ya'_z, b_j)$ denote the function that specifies the score of aligning $a'_xa'_ya'_z$ with the amino acid b_j.

We define gaps in an alignment with respect to A'. A maximal substring of gap characters in A' aligned with a substring of B is called a *deletion gap*. A maximal substring of nucleotide characters in A' that is not aligned with any amino acid in B is an *insertion gap*. Each nucleotide in an insertion gap is referred to as a *nucleotide insertion*. Let $G_D = \{g_1, g_2, \ldots, g_{|G_D|}\}$ be the set of all deletion gaps, and $G_I = \{g_1, g_2, \ldots, g_{|G_I|}\}$ be the set of all insertion gaps. Let q be the gap opening penalty and r be the gap extension penalty, and let k be the minimum length for an insertion gap to be consid-

IEEE COMPUTER SOCIETY

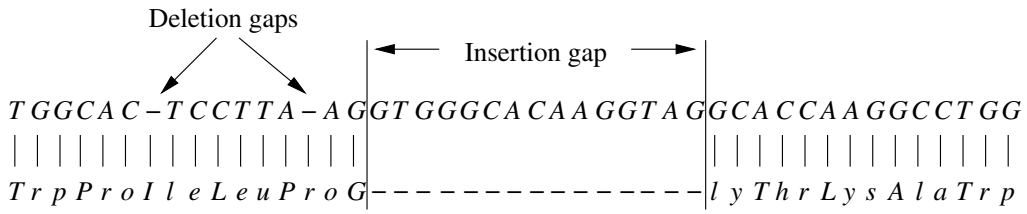

Figure 1. Example of a DNA-protein alignment. The alignment of the amino acid Gly corresponds to a codon-split.

ered an intron. The gap penalty functions γ_I and γ_D are defined as follows:

$$\gamma_I(|g_i|) = q + \begin{cases} r|g_i|, & \text{if } |g_i| < k,\ g_i \in G_I, \\ p_I, & \text{if } |g_i| \geq k,\ g_i \in G_I, \end{cases}$$

where $0 \leq p_I \leq rk$ is referred to as the *intronic penalty*. For convenience, we set $p_I = rk$ in the rest of the paper.

$$\gamma_D(|g_d|) = q + \ \ r|g_d|, \qquad g_d \in G_D$$

The score of a valid alignment, $f \in F_{A',B}$ is:

$$score(f) \ = \sum_{1 \leq j \leq n} \sigma(a'_x a'_y a'_z, b_j) - \sum_{\forall g_i \in G_I} \gamma_I(|g_i|) - \sum_{\forall g_i \in G_D} \gamma_D(|g_i|)$$

An optimal alignment of sequences A and B is:

$$\max_{A' \in S_{A,B}} \left\{ \max_{f \in F_{A',B}} \{score(f)\} \right\}$$

See Figure 1 for an example of a DNA-protein alignment. For illustrative purposes, each insertion gap is shown as being aligned with gaps in the amino acid sequence. Also, each amino acid is denoted by its standard three-letter abbreviation.

3. Optimal Solution

Our dynamic programming solution to compute an optimal alignment uses only four $(m+1)\times(n+1)$ tables. In each table, entry $[i, j]$ contains the score of an optimal alignment between $a_1, a_2, \ldots, a_i$ and $b_1, b_2, \ldots, b_j$ subject to certain restrictions. $I[i, j]$ stores the score of an optimal alignment such that a_i is a nucleotide insertion, and $II[i, j]$ stores the score of an optimal alignment such that a_i is part of an insertion gap of length greater or equal to the minimum length of an intron k. Both tables I and II do not need to be stored completely, only 2 columns of each table are needed. If a_i is not a nucleotide insertion, then two possibilities arise: (i) a_i is the third character that is aligned to the amino acid b_j, and the corresponding optimal score is stored in $T[i, j]$, (ii) a_i is not the third character aligning with b_j, implying that the third character is a deletion gap character, and the corresponding optimal score is stored in $\alpha[i, j]$. Figure 2 illustrates the list of cases we consider. Note that we do not address an alignment where a codon is split by two insertion gaps, one between its first and second characters, and another between its second and third characters, any one or two of the nucleotides of the codon could be a gap character. Biologically, a necessary condition for a codon to be split twice is the presence of a unit-length exon, a rare event [2]. For all other cases not listed in Figure 2 there is a case in our formulation that produces equal or better score.

The tables I, II, T, and α are computed one column at a time. All the following recurrences assume that column $j-1$ of each table has been computed and column j is currently being computed. Initialization of the leftmost column and the top row of each table is straightforward, and is omitted for brevity. The recurrence for computing table I is given by:

$$I[i, j] = max \begin{cases} I[i-1, j] - r, \\ T[i-1, j] - q - r, \\ \alpha[i-1, j] - q - r \end{cases}$$

The recurrence for computing table II is given by:

$$II[i, j] = max \begin{cases} II[i-1, j], \\ T[i-k, j] - q - kr, \\ \alpha[i-k, j] - q - kr, \end{cases}$$

The recurrences for tables T and α are shown in Figure 3 and Figure 4 in correspondence to the cases they handle. We refrain from explaining the recurrences for cases 2 through 5 and for cases 13

IEEE COMPUTER SOCIETY

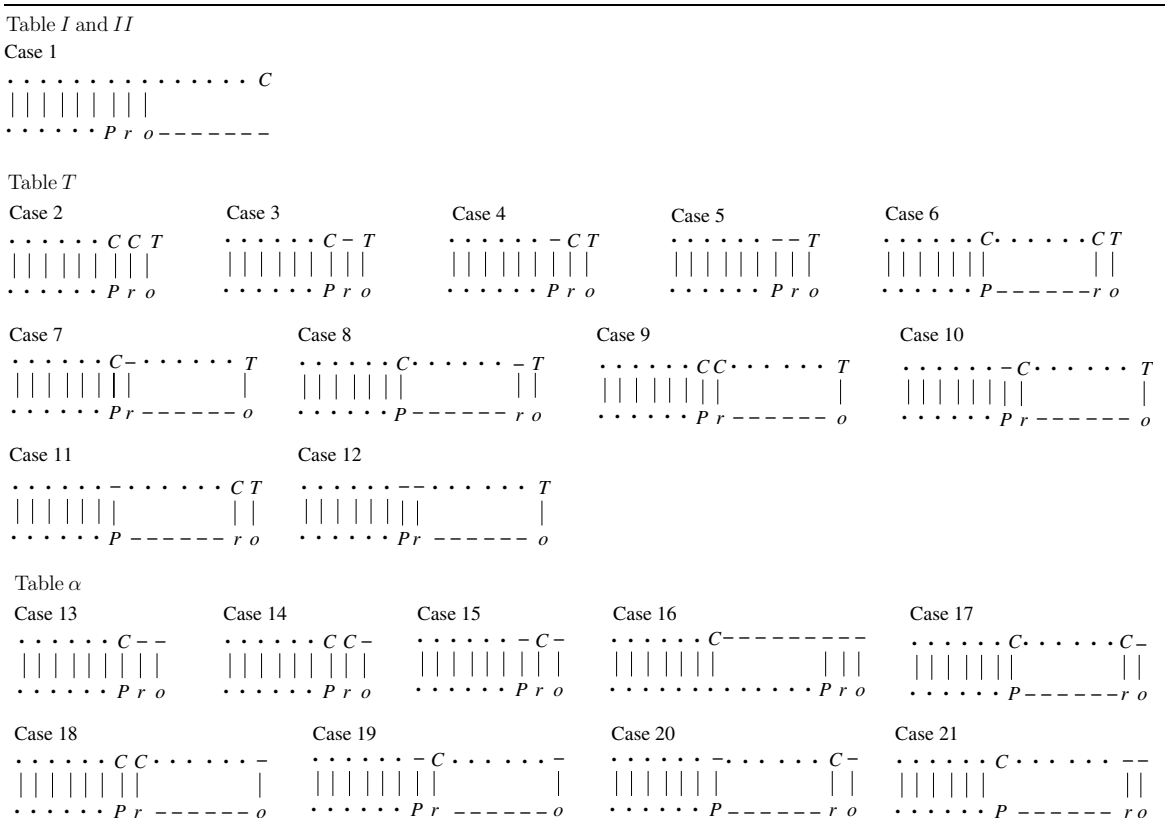

Figure 2. An example for each case considered by our formulation. Case 1 corresponds to table I and table II. Cases 2 through 12 correspond to table T, and the remaining correspond to table α. The alignment of an amino acid to each of the three characters in the nucleotide sequence is denoted by '|'. Pro denotes the amino acid Proline.

through 16 as they are straightforward and conform to techniques that are typical to other dynamic programming approaches of the problem. However, the recurrences for cases 6 through 10 and cases 17 through 21 may not be evident, and to that purpose, we explain the notion of *M vectors* and *IM vectors*. Observe that, in these cases, the three characters that are aligned with the amino acid b_j exhibit codon-splitting although in different flavors. For a thorough discussion of a specific case, consider case 6. For the remaining codon-splitting cases, a similar explanation holds.

Let $Opt(i,j) = max\{I[i,j], T[i,j], \alpha[i,j], II[i,j]\}$, and let $score_{i'}(i,j) = Opt(i'-1, j-1) - \gamma_I(i-i'-2) + \sigma(a_{i'}a_{i-1}a_i, b_j)$. $Opt(i,j)$ is used only for exposition and is not stored. The optimal score corresponding to case 6 can be expressed as: $S_{ij} = \max_{1 \le i' \le i-3} \{score_{i'}(i,j)\}$.

In terms of the dynamic programming table, S_{ij} corresponds to finding an i' from column $j-1$ such that $score_{i'}(i,j)$ is maximized. Although $1 \le i' \le i-3$, $a_{i'} \in \{A, C, G, T\}$.

Consider i'_1 and i'_2, such that $i-k-2 < i'_1, i'_2 < i-1$ and $a_{i'_1} = a_{i'_2}$, i.e., both i'_1 and i'_2 are within the intronic length from i. If $score_{i'_1}(i,j) < score_{i'_2}(i,j)$, then $score_{i'_1}(i+1,j) < score_{i'_2}(i+1,j)$. This is because, by increasing the value of i by 1, the value of the γ_I term increases by r in both the cases of i'_1 and i'_2, and $\sigma(a_{i'_1}a_i a_{i+1}, b_j) = \sigma(a_{i'_2}a_i a_{i+1}, b_j)$ (because $a_{i'_1} = a_{i'_2}$). Thus, the choice of i' that yielded the optimal score at i for nucleotide $a_{i'}$ will still hold to be the choice at $i+1$, unless the new candidate for i', $i-3$, results in a better score. Based on this intuition, we define four vectors $M_A[i], M_C[i], M_G[i], M_T[i]$, one for each of the four possibilities of $a_{i'}$. For $x \in \{A, C, G, T\}$,

IEEE COMPUTER SOCIETY

Case 2:
$\alpha[i-3,j-1]+\sigma(a_{i-2}a_{i-1}a_i,b_j)$
$T[i-3,j-1]+\sigma(a_{i-2}a_{i-1}a_i,b_j)$
$I[i-3,j-1]+\sigma(a_{i-2}a_{i-1}a_i,b_j)$
$II[i-3,j-1]+\sigma(a_{i-2}a_{i-1}a_i,b_j)$

Case 3:
$\alpha[i-2,j-1]+\sigma(a_{i-1}-a_i,b_j)-q-r$
$T[i-2,j-1]+\sigma(a_{i-1}-a_i,b_j)-q-r$
$I[i-2,j-1]+\sigma(a_{i-1}-a_i,b_j)-q-r$
$II[i-2,j-1]+\sigma(a_{i-1}-a_i,b_j)-q-r$

Case 4:
$\alpha[i-2,j-1]+\sigma(-a_{i-1}a_i,b_j)-r$
$T[i-2,j-1]+\sigma(-a_{i-1}a_i,b_j)-q-r$
$I[i-2,j-1]+\sigma(-a_{i-1}a_i,b_j)-q-r$
$II[i-2,j-1]+\sigma(-a_{i-1}a_i,b_j)-q-r$

Case 5:
$\alpha[i-1,j-1]+\sigma(--a_i,b_j)-2r$
$T[i-1,j-1]+\sigma(--a_i,b_j)-q-2r$
$I[i-1,j-1]+\sigma(--a_i,b_j)-q-2r$
$II[i-1,j-1]+\sigma(--a_i,b_j)-q-2r$

Case 6:
$M_A[i]+\sigma(Aa_{i-1}a_i,b_j)$
...
$M_T[i]+\sigma(Ta_{i-1}a_i,b_j)$
$IM_A[i]+\sigma(Aa_{i-1}a_i,b_j)$
...
$IM_T[i]+\sigma(Ta_{i-1}a_i,b_j)$

Cases 7, Case 8:
$M_A[i+1]+\sigma(A-a_i,b_j)-q-r$
...
$M_T[i+1]+\sigma(T-a_i,b_j)-q-r$
$IM_A[i+1]+\sigma(A-a_i,b_j)-q-r$
...
$IM_T[i+1]+\sigma(T-a_i,b_j)-q-r$

Case 9:
$M_{AA}[i]+\sigma(AAa_i,b_j)$
...
$M_{TT}[i]+\sigma(TTa_i,b_j)$
$IM_{AA}[i]+\sigma(AAa_i,b_j)$
...
$IM_{TT}[i]+\sigma(TTa_i,b_j)$

Case 10:
$M_{-A}[i]+\sigma(-Aa_i,b_j)$
...
$M_{-T}[i]+\sigma(-Ta_i,b_j)$
$IM_{-A}[i]+\sigma(-Aa_i,b_j)$
...
$IM_{-T}[i]+\sigma(-Ta_i,b_j)$

Case 11:
$M_*^\alpha[i]+\sigma(-a_{i-1}a_i,b_j)-q-r$
$IM_*^\alpha[i]+\sigma(-a_{i-1}a_i,b_j)-q-r$

Case 12:
$M_*^\alpha[i+1]+\sigma(--a_i,b_j)-q-2r$
$IM_*^\alpha[i+1]+\sigma(--a_i,b_j)-q-2r$

Figure 3. Recurrences for computing $T[i,j]$. The value chosen is the maximum of all the entries shown.

If $a_{i-3}=x$, then

$$M_x[i]=max\begin{cases}M_x[i-1]-r,\\ \alpha[i-4,j-1]-q-r,\\ T[i-4,j-1]-q-r,\\ I[i-4,j-1]-q-r,\\ II[i-4,j-1]-q-r,\end{cases}$$

Otherwise

$$M_x[i]=M_x[i-1]-r$$

Note for M_x, we only consider the values of α, T, I and II if $a_{i-3}=x$, and for the case $a_{i-3}\neq x$ the equation is a subset of the case $a_{i-3}=x$. For the M and IM vectors below we only present the case where the appropriate characters match, and the other case is implied.

Each of the above vectors can be stored as a single variable by overwriting $M_x[i-1]$ with $M_x[i]$. These values are then used to compute the optimal score at $T[i,j]$ as shown in Figure 3 (for case 6).

These recurrences only work for the case where length of the insertion gap is less than k. For addressing intronic insertion gaps (of length $\geq k$), column $j-1$ above row i can be partitioned into two segments: S_1 $(=1\ldots i-k-2)$ and S_2 $(=i-k-1\ldots i-1)$. As we proceed from entry $[i-1,j]$ to $[i,j]$, the entry $[i-k-3,j-1]$ moves from S_2 to S_1. Based on this observation, we define four more vectors $IM_A[i], IM_C[i], IM_G[i], IM_T[i]$, which correspond to cases where the choice for i' comes from segment S_1. These values can be updated as shown below, prior to their usage as shown in Figure 3 (for case 6). For $x\in\{A,C,G,T\}$,
If $a_{i-k-2}=x$, then

$$IM_x[i]=max\begin{cases}IM_x[i-1],\\ \alpha[i-k-3,j-1]-q-kr,\\ T[i-k-3,j-1]-q-kr,\\ I[i-k-3,j-1]-q-kr,\\ II[i-k-3,j-1]-q-kr,\end{cases}$$

For cases 7 and 8 of table T, and cases 17 and 21 of table α, vectors M_x and IM_x can be used, but since in each of those cases, the number of characters used is less than that of case 6, an index greater than i is needed. Although in the recurrences of tables T and α we use $M_x[i+1]$, $M_x[i+2]$, and their IM counterparts, it is important to note that all those values can be calculated with existing information due to the recurrence relation of M_x and IM_x.

A similar technique as outlined for case 6 can be adapted for case 9 and case 18. The only difference is that we need 16 vectors in place of 4, because there are 16 possibilities for first and second nucleotides aligned with the amino acid b_j. The following recurrences apply for case 9. For $x,y\in\{A,C,G,T\}$,

IEEE COMPUTER SOCIETY

Case 13:
$\alpha[i-1,j-1]+\sigma(a_i--,b_j)-q-2r$
$T[i-1,j-1]+\sigma(a_i--,b_j)-q-2r$
$I[i-1,j-1]+\sigma(a_i--,b_j)-q-2r$
$II[i-1,j-1]+\sigma(a_i--,b_j)-q-2r$

Case 14:
$\alpha[i-2,j-1]+\sigma(a_{i-1}a_i-,b_j)-q-r$
$T[i-2,j-1]+\sigma(a_{i-1}a_i-,b_j)-q-r$
$I[i-2,j-1]+\sigma(a_{i-1}a_i-,b_j)-q-r$
$II[i-2,j-1]+\sigma(a_{i-1}a_i-,b_j)-q-r$

Case 15:
$\alpha[i-1,j-1]+\sigma(-a_i-,b_j)-q-2r$
$T[i-1,j-1]+\sigma(-a_i-,b_j)-2q-2r$
$I[i-1,j-1]+\sigma(-a_i-,b_j)-2q-2r$
$II[i-1,j-1]+\sigma(-a_i-,b_j)-2q-2r$

Case 16:
$\alpha[i,j-1]-3r$
$T[i,j-1]-q-3r$
$I[i,j-1]-q-3r$
$II[i,j-1]-q-3r$

Case 17:
$M_A[i+1]+\sigma(Aa_i-,b_j)-q-r$
...
$M_T[i+1]+\sigma(Aa_i-,b_j)-q-r$
$IM_A[i+1]+\sigma(Aa_i-,b_j)-q-r$
...
$IM_T[i+1]+\sigma(Aa_i-,b_j)-q-r$

Case 18:
$M_{AA}[i+1]+\sigma(AA-,b_j)-q-r$
...
$M_{TT}[i+1]+\sigma(TT-,b_j)-q-r$
$IM_{AA}[i+1]+\sigma(AA-,b_j)-q-r$
...
$IM_{TT}[i+1]+\sigma(TT-,b_j)-q-r$

Case 19:
$M_{-A}[i+1]+\sigma(-A-,b_j)-q-r$
...
$M_{-T}[i+1]+\sigma(-T-,b_j)-q-r$
$IM_{-A}[i+1]+\sigma(-A-,b_j)-q-r$
...
$IM_{-T}[i+1]+\sigma(-T-,b_j)-q-r$

Case 20:
$M_*^{\alpha}[i+1]+\sigma(-a_i-,b_j)-2q-2r$
$IM_*^{\alpha}[i+1]+\sigma(-a_i-,b_j)-2q-2r$

Case 21:
$M_A[i+2]+\sigma(A--,b_j)-q-2r$
...
$M_T[i+2]+\sigma(T--,b_j)-q-2r$
$IM_A[i+2]+\sigma(A--,b_j)-q-2r$
...
$IM_T[i+2]+\sigma(T--,b_j)-q-2r$

Figure 4. Recurrences for computing $\alpha[i,j]$. The value chosen is the maximum of all the entries shown.

If $a_{i-3}a_{i-2}=xy$, then

$$M_{xy}[i]=max\begin{cases} M_{xy}[i-1]-r, \\ \alpha[i-4,j-1]-q-r, \\ T[i-4,j-1]-q-r, \\ I[i-4,j-1]-q-r, \\ II[i-4,j-1]-q-r, \end{cases}$$

If $a_{i-k-2}a_{i-k-1}=xy$, then

$$IM_{xy}[i]=max\begin{cases} IM_{xy}[i-1], \\ \alpha[i-k-3,j-1]-q-kr, \\ T[i-k-3,j-1]-q-kr, \\ I[i-k-3,j-1]-q-kr, \\ II[i-k-3,j-1]-q-kr, \end{cases}$$

Case 10 can be viewed as a variation of case 9, where the first character aligned with the amino acid b_j is a deletion gap character. The following gives the recurrences for computing the M vectors corresponding to case 10. For $x\in\{A,C,G,T\}$,

If $a_{i-2}=x$, then

$$M_{-x}[i]=max\begin{cases} M_{-x}[i-1]-r, \\ \alpha[i-3,j-1]-q-2r, \\ T[i-3,j-1]-2q-2r, \\ I[i-3,j-1]-2q-2r, \\ II[i-3,j-1]-2q-2r, \end{cases}$$

If $a_{i-k-1}=x$, then

$$IM_{-x}[i]=max\begin{cases} IM_{-x}[i-1], \\ \alpha[i-k-2,j-1] \\ \qquad -q-(k+1)r, \\ T[i-k-2,j-1] \\ \qquad -2q-(k+1)r, \\ I[i-k-2,j-1] \\ \qquad -2q-(k+1)r, \\ II[i-k-2,j-1] \\ \qquad -2q-(k+1)r, \end{cases}$$

For cases 11, 12, and 20, a similar case already exists. For example, case 11 is similar to case 4, because we can move the insertion gap in case 11 to the front of the leading deletion gap. This shuffling of insertion and deletion gap will not affect the score of the alignment, unless the optimal alignment of $A[i-3]$ and $B[j-1]$ ends with a deletion gap. In that case, case 11 will have higher score than case 4. So to handle those cases, we need to apply the same technique as before to only the α table. Since all of those cases begin with a deletion gap, it does not matter which nucleotide it is. Thus we have,

$$M_*^{\alpha}[i]=max\begin{cases} M_*^{\alpha}[i-1]-r, \\ \alpha[i-3,j-1]-q-r, \end{cases}$$

IEEE COMPUTER SOCIETY

Gene Name	Ahsg	Bf	CD4	Cyp21	DRPLA	ENO2	IL6	ISOT	Pim1	UPA
GeneID	197	629	920	1589	1822	2062	3569	8078	5292	118471

Table 1. List of genes used in our experiments

$$IM_*^{\alpha}[i] = max \begin{cases} IM_*^{\alpha}[i-1], \\ \alpha[i-k-2, j-1] - q - kr, \end{cases}$$

Even though not explicitly stated, each M vector is stored as a single variable. Each of the tables I, T, and α can be augmented to allow traceback operations for retrieving an alignment that resulted in the optimal score. For each entry of a table, we keep track of the table and the coordinate that was used to obtain its score. The coordinates are needed because if the previous value is from the M or IM values, we cannot recalculate the row number from which they originate.

4. Experimental Results

Our algorithm is implemented in C++ as a program named *PanDA* (Protein and DNA Alignment). Exploiting the fact that the tables I and II represent a single case each, it is sufficient to keep track of traceback information only on tables T and α. This further reduces space usage because a column of table I or table II can be discarded as soon as the next column in the table is computed. In addition to the parameters q, r, k, and σ, *PanDA* also awards introns that begin and/or end with canonical splice sites (e.g., 'GT','AG'). Users can choose among the alignment types: global, semi-global, or local alignment. For comparative evaluation, we tested two other programs: *aln* [4] and *NAP* [6]. Comparison with *NAP* is especially meaningful because both the underlying formulations use the same scoring of alignments. The only difference is that *PanDA* always finds an optimal scoring alignment.

We classify our experiments into two groups: (i) *homologous alignments*, which are experiments performed on human genes and their corresponding human proteins, and (ii) *orthologous alignments*, which are experiments performed on human genes and orthologous proteins. Ten human genes and their corresponding proteins were obtained from GenBank. For each of these human genes, we also obtained at least one orthologous protein identified by a gene name search in GenBank. Table 1 summarizes the genes selected for our experiments.

To measure the quality of the alignments produced, we compared the predicted results with GenBank annotation. If a nucleotide is part of both a predicted exon (intron), and an exon (intron) in the GenBank annotation, it is a true positive (negative), denoted by TP (TN). If a nucleotide is a part of a predicted exon (intron), but is not part of an exon (intron) in GenBank annotation, it is called a false positive (negative), denoted by FP (FN). The following quality measures are then derived: *Specificity* ($SP = \frac{TP}{TP+FP}$) is the percentage of correctly predicted exon nucleotides over the number of predicted exon nucleotides. *Sensitivity* ($SN = \frac{TP}{TP+FN}$) is the percentage of correctly predicted exon nucleotides over the number of annotated exon nucleotides. *Overlap quality* is given by $OQ = \frac{TP}{TP+FP+FN}$, and *correlation coefficient* is given by $CC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)\cdot(TN+FN)\cdot(TP+FN)\cdot(TN+FP)}}$.

4.1. Homologous alignments

The results produced by *PanDA* for homologous alignments is as follows: Of the 10 homologous alignments, all show complete agreement with their corresponding GenBank annotation, with the exception of DRPLA. For this case our alignment shows an insertion gap of length 15 from position 8398 to 8412. The total length of all coding sequences in the GenBank annotation is 3570 nucleotides, while the corresponding protein has 1184 amino acids. Taking the stop codon into account, the total length of all coding sequences in the annotation is 15 nucleotides longer than its corresponding protein sequence indicates. This justifies the observed behavior of our result. The results of the homologous alignments using *PanDA* validates both our formulation and algorithm applied to real biological data. As an additional test, we aligned 76 *Zea mays* genes with their corresponding homologous proteins, and in all but 2 cases *PanDA* computed perfect alignments, further validating our algorithm.

4.2. Orthologous alignments

Results for orthologous alignments show the ability of a program to predict coding sequences by using proteins from related species. Table 2 shows

IEEE COMPUTER SOCIETY

Gene	*PanDA*				*NAP*			
	SP	SN	OQ	CC	SP	SN	OQ	CC
$Ahsg_B$	**100**	**96.73**	**96.73**	**98.11**	95.92	89.49	86.21	91.57
$Ahsg_R$	**100**	**95.92**	**95.92**	**97.63**	**100**	95.83	95.83	97.58
Bf_R	**100**	**99.87**	**99.87**	**99.90**	**100**	99.74	99.74	99.79
$CD4_R$	**97.66**	**97.02**	**94.81**	**97.21**	95.61	93.46	89.62	94.29
$Cyp21_R$	98.92	**98.12**	**97.07**	**97.33**	**99.16**	95.43	94.66	95.18
$Cyp21_S$	**100**	**99.13**	**99.13**	**99.21**	**100**	97.98	97.98	98.19
$DRPLA_R$	**100**	**99.24**	**99.24**	**99.49**	82.86	78.85	67.80	74.50
$ENO2_R$	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
$IL6_S$	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
$IL6_B$	96.33	**94.52**	91.24	94.73	**96.64**	**94.52**	**91.52**	**94.90**
$IL6_R$	95.25	**94.05**	**89.84**	**93.83**	**96.42**	92.80	89.71	93.79
$ISOT_X$	**97.15**	**99.20**	**99.40**	**98.00**	96.06	97.40	93.67	96.43
$Pim1_B$	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
$Pim1_R$	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
UPA_R	95.03	**92.32**	88.08	93.06	**99.51**	89.25	**88.86**	**93.73**
Avg.	**99.15**	**98.56**	**97.74**	**98.66**	96.31	94.26	90.97	94.46

Table 2. Comparative evaluation of *PanDA*, and *NAP*. The subscript of a gene name denotes the species corresponding to the protein. *B* denotes *Bos taurus*; *R* denotes *Rattus norvegicus*; *S* denotes *Sus scrofa*; and *X* denotes *Xenopus laevis*. The best values in each experiment are displayed in bold.

a comparative evaluation of *PanDA*, and *NAP* for orthologous DNA-protein alignments. For both of the programs, the results presented are under the parameter settings which we empirically found to yield the best results when compared to the GenBank annotation. The parameter values used are $q = 9$, $r = 3$, and $k = 9$ for both *PanDA* and *NAP*. For both of the programs, Blosum62 was used as the substitution matrix. As can be seen from Table 2, *PanDA* produces better results in more experiments. It is important to note that *PanDA* never produces an alignment with lower score than *NAP*. Thus, in cases where *NAP* produces a better result than *PanDA* according to GenBank annotation (presumed to be correct), it is interesting to note that a combinatorially suboptimal solution happens to be the biologically preferred solution. Of course, *NAP* produces such a solution not because it is biologically correct, but because it is the optimal of all the cases considered by it. A graphical view of the orthologous alignments produced by *PanDA* is shown in Figure 5. We also tested two versions of *aln* with the same set of sequences as in Table 2. When sequences from *H. sapiens* is used as the training set, *aln* produced perfect alignments in all but two cases. However, when *C. elegans* is used as the training set the results were not good. This shows that the quality of results produced by *aln* and perhaps other hybrid methods is heavily dependent on the training set used. Therefore, for newly sequenced organisms where organism specific information is not available, *PanDA* produces better results than *NAP* and *aln*. On the other hand, *aln* seems to be superior when such information is readily available.

5. Conclusions

We presented a space-conserving optimal alignment algorithm between a DNA sequence and a protein sequence. Our main contributions include a comprehensive treatment of all possible scenarios relevant to DNA-protein alignments and computing optimal alignments using only four $O(mn)$ sized dynamic programming tables, where m and n are the lengths of the input sequences. Thus, we do not need to use Hirschberg's [5] memory reduction technique to make our algorithm practically useful. This makes the implementation much easier. In our experiments *NAP* runs twice as fast as *PanDA*, this is because *PanDA* uses a more comprehensive formulation than *NAP*, and *PanDA* calculates the optimal solution while *NAP* could produce a sub-optimal solution. Experimental results indicate the robustness and accuracy of our algorithm for both homologous and orthologous align-

IEEE COMPUTER SOCIETY

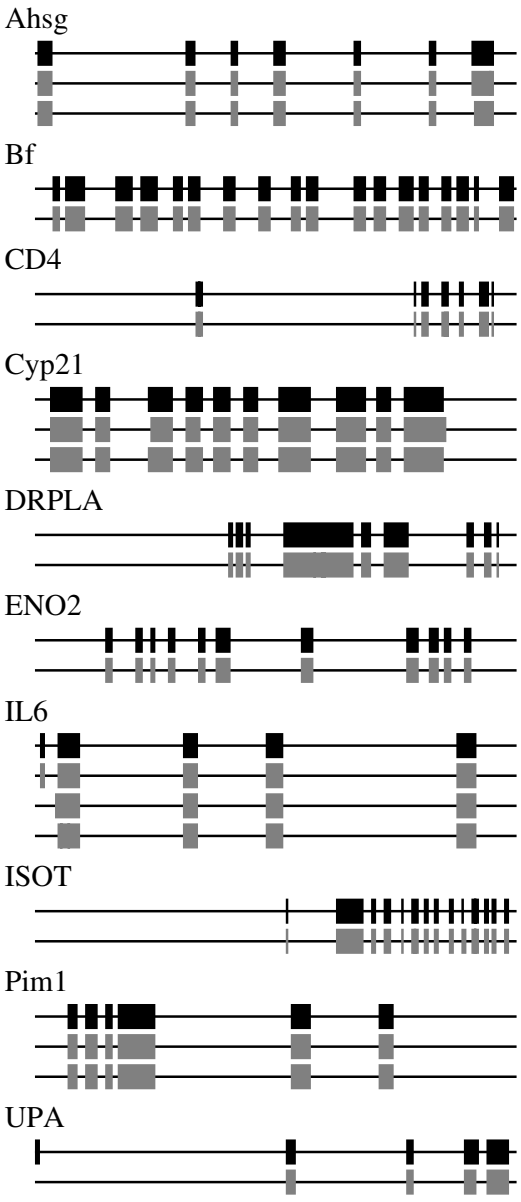

Figure 5. Visualization of the orthologous alignments produced by *PanDA*. For each gene the dark line is the GenBank annotation, the following lines are coding sequences identified by *PanDA* using the orthologous proteins from other species. The order of the proteins is the same as in Table 2.

ments. The software is available for public use by contacting the authors.

Acknowledgments

We would like to thank Scott Emrich for helpful discussions and comments on our paper. We sincerely thank Dr. Osamu Gotoh for making his program *aln* available, and for his prompt responses to our queries regarding his program. We thank Dr. Xiaoqiu Huang for making his program *NAP* available for our use. We also wish to acknowledge Dr. Patrick Schnable and Yan Fu for providing maize test data.

References

[1] E. Birney and R. Durbin. Dynamite: a flexible code generating language for dynamic programing methods used in sequence comparison. In *Proc Int Conf Intell Syst Mol Biol.*, volume 5, pages 56–64, 1997.

[2] M. Deutsch and M. Long. Intron-exon structures of eukaryotic model organisms. *Nucleic Acids Res.*, 27(15):3219–28, 1999.

[3] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. *Proc Natl Acad Sci U.S.A*, 93(17):9061–6, 1996.

[4] O. Gotoh. Homology-based gene structure prediction: Simplified matching algorithm using a translated codon (tron) and improved accuracy by allowing for long gaps. *Bioinformatics*, 16(3):190–202, 2000.

[5] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–3, 1975.

[6] X. Huang and J. Zhang. Methods for comparing a DNA sequence with a protein sequence. *Comput Appl Biosci.*, 12(6):497–506, 1996.

[7] C. Mathé, M. Sagot, T. Schiex, and P. Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Res.*, 30(19):4103–17, 2002.

[8] L. E. Palmer, P. D. Rabinowicz, and A. L. O'Shaughnessy *et al.* Maize genome sequencing by methylation filtration. *Science*, 302(5653):2115–7, 2003.

[9] C. A. Whitelaw, W. B. Barbazuk, and G. Pertea *et al.* Enrichment of gene-coding sequences in maize by genome filtration. *Science*, 302(5653):2118–20, 2003.

[10] Z. Zhang, W. R. Pearson, and W. Miller. Aligning a DNA sequence with a protein sequence. *J Comput Biol.*, pages 339–49, 1997.

IEEE COMPUTER SOCIETY