A Fast and Sensitive Algorithm for Aligning ESTs to the Human Genome

Jun Ogasawara
Department of Computer Science, University of Tokyo
jun@gi.k.u-tokyo.ac.jp

Shinichi Morishita
Department of Computer Science, University of Tokyo
moris@gi.k.u-tokyo.ac.jp

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

There is a pressing need to align the growing set of expressed sequence tags (ESTs) with the newly sequenced human genome. However, the problem is complicated by the exon/intron structure of eukaryotic genes, misread nucleotides in ESTs, and the millions of repetitive sequences in genomic sequences. To solve this problem, algorithms that use dynamic programming have been proposed. In reality, however, these algorithms require an enormous amount of processing time. In an effort to improve the computational efficiency of these classical DP algorithms, we developed software that fully utilizes lookup-tables to detect the start- and endpoints of an EST within a given DNA sequence efficiently, and subsequently promptly identify exons and introns. In addition, the locations of all splice sites must be calculated correctly with high sensitivity and accuracy, while retaining high computational efficiency. This goal is hard to accomplish in practice, due to misread nucleotides in ESTs and repetitive sequences in the genome. Nevertheless, we present two heuristics that effectively settle this issue. Experimental results confirm that our technique improves the overall computation time by orders of magnitude compared with common tools, such as sim4 and BLAT, and simultaneously attains high sensitivity and accuracy against a clean dataset of documented genes.

1. Introduction

The Human Genome Project is an international collaboration, designed to investigate the genetic complexity of humans. Initially, the roughly three billion nucleotides of the human genome were elucidated (Celera Genomics [1], International Human Genome Sequencing Consortium [2]). The next step involves interpreting the encoded sequences. In order to identify the coding regions, i.e., regions containing exons and introns, of any given DNA sequence, many ESTs must be aligned with genomic DNA to reveal these complex structures, while verifying alternative splicing transcripts. The alignment of full-length cDNAs gives clues to regulatory elements in the upstream regions. Furthermore, the annotation of the upstream regions using Transfac data identifies the candidates of cis-elements. The alignment of both sequences associated with expression patterns and sequences from dbSNP with the identified locations of SNPs near the gene, enables more detailed functional

analysis.

Indeed, a variety of sequence alignment algorithms have been proposed. One technique for computing the similarity between two sequences is to assign penalties, designated by letters, to insertions, deletions, and substitutions present in one sequence, but not in the other (Needleman and Wunsch [3], Smith and Waterman [4]). Heuristic algorithms such as FASTA [5] and BLAST [6]are used because of their higher speed compared with dynamic programming methods.

However, these algorithms either require a very large amount of time for processing or they fail to align ESTs to genomes that are known to encode them, because they are designed to determine only the similarity between two sequences, and not to determine eukaryotic gene structure (exon/intron structure) by identifying splice sites between exons and introns. Figure 1 illustrates the problem. We need to decompose an EST into exons and then to align each exon onto the DNA sequence, while preserving the order of exons. The difficulty with the problem is that there are potentially a huge number of ways to decompose an EST. To settle this problem, it is reasonable to define scores for matches and penalties for introducing introns, and then to select the optimal decomposition as that with the best score.

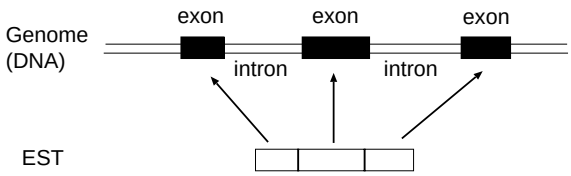

Fig. 1. Decomposing and mapping an EST.

For this optimization problem, many dynamic programming algorithms that consider exon/intron structure have been developed. Gotoh's algorithm [7] defines an affine gap penalty for introns to identify very long introns, since introns correspond to long insertions. Although Gotoh's algorithm runs in $O(MN)$-time complexity and requires $O(MN)$ space for a genomic sequence of length M and for an EST of length N, the space complexity can be reduced to $O(\min(M, N))$ using Hirschberg's technique [8]. Several software tools have been developed that utilize this dynamic programming technique [9,10,11], but in practice, it is computationally infeasible to apply these tools to long genomic sequences of length greater than one million. Conversely, Sim4 [12] and BLAT [13] are improved methods based on BLAST, which can extend multiple exons. Although they are able to decompose a given sequence into its exons, they are not designed to guarantee to output the optimal alignment that maximizes the sum of matching scores.

Since high-performance software is needed to solve this problem, we designed software that shortens the calculation time, while retaining sensitivity and accuracy. This software can align more than 20 ESTs per second, on average, to the draft human genome using a

single processor. In practice, it can process more than 100 ESTs per second using several processors, or about four million ESTs in less than half a day. Regarding sensitivity and accuracy, special care must be taken to identify splice sites in the final step of the software. Clean datasets of splice sites have been collected from various species to derive statistically confirmed rules for improving gene-finding algorithms [15,16,17,18,19]. These clean datasets are also valuable for evaluating the sensitivity and accuracy of alignment software. We used the HMR195 dataset [19], a collection of mammalian sequences, for this. We demonstrate the high sensitivity and accuracy of our method against human genes in HMR195.

2. Algorithm

We start by introducing basic data-structures and simpler algorithms, and move to more elaborate ones by improving the algorithms step by step.

Mapping the millions of ESTs in the GenBank and EST databases (dbEST) to the human genomic sequence is an arduous task. ESTs are as much as tens of thousands of bases long, while the genomic sequence is about 3 billion bases long. In order to shorten the overall calculation time, we pre-process the genomic sequences and create an auxiliary look-up table called a *MapTable*, that stores the position at which each primary key (a subsequence of length K) occurs in the genomic sequence. The idea of using lookup tables has been incorporated in BLAST and FASTA to boost their search performance.

There are other elaborated data structures for strings, such as suffix tree and suffix array. However, suffix tree requires $6 \times G \times 4$ (size of integer or pointer) bytes for a string of length G, which is too large to store in the main memory. Suffix array, by contrast, requires only $G \times 4$ bytes in space, which equals the size of a hash table, but it needs $O(\log G + ooc)$ time for searching (ooc is the candidate query string), while a hash table needs only $O(ooc)$ time for searching. Although these data structures (suffix tree and suffix array) are very useful for comparing sequences, comparison of a genome and its ESTs can be solved using the technique of perfect matching with a hash table. Figure 2 shows how to generate a MapTable when the primary key length is 2, for simplicity.

Referring to the MapTable, it is obvious that "TA" exists at the 9th position and that "GC" occurs at the 3rd and 12th positions in the genome sequence. When considering a particular EST sequence, the position of the L length prefix and the suffix of the EST are inferred by referring to the MapTable in the main memory. Assuming that the four characters (nucleotides) appear at random in the genome sequence, we can deduce the position of a sequence of length L from the MapTable by accessing the main memory about $(G/4^K) \times \lceil L/K \rceil$ times (G is the length of the genome sequence). Care must be taken in selecting appropriate values for K. Smaller K values, say 5, result in aligning an enormous number of positions for each 5-mer, while larger K values increase the number of K-mers, yielding a huge index. We will discuss the selection of an appropriate value for K in Section 3.

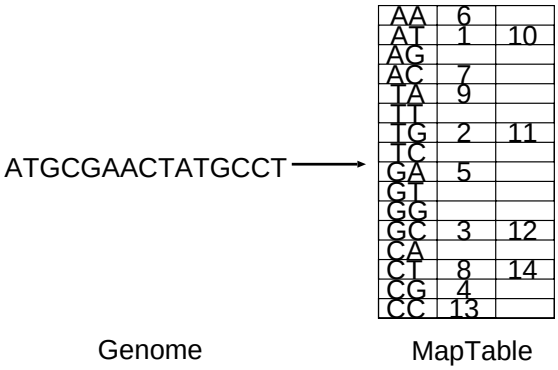

Fig. 2. How to generate a lookup table (MapTable).

2.1. *Simple algorithm assuming no mismatches or gaps*

First, we present a slow, but simple, algorithm that assumes that an EST sequence exactly maps to the genomic sequence with a 100% match ratio. We then improve the algorithm in a stepwise manner to accelerate performance, while allowing mismatches and gaps.

In the following discussion, let us denote the genome sequence G using the sequence $g_1, g_2, ..., g_M$ $(g_i \in \{A, T, G, C, N\})$, and the EST sequence E by the sequence $e_1, e_2, ..., e_N$ ($e_i \in \{A, T, G, C, N\}$). $G_{[i,j]}$ represents a substring, $g_i, g_{i+1}, ..., g_j$, so does $E_{[i,j]}$. We express the alignment of the i-th nucleotide in the EST with genome G as position $f(i)$ in genome G. Here, we assume that each nucleotide e_i maps to a location in G. In practice, however, e_i might be skipped, requiring its position $f(i)$ to be undefined. This case will be considered later in this section. The simplest version of the algorithm (Figure 3) can be written as:

Step 1.(Detection of start- and endpoints)

Let K be the length of a primary key in the MapTable and L be the length of the key used to detect first and last exons. First, we consider a prefix of length L ($E_{[1,L]}$) and a suffix of length L ($E_{[N-L+1,N]}$) in EST E. We align the prefix and suffix with the genome by accessing the MapTable, i.e., associate $f(i)$ for each $i = 1, ..., L$ and $i = N - L + 1, ..., N$. The remainder of the EST, $E_{[L+1,N-L]}$, that should be aligned in the next step, is called the *dangling part*.

Step 2.(Alignment by dynamic programming)

Align the unassigned interval of the EST $E_{[L+1,N-L]}$ with the genome interval $G_{[f(L)+1,f(N-L+1)-1]}$ using Gotoh's dynamic programming, which yields $f(i)$ for each $i = L + 1, ..., N - L$.

In Step 1, there could be multiple locations for $E_{[1,L]}$ and $E_{[N-L+1,N]}$, which will be discussed later in this section. In Step 2, Gotoh's dynamic programming [7] is an algorithm that finds the optimal alignment of an EST that has long introns with the genome. Smith-

Waterman and Needleman-Wunsch methods do not work well for this problem, because they impose a high penalty on long introns and miss alignments with long introns. To overcome this issue, Gotoh's method assigns a small constant or an affine gap penalty to introns that could be very long, and it can output the optimal solution.

Although Step 1 is very efficient at detecting the start- and endpoints of a given EST in the genome sequence because it can be achieved by accessing the main memory, Step 2 sometimes requires a large amount of execution time when the length of the genome interval $G_{[f(L)+1,f(N-L+1)-1]}$ is still long for Gotoh's dynamic programming.

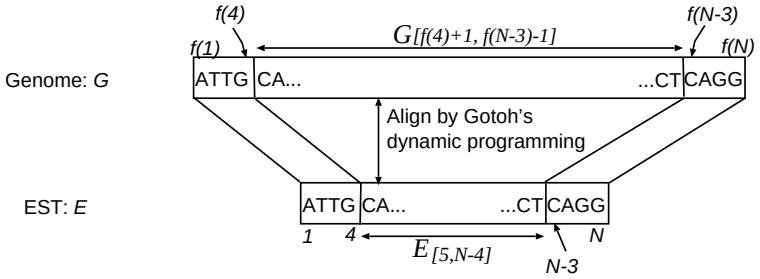

Fig. 3. Simplest algorithm with a MapTable($L = 4$).

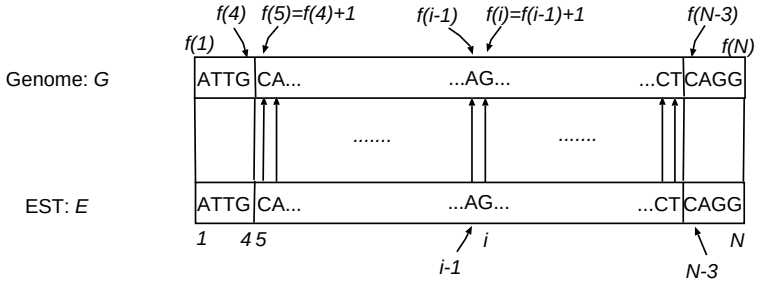

Fig. 4. Fast algorithm with a MapTable (Case of a single exon, $L = 4$).

2.2. *Fast algorithm assuming no mismatches or gaps*

To accelerate the dynamic programming in Step 2, we determine $f(i)$ by elongating the exon and skipping long introns using MapTable. This greatly increases performance because of its practicability in considering exon/intron structure. The algorithm used to align an EST with a single exon is shown.

Step 2 (Identification of a single exon)
For each i from $L+1$ to $N-L$, set $f(i)$ to $f(i-1)+1$ (see Figure 4).

We call Step 2 the *Elongation Step*, because this step adds one nucleotide to the end of the exon per iteration.

To handle ESTs with multiple exons, we incorporate a process that skips an intron when extension of the exon fails (Figure 5).

Step 2.1 (Identification of one exon)
While the i-th nucleotide in E coincides with the $f(i)$-th nucleotide in G, set $f(i) = f(i-1)+1$ and increment i.

Step 2.2 (Search for the next exon) Since Step 2.1 confirmed that the exon terminates at the i-th nucleotide in the EST, detect the position of the next exon by referring to the MapTable with $E_{[i,i+K-1]}$ as a primary key. After determining the locus of the next exon, to which $f(j)$ ($j = i, ..., i+K-1$) is set, increment i by K and return to Step 2.1.

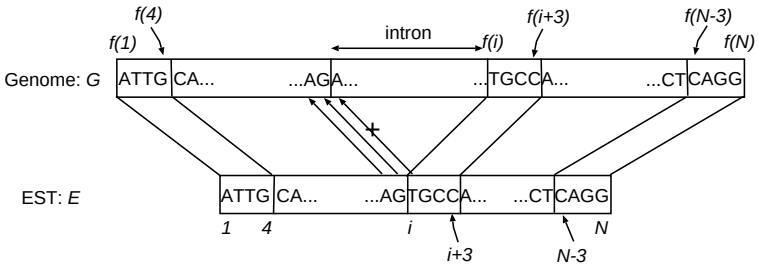

Fig. 5. Fast algorithm with MapTable (Case of multiple exons, $L = 4, K = 4$).

2.3. *Allowing mismatches and gaps in alignment*

In practice, ESTs usually cannot be fully aligned with 100% identity, resulting in mismatches or gaps in the alignment. To allow for these mismatches and gaps, we revise the algorithm in Section 2.2. We allow e_i to be mismatched with a different nucleotide at the $f(i)$-th position in the genome. Alternatively, $f(i)$ is undefined when e_i is skipped and is associated with a gap. In this general setting, the start- and endpoints of the EST in the genome sequence cannot be detected if the prefix $E_{[1,L]}$ (or the suffix $E_{[N-L+1,N]}$) contains mismatches or gaps. To resolve this problem, we scan the EST sequence E from the start until the position of a subsequence of length L is found in the MapTable, and scan E from the end in the same way (Figure 6). This method is described below.

Step 1.1 (Approximation of the startpoint of an EST)
Initialize $i = 1$, and increment i until i equals 300 or the position of $E_{[i,i+L-1]}$ is

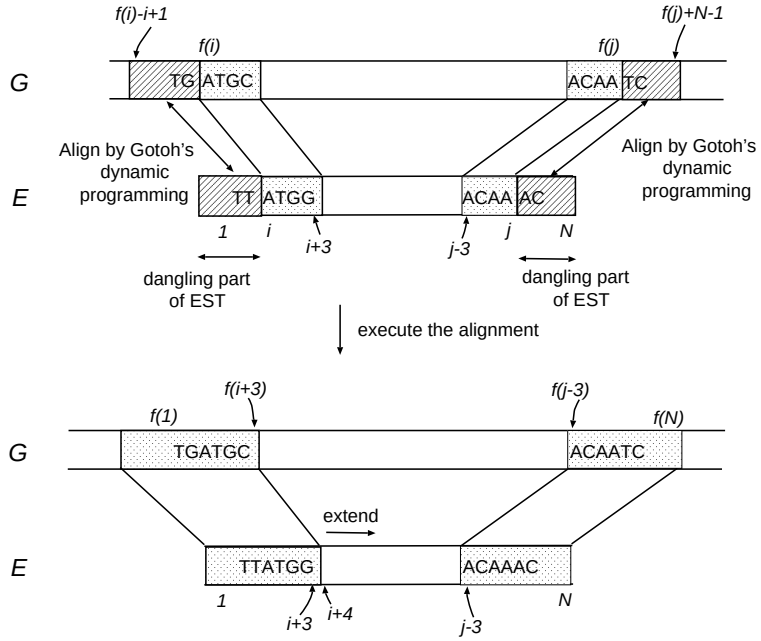

Fig. 6. Approximation of the start- and endpoints ($L = 4$).

found in the MapTable. The rationale behind the choice of 300 bases is presented in Section 3. As mentioned above, $E_{[1,i-1]}$ is called the *dangling part* of the EST, i.e., it is the part that still remains to be aligned. Align this dangling part of EST $E_{[1,i-1]}$ with genome $G_{[f(i)-i+1,f(i)-1]}$ to decide $f(h)$ for each $h = 1,...,i-1$ using Gotoh's dynamic programming.

Step 1.2 (Approximation of the endpoint of an EST)

Similarly, initialize $j = N$, and decrement j until j equals N-300 or the position of $E_{[j-L+1,j]}$ is found in the MapTable. After this, $E_{[j+1,N]}$ is called the dangling part of the EST that still remains to be aligned. Then, align this dangling part of EST $E_{[j+1,N]}$ with genome $G_{[f(j)+1,f(j)+N-j]}$ to decide $f(h)$ for each $h = j+1,...,N$ using Gotoh's dynamic programming.

Mismatches or gaps in an EST make it more complicated to extend an exon or skip an intron. While the elongation stops only when the exon terminates in the case of no mismatches, elongation of the exon fails when it reaches the end of the exon or encounters a mismatch or a gap in the alignment.

Step 2.1 (Identification of one exon)

Initialize i, which is the smallest position in the EST that is not aligned (e.g., the (i+4)-th position of E in Figure 6) and while the i-th nucleotide in E coincides with the $(f(i-1)+1)$-th nucleotide in G, set $f(i) = f(i-1)+1$ and increment i.

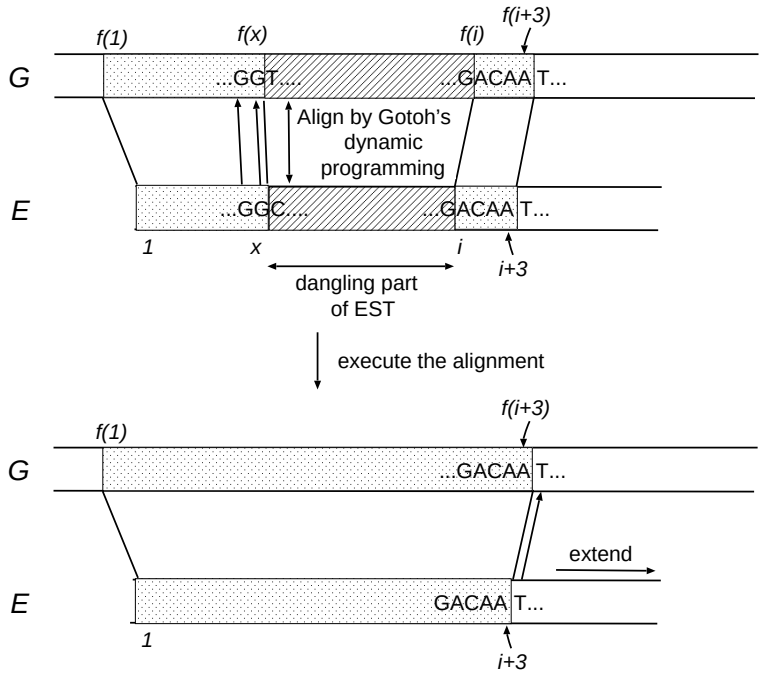

Fig. 7. Alignment of the dangling part of an EST($L = 4, K = 4$).

Then, set $x = i-1$ to memorize the position $i-1$ in the EST where the elongation ends.

Step 2.2 (Search for the next exon)

Increment i until the position of $E_{[i,i+K-1]}$ is found in the MapTable. $E_{[x+1,i-1]}$ is called the dangling part of EST, that remains to be aligned.

Step 2.3 (Alignment of the dangling part of an EST)

Align the dangling part of EST $E_{[x+1,i-1]}$ with the part of genome $G_{[f(x)+1,f(i)-1]}$ using Gotoh's dynamic programming. Figure 7 illustrates the case when no intron is detected after the dynamic programming, and Figure 8 shows when an intron is observed.

The issue of selecting of appropriate values for L and K has be resolved by considering the sensitivity and specificity of the alignment. This will be investigated statistically in Section 3. For the human genome, we assign L=18, partly because we have observed that most first and last exons are longer than 18, although some internal exons are shorter than 18. If the location of the coding region of the exon is figured out, we do not need L to be as long, and we can select smaller values for K, e.g., 11, 12, or 13.

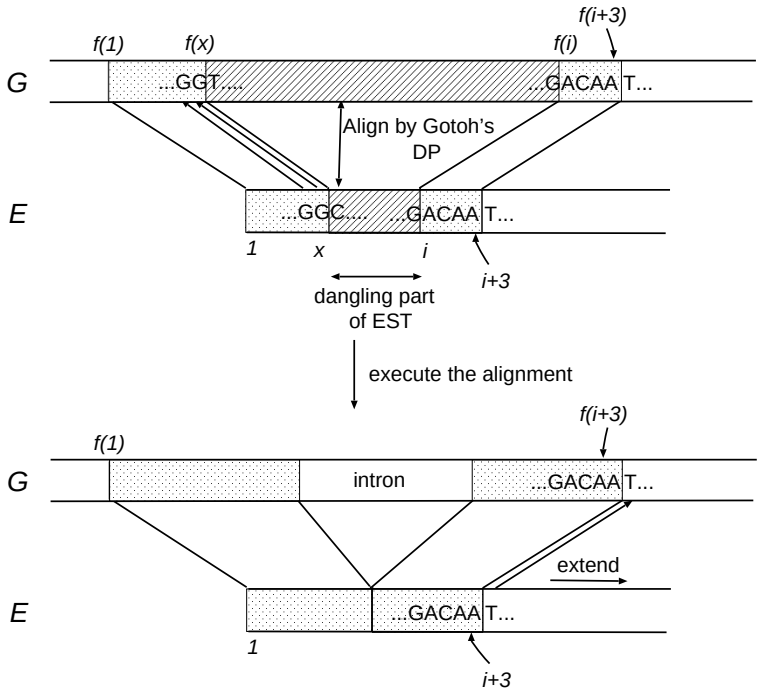

Fig. 8. Alignment of the dangling part of an EST ($L = 4, K = 4$) when part of the genome contains an intron.

2.4. *Further acceleration by preprocessing long introns*

In practice, an intron can be thousands of base pairs long, while the dangling part is at most hundreds of base pairs long. Naive application of dynamic programming to the alignment of the dangling part of the EST $E_{[x+1,i-1]}$ with $G_{[f(x)+1,f(i)-1]}$ is computation intensive when $G_{[f(x)+1,f(i)-1]}$ contains an intron in Step 2.3. To accelerate this step, we examine whether an intron is included in the part of genome $G_{[f(x)+1,f(i)-1]}$ by comparing the length of $E_{[x+1,i-1]}$ with the length of $G_{[f(x)+1,f(i)-1]}$ before applying Gotoh's dynamic programming. If $G_{[f(x)+1,f(i)-1]}$ is much longer than $E_{[x+1,i-1]}$, and the length of $E_{[x+1,i-1]}$ is less than ten, we assume that $G_{[f(x)+1,f(i)-1]}$ contains only one intron, because the length of most exons is much greater than ten. In this case, $E_{[x+1,i-1]}$ should be aligned to $G_{[f(x)+1,f(x)+(i-x)]}$ and $G_{[f(i)-(i-x),f(i)-1]}$. Therefore, we align the dangling part of EST $E_{[x+1,i-1]}$ with the concatenation of two sequences of length $i - x$, which is $G_{[f(x)+1,f(x)+(i-x)]} + G_{[f(i)-(i-x),f(i)-1]}$ (See Figure 9).

2.5. *Detecting splice sites*

We have presented a way to determine approximate exon and intron regions. However, decisions regarding exon/intron boundaries need rigorous investigation, because most boundaries follow the GT-AG rule, while other patterns, such as GC-AG and AT-AC, are also ob-

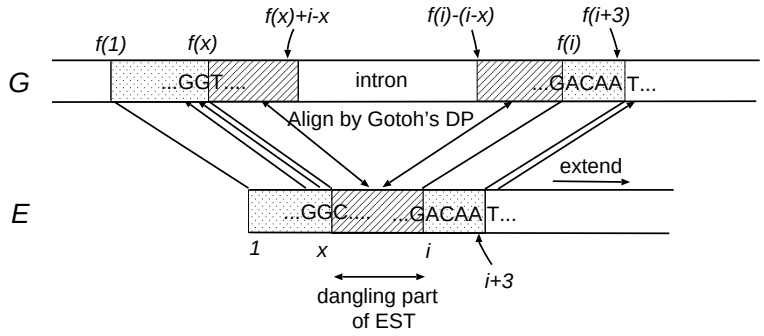

Fig. 9. Acceleration of dynamic programming.

served. In the literature, considerable effort has been made to comprehend variants of the GT-AG rule statistically [15,16,17,18] and to make a comparative analysis of gene-finding programs [19]. For instance, Thanaraj [16] derived decision trees for inferring human exon-intron junctions from a number of EST-confirmed splice sites. Burset et al. [18] also presented statistical rules for mammalian splice sites, which also confirm that most sites obey the GT-AG rule or its variants, such as GC-AG and AT-AC.

Figure 10 illustrates some alternative solutions for deciding splice sites, and the lower alignment should be selected according to the GT-AG rule. To this end, as shown in Figure 11, we shift the intron frame locally along the genome if no more mismatches are introduced in the alignment, for the purpose of detecting GT-AG or its variants. More precisely, the intron frame is moved locally to maximize the number of matches between GT-AG and the four letters at both ends of the intron (two letters at the right end and two letters at the left end).

We will demonstrate the high sensitivity and accuracy of this method later.

2.6. *Match ratio*

Having determined the intron regions, the match ratio between the genome and the mapped EST sequence is examined. Since the match ratio between the genome and a coded EST is 99.9% if the EST sequence is read precisely (the residual 0.1% is the difference between each human genome, i.e., SNP, sequencing errors, or other artifacts), we assume that an EST with a low match ratio is not encoded. In effect, an EST whose match ratio is low contains many misread nucleotides, or is not encoded in the genome sequence. The lower boundary for the match ratio was set at 90%.

2.7. *Solution for two or more alignments*

In this section, we consider cases when there are two or more distinct alignments of an EST in a genome sequence, because an EST is often aligned with many different regions in a chromosome because of retro-transposition or gene-duplication (Figure 12). Furthermore,

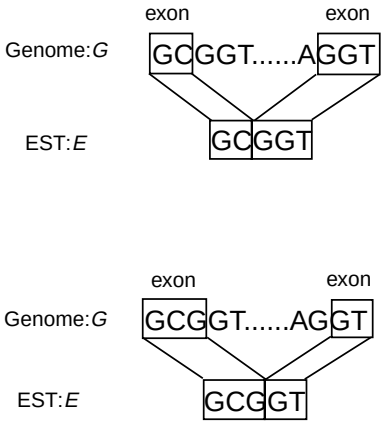

Fig. 10. Alternative solutions for splice sites.

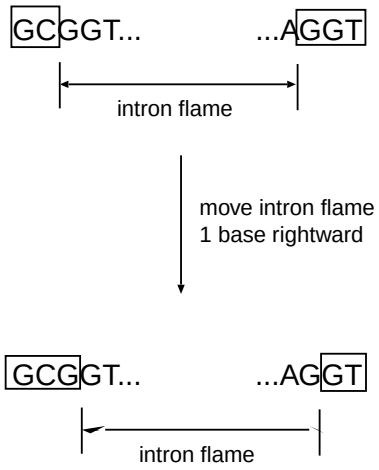

Fig. 11. Shifting the intron frame to detect GT-AG or its slight variants.

the millions of repetitive sequences in the human genome complicate correct identification of start- and endpoints.

Although the start- and endpoints of an EST are inferred in Steps 1.1 and 1.2, the start- and endpoints are not determined uniquely if the MapTable has many candidates for the primary key. Let $StartSet$ denote the set of candidate start-points calculated in Step 1.1, and $EndSet$ denote the set of candidate endpoints in Step 1.2. We must compute alignments for all the pairs in $StartSet \times EndSet$. In practice, however, the size of $StartSet$ or $EndSet$ often exceeds 1000 due to repetitive sequences. To avoid such difficult cases, we

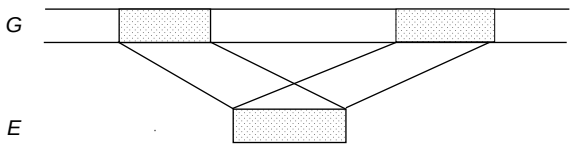

Fig. 12. EST Alignment with multiple regions in a genome.

focus on the fact that the number of 18-mers appearing no more than ten times in the human genome is about 2.2 billion, which is about 64.6% of all the 18-mers in the human genome. In addition, sub-sequences in exons are typically less frequent in the human genome than other sub-sequences in non-coding regions. These facts imply that scanning three hundred bases in a human EST could find, with a high probability, such 18-mers of frequency no more than ten. We therefore revised our algorithm to search for a subsequence of length L until the size of $StartSet$ (or $EndSet$) becomes no more than ten. Consequently, our algorithm is described as:

Step 1.3 (Solution for many start- and endpoints)
Assume that Steps 1.1 and 1.2 output $StartSet$ and $EndSet$ that are of size no more than ten. Solve the alignment of $E_{[i,j-L]}$ with $G_{[start,end]}$ ($start \in StartSet$, $end \in EndSet$) by executing Steps 2.1-2.3, if the distance between $start$ and end is smaller than 3,000,000 bp.

This technique seems to involve brute force, but it solves all the alignments of a given EST reliably.

3. Selecting appropriate parameter values for better alignments

3.1. *Approximation of the start- and endpoints of an EST*

In order to approximate the startpoint of an EST E in Step 1.1, we scan the first 300 bases of the EST to look for an L-mer $E_{[i,i+L-1]}$ that perfectly matches the genome. We selected 18 for L, and 300 for the length of the range to search. In what follows, we discuss the rationale behind this selection from a statistical viewpoint.

Let us call the range to search in the EST the *head sequence*. Given a head sequence of length H (we used 300 for H), our goal is to efficiently associate the head with its homologous range in the genomic sequence by finding one L-mer in the head that perfectly matches the genome. To accelerate the overall performance of this approach, we must exclude false positive L-mers in the head that happen to match the genomic sequence by chance. To reduce such false positive candidates, we can simply use larger values for L in order to increase the specificity of L-mers and hence the computational efficiency of our alignment algorithm. However, longer L-mers may decrease the number of head sequences that successfully map to the genome by using L-mers and may make our algorithm less sensitive, because of noise in both the head and genomic sequences.

Suppose that the match ratio between the head and its homologous region is M_N. When

M_N becomes low, say 85%, it becomes difficult to find one L-mer shared by the head and its homologous region in the genome. Therefore, depending on typical values of M_N, we must decide an appropriate value of L, considering the trade-off between sensitivity and specificity. To resolve this, we present a way of computing the probability that a head of length H and its homologous region share at least one common L-mer for various values of M_N. In the literature, Kent [13] studied this problem when L-mers are distinct and non-overlapping, which is not applicable to our case, since we use overlapping L-mers. Kasahara [21] investigated the statistical properties of overlapping L-mers, and we follow the line suggested by Kasahara.

Let $P(h, m, l)$ denote the number of head sequences of length h, such that each head contains m mismatched nucleotides with its homologous region and the longest subsequence of length l that perfectly matches its homologous region. Observe that the following recurrences hold for $P(h, m, l)$ when $h \geq 1, m \geq 0, l \geq 0$, and $h \geq m + l$:

$$
\begin{aligned}
P(h,0,l) &= 1 \qquad (h = l) \\
P(h,0,l) &= 0 \qquad (h > l) \\
P(h,m,l) &= \sum_{i=0}^{min(l,h-(m+l))} P(h-l-1, m-1, i) \\
&\quad + \sum_{i=0}^{min(l-1,h-(m+l))} P(h-i-1, m-1, l) \\
&\qquad\qquad (m \geq 1)
\end{aligned}
$$

The first two equations express the case when there are no mismatched nucleotides (namely $m = 0$). Hence, h must be equal to l. The last equation handles the case when the right l (i $(< l)$, respectively) nucleotides do not contain any mismatches, but the $l+1$-th ($i+1$-th) nucleotide from the right end is a mismatch.

Then, the probability that a head of length H and its homologous region share at least one L-mer when the match ratio is M_N is:

$$\sum_{m=0}^{H} \sum_{l=L}^{H} P(H, m, l) M_N^{H-m} (1 - M_N)^m.$$

In this formula, we simply assume that the head is not split into more than one exon when the head is aligned with the genome. Table 1 displays the probabilities for various values of M_N and L. Furthermore, to measure the specificity of using each L-mer, the lowest row gives the number F of L-mers in the head that are expected to match by chance, under the condition that all nucleotides occur equally in the sequences. F is defined as the following formula, and the values in the lowest row are calculated for $G = 3 \times 10^9$ and $H = 300$.

$$F = (H - L + 1) \times 3 \times 10^9 \times (1/4)^L$$

Observe that even if the match ratio is 90%, our choice of assigning 18 to L is able to locate a head of length 300 in the genome with a probability of 99.8%. Furthermore, the value of F is sufficiently low and false positive predictions of head locations are effectively reduced.

Table 1. The specificity and sensitivity of L-mer perfect matching. The table displays the probability that a head of length $H = 300$ and its homologous region share at least one L-mer when the match ratio is M_N. The bottom row presents the number F of L-mers in the head that are expected to match a genomic sequence of length $G = 3 \times 10^9$ by chance, given that the nucleotides are equally distributed.

	L					
M_N	18	19	20	21	22	23
100%	1.000	1.000	1.000	1.000	1.000	1.000
99%	1.000	1.000	1.000	1.000	1.000	1.000
98%	1.000	1.000	1.000	1.000	1.000	1.000
97%	1.000	1.000	1.000	1.000	1.000	1.000
96%	1.000	1.000	1.000	1.000	1.000	1.000
95%	1.000	1.000	1.000	1.000	0.999	0.999
94%	0.999	0.999	0.999	0.999	0.999	0.999
93%	0.999	0.999	0.999	0.999	0.999	0.998
92%	0.999	0.999	0.999	0.998	0.996	0.993
91%	0.999	0.998	0.997	0.994	0.990	0.983
90%	0.998	0.996	0.992	0.986	0.976	0.962
89%	0.996	0.991	0.983	0.970	0.953	0.929
88%	0.990	0.980	0.965	0.944	0.916	0.882
87%	0.979	0.963	0.938	0.906	0.866	0.820
86%	0.962	0.935	0.899	0.855	0.804	0.748
85%	0.936	0.897	0.849	0.793	0.732	0.667
F	12.3	3.08	0.767	0.191	0.0475	0.0119

3.2. *Search for internal exons*

When we search for the next internal exon, we only need to consider candidates in positions between the start- and endpoint of an EST, which is called the *coding region* of the EST in what follows. The length of most coding regions is bounded by one million, and is typically no more than one hundred thousand.

While we use long L-mers, say 18-mers, to approximate the start- and endpoints of an EST in a huge genomic sequence of length three billion, we should consider shorter K-mers when searching for internal exons inside coding regions, in order to get better sensitivity and specificity of alignment and to improve the overall performance. We now study appropriate values for K.

Suppose that $P(h, m, l)$ is defined as before. The probability that a subsequence of length H and its homologous region have in common at least one K-mer when the match ratio M_N is:

$$\sum_{m=0}^{H} \sum_{l=K}^{H} P(H, m, l) M_N^{H-m} (1 - M_N)^m.$$

We here set H to 100, which is slightly smaller than the average length of an exon, 150.

Table 2 shows the probabilities for various values of M_N and K. We then calculate the frequency of K-mers in a subsequence of length H that are expected to match a coding region of length G by chance, under the assumption that all nucleotides are distributed equally in the coding region. Let F_G denote the frequency. Then, observe that

$$F_G = (H - L + 1) \times G \times (1/4)^L.$$

We here assign 10^6 or 10^5 to G.

From Table 2, we see that when $K = 12$, the detection of the next exon succeeds with probability $\geq$ 0.99, even if the match ratio is 89%. Furthermore, the frequencies of both F_{10^5} and F_{10^6} are sufficiently low as to achieve high specificity.

Table 2. The specificity and sensitivity of perfect matches for a K-mer when $H = 100$ and $G = 10^5$ or 10^6.

	K					
M_N	10	11	12	13	14	15
100%	1.000	1.000	1.000	1.000	1.000	1.000
99%	1.000	1.000	1.000	1.000	1.000	1.000
98%	1.000	1.000	1.000	1.000	1.000	1.000
97%	1.000	1.000	1.000	1.000	1.000	1.000
96%	1.000	1.000	1.000	1.000	1.000	1.000
95%	1.000	1.000	1.000	1.000	1.000	0.999
94%	1.000	1.000	1.000	1.000	0.999	0.998
93%	1.000	1.000	1.000	0.999	0.997	0.994
92%	1.000	1.000	0.999	0.998	0.994	0.988
91%	1.000	0.999	0.998	0.995	0.988	0.977
90%	1.000	0.999	0.996	0.989	0.978	0.961
89%	0.999	0.997	0.992	0.981	0.964	0.938
88%	0.998	0.994	0.986	0.969	0.944	0.909
87%	0.997	0.99	0.976	0.952	0.918	0.873
86%	0.994	0.984	0.963	0.931	0.886	0.831
85%	0.991	0.974	0.946	0.903	0.848	0.783
F_{10^5}	8.68	2.15	0.53	0.13	0.03	0.01
F_{10^6}	86.78	21.46	5.30	1.31	0.32	0.08

In Table 2, assuming that the length H of an internal exon is 100, 12 is derived as an appropriate value for K. One might however be interested in what happens if H becomes smaller. Table 3 shows the case when $H = 50$. Note that when $K = 10$, an internal exon of length 50 can be found with probability $\geq$ 0.99, even if the match ratio is 90%, but the frequency F_{10^5} is fairly large. Selecting 12 for K reduces the frequency F_{10^5}, but requires a higher match ratio, 93%, to achieve a probability exceeding 0.99.

Table 3. The specificity and sensitivity of perfect matches for a K-mer when $H = 50$ and $G = 10^5$ or 10^6.

	K					
M_N	8	9	10	11	12	13
100%	1.000	1.000	1.000	1.000	1.000	1.000
99%	1.000	1.000	1.000	1.000	1.000	1.000
98%	1.000	1.000	1.000	1.000	1.000	1.000
97%	1.000	1.000	1.000	1.000	1.000	1.000
96%	1.000	1.000	1.000	1.000	0.999	0.998
95%	1.000	1.000	1.000	0.999	0.998	0.995
94%	1.000	1.000	0.999	0.998	0.995	0.989
93%	1.000	1.000	0.999	0.996	0.990	0.980
92%	1.000	0.999	0.997	0.992	0.982	0.966
91%	1.000	0.998	0.994	0.986	0.970	0.947
90%	0.999	0.997	0.990	0.977	0.954	0.923
89%	0.999	0.994	0.984	0.964	0.934	0.893
88%	0.997	0.991	0.975	0.948	0.909	0.859
87%	0.996	0.985	0.963	0.928	0.879	0.819
86%	0.993	0.978	0.949	0.904	0.845	0.776
85%	0.989	0.968	0.930	0.876	0.807	0.730
F_{10^5}	80.8	19.8	4.86	1.192	0.292	0.072
F_{10^6}	808.7	198.3	48.63	11.921	2.921	0.715

4. Execution Time, Sensitivity, and Accuracy

The algorithm described here has been implemented in C++. In what follows, we call our software *Squall*. We installed and evaluated the performance of Squall, sim4, and BLAT on the single processor of a PrimePower 1000 with a clock rate of 675 MHz, 64 Gbytes of main memory, and running Solaris 8.

4.1. *Comparison of the execution time and memory requirements of sim4 and BLAT*

The performance of Squall was compared with sim4 and BLAT using chromosome 22 of the NCBI draft human genome (Build 30). We aligned the RefSeq sequences [14] (16,133 sequences extracted from the UniGene database of Build #154, with an average length of 2697 bases) with the human chromosome 22. Table 4 shows the average time in seconds required to align an EST, which makes it clear that our method improves the computation time by orders of magnitude, although Squall requires more memory than BLAT. We think that our major improvement is avoiding the execution of dynamic programming as much as possible. We assume that all exons of a gene are found in the positional approximation using MapTable, and this prevents us from computing dynamic programming for overall sequences when we compare an EST with a genomic region containing an intron. Both

BLAT and sim4 initially feed genomic sequences, and then align ESTs. In Table 4, we excluded execution times for the data-reading step, so that this comparison was fair to both sim4 and BLAT, because sim4 and BLAT require much more execution time to feed the genome than Squall did.

Table 4. Comparison of the performance of Squall, BLAT, and sim4. Sequences in the HMR195 dataset were aligned with human Chr. 22 using these three programs.

	Average time to align an EST (sec)	Memory Requirement (MB)
Squall (our software, $K = 12$,$L = 18$)	0.012	760
BLAT(tileSize=11,repMatch=100)	0.263	130
sim4	14.732	700

4.2. *Execution time for aligning millions of ESTs with a genome*

Using Squall, we evaluated the average time required to check whether each EST in the UniGene (Build #154) database was aligned with the NCBI draft human genome (Build 30) and calculated the alignment when the EST mapped to the genome. Table 5 lists the average time in seconds. Observe that the average time required to process one chromosome is roughly proportional to the size of the chromosome. The average time was 0.0568 seconds. This implies that three million ESTs can be aligned in about 150,000 seconds on a single processor, and we can typically complete this task in less than half a day by using several processors.

4.3. *Sensitivity and accuracy*

To validate the sensitivity and accuracy of Squall, we considered clean datasets of splice sites [15,16,17,18,19]. These datasets have been collected from various species to derive statistically confirmed rules for improving gene-finding algorithms. Of these, we used HMR195 [19], a collection of 195 mammalian genomic sequences that are annotated with the exact locations of splice sites, which is available at `http://www.cs.ubc.ca/~rogic/evaluation/`. HMR195 was useful for our experiment, because it includes 103 human genomic sequences, from which we extracted 103 mRNA sequences by consulting the locations of splice sites. We then aligned these 103 sequences to the NCBI draft sequence of the human genome (Build 30) using Squall and BLAT [13]. We did not perform the same task with sim4 or Spidey because they cannot process very long genomic sequences, such as human chromosome 1. Although we used human chromosome 22, they were not able to compute the appropriate positions of two cDNAs (AF058293, AB002059) in the HMR195 dataset (See Table 6). Table 6 presents the quality of each alignment using Squall and BLAT (as of December 1st, 2002). '$\circ$' indicates that an exactly correct alignment was computed. '$\cdot$' means that the positions of one

Table 5. Size of chromosome and average execution time.

Chr. Number	Number of bases in chromosome	Average time to align an EST (sec)
1	246,874,334	0.0044
2	240,681,600	0.0043
3	194,908,136	0.0042
4	192,019,378	0.0025
5	180,966,400	0.0031
6	170,309,517	0.0034
7	157,432,793	0.0035
8	143,874,322	0.0017
9	132,438,756	0.0017
10	134,416,750	0.0023
11	137,442,545	0.0021
12	131,300,572	0.0026
13	113,446,104	0.0010
14	104,324,908	0.0013
15	99,217,355	0.0011
16	81,671,585	0.0015
17	80,052,782	0.0019
18	77,516,809	0.0008
19	60,013,307	0.0020
20	62,842,998	0.0009
21	44,626,493	0.0004
22	47,748,585	0.0006
X	149,249,818	0.0029
Y	58,368,225	0.0002

or two exons are incorrect, while '$\times$' indicates that more than two exons are located incorrectly. '-' indicates that no alignments were calculated for the mRNA. Table 7 summarizes the numbers of alignments of each quality category. Observe that Squall is much superior to BLAT in both sensitivity and accuracy.

4.4. *Alignment of NCBI Reference Sequences*

Known genes other than the HMR195 dataset are available in the RefSeq database [14], a manually curated collection designed to contain non-redundant representatives of most of the full-length human mRNA sequences in GenBank.

We aligned a current version of the RefSeq sequences (16,133 sequences) with the NCBI working draft sequence of the human genome (Build 30). As a result, more than 94.8% of the RefSeq entries were aligned given the conditions that: (i) The match ratio

Table 6. Quality of aligning 103 human mRNAs in HMR195 to the NCBI human draft genome (Build 30) by using Squall(K=12,L=18) and BLAT (as of December 1st, 2002, the options: tileSize=11,repMatch=100). '○' indicates the exacly correct alignment is computed for the mRNA. '·' means that positions of one or two exons are incorrect, while '×' indicates that more than two exons are located incorrectly . '-' implies that no alignments are calculated for the mRNA. The last column shows the chromosome number in which each mRNA is located.

Acc numer	Squall	BLAT	Chr.
AB016625	○	×	5
AF008216	○	○	4
AF092047	○	○	2
AF096303	·	·	11
AF019563	○	·	19
AB012922	○	×	11
U25134	×	×	16
U17081	○	·	1
AB021866	·	·	15
AF039704	×	×	11
AF082802	○	×	19
AF039954	○	·	17
AB018249	○	·	17
AF099731	○	○	1
AF099730	○	○	1
AF039401	·	×	1
AF084941	○	○	6
AF059675	○	×	6
AF007189	○	○	7
AF016898	○	○	14
AF076214	○	·	5
AB012113	○	·	17
AB019534	○	×	9
AF080237	○	×	16
AF071596	○	○	6
U43842	○	·	14
AF053455	○	·	4
AF071216	○	○	8
AF058761	○	○	19
Y16791	·	×	17
AB016492	○	×	1
AF001689	○	○	17
AF029081	○	○	1
U96846	○	·	12
AF027152	○	×	12
U55058	○	·	1
AF068624	○	×	X
AF053069	○	·	11
AF032437	○	·	12
AF007876	○	×	17
AF051160	○	·	6
AF022382	○	×	1
AF045999	○	×	2
U53447	·	○	4
AF009356	○	×	1
AF019409	○	×	11
AF015224	○	·	11
AF042782	○	○	17
AF037207	○	·	10
AB016243	○	·	16
AF049259	○	×	17

Acc numer	Squall	BLAT	Chr.
AB012668	-	-	?
AF052572	-	-	?
AF042001	○	·	8
AF055475	○	·	X
AF055903	○	×	11
AF053630	○	×	6
AF027148	○	·	11
AB007828	○	○	15
AF044311	○	·	10
AF061327	○	○	19
AF059650	○	×	5
AB010874	-	-	?
AF071552	○	○	8
AF059734	○	·	3
AF009962	○	○	3
AB013139	○	×	8
AF013711	○	·	11
AF065396	○	×	6
AF058762	○	○	17
AF043105	○	×	1
AF065988	○	○	12
AF026564	·	·	Y
AF037438	○	○	X
AF028233	○	·	17
AB007546	○	·	5
AF058293	○	·	22
AF055080	○	○	15
AF037062	○	·	12
AB009589	○	·	9
AF047383	×	×	1
U31468	○	○	2
AF036329	○	·	20
AF042084	○	×	10
AF040714	○	○	7
AF039307	○	○	7
AF031237	○	○	3
AF005058	-	-	?
AF037372	○	○	1
D89060	○	×	1
AF032455	○	×	7
AB003730	○	○	12
AB006987	○	×	12
AF015812	○	×	17
AF015954	·	×	11
D67013	·	×	3
D38752	·	○	10
D83956	○	·	6
AF016052	○	·	18
AB002059	○	·	22
AJ223321	○	○	1
U76254	○	○	4
AF017115	○	×	16

Table 7. Statistics of alignment quality in Table 6. For instance, the number of exactly correct answers by Squall is 83 among 103 genes in HMR195.

Quality	Squall	BLAT
○	87	31
·	9	34
×	3	34
—	4	4

was at least 70%, and (ii) The coverage ratio (the length of the aligned part / the entire length) was at least 50%. This shows that the sensitivity of our software Squall is high. In addition, we confirmed that Squall accurately aligned each RefSeq sequence with multiple exons and its boundaries, by checking 300 randomly selected alignments.

5. Graphical Viewer

Here, we present a graphical viewer for browsing the intron and exon structures resolved in our implementation. The viewer is available at

`http://grl.gi.k.u-tokyo.ac.jp/`.

This browser is called the *Gene Resource Locator viewer* (*GRL viewer*, for short) [20] and it shows the alignment of each EST in the genomic sequence with a user-friendly interface. Figure 13, for instance, shows a group of ESTs mapped onto the same locus. Each thick line represents the alignment of one EST (an EST alignment) in which the narrow yellow boxes are exons and the blue boxes are introns. Note that the alignments share some common exons. Some alternatively splicing transcripts are also observed. The details of biological results discovered by our software are presented in another article [20].

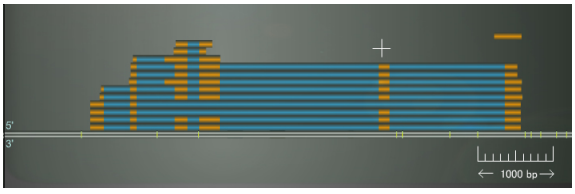

Fig. 13. GRL viewer

6. Conclusion and Future Work

Our software Squall shows excellent speed and high precision, as described in Section 4. Gene structures can be resolved faster and more precisely using this algorithm than with

other methods. In the field of medicine, there are many instances in which a disease-related gene is known. The precise localization of such genes will be possible in future by referring to the complete genetic map. Moreover, the generation of precise genetic maps for many creatures is a prerequisite for inter- and intra-species genome comparisons. Our algorithm can be used to generate genetic maps that combine the genomes and genes of various species.

Acknowledgments

We thank Masahiro Kasahara, Toshihiko Honkura, and Tomoyuki Yamada for stimulating discussion. This research was supported by grant 12208003, a Grant-in-Aid for Scientific Research on Priority Areas, from the Ministry of Education, Science, and Culture, Japan.

References

1. J. Craig Venter *et al*. The sequence of the human genome *Science*, 291:1304-1351 (2001).
2. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome *Nature*, 409:860-921 (2001).
3. S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443-453 (1970).
4. T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195-197 (1981).
5. W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444-2448 (1988).
6. S.F. Altschul, W. Gis, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. M. Biol.*, 215:403-410 (1990).
7. O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705-708 (1982).
8. Daniel S. Hirschberg. A Linear Space Algorithm for Computing Maximal Common Subsequences. *CACM* 18(6): 341-343 (1975).
9. M.S. Gelfand, A.A. Mironov, and P.A. Pevzner. Spliced alignment: A new approach to gene recognition. *Proc. Natl. Acad. Sci. USA* 93:9061-9066 (1996).
10. E. Birney and R. Durbin. Dynamite: a flexible code generating language for dynamic programming methods used in sequence comparison. *Proc. Fifth Int. Conf. Intelligent Systems Mol. Biol.* 5:55-64 (1997).
11. R. Mott. EST_GENOME: A program to align spliced DNA sequences to unspliced genomic DNA. *Comput. Appl. Biosci.* 13:477-478 (1997).
12. L. Florea, G. Hartzell, Z. Zhang, G.M. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic sequence. *Genome Research*, 8(9):967-974 (1998).
13. W. James Kent. BLAT - The BLAST-Like Alignment Tool. *Genome Res.*, 12: 656-664 (2002).
14. RefSeq and LocusLink: NCBI gene-centered resources. Pruitt KD, Maglott DR *Nucl. Acids Res.*, 29(1):137-140 (2001).
15. M. Burset and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 34:353-357 (1996).
16. T.A. Thanaraj. A clean data set of EST-confirmed splice sites from Homo sapiens and standards for clean-up procedures. *Nucl. Acids Res.* 27: 2627-2637 (1999).
17. F. Clark and T.A. Thanaraj. Categorization and characterization of transcript-confirmed constitutively and alternatively spliced introns and exons from human. *Hum. Mol. Genet.* 11: 451-464 (2002).

18. M. Burset, I. A. Seledtsov, and V. V. Solovyev. SpliceDB: database of canonical and non-canonical mammalian splice sites. *Nucl. Acids Res.* 29: 255-259 (2001).
19. S. Rogic, A. Mackworth and F. Ouellette. Evaluation of gene finding programs. *Genome Research*, 11: 817-832 (2001).
20. T. Honkura, J. Ogasawara, T. Yamada, and S. Morishita. The Gene Resource Locator: gene locus maps for transcriptome analysis. *Nucl. Acids. Res.*, 30(1):221-225 (2002).
21. M. Kasahara. Non-consecutive sequence indexes for cross-species alignment. *Proceedings of Genome Sequencing & Biology 2002 at CSHL*,

Appendix

Use of amino acid L-mers to approximate EST location

In Section 3, we presented the statistical analysis for selecting the appropriate values for K and L. The amino acid sequences of proteins, however, are likely preserved in coding regions across species, and using amino acid L-mers in place of nucleotide L-mers as positional indexes might improve the sensitivity of alignment. We here examine the validity of this approach.

Consider a nucleotide sequence of length $3H$ and its corresponding amino acid sequence of length H. Let M_N (or the equivalent M_A) denote the match ratio between the nucleotide (amino acid) sequence and its homologous region. Here, we calculate M_A from M_N. Let $c1$ and $c2$ be codons (sequences of three nucleotides), and let $\delta(c1, c2)$ denote the number of distinct nucleotides at the same positions. For instance, $\delta(\mathrm{atg}, \mathrm{tta}) = 2$, and $\delta(\mathrm{atg}, \mathrm{agg}) = 1$. Define $(c1 =_A c2) = 1$ if both $c1$ and $c2$ encode the same amino acid. Otherwise, $(c1 =_A c2) = 0$. Assume that any codon occurs equally and any nucleotide in a codon can be altered with probability $(1 - M_N)/3$. Then, M_A is computed from M_N according to the following formula, and Table 8 shows the relation for typical values of M_N and M_A.

$$M_A = \sum_{c1,c2:\mathrm{codons}} (c1 =_A c2) M_N^{3-\delta(c1,c2)} \left(\frac{1 - M_N}{3}\right)^{\delta(c1,c2)} \frac{1}{64}$$

Suppose that $P(h, m, l)$ has the same meaning as before, except that nucleotides are replaced with amino acids. Then, the probability that the amino acid sequence and its homologous region share at least one sequence of consecutive L amino acids is:

$$\sum_{m=0}^{H} \sum_{l=L}^{H} P(H, m, l) M_A^{H-m} (1 - M_A)^m.$$

Let F_A denote the frequency of amino acid L-mers in a sequence of length H that are expected to match the translated genomic sequence, when all amino acids, including the stop symbol, equally occur, then,

$$F_A = (H/3 - L + 1) \times 3 \times 10^9 \times (1/18)^L.$$

Table 9 lists the probabilities and frequencies for various values of L and M_N when the number of amino acids H is 100. Recall that Table 1 shows the case when the number of nucleotides is 300, and we selected 18 as an appropriate value for L, which corresponds to $L = 18/3 = 6$ in Table 9. Observe that the sensitivity of using amino acid 6-mers is better than that with nucleotide 18-mers, because the probability is greater than 0.99 even if $M_N = 86\%$. However, the frequency of amino acid 6-mers is much greater than that of nucleotide 18-mers, because of the lower specificity of amino acids versus nucleotides.

Our analysis assumes that the nucleotide mutations occur equally at random, but this may not be the case in practice. In reality, mutations tend to be observed at the third nucleotides of codons. Then, the values of M_A in Table 8 could be slightly greater, thereby improving the sensitivity of amino acid L-mers in Table 9.

Table 8. Relationship between the match ratio for nucleotides and amino acids.

M_N	M_A
100%	100.000%
99%	97.735%
98%	95.502%
97%	93.301%
96%	91.131%
95%	88.993%
94%	86.887%
93%	84.811%
92%	82.767%
91%	80.753%
90%	78.769%
89%	76.816%
88%	74.894%
87%	73.001%
86%	71.138%
85%	69.304%
84%	67.500%
83%	65.725%
82%	63.978%
81%	62.261%
80%	60.572%

Table 9. Specificity and sensitivity of perfect matches foramino acid L-mer when $H = 100$ and $G = 3 \times 10^9$.

	L					
M_N	5	6	7	8	9	10
100%	1.000	1.000	1.000	1.000	1.000	1.000
99%	1.000	1.000	1.000	1.000	1.000	1.000
98%	1.000	1.000	1.000	1.000	1.000	1.000
97%	1.000	1.000	1.000	1.000	1.000	1.000
96%	1.000	1.000	1.000	1.000	1.000	1.000
95%	1.000	1.000	1.000	1.000	1.000	0.999
94%	1.000	1.000	1.000	1.000	0.999	0.997
93%	1.000	1.000	1.000	1.000	0.997	0.990
92%	1.000	1.000	1.000	0.998	0.992	0.974
91%	1.000	1.000	0.999	0.995	0.980	0.948
90%	1.000	1.000	0.998	0.988	0.960	0.907
89%	1.000	1.000	0.995	0.976	0.930	0.853
88%	1.000	0.999	0.989	0.956	0.887	0.786
87%	1.000	0.997	0.979	0.926	0.833	0.711
86%	1.000	0.994	0.963	0.887	0.769	0.631
85%	0.999	0.988	0.939	0.838	0.698	0.549
F_A	70517	3322	156.5	7.376	0.347	0.016