Fast and Sensitive Algorithm for Aligning ESTs to Human Genome

Jun Ogasawara and Shinichi Morishita
Department of Computer Science, University of Tokyo
$\{$*jun, moris*$\}$*@gi.k.u-tokyo.ac.jp*

Abstract

There is a pressing need to align growing set of expressed sequence tags (ESTs) to newly sequenced human genome. The problem is, however, complicated by the exon/intron structure of eucaryotic genes, misread nucleotides in ESTs, and millions of repeptive sequences in genomic sequences. Indeed, to solve this, algorithms that use dynamic programming have been proposed, but in reality, these algorithms require an enormous amount of processing time. In an effort to improve the computational efficiency of these classical DP algorithms, we develop software that fully utilizes the lookup-table for allowing the efficient detection of the start- and endpoints of an EST within a given DNA sequence, and subsequently, the prompt identification of exons and introns. In addition, high sensitivity and accuracy must be achieved by calculating locations of all spliced sites correctly for more ESTs while retaining high computational efficiency. This goal is hard to accomplish in practice, owing to misread nucleotides in ESTs and repeptive sequences in the genome, but we present a couple of heuristics effective in settling this issue. Experimental results have confirmed that our technique improves the overall computation time by orders of magnitude compared with common tools such as sim4 and BLAT, and attains high sensitivity and accuracy against datasets of clean and documented genes at the same time.

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 second step involves the interpretation of the encoded sequences. For the purpose of identifying the coding regions, i.e., regions containing exons and introns, of any given DNA sequence, the alignment of many ESTs to genomic DNA is helpful to reveal these complex structures while verifying the alternative spliced transcripts. The alignment of full-length cDNAs gives a clue to some regulatory elements in its upstream regions, and further the annotations of the upstream regions using Transfac data line up candidates of cis-elements. The alignment both of sequences associated with expression pattern and of sequences from dbSNP with identifying locations of SNPs around the gene make it possible to step forward more in function 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]). Recently, heuristic algorithms such as FASTA [5] and BLAST [6] have been used because of their higher speed compared to dynamic programming methods.

These algorithms however require a very large amount of time for processing or they fail to align ESTs to genome that are known to be encoded, because they are designed only to solve the similarity between two sequences, but not to decide eucaryotic gene structure (exon/intron structure) through the identification of spliced sites between exons and introns. Figure 1 illustrates our problem that 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 the EST. To settle this problem, it is reasonable to define scores of matching and penalty for introducing introns and then to select the optimal decomposition as that with the best score.

For this optimization problem, many dynamic programming algorithms that consider exon/intron structure have been developed. Gotoh's algorithm [7] defines the 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 $O(\min(M, N))$ by Hirschberg's technique [8]. There have been developed several software tools that utilize the idea of this dynamic programming technique [9, 10, 11], but in practice, these tools are computationally infeasible to apply to long genomic sequences of length greater than one million. One the other hand, Sim4 [12] and BLAT [13] are also improved methods based on BLAST, which extend not only single exon but also multiple exons. Although they are able to decompose a given sequence into its exons, they are not designed to compute the optimal alignment.

Therefore, since high performance software is needed to solve this problem, we invented software that shortens the calculation time, while retaining sensitivity and accuracy. This software is able to align more than 20 ESTs per second on average to a human draft genome by using a single processor, but in practice it can process more than 100 ESTs per second on several processors, and hence about three millions of ESTs in less than half a day. With regarding sensitivity and accuracy, special care has to be taken

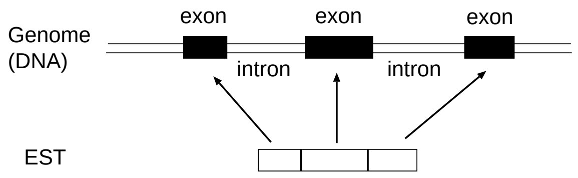

Figure 1: Decomposing and mapping an EST.

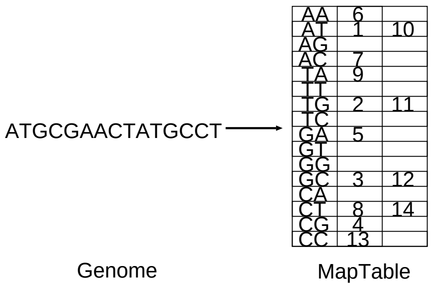

Figure 2: How to generate a lookup table (MapTable).

to identify spliced sites in the final step of the software. Clean datasets of spliced 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 to evaluate sensitivity and accuracy of alignment software, and we use the HMR195 dataset [19], a collection of Mammalian sequences. We will demonstrate high sensitivity and accuracy of our method against human genes in HMR195.

2. Algorithm

We start with the introduction of basic data-structures and simpler algorithms, and move to elaborated one by improving algorithms step by step.

Mapping of the millions of ESTs in the GenBank and EST databases (dbEST) to the human genomic sequence is an arduous task. ESTs are a maximum of tens of thousands of bases long, while genomic sequences are about 3 billion bases long. In order to shorten the overall calculation time, we pre-process the genomic sequences. A DNA sequence of length K is defined as a *primary key*. Our idea is to create an auxiliary look-up table called *MapTable* that stores the position at which each primary key occurs in the genomic sequence. 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 suffix of the gene is 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 from the MapTable by accessing the main memory about $M/4^K$ times (M is the length of the genome sequence). Care had to be taken in selecting appropriate values for K. Smaller K values, say 5, aligned an enormous number of positions for each 5-mer, while a larger K value increased the number of K-mers, yielding a huge index. We will mention this issue of selecting an appropriate value for K.

2.1. Simple algorithm assuming no mismatches or gaps

First we present a slow but simple algorithm which assumes that an EST sequence exactly maps to the genomic sequence with 100% matching 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 by 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. We here assume that each nucleotide e_i maps to a location in G, but in general e_i might be skipped, requiring its position $f(i)$ 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 a key to detect first exon and last exon. 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. Align the prefix and suffix with the genome by accessing the MapTable, i.e., associate $f(i)$ for each $i = 1, ..., L, N-L+1, ..., N$.

Step 2.(Alignment by dynamic programming)
Align 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 the case of human genome, we assign 14 to K and 21 to L, because most first exons and last exons are of length more than 21, though it is known that some internal exons are of length less than 21. If the location of coding region of the exon is figured out, we don't need to have such long key of length L, which encourages us to select 14 for K. 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 the algorithm that finds the optimal alignment of EST that has long introns with genome. Smith-Waterman and Needleman-Wunsch do not work well for this problem, because they pose high penalty on long introns and miss alignments with long introns. To overcome this issue, Gotoh's method allows the assignment of a small constant or an affine gap penalty to introns that could be very long, and it can output the optimal solution.

Although the detection of the start- and endpoints of a given EST in the genome sequence in Step 1 works very efficiently because it can be achieved by accessing the main memory, Step 2 could sometimes require 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.

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 intron using MapTable. This greatly increases performance because of its practicability in considering exon/intron structure. The algorithm for aligning an EST with a single exon is shown.

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

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

To handle ESTs with multiple exons, we incorporate the 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.

2.3. Allowing mismatches and gaps in alignment

In practice, any EST cannot be fully aligned with 100 % identity, resulting in mismatches or gaps in the alignment. To allow for these mismatches and gaps, we here revise the algorithm in Section 2.3. We allow that e_i is mismatched with a different nucleotide at the $f(i)$-th position on the genome. Or, $f(i)$ 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 $E_{[N-L+1,N]}$) might contain 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 is equal to 300 or the position of $E_{[i,i+L-1]}$ is found in the MapTable. The rationale behind the choice of 300 bases will be presented in the appendix. After this, $E_{[1,i-1]}$ is called the *dangling part* of the EST that still remains to be aligned. Then 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)
In a similar way, initialize $j=N$, and decrement j until j is equal to 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 also make it more complicated to extend an exon and 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 is the smallest position in the EST that is not aligned (for instance, (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. 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. After this, $E_{[x+1,i-1]}$ 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.

2.4. Further acceleration by preprocessing long introns

In practice, an intron could 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 EST $E_{[x+1,i-1]}$ with $G_{[f(x)+1,f(i)-1]}$ is computation intense 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 $E_{[x+1,i-1]}$, we assume that $G_{[f(x)+1,f(i)-1]}$ contains an intron in it. In this case, since $E_{[x+1,i-1]}$ should be aligned to $G_{[f(x)+1,f(x)+(i-x)]}$ and $G_{[f(i)-(i-x),f(i)-1]}$, 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 spliced sites

We have presented how to determine the approximate exon and intron regions. However, decisions regarding exon/intron bound-

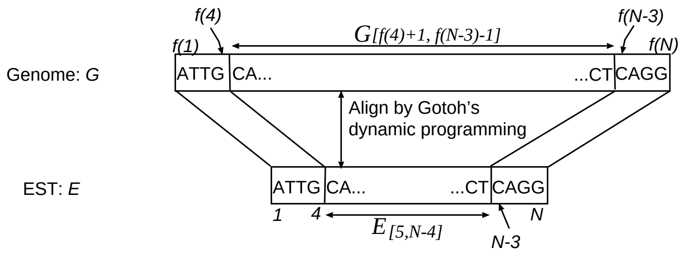

Figure 3: Simplest algorithm with a MapTable($L = 4$).

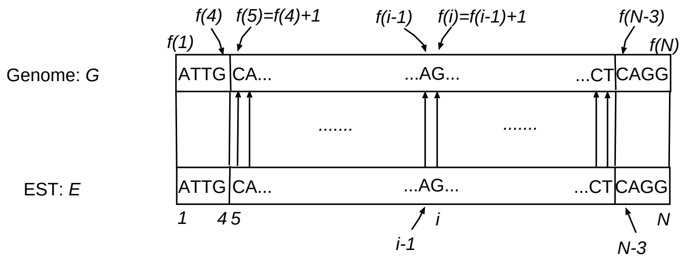

Figure 4: Fast algorithm with a MapTable (Case of a single exon, $L = 4$).

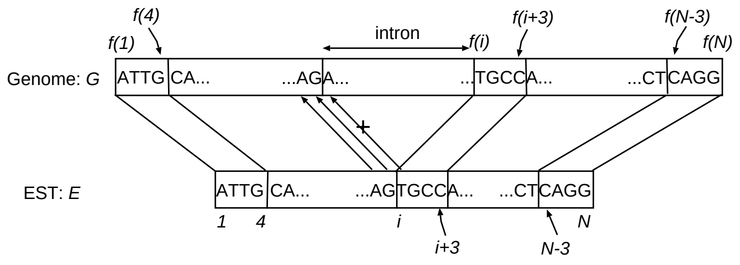

Figure 5: Fast algorithm with MapTable (Case of multiple exons, $L = 4, K = 4$).

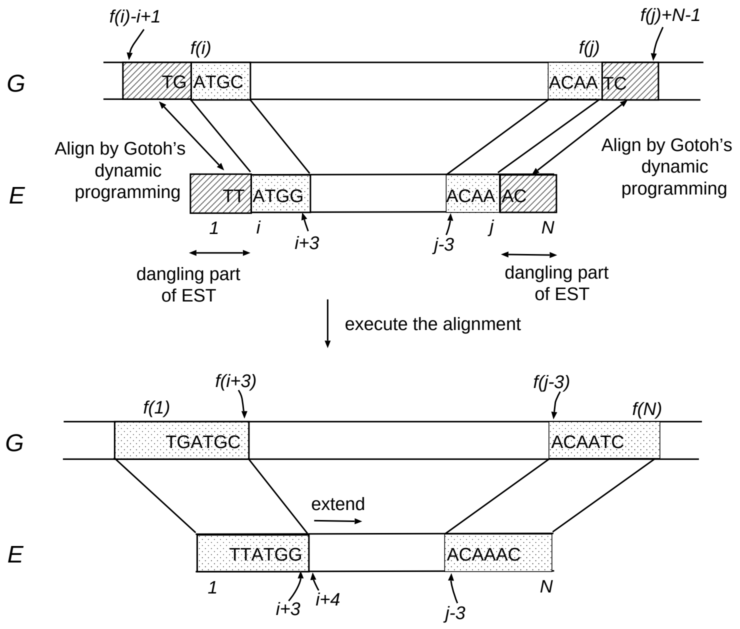

Figure 6: Approximation of start- and endpoints ($L = 4$).

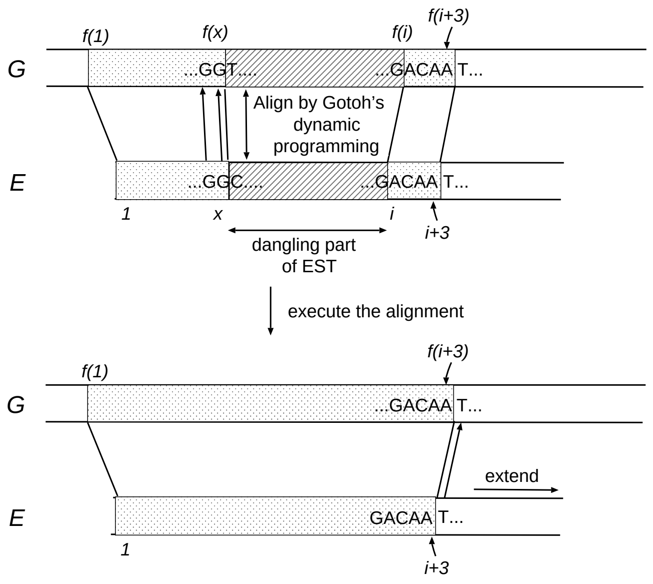

Figure 7: Alignment of the dangling part of an EST($L = 4, K = 4$).

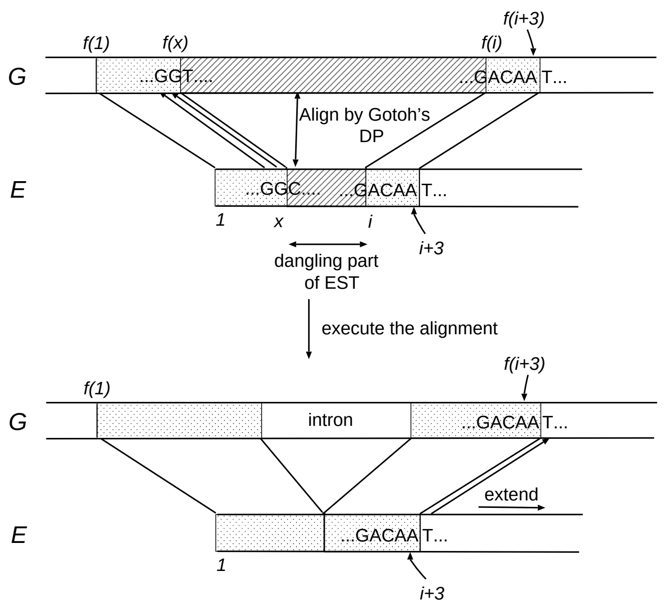

Figure 8: Alignment of the dangling part of an EST ($L = 4, K = 4$) when part of the genome contains an intron.

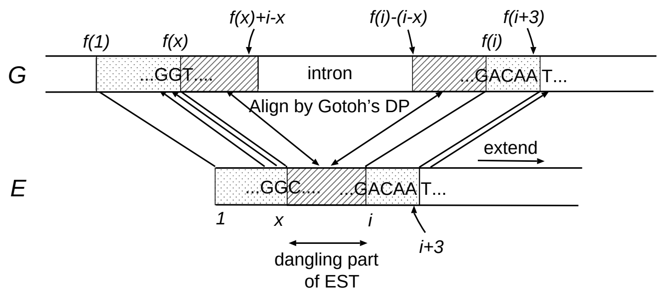

Figure 9: Acceleration of dynamic programming.

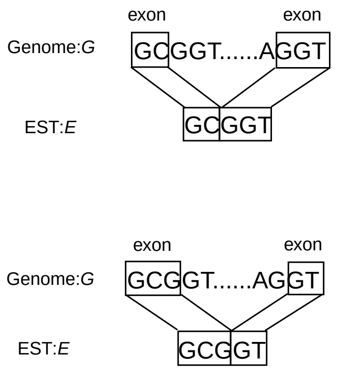

Figure 10: Alternative solutions for spliced sites.

aries need rigorous investigation, because most boundaries follow the GT-AG rule, but some other patterns such as GC-AG and AT-AC are also observed. In the literature, considerable efforts have 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] derives decision trees for inferring human exon-intron junctions from a number of EST-confirmed splice sites. Burset et al.[18] also present statistical rules for mammalian spliced sites, which also confirms that most sites obey GT-AG rule or its variants such as GC-AG and AT-AC.

Figure 10 illustrates some alternative solutions in deciding spliced sites, and the lower alignment ought to 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 the GT-AG or its slight variants. In more precise, the intron frame is moved locally so that the number of matches between GT-AG and the two plus two letters at the boundary of the intron is maximized. We will demonstrate the high sensitivity and accuracy of this method later.

2.6. Matching ratio

Having determined the intron regions, the matching ratio between the genome and the mapped EST sequence is examined. Since the matching 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), we assume that an EST with a low matching ratio is not encoded. In effect, an EST whose matching ratio is low contains many misread nucleotides, or is not encoded in the genome sequence. Depending on our implementation result, the lower boundary for the matching ratio was

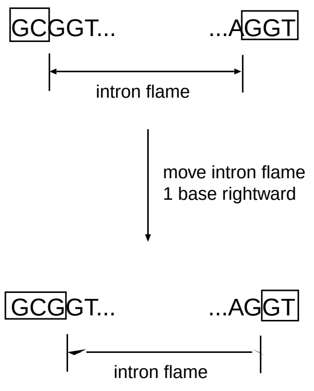

Figure 11: Shifting the intron frame to detect GT-AG or its slight variants.

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 to many different regions in a chromosome because of retro-transposition or gene-duplication (Figure 12). Furthermore, millions of repeptive sequences in human genome disturb correct identification of start-points and end-points.

Although the start- and endpoints of an EST are inferred in Step 1.1 and Step 1.2, the start- and endpoints is not determined uniquely if the MapTable has many candidates for the primary key. Let $StartSet$ denote the set of candidate startpoints calculated in Step 1.1, and $EndSet$ the set of candidate endpoints in Step 1.2. We then need to compute alignments for all the pairs in $StartSet \times EndSet$. In practice, however, the size of $StartSet$ or $EndSet$ could be often more than one thousand owing to repeptive sequences. To avoid such difficult cases, we focus on the fact that the number of 21-mers appearing no more than ten times in the human genome is about 2.2 billions, which is about 73% of all the 21-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 hundreds bases in a human EST could find, with a high probability, such a 21-mer of frequency no more than ten. We therefore revise our algorithm to search 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 such $StartSet$ 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 Step 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 the all alignments of a given EST reliably.

Table 1: Performance comparison between Squall, BLAT, and sim4. All of RefSeq sequences are aligned with human Chr. 22 by using these three software.

	Average time to align an EST (sec)
Squall (our software)	0.03
BLAT	1.69
sim4	12

3. 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 performance of our software Squall, sim4, and BLAT on a single processor of PrimePower 1000 with a clock rate of 675 MHz, 64 Gbytes of main memory, and running Solaris 8.

3.1. Execution time comparison with sim4 and BLAT

The performance of Squall was compared with sim4 and BLAT using the chromosome 22 of the NCBI draft human genome (Build 28) . We have aligned current version of RefSeq sequences [14] (14784 sequences, their average length was 2415 bases) to the human chromosome 22. Table 1 presents average time in seconds to align an EST, which makes it clear that our method has improved the computation time by orders of magnitude. Both BLAT and sim4 initially feed genomic sequences, and then perform alignments of ESTs. In Table 1, we excluded execution times of the former feeding step, so that this comparison was fair to both sim4 and BLAT, because sim4 and BLAT required much more execution time to feed the genome than our software Squall did.

3.2. Execution time to align millions of ESTs with genome

Using Squall, we evaluated average time of checking whether or not each EST in the UniGene database was aligned with the NCBI draft human genome (Build 28) and calculating the alignment when the EST mapped to the genome. Table 2 illustrates average time in seconds, and observe that average time to process one chromosome is almost proportional to the size of the chromosome. The total of average time is 0.0504 seconds. Thus, for instance, aligning three millions ESTs can be done in about 150,000 seconds on a single processor, but in practice, we can typically complete this task in less than half a day with using several processors.

3.3. Sensitivity and accuracy

To validate sensitivity and accuracy of Squall, we considered to use clean datasets of spliced sites [15, 16, 17, 18, 19]. These datasets have been collected from various species to derive statistically confirmed rules for improving gene finding algorithms. Among these, we used HMR195 [19], a collection of 195 Mammalian genomic sequences that are annotated

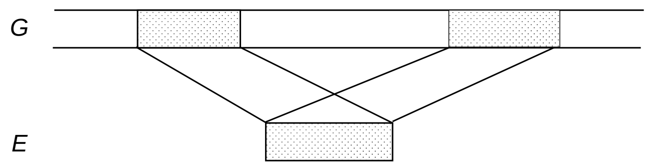

Figure 12: EST Alignment to plural regions in genome.

with exact locations of spliced sites, which are available at `http://www.cs.ubc.ca/˜rogic/evaluation/`. HMR195 is useful for our experiment, because it includes 103 human genomic sequences, from which we extracted 103 mRNA sequences by consulting the locations of spliced sites. We then attempted to align these 103 sequences to the NCBI draft sequence of human genome (Build 28) by using Squall and BLAT [13]. We abandoned the same task by executing sim4 and Spidey because the task was very time-consuming and impossible to complete.

Table 3 presents the quality of each alignment by Squall and BLAT. '○' indicates the exactly correct alignment is computed. '·' 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. Table 4 summarizes the numbers of alignments in each quality category. Observe that Squall is much superior to BLAT both in sensitivity and in accuracy.

3.4. Alignment of NCBI Reference Sequences

The known genes other than HMR195 datasets are available at the RefSeq database [14], a manually curated collection designed to contain nonredundant representative of most full-length human mRNA sequences in GenBank.

We have aligned current version of RefSeq sequences (14784 sequences) to NCBI working draft sequence of human genome (Build 28), and as a result, more than 94.5% of the RefSeq entries were aligned under the condition that: (i) the matching ratio was at least 70%; (ii) the coverage ratio (the length of aligned part / the entire length) was at least 50%;, showing that our software Squall have high sensitivity. Also, we confirm that Squall is able to obtain the accurate alignment of each RefSeq sequence with multiple exons and its boundaries, by checking 300 randomly-selected alignments.

4. Graphical Viewer

We here 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 *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, presents a group of ESTs mapped to 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 alignments share some common exons. Some alternatively spliced transcripts are also observed. The details of biological results discovered by our software can be found in [20].

5. Conclusion and Future Work

Our software Squall shows excellent speed and high precision, as described in Section 3. 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.

Acknowledgements

We are grateful to Toshihiko Honkura and Tomoyuki Yamada for stimulating discussions. This research is 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. *Journal of Molecular Biology*, 48:443-453 (1970).

[4] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195-197 (1981).

[5] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceeding of the National Academy of Sciences*, 85:2444-2448 (1988).

Table 3: Quality of aligning 103 human mRNAs in HMR195 to the NCBI human draft genome (Build 28) by using Squall and BLAT. '○' 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	-	-	?
AF039704	×	×	11
AF082802	○	×	19
AF039954	○	·	17
AB018249	○	·	17
AF099731	○	○	1
AF099730	○	○	1
AF039401	·	×	1
AF084941	○	○	6
AF059675	○	×	6
AF007189	○	○	7
AF016898	○	○	14
AF076214	-	-	?
AB012113	○	·	17
AB019534	○	×	9
AF080237	○	×	16
AF071596	○	○	6
U43842	○	·	14
AF053455	-	-	?
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	-	-	?
AB016243	○	·	16
AF049259	○	×	17
AB012668	-	-	9
AF052572	·	·	2
AF042001	○	·	8
AF055475	○	·	X
AF055903	○	×	11
AF053630	○	×	6
AF027148	○	·	11
AB007828	○	○	15
AF044311	-	-	?
AF061327	○	○	19
AF059650	○	×	5
AB010874	×	·	12
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	×	×	2
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

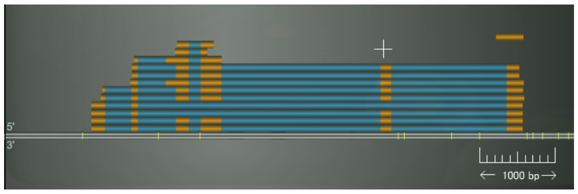

Figure 13: GRL viewer

Table 2: Size of chromosome and average execution time.

Chr. Number	Number of bases in chromosome	Average time to align an EST (sec)
1	256,422,226	0.0044
2	241,269,072	0.0043
3	204,853,362	0.0042
4	191,388,243	0.0025
5	184,841,686	0.0031
6	178,317,572	0.0034
7	163,810,217	0.0035
8	145,730,583	0.0017
9	133,027,393	0.0017
10	142,094,824	0.0023
11	141,434,656	0.0021
12	139,632,178	0.0026
13	115,090,884	0.0010
14	106,469,138	0.0013
15	99,058,357	0.0011
16	93,800,608	0.0015
17	83,863,704	0.0019
18	81,800,002	0.0008
19	76,926,326	0.0020
20	62,963,852	0.0009
21	44,620,380	0.0004
22	47,848,586	0.0006
X	151,672,894	0.0029
Y	58,368,226	0.0002

[6] S.F. Altschul, W. Gis, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403-410 (1990).

[7] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 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.* 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).

Table 4: Statistics of alignment quality in Table 3. For instance, the number of exactly correct answers by Squall is 83 among 103 genes in HMR195.

Quality	Squall	BLAT
$\circ$	83	30
$\cdot$	8	32
$\times$	6	35
$-$	7	6

[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 *Nucleic Acids Res 2001 Jan 1*, 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

In order to approximate the startpoint of an EST E in Step 1.1 of our algorithm, we scan the first 300 bases of the EST to look for an L-mer $E_{[i,i+L-1]}$ that perfectly matches with the genome. We have selected 21 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 with the head its homologous range in the genomic sequence by finding one L-mer in the head that perfectly matches with the genome. To accelerate the overall performance of this approach, we have to exclude false positive L-mers in the head that happen to match with 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 through the use of L-mers and may make our algorithm less sensitive, because of noises both in head and genomic sequences.

Suppose that the match ratio between the head and its homologous region is M. When M gets lower, say 85%, it becomes hard to find one L-mer shared by the head and its homologous region in the genome. Thus, depending on typical values of M, we have to decide an appropriate value of L in the consideration of the trade off between sensitivity and specificity. To resolve this, we here present a way of computing the probability that the head of length H and its homologous region share at least one common L-mer for various values of M. In the literature, Kent [13] studies 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] investigates the statistical property of overlapping L-mers, and we follow the line suggested by Kasahara.

Let $F(h, m, l)$ denote the number of head sequences of length h such that each head contains m mismatch nucleotides with its homologous region and the longest subsequence of length l that perfectly match its homologous region. Observe that the following recurrences hold for $F(h, m, l)$:

$$\begin{aligned} F(h,0,l) &= 1 \qquad (h = l) \\ F(h,0,l) &= 0 \qquad (h \neq l) \\ F(h,m,l) &= \sum_{i=0}^{l} F(h-l-1, m-1, i) \\ &+ \sum_{i=0}^{l-1} F(h-i-1, m-1, l) \end{aligned}$$

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

$$\sum_{m=0}^{H}\sum_{l=L}^{H} F(H,m,l) M^{H-m}(1-M)^m$$

In the above formula, we simply assume that the head is not split into more than one exons when the head is aligned with the genome. Table 5 displays the probabilities for various values of M and L. Furthermore, to measure the specificity of using each L-mer, the lowest row presents the number F of L-mers in the head that are expected to match by chance, under the condition that all nucleotides equally occur in the sequences. F is defined as the following formula, and 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 21 to L is able to locate the head of length 300 in the genome with a probability of 98.6%. Furthermore, the value of F is sufficiently low, and hence false positive predictions of head locations would be reduced effectively.

Table 5: Specificity and sensitivity of L-mer perfect matching

	L					
M	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