Efficient mapping of large cDNA/EST databases to genomes: A comparison of two different strategies

C. Wawra* M.I. Abouelhoda* E. Ohlebusch*

Abstract: This paper presents a comparison of two strategies for cDNA/EST mapping: The seed-and-extend strategy and the fragment-chaining strategy. We derive theoretical results on the statistics of fragments of type maximal exact match. Moreover, we present efficient fragment-chaining algorithms that are simpler than previous ones. In experiments, we compared our implementation of the fragment-chaining strategy with the seed-and-extend strategy implemented in the software tool BLAT.

1 Introduction

The first step in gene expression is transcription of the genetic information contained in DNA into RNA. In this process, the RNA polymerase generates a primary RNA transcript that extends from the initiation site to the termination site in a perfect complementary match to the DNA sequence used as a template. In eukaryotes, however, not all transcribed RNA is destined to arrive in the cytoplasm as mRNA. Rather, by an incompletely understood process, sequences complementary to introns are excised from the primary transcript, and the ends of exon sequences are joined together in a process termed "splicing." The exons are short segments ranging from tens to hundreds of base pairs, while the introns are normally several orders of magnitude longer. To make a cDNA library, one isolates all the mRNA from a cell or tissue. Then, using this mRNA as a template, reverse transcriptase makes cDNA copies of each mRNA molecule in the mixture. A completely sequenced cDNA is termed "full-length" cDNA. For economical reasons, however, cDNA is often only partially sequenced, yielding expressed sequence tags (ESTs). As mentioned above, cDNA consists only of the exons of the transcribed gene because the introns have been spliced out. The problem of cDNA mapping is to find the gene and its exon/intron structure on the genome from which the cDNA originated; see Figure 1. In this way, cDNA libraries can be used to identify previously unknown genes or to annotate a genomic sequence.

The software tool BLAT (the BLAST-Like Alignment Tool) [Ken02] allows a fast mapping of a cDNA/EST sequence to a genomic sequence as follows. First, the genomic sequence (the database) is divided into consecutive non-overlapping K-mers (subsequences of K contiguous bases). Then, the position of each occurrence of each K-mer is stored in a

*Computer Science Faculty, Theoretical Bioinformatics, University of Ulm, 89069 Ulm, Germany. Email: eo@informatik.uni-ulm.de

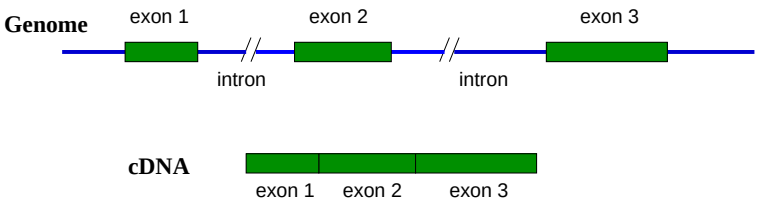

Figure 1: An example of cDNA mapped to a genomic sequence.

hash table. Searching for a cDNA/EST sequence (the query sequence) in the genome is done by obtaining from the hash table the "hits" for each K-mer in the query sequence. In other words, in the "search stage" one looks for all K-mers that are shared by the query sequence and the database. Other software tools like e.g. SSAHA [NCM01] use the same strategy. Because extensions of these shared regions are likely to be homologous, BLAT examines these regions in more detail and in an "alignment stage" produces alignments for the regions that are indeed homologous according to some criteria. Thus, BLAT uses a seed-and-extend strategy: The exact K-mer matches are the seeds that are extended in the alignment stage. BLAT also supports alternative search methods, in which a seed consists of either a *near perfect* match (with at most one mismatch) or multiple exact K-mer matches that are constrained to be near each other.

Our method is not a seed-and-extend method; it rather resembles the anchor-based multiple alignment methods. To be precise, we first build a suffix tree from the genomic sequence. Then, maximal exact matches[1] (MEMs)—exceeding a length threshold k—between the genomic sequence and the query sequence are determined by matching the query against the suffix tree. In fact, instead of a suffix tree, we use a data structure that requires less memory. This data structure, called enhanced suffix array [AKO04], requires only 5 bytes per character in the database. Finally, the MEMs computed in the search stage are clustered by a suitable chaining algorithm.

2 Searching with single exact matches

Besides K (the K-mer size) and k (the length threshold on MEMs), we will use the following parameters:

H: The size of a homologous region. For a human exon this is typically 50-200 bp.

M: The probability that two corresponding nucleotides in two homologous regions coincide. Roughly speaking, M is the match ratio between homologous regions.

G: Length of the genomic sequence. For example, the human genome contains $3 \cdot 10^9$ bp.

Q: Length of the query sequence. For cDNA/EST mapping this is typically 500 bp.

[1] A maximal exact match is an exact match that is bounded by mismatches.

A: The alphabet size. Here $A = 4$ because we solely consider nucleotide sequences.

P: Probability that homologous regions of length H will be found.

F: Expected number of (random) matches, based on the assumption that G and Q are random sequences (i.e., at each position each nucleotide occurs with probability $\frac{1}{A} = \frac{1}{4}$).

For the single K-mer match strategy described above, the values of P and F can be computed by the equations [Ken02]: $P = 1 - (1 - M^K)^{\lfloor \frac{H}{K} \rfloor}$ and $F = (Q - K + 1)(G/K)(1/A)^K$. For example, if $M = 97\%$ and $H = 100$, then the probability that two homologous regions contain an exact match of length $K = 16$ exceeds 99%. In other words, if one searches for all K-mers of length 16 and extends these seeds appropriately, then two homologous regions will be found with probability $P \geq 99\%$. To put it differently, given M and H we can asked for the largest K such that $P \geq 99\%$. Table 1 shows the values of K for varying M. Furthermore, it shows the respective values of F for $G = 3 \cdot 10^9$ and $Q = 500$. The value of F gives a hint of how many false positive seeds are to be expected by chance. A large value of F means that one can expect that a lot of work in the alignment stage will be wasted, because many alignments of extended seeds will be "thrown away" because of poor quality.

	single exact K-mer match		single MEM		near perfect K-mer match		near perfect MEM		two exact K-mer matches		two MEMs	
M	K	F	k	F	K	F	k	F	K	F	k	F
81%	6	6.0e+7	8	1.7e+7	10	4.4e+6	13	5.9e+5	5	4.8e+6	7	2.4e+5
83%	6	6.0e+7	9	4.2e+6	11	1.1e+6	14	1.6e+5	5	4.8e+6	7	2,4e+5
85%	7	1.3e+7	10	1.1e+6	12	2.7e+5	15	42773	6	2.0e+5	8	1.5e+4
87%	8	2.8e+6	11	2.6e+5	12	2.7e+5	17	3043	6	2.0e+5	9	905
89%	9	6.3e+5	12	65625	13	67124	18	807	7	9233	10	55
91%	10	1.4e+5	13	16373	16	1037	21	15	8	427	11	3.3
93%	11	31861	15	1019	19	16	24	0.3	9	21	12	0.2
95%	14	389	18	16	22	0.2	29	0.0	11	0.1	14	0.0
97%	16	21	23	0.0	29	0.0	36	0.0	14	0.0	18	0.0

Table 1: Given M, the table shows the largest K (k, respectively) for which $P \geq 99\%$, where $H = 100$, $G = 3 \cdot 10^9$ and $Q = 500$. The F columns show the corresponding numbers of matches that are expected by chance.

In order to compare the single K-mer match strategy with the single MEM strategy, we will show how P and F can be computed for the latter. The probability that two homologous regions of length H contain an exact match of length $\geq k$ is the same as the probability that H coin flips (where the probability of the event head is M and that of tail is $1 - M$) produce a sequence of $\geq k$ consecutive heads. The probability of such a *headrun* can be obtained by the 0-order Markov model depicted in Figure 2.

The Markov model starts in state 0 and the transition probabilities $p_{i,i+1}$ from state i to state $i + 1$ are M for $0 \leq i < k$. Moreover, we have $p_{i,0} = 1 - M$ for $0 \leq i < k$ and $p_{k,k} = 1$, while all other transition probabilities are 0. Thus the transition matrix

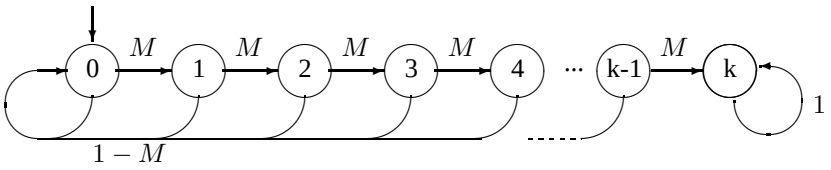

Figure 2: Markov model for the probability of a headrun of length $\geq k$.

$\mathcal{P} = (p_{i,j})$ is:

$$\mathcal{P} = \begin{pmatrix} p_{0,0} & p_{0,1} & ... & p_{0,k} \\ p_{1,0} & p_{1,1} & ... & p_{1,k} \\ ... & ... & ... & ... \\ p_{k-1,0} & p_{k-1,1} & ... & p_{k-1,k} \\ p_{k,0} & p_{k,1} & ... & p_{k,k} \end{pmatrix} = \begin{pmatrix} 1-M & M & 0 & 0 & ... & 0 & 0 \\ 1-M & 0 & M & 0 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 1-M & 0 & 0 & 0 & ... & 0 & M \\ 0 & 0 & 0 & 0 & ... & 0 & 1 \end{pmatrix}$$

After H time steps, the Markov model is in state k if and only if a headrun of length $\geq k$ occurred. Let $P = p_{0,k}^{(H)}$ denote the probability of this event. It is a consequence of the *Chapman-Kolmogorov equations* [Pap84] that the entries in the Hth power of the transition matrix give the H step transition probabilities. Hence $p_{0,k}^{(H)} = (\mathcal{P}^H)_{0,k}$. Again, for $H = 100$ and varying M, Table 1 shows the largest k such that $P \geq 99\%$.[2] It is not difficult to verify that if two homologous regions of 100bp contain exactly 3 mismatches, then they contain at least one MEM of length $\geq \lfloor 100/(3+1) \rfloor = 25$; see [KCO$^+$01]. Thus the reader may wonder why, according to Table 1, for $M = 97\%$ the largest k such that $P \geq 99\%$ is 23 and not 25. The explanation for this discrepancy is that $M = 97\%$ (recall that M is the probability that two corresponding nucleotides in two homologous regions coincide) does not mean that any two homologous regions have 3% mismatches.

In order to compute F for the single MEM-strategy, let $I_{l,i,j}$ be the following random variable:

$$I_{l,i,j} = \begin{cases} 1, & \text{if there is a MEM of length } l \text{ ending at position } i \text{ in } Q \text{ and at position } j \text{ in } G \\ 0, & \text{otherwise} \end{cases}$$

The expected value of $I_{l,i,j}$ is

$$E(I_{l,i,j}) = P(I_{l,i,j} = 1) = \left(1 - \frac{1}{A}\right)\left(\frac{1}{A}\right)^l\left(1 - \frac{1}{A}\right)$$

because this is the probability of l matching characters bounded by mismatches. Thus, F

[2] If a headrun is a rare event, then the probability that headrun of length $\geq k$ occurs can be obtained by the Chen-Stein method; see [Wat95]. However, these events are not rare in our context, because we deal with homologous regions.

can be calculated as follows.

$$\begin{aligned} F &= E\left(\sum_{l=k}^{Q}\sum_{i=l}^{G}\sum_{j=l}^{Q} I_{l,i,j}\right) = \sum_{l=k}^{Q}\sum_{i=l}^{G}\sum_{j=l}^{Q} E(I_{l,i,j}) \leq \sum_{l=k}^{Q}\sum_{i=k}^{G}\sum_{j=k}^{Q}\left(1-\frac{1}{A}\right)^2\left(\frac{1}{A}\right)^l \\ &= \sum_{l=k}^{Q}(G-k+1)(Q-k+1)\left(1-\frac{1}{A}\right)^2\left(\frac{1}{A}\right)^l \end{aligned}$$

Using the formula $\sum_{i=0}^{n} c^i = \frac{1-c^{n+1}}{1-c}$ for $c \neq 1$ yields $\sum_{l=k}^{Q}(\frac{1}{A})^l =$

$$\sum_{l=0}^{Q}\left(\frac{1}{A}\right)^l - \sum_{l=0}^{k-1}\left(\frac{1}{A}\right)^l = \left(\frac{1-\left(\frac{1}{A}\right)^{Q+1}}{1-\left(\frac{1}{A}\right)} - \frac{1-\left(\frac{1}{A}\right)^{k}}{1-\left(\frac{1}{A}\right)}\right) = \left(\left(\frac{1}{A}\right)^k - \left(\frac{1}{A}\right)^{Q+1}\right)$$

Since $\left(\frac{1}{A}\right)^{Q+1}$ is really small ($GQ\left(\frac{1}{A}\right)^{Q+1} = 3.5{\cdot}10^{-290}$ for $Q = 500,\ G = 3{\cdot}10^9,\ A = 4$), we can compute F approximately by

$$F \approx (G-k+1)(Q-k+1)\left(1-\frac{1}{A}\right)\left(\frac{1}{A}\right)^k$$

Table 1 shows the values of F for varying M.

Kent [Ken02] also derived formulas for the computation of P and F for the cases in which a seed consists of either a near perfect match (with at most one mismatch) or two exact K-mer matches that are constrained to be near each other. We did the same for MEMs instead of K-mer matches. For space reasons, the derivations of the corresponding formulas are omitted, but the respective values of P and F can be found in Table 1. The strategy of using MEMs instead of K-mers has the advantage that less matches are expected by chance. For example, if $M = 93\%$, then $K = 11$ and $k = 15$ guarantee that homologous regions are found with probability $P \geq 99\%$ by the single match strategy. In this case 31861 random K-mer matches are expected, but only 1019 random MEMs. If one uses a seed-and-extend method, then this implies that 30 times more random seeds have to be examined in the single K-mer match strategy than in the single MEM strategy.

By default BLAT uses a two K-mer matches strategy with $K = 11$. Our chaining algorithm that will be explained in the next section can be viewed as a single MEM strategy.

3 Chaining instead of seed-and-extend

In contrast to BLAT, we do not use a seed-and-extend method. Instead we search for the highest scoring chain of colinear matches (MEMs) between the cDNA and the genomic sequence. To make this precise, we need some preliminaries.

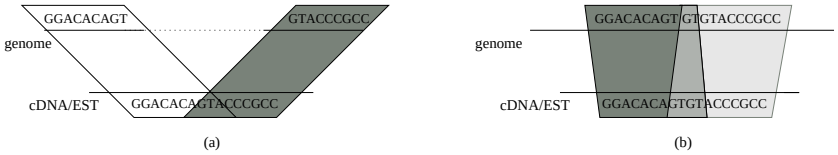

Figure 3: Overlapping MEMs.

3.1 Preliminaries

Definition 3.1 An *exact match* between two sequences G and Q is a triple (l, p, q) such that $G[p..p + l - 1] = Q[q..q + l - 1]$, i.e., the l-character-long substring of G starting at position p coincides with the l-character-long substring of Q starting at position q. An exact match is *left maximal* if $G[p-1] \neq Q[q-1]$ and *right maximal* if $G[p+l] \neq Q[q+l]$. A *maximal exact match* (*MEM*) is a left and right maximal exact match.

A *maximal unique match* (*MUM*) is a MEM (l, p, q) such that the substring $G[p..p + l - 1] = Q[q..q + l - 1]$ occurs exactly once in G and exactly once in Q.

Definition 3.2 Let (l_1, p_1, q_1) and (l_2, p_2, q_2) be two MEMs with $p_1 < p_2$ and $q_1 < q_2$. We say that (l_1, p_1, q_1) *overlaps* with (l_2, p_2, q_2)

- in G if and only if $p_2 \leq p_1 + l_1 - 1 < p_2 + l_2 - 1$
- in Q if and only if $q_2 \leq q_1 + l_1 - 1 < q_2 + l_2 - 1$

Figure 3 (a) shows two MEMs that overlap in Q but not in G, while Figure 3 (b) shows two MEMs that overlap in both G and Q.

A MEM (l, p, q) can be represented by a rectangle in $\mathbb{R}^2$ with the two extreme corner points (p, q) and $(p + l - 1, q + l - 1)$; see Figure 4. Such a rectangle is also called *fragment*. In the following, we will identify a MEM (l, p, q) with its fragment f in $\mathbb{R}^2$ and denote the two extreme corner points by $beg(f) = (beg(f).x, beg(f).y) = (p, q)$ and $end(f) = (end(f).x, end(f).y) = (p + l - 1, q + l - 1)$. Furthermore, we define $f.length = l$. That is, $f.length$ denotes the length of the MEM corresponding to f.

Definition 3.3 The relation $\ll$ on the set of fragments is defined as follows. $f' \ll f$ if and only if the following two conditions hold:

1. $beg(f').x < beg(f).x$ and $beg(f').y < beg(f).y$.
2. $end(f').x < end(f).x$ and $end(f').y < end(f).y$.

If $f' \ll f$, then we say that f' *precedes* f. The fragments f' and f are *colinear* if either f' precedes f or f precedes f'.

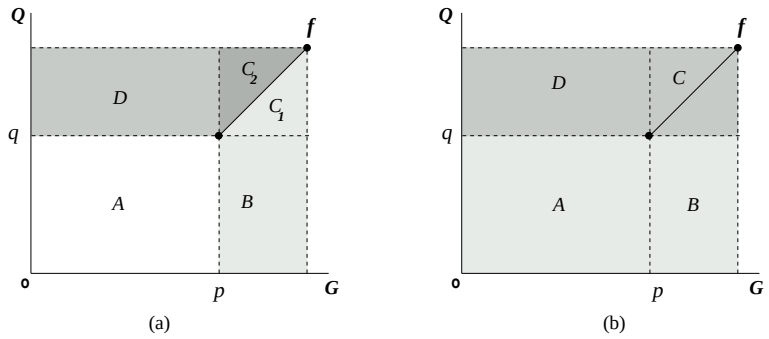

Figure 4: (a) The four cases that have to be considered in Shibuya and Kurochkin's approach. (b) The two cases $A \cup B$ and $C \cup D$ that have to be considered in our approach.

Thus, two fragments are colinear if they appear in the same order in both sequences. Note that if we further have $end(f').x < beg(f).x$ and $end(f').y < beg(f).y$, then f' and f are colinear and non-overlapping.

3.2 A new chaining algorithm for cDNA mapping

In cDNA mapping, the score $score(C)$ of a chain C of colinear and non-overlapping MEMs is the sum of the lengths of the MEMs in C. That is, gaps between the matches are not penalized because large gaps correspond to introns. Thus, a highest-scoring chain contains MEMs that best "cover" the cDNA and hence the locations of these matches in the genomic sequence are the most promising exon candidates. It is well-known that a highest-scoring chain can be computed in $O(m \log m)$ time using the technique of *sparse dynamic programming* [EGGI92], where m denotes the number of matches. We have shown in [AO05] that highest-scoring chains for more than two sequences can also be computed in subquadratic time using *range maximum queries* (RMQs) based on range trees or kD-trees. Contemporaneously, Shibuya and Kurochkin [SK03] showed that the chaining problem for two sequences can be solved with dynamic RMQs based on AVL trees. To take overlaps into account, they defined the *overlap length* of two MEMs to be the *maximum* of the amount of overlap in G and in Q. In their paper, the score $score(C)$ of a chain C of colinear MEMs is the sum of the lengths of the MEMs in C minus their overlap lengths. As a consequence of their definition of the overlap length, in the computation of a highest-scoring chain C_f of colinear MEMs ending with MEM f, one has to consider four different regions, namely A, $B \cup C_1$, C_2, and D; see Figure 4 (a). Each of these cases yields a candidate MEM f_i and a highest-scoring chain C_{f_i} of colinear MEMs ending with f_i, $1 \leq i \leq 4$. Then, $score(C_f)$ is computed by $score(C_f) = \max_i\{score(C_{f_i}) + f.length - overlap\ length(f, f_i)\}$; see [SK03] for details. Shibuya and Kurochkin's algorithm runs in $O(m \log m)$ time, but it is rather complicated because it is a mixture of RMQs and the candidate list paradigm.

We argue that for cDNA mapping Shibuya and Kurochkin's penalty for overlaps is not suitable. In our opinion, the overlap length of two MEMs should be defined solely as the amount of their overlap in the cDNA because in the problem at hand one wants to maximize the coverage of the cDNA. In other words, we suggest to *not* penalize an overlap in the genomic sequence G. The consequences of our approach are illustrated in Figure 4 (b). The x-axis corresponds to the genomic sequence G, while the y-axis corresponds to the cDNA sequence Q. Each MEM that can precede the MEM f in a chain must start in region A. Those that do not overlap with f in Q must end in region A or B. Those that do overlap with f in Q must end in region C or D. We will show next that this means that the cDNA mapping problem can be solved by two two-dimensional RMQs in the regions $A \cup B$ and $C \cup D$.

Definition 3.4 For any two fragments $f' \ll f$, the amount of overlap in the cDNA sequence is

$$overlap_y(f', f) = \begin{cases} end(f').y - beg(f).y + 1, & \text{if } end(f').y \geq beg(f).y \\ 0, & \text{otherwise} \end{cases}$$

In our opinion, the cDNA mapping problem should be formulated as follows.

Definition 3.5 Given a set of m fragments, find a chain C of colinear fragments $f_1, f_2, \ldots, f_t$ (i.e., $f_1 \ll f_2 \ll \ldots \ll f_t$) such that $score(C) = \sum_{i=1}^{t} f_i.length - \sum_{i=1}^{t-1} overlap_y(f_i, f_{i+1})$ is maximal.

Thus, we want to maximize the amount of cDNA sequence mapped to the genomic sequence. It is easy to see that a perfect mapping has a score that equals the cDNA length.

In the following, $f.score$ denotes the maximum score of all chains ending with the fragment f. When we speak about the score of the points $beg(f) = (beg(f).x, beg(f).y)$ and $end(f) = (end(f).x, end(f).y)$ we implicitly mean the score of f. Clearly, $f.score$ can be computed by the recurrence

$$f.score = f.length + \max\{f'.score - overlap_y(f', f) | f' \ll f\}$$

Algorithm 3.6 is a geometric solution to this recurrence. It uses a line sweep procedure w.r.t. the genomic sequence and range maximum queries to find the fragment that maximizes the score. There are two data structures D_1 and D_2 to efficiently answer 2-dimensional range maximum queries with activation. In this algorithm, the function $RMQ_{D_i}([x_1..x_2], [y_1..y_2])$, $i \in \{1, 2\}$, is a range maximum query that retrieves the point of maximum score in the set of active points stored in the data structure D_i and within the rectangular region defined by the intervals $[x_1..x_2]$ in G and $[y_1..y_2]$ in Q. For greater details, we recommend the reader to consult our paper [AO05].

Algorithm 3.6
Sort all start points of the m fragments in ascending order w.r.t. their x coordinate

and store them in the array points.
Store all the end points as inactive in the data structures D_1 *and* D_2.
for $1 \leq i \leq m$
  determine the fragment f *with* $beg(f).x = \mathsf{points}[i]$
  $q_1 := \mathsf{RMQ}_{D_1}([0..end(f).x-1],[0..beg(f).y-1])$
  $q_2 := \mathsf{RMQ}_{D_2}([0..end(f).x-1],[beg(f).y..end(f).y-1])$
  determine the fragment f_1 *with* $end(f_1) = q_1$
  if $f_1 = \perp$ **then** $score_1 = 0$
  else $score_1 = f_1.score$
  determine the fragment f_2 *with* $end(f_2) = q_2$
  if $f_2 = \perp$ **then** $score_2 = 0$
  else
    if $beg(f_2).y < beg(f).y$) **then**
      $score_2 = f_2.score - (end(f_2).y - beg(f).y)$
    else $f_2 = \perp$, *and* $score_2 = 0$
  $f.score = f.length + \max\{score_1, score_2\}$
  if $score_1 \geq score_2 > 0$ **then** *connect* f_1 *to* f **else if** $score_2 > 0$ **then** *connect* f_2 *to* f
  activate $(end(f).x, end(f).y)$ *in* D_1 *with score* $f.score$
  activate $(end(f).x, end(f).y)$ *in* D_2 *with score* $f.score - end(f).y$

Before proving the correctness of this algorithm, we would like to explain it. When the start point of a fragment f is scanned, we search for a fragment f' that precedes f and maximizes $f'.score - overlap_y(f', f)$. To take overlaps into account, we have to divide the search region into two subregions. The first subregion is the rectangle $([0..end(f).x - 1], [0..beg(f).y - 1])$; this is the region $A \cup B$ in Figure 4 (b). In the following this region will be denoted by $AB(f)$ to emphasize its dependence on f. Any fragment in this region that precedes f does not overlap with f in the cDNA sequence Q. The second subregion is the rectangle $([0..end(f).x - 1], [beg(f).y..end(f).y - 1])$, denoted by $CD(f)$; see region $C \cup D$ in Figure 4 (b). Any fragment in this region that precedes f overlaps with f in the cDNA sequence Q. In order to penalize this overlap, we activate each end point in D_2 with $f.score - end(f).y$ instead of $f.score$. This guarantees, as we shall see in the correctness proof, that a fragment f_2 will be found such that $f_2.score - overlap_y(f_2, f)$ is maximal in $CD(f)$. However, it may happen that $beg(f_2).y \geq beg(f).y$. That is, $beg(f_2) \notin A(f)$, where $A(f)$ denotes the rectangle $([0..beg(f).x - 1], [0..beg(f).y - 1])$. In this case, we simply ignore f_2. This does not affect the correctness of the algorithm, because then there is a fragment in $AB(f)$ whose score is at least as high as that of f_2. Finally, if $f_1.score \geq f_2.score - overlap_y(f, f_2)$, then f_1 is connected to f. Otherwise, f_2 is connected to f.

For a formal correctness proof, we need the following definition and lemmata.

Definition 3.7 The priority of a fragment f', denoted by $f'.priority$, is defined as $f'.priority = f'.score - end(f').y$.

Lemma 3.8 *Let* f' *and* f'' *be fragments with* $end(f') \in CD(f)$ *and* $end(f'') \in CD(f)$.

We have $f''.priority < f'.priority$ *if and only if* $f''.score - overlap_y(f'', f) < f'.score - overlap_y(f', f)$.

Proof

$$\begin{array}{rrcl} & f''.priority & < & f'.priority \\ \Leftrightarrow & f''.score - end(f'').y & < & f'.score - end(f').y \\ \Leftrightarrow & f''.score - (end(f'').y - beg(f).y) & < & f'.score - (end(f').y - beg(f).y) \\ \Leftrightarrow & f''.score - overlap_y(f'', f) & < & f'.score - overlap_y(f', f) \end{array}$$

Note that in the lemma $<$ can be replaced with $\leq$. □

Thus, if f' is a fragment with highest priority in $CD(f)$, then $f'.score - overlap_y(f', f)$ is maximal in $CD(f)$. The priority of a fragment f' is independent of f. Hence, it can be computed in constant time when f' is scanned. This has the advantage that the overlaps between all fragments need not be computed in advance (note that this would yield a quadratic time algorithm).

Lemma 3.9 *Let* C *be a chain composed of the fragments* $f_1, \ldots, f_t$. *For every index* i, $1 \leq i \leq t-1$, *we have* $f_{i+1}.priority \leq f_i.priority$.

Proof

$$\begin{array}{rcl} f_{i+1}.priority & = & f_{i+1}.score - end(f_{i+1}).y \\ & = & f_i.score + f_{i+1}.length - overlap_y(f_i, f_{i+1}) - end(f_{i+1}).y \\ & = & f_i.score - beg(f_{i+1}).y - overlap_y(f_i, f_{i+1}) + 1 \\ & \leq & f_i.score - end(f_i).y \\ & \leq & f_i.priority \end{array}$$

Not that if $overlap_y(f_i, f_{i+1}) = 0$, then $f_{i+1}.priority < f_i.priority$. □

Theorem 3.10 *Algorithm 3.6 correctly computes an optimal chain.*

Proof When the sweep-line reaches the start point $beg(f)$ of fragment f, the end points of all fragments that started before $beg(f).x$ are already activated in the data structures D_1 and D_2. The end points of the remaining fragments are still inactive. This guarantees that each fragment whose end point lies in $AB(f)$ or $CD(f)$ but whose start point occurs after $beg(f).x$ will not be considered in what follows. Because each fragment with end point in region $AB(f)$ does not overlap with f in the cDNA sequence, the two-dimensional $\mathtt{RMQ}_{D_1}$ retrieves the fragment f_1 of maximum score in the region $AB(f)$ by searching in the set of active end points in D_1. Analogously, because a fragment f' with end point in region $CD(f)$ overlaps with f in the cDNA sequence, it is activated in D_2 with priority $f'.score - end(f').y$. The two-dimensional $\mathtt{RMQ}_{D_2}$ yields the fragment f_2 of the highest priority in $CD(f)$, by searching in the set of active end points in D_2.

By Lemma 3.8, $f_2.score - overlap_y(f_2, f)$ is maximal in $CD(f)$. Suppose first that $beg(f_2).y < beg(f).y$, i.e., the start point of f_2 lies in $A(f)$. In this case we connect f_2 to f, if $f_2.score - overlap_y(f_2, f) > f_1.score$. Otherwise, we connect f_1 to f provided that $f_1 \neq \perp$, and we are done. Now suppose that $beg(f_2).y \geq beg(f).y$, i.e., the start point of f_2 lies in $CD(f)$. This implies $overlap_y(f_2, f) \geq f_2.length$. According to Lemma 3.9, if there is a fragment f' connected to f_2 (i.e., f' is the predecessor of f_2 in a highest-scoring chain ending with f_2), then the end point $end(f')$ of f' must lie in $A(f)$. Hence, $overlap_y(f', f_2) = 0$ and we have

$$\begin{aligned} f'.score &= f'.score - overlap_y(f', f_2) \\ &= f_2.length + f'.score - overlap_y(f', f_2) - f_2.length \\ &= f_2.score - f_2.length \\ &\geq f_2.score - overlap_y(f_2, f) \end{aligned}$$

Recall from Lemma 3.8 that $f_2.score - overlap_y(f_2, f)$ is maximal in $CD(f)$. Now it follows from $f'.score \geq f_2.score - overlap_y(f_2, f)$ in conjunction with $f'.score \leq f_1.score$ that f_2 can safely be ignored.

There is one remaining case: f_2 has no predecessor. Again, f_2 can be ignored by Lemma 3.8 because $f_2.score - overlap_y(f_2, f) = f_2.length - overlap_y(f_2, f) \leq 0$. □

The complexity of Algorithm 3.6 depends on the complexity of the RMQs with activation supported by the data structures D_1 and D_2. If D_1 and D_2 are implemented as range trees supported by the technique of fractional cascading and enhanced with priority queues as shown in [AO05], then the complexity of the algorithm is $O(m \log m \log \log m)$ time and $O(m \log m)$ space. If the *kd*-tree is used instead of the range tree, then the algorithm takes $O(m^{1.5})$ time in the worst case and $O(m)$ space. Interestingly, the query time of *kd*-tree can be improved in practice using a set of programming tricks [Ben90]. If the gaps between successive fragments in a chain are constrained to be at most W characters long (e.g., W could be set to the estimated maximum intron length), then RMQ_{D_1} and RMQ_{D_2} have to be limited to the regions $([end(f).x - W..end(f).x - 1], [beg(f).y - W..beg(f).y - 1])$, and $([end(f).x - W..end(f).x - 1], [beg(f).y..end(f).y - 1])$, respectively. This restriction increases the complexity of the algorithm using the range tree to $O(m \log^2 m)$ time and $O(m \log m)$ space (because the priority queues can no longer be used). The complexity using the *kd*-tree remains the same.

3.3 Special cases

In Algorithm 3.6, the range maximum queries are two-dimensional to guarantee that $beg(f') \in A(f)$ and $end(f') \in \{A \cup B \cup C \cup D\}$. However, one-dimensional RMQs suffice if $beg(f') \in A(f)$ implies $end(f') \in \{A \cup B \cup C \cup D\}$ and vice versa. There are two interesting special cases that meet this requirement. The first is the usage of *MUMs* or *rare-MEMs* instead of *MEMs* and the second is the restriction to a certain amount of overlapping.

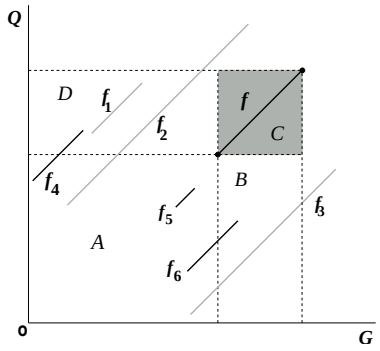

Figure 5: Fragment f_1 is embedded in fragment f in Q, while f is embedded in f_2 in Q and in f_3 in G. Such embeddings cannot occur if *MUMs* are used instead of *MEMs*.

3.3.1 *MUMs* or *rare-MEMs*

Suppose one uses *MUMs* instead of *MEMs*. When the sweep-line reaches the start point $beg(f)$ of fragment f, then it is sufficient to use the one-dimensional range maximum queries $\mathtt{RMQ}_{D_1}([0..beg(f).y-1])$ and $\mathtt{RMQ}_{D_2}([beg(f).y..end(f).y-1])$ instead of the two-dimensional RMQs of Algorithm 3.6. This is because $\mathtt{RMQ}_{D_1}([0..beg(f).y-1])$ considers only the end points that where activated before, that is, the corresponding start point must occur before $beg(f).x$. In summary, the RMQ yields a fragment f' with $beg(f')$ in $A(f)$. If $end(f')$ were not contained in the region $AB(f)$, then f would be embedded in f'; cf. fragment f_3 in Figure 5. However, a *MUM* cannot be embedded in another *MUM*. (If such *MUMs* would exist, then the substring of the embedded *MUM* would occur more than once in G or Q, which contradicts with the definition of *MUMs*). Analogously, one can show that $\mathtt{RMQ}_{D_2}([beg(f).y..end(f).y-1])$ retrieves the end point of a fragment f'' such that $beg(f'') \in A(f)$ and $end(f'') \in CD(f)$. Clearly, this implies that the algorithm sketched above requires $O(m \log m)$ time and $O(m)$ space. (For a one-dimensional RMQ the range tree and *kd*-tree are equivalent.) The algorithm can be modified to deal with rare-MEMs in $O(rm \log m)$ time and $O(m)$ space.

3.3.2 Restricting the amount of overlapping

Suppose that the minimum fragment length is k. If we tolerate overlappings of at most $k-1$ characters between any two successive fragments f' and f in a chain, (i.e., $end(f').x < beg(f).x + k$ and $end(f').y < beg(f).y + k$), then it follows that $beg(f') \in AB(f)$ (i.e., $beg(f').x < beg(f).x$ and $beg(f').y < beg(f).y$). This property can be used to reduce the dimension of the RMQs to one. To this end, we attach to each fragment f the virtual point $v(f) = (beg(f).x+k, beg(f).y+k)$. When the sweep-line reaches $v(f)$, we launch $\mathtt{RMQ}_{D_1}([0, beg(f).y-1])$ and $\mathtt{RMQ}_{D_2}(beg(f).y, v(f).y-1])$ to find the fragments of highest score. Algorithm 3.11 shows how to compute a chain of maximum score allowing only overlaps of at most $k-1$ characters. (For ease of presentation, we ignore the case in which the fragments retrieved by $\mathtt{RMQ}_{D_1}$ or $\mathtt{RMQ}_{D_2}$ are $\perp$.)

Algorithm 3.11
Sort all virtual and all end points of the m fragments in ascending order w.r.t. their x_1 coordinate and store them in the array points*.*
Store all the end points (ignoring their x_1 coordinate) as inactive in D_1 and D_2.
for $1 \leq i \leq 2m$
  if points$[i]$ *is the point* $v(f)$ **then**
    $q_1 := \mathsf{RMQ}_{D_1}(0, beg(f).y - 1)$
    $q_2 := \mathsf{RMQ}_{D_2}(beg(f).y, beg(f).y + k - 1)$
    determine the fragment f_1 *with* $end(f_1) = q_1$
    determine the fragment f_2 *with* $end(f_2) = q_2$
    $score_1 = f.length + f_1.score$
    $score_2 = f.length + f_2.score - (end(f_2).y - beg(f).y)$
    $f.score = \max\{score_1, score_2\}$
    if $f.score = score_1$ **then** *connect* f *to* f_1 **else** *connect* f *to* f_2
  else $/\star$ points$[i]$ *is the end point of a fragment* f $\star/$
    activate $end(f).y$ *in* D_1 *with score* $f.score$
    activate $end(f).y$ *in* D_2 *with score* $f.score - end(f).y$

4 Experimental results

We mapped the Fantom database [OFK+02] (version 2.1.1, 60770 full cDNA sequences of total length 120 Mbp) to the mouse chromosome 19 (UCSC Genome Browser) using BLAT and our method. More precisely, we used the *Vmatch* package developed by Stefan Kurtz (`http://www.vmatch.de`) to generate all *MEMs* of minimum length 20 (approx. 1.4 million) and the program CHAINER [AO04] to compute the highest-scoring chains of MEMs. (Although Table 1 suggests to use $k = 23$ for the experiments, we found that $k = 20$ results in a better coverage of the chains.) We tested BLAT using the default options (in particular $K = 11$) with and without the option that excludes K-mers that occur too often. BLAT took 442 minutes without this option and 12 minutes with this option. The running time of *Vmatch* and CHAINER was 18 minutes: 16 minutes to obtain the *MEMs* and 2 minutes to build the chains. Using the unmasked chromosome, BLAT took 39 hours even when over-repeated K-mers were excluded, while *Vmatch* and CHAINER took 2 hours when over-repeated MEMs were excluded. The experiments were performed on a Sun Fire 280 computer equipped with 6 GB RAM and two processors (UltraSPARC III Cu 1.015 GHz).

We compared our results on two levels with those obtained by BLAT, taking the annotation as a reference. The first level measures whether the cDNA is mapped or not. The second level measures the amount of correctly mapped exons in each cDNA. Table 2 shows the results of this comparison. These results are w.r.t. the positive strand only. From the table, one can see that BLAT and CHAINER have the same sensitivity on the gene level. However, the specificity of BLAT on that level is lower than that of CHAINER. Although we removed BLAT hits whose percentage identity is less than 50%, there are still 113 false positives. There are 41 genes that are contained solely in the annotation; this is due to the

Gene Level							Exon Level						
$a \wedge b \wedge c$	$a \wedge b$	$a \wedge c$	$b \wedge c$	a	b	c	$a \wedge b \wedge c$	$a \wedge b$	$a \wedge c$	$b \wedge c$	a	b	c
985	1	1	123	41	113	0	5056	8	9	53	43	20	11

Table 2: Accuracy of the cDNA mapping on the gene and exon level. The symbol a stands for the annotation, b refers to BLAT results, and c denotes CHAINER results. The term $a \wedge b \wedge c$ stands for the set of genes/exons contained in a, b, and c.

$0-$	$50-$	$60-$	$70-$	$80-$	$90-$	100	$0-$	$50-$	$60-$	$70-$	$80-$	$90-$	100
0	42	63	89	147	616	29	0	17	23	20	34	29	0

Table 3: Coverage w.r.t. CHAINER. Left: Coverage of the chains that are in the annotation. Right: Coverage of the chains that are not in the annotation.

fact that their regions were masked in the chromosomal sequence used.

On the exon level, we considered only the exons of the genes that occur in the annotation and that were found by both BLAT and CHAINER. On this level, BLAT also has less specificity than CHAINER. We examined the 20 hits found only by BLAT and found that all of them are false positives. Usually these hits correspond to the boundaries of the mapped cDNA and the hits lie far away from the corresponding annotated position in the genomic sequence. Interestingly, some of the 11 exons found only by CHAINER seem to be missing from the annotation. All 53 exons found by BLAT and CHAINER that are not in the annotation seem to be true positives.

Table 3 shows the coverage (percentage identity) of the chains obtained by CHAINER. We measured the coverage to estimate the amount of work needed to post-process the chains by means of a standard dynamic programming algorithm on the character level that takes splice site signals into account. From the table, one can see that about 90% of the annotated cDNAs have chains whose coverage is higher than 70%. This shows that dynamic programming has to be applied only to short regions. The cDNA sequences not occurring in the annotation that were mapped by BLAT and CHAINER have varying coverage levels. Table 3 (right) shows the coverage of these potentially false positives. Note that some of them have a rather high coverage, so one cannot rule out the possibility that these are in fact genes.

Finally, Table 4 quantifies the number and the amount of overlaps. It is interesting to note that most of the overlaps occur in the cDNA sequences and the size of the overlap is usually two, due to the splice site signals.

References

[AKO04] M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing Suffix Trees with Enhanced Suffix Arrays. *Journal of Discrete Algorithms*, 2:53–86, 2004.

	In the Annotation														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	$14-19$
Q	3071	796	1187	857	465	169	84	27	13	8	16	17	3	6	28
G	6472	69	73	29	9	4	12	5	13	11	11	8	7	4	20

	Not in the Annotation														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	$14-19$
Q	1844	25	29	13	5	2	7	0	7	0	0	0	0	0	2
G	1892	6	3	13	5	0	10	3	1	0	0	0	0	1	0

Table 4: Amount of overlapping between the *MEMs* in the chains. The row titled Q contains overlaps in the cDNA sequence, while that titled G contains overlaps in the genomic sequence. The total number of overlaps occurring in both the cDNA and the genome is 264 (226 occur in the annotation and 38 do not occur in the annotation).

[AO04] M.I. Abouelhoda and E. Ohlebusch. CHAINER: Software for Comparing Genomes. In *12th International Conference on Intelligent Systems for Molecular Biology/3rd European Conference on Computational Biology*, 2004. Short paper available at `http://www.iscb.org/ismbeccb2004/short%20papers/19.pdf`.

[AO05] M.I. Abouelhoda and E. Ohlebusch. Chaining Algorithms for Multiple Genome Comparison. *Journal of Discrete Algorithms*, 3:321–341, 2005.

[Ben90] J.L. Bentley. K-d trees for Semidynamic Point Sets. In *Proc. 6th Annual ACM Symposium on Computational Geometry*, pages 187–197, 1990.

[EGGI92] D. Eppstein, Z. Galil, R. Giancarlo, and G.F. Italiano. Sparse dynamic programming. I: Linear cost functions; II: Convex and concave cost functions. *Journal of the ACM*, 39:519–567, 1992.

[KCO+01] S. Kurtz, J.V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. REPuter: The Manifold Applications of Repeat Analysis on a Genomic Scale. *Nucleic Acids Research*, 29(22):4633–4642, 2001.

[Ken02] W.J. Kent. BLAT—The BLAST-Like Alignment Tool. *Genome Research*, 12:656–664, 2002.

[NCM01] Z. Ning, A.J. Cox, and J.C. Mullikin. SSAHA: A Fast Search Method for Large DNA Databases. *Genome Research*, 11(10):1725–1729, 2001.

[OFK+02] Y. Okazaki, M. Furuno, T. Kasukawa, J. Adachi, H. Bono, S. Kondo, I. Nikaido, N. Osato, R. Saito, and H. Suzuki et al. Analysis of the mouse transcriptome based on functional annotation of 60770 full-length cDNAs. *Nature*, 420(6951):563–573, 2002.

[Pap84] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 1984.

[SK03] S. Shibuya and I. Kurochkin. Match Chaining Algorithms for cDNA Mapping. In *Proc. 3rd International Workshop on Algorithms in Bioinformatics*, volume 2812 of *Lecture Notes in Bioinformatics*, pages 462–475, Berlin, 2003. Springer-Verlag.

[Wat95] M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman Hall, 1995.