Sequence Alignment with Tandem Duplication

Gary Benson*

Abstract

Algorithm development for comparing and aligning biological sequences has, until recently, been based on the SI model of mutational events which assumes that modification of sequences proceeds through any of the operations of substitution, insertion or deletion (the latter two collectively termed *indels*). While this model has worked fairly well, it has long been apparent that other mutational events occur.

In this paper, we introduce a new model, the DSI model which includes another common mutational event, *tandem duplication*. Tandem duplication produces *tandem repeats* which are common in DNA, making up perhaps 10% of the human genome. They are responsible for some human diseases and may serve a multitude of functions in DNA regulation and evolution.

Using the DSI model, we develop new exact and heuristic algorithms for comparing and aligning DNA sequences when they contain tandem repeats.

1 Introduction

Much research has been devoted to developing efficient algorithms for determining the similarity of two strings. An obvious motivation for this work comes from molecular biology. DNA sequences that are found to be similar often share a common origin and have diverged through a series of evolutionary events. The recognition of similar sequences can reveal structural and functional properties of major importance.

The method used to compute similarity should mirror the set of mutational events by which sequences can diverge. Until recently, the most common model of mutational events has allowed just three operations to transform one sequence into another [16]. These are substitutions (whereby a single *nucleotide base*–one of the four building blocks A, C, G, T of DNA–is replaced by a different base), and insertions and deletions (collectively termed *indels*, whereby a new sequence of bases is inserted between two preexisting bases or a preexisting subsequence is removed and the gap closed). We will refer to this as the **SI model** for substitutions and indels. The SI model reflects frequent mutational events and within this framework, dynamic programming algorithms are used to compute an optimal *similarity score* for two sequences and to produce an *alignment* of their characters [21, 26, 28, 9]. (An alignment is a pairing up of the characters from the two sequences to show which character in the first sequence occupies the same "position" as each character in the second sequence. See section 5 for an example.)

The SI model has worked fairly well, yet, it has long been apparent that there is a panoply of other mutational events that occur, including 1) **inversions** – the removal of a subsequence of DNA and the replacement by its reversed sequence, 2) **translocations** – the excision (*removal*) of a subsequence of DNA and its insertion in another location, 3) **transpositions** – the movement or copying of a subsequence of DNA into another location, often facilitated by structures within the subsequence and 4) **tandem duplications** – the copying

*Department of Biomathematical Sciences, Mount Sinai School of Medicine, Box 1023, One Gustave Levy Place, New York, NY 10029-6574; (212)241-5777; benson@ecology.biomath.mssm.edu. Partially supported by NSF grant CCR-9623532.

of a subsequence of DNA to a position immediately adjacent to the original copy. With ever more DNA sequence data becoming available for analysis, **the need to accurately compare sequences which have clearly undergone more complicated types of mutational processes is becoming critical. It is important therefore, that algorithms be developed that determine similarity and produce alignments in terms of a more complete set of mutational operations**.

In this paper, we address tandem duplication. Tandem duplication is a mutational process in which a stretch of DNA is duplicated to produce one or more new copies, each copy following the preceding one in a contiguous fashion. For example:

$$ABC \rightarrow ABBBC$$

(Here each letter represents a fixed but unspecified number of bases.) The result of a tandem duplication is a *tandem repeat*. Over time, tandem repeats undergo additional mutations so that typically, only *approximate* copies are present. Tandem repeats occur frequently (comprising perhaps 10% or more of the human genome) and form major chromosomal structures including centromeres and telomeres. They have recently been implicated in the causation of at least eight inherited human diseases including fragile-X mental retardation [30], Huntington's disease [13], myotonic dystrophy [8] and Friedreich's ataxia [5]. Tandem repeats may play a significant role in gene regulation [17, 10, 22], DNA-protein binding [23, 32], and evolution [11].

Besides their importance in DNA function and expression, tandem repeats are useful laboratory tools. The number of copies of the pattern in a tandem repeat is often variable among individuals (*polymorphic*). Polymorphic sites are useful in localizing genes to specific regions of the chromosome (linkage studies) and in DNA fingerprinting [6, 31]. Recently, polymorphic tandem repeats have been used to support the "Out of Africa" hypothesis of human evolution and migration [29, 1] and to suggest the evolutionary history of a microsatellite locus in primates [18].

In this paper, we **formalize a new model of mutation, the DSI model**, for tandem duplications, substitutions and indels. We present the **first algorithms for computing sequence similarity and alignment when tandem duplication is allowed as a mutational event.** We give both exact and heuristic algorithms.

Earlier research on tandem repeats has focused on 1) producing alignments of tandem repeats with a known pattern (wraparound dynamic programming - WDP), and 2) detecting unknown tandem repeats. WDP [19, 7] aligns an unknown number of tandem copies of a pattern sequence B with a sequence A. This is not the same as producing a local alignment of two sequences either of which may contain tandem repeats as subsequences. WDP is described more fully in Appendix A. Algorithms for detecting tandem repeats can be divided into exact algorithms [14, 27, 2, 24, 15] and heuristic algorithms [20, 4, 3]. The exact algorithms search either for tandem repeats or for highest scoring non-overlapping aligned regions (a broader category of repeat). Scoring methods include Hamming distance (substitutions only), unit cost edit distance (substitutions and indels have equal cost), and arbitrarily weighted edit operations. None of the exact algorithms can use the common affine gap penalty scheme (section 2). The best time for the exact algorithms is $O(n^2 \log n)$ for sequences of length n [24].

The heuristic algorithms search either for tandem repeats using matching k-tuples, or for "simple sequences" (again, a broader category) with data compression techniques. We make use of the newest algorithm that searches specifically for tandem repeats [3] in our heuristic approach.

The remainder of this paper is organized as follows. In section 2 we formalize the DSI model and give an overview of the problem we address. In section 3 we develop an exact algorithm for alignment with tandem repeats. Finally, in section 4 we present a heuristic algorithm which is designed to align DNA sequences containing tandem repeats in an efficient, yet biologically meaningful way. Two examples are presented in section 5.

2 Local alignment under the SI and DSI models

In what follows, we will examine alignment using *affine gap penalties*. In this formulation, an indel of length k has an associated cost $c(k) = \alpha + k * \beta$, where α is the cost for opening a gap and β is the cost for extending the gap by one character. (α and β are negative values.)

An alignment is an optimal pairing up of the characters from two sequences based on a scoring function. In *global alignment*, the entirety of both sequences must be aligned. In *local alignment* [28] the best scoring alignment of any pair of subsequences is determined. In what follows, we are concerned with local alignment. The modifications for global alignment are minor.

For sequences S_1 and S_2, under the SI model, with affine gap penalties, local alignment score $S[i,j]$, the optimal score when aligning suffixes of $S_1[1..i]$ and $S_2[1..j]$, is the maximum of four values:

$$S[i,j] \quad = \quad max \begin{cases} E[i,j] \\ F[i,j] \\ M[i,j] \\ 0 \end{cases}$$

where

- $E(i,j)$ is the highest score given that the alignment ends with a deletion at the right end of $S_2[1..j]$,
- $F(i,j)$ is the highest score given that the alignment ends with a deletion at the right end of $S_1[1..i]$,
- $M[i,j]$ is the highest score given that the alignment ends with a match (or substitution) between $S_1[i]$ and $S_2[j]$, and
- zero allows the suffixes that participate in the alignment to be optimally selected.

For the DSI model, we add another option, the **duplication option**. Thus, $Dup[i,j]$ is the highest score given that the alignment ends with a duplication of the right end of $S_1[1..i]$ or a duplication of the right end of $S_2[1..j]$ (see figure 1). We do not here subcategorize Dup as with E and F. Its meaning is explained more fully in section 3. Our new recurrence in the DSI model is therefore:

$$S[i,j] \quad = \quad max \begin{cases} Dup[i,j] \\ E[i,j] \\ F[i,j] \\ M[i,j] \\ 0 \end{cases} \tag{1}$$

The remainder of the paper solves the following problem:

Local Alignment with Tandem Duplication

Input: Strings $S_1[1 \ldots m]$ and $S_2[1 \ldots n]$ each containing zero, one or more occurrences of tandem repeats interspersed with regions that are not tandem repeats.

Output: Best scoring *local* alignment of S_1 and S_2 under the DSI model.

In the following sections, we handle the problem of efficiently computing Dup to obtain biologically meaningful alignments. Our main result concerns a modification of Wraparound Dynamic Programming (WDP) [19, 7]. WDP is explained more fully in Appendix A. We are concerned especially with GWDP, global alignment using WDP, a full recurrence of which is provided in Appendix B.

3 An exact duplication alignment algorithm

In this section we investigate the time and space complexity of two exact algorithms for producing alignments under the DSI model. We make the following assumptions about mutation type and order:

Assumption 1: There are *no excisions* (removal of a copy from a tandem repeat region).

Assumption 2: Duplication occurs *before* other types of mutations (indels and substitutions).

Assumptions 1 and 2 simplify the biology, but they do not affect the *general* alignment of the sequences so much as they affect the *score* of the alignment. That is, we are seeking an algorithm that recognizes a large gap as the result of a duplication rather than an indel. Even if these assumptions force the wrong copy of the tandem repeat pattern to be duplicated in the alignment, the dynamic programming algorithms, when presented with a large contiguous gap, will preferentially select the modest number of mismatch and indel errors required for duplication and alignment rather than select a large indel. Elimination of these assumptions would require confounding alignment with the much more complicated problem of tracing the evolutionary history of a tandem repeat and specifically with respect to Assumption 2 would require aligning hypothetical sequences.

3.1 Including the Duplication Option

Given two strings S_1 and S_2, at each index pair (y, x) we want to find the maximum score obtained by a duplication. Let U be a substring that is duplicated and let T be the substring aligned with the duplicate copies of U ($|U| = k$, $|T| = h$). We are interested in computing the following:

$$Dup[y,x] = \max_{U,T} \begin{cases} S[y-h, x-k] + \gamma + dupcost(U,T) - \delta, \\ \qquad\qquad T = S_1[y-h+1, \ldots, y] \\ \qquad\qquad U = S_2[x-k+1, \ldots, x] \\ \\ S[y-k, x-h] + \gamma + dupcost(U,T) - \delta, \\ \qquad\qquad U = S_1[y-k+1, \ldots, y] \\ \qquad\qquad T = S_2[x-h+1, \ldots, x] \end{cases}$$

where both γ and δ are negative valued parameters. In order to avoid redundancy, throughout the remainder of this paper, we will assume that U comes from S_2 and T comes from S_1. Dup is an alignment score based on the premise that the alignment ends with a tandem duplication of U aligned with T. Here, $S[i, j]$ is a local alignment score under the DSI model, γ is a duplication initiation cost and $dupcost(U, T)$ is the maximum alignment score for duplicating and aligning substring U with substring T. It is composed of:

1. a duplication extension cost δ, for *each copy* of U used in the alignment, and
2. an SI model cost to transform the copies of U into T.

The $-\delta$ term accounts for the fact that if the number of copies of U is c, the extension cost should be only $(c-1)*\delta$ because one copy is not a duplicate. Since we formulate *dupcost* to include a δ term for each copy of U, our definition includes the correction term.

Note that the γ term should not be included if only one copy of U is aligned with T. We do not modify the Dup score to account for this possibility because the correct value will be reflected in the separate SI model calculations (equation 1). Below is an outline of the program framework on which we build Algorithms 1 and 2. The difference in the algorithms is in the way we compute Dup.

INPUT: Strings $S_1[1\ldots m]$ and $S_2[1\ldots n]$

OUTPUT: Best scoring local alignment of S_1 and S_2 under the DSI model.

begin program
  for $y = 1$ **to** m
    for $x = 1$ **to** n
      $S[y][x] =$ maxscore{SI model options
        (substitutions and indels)}
      for $k = 1$ **to** x
        select U: $U = S_2[x-k+1,\ldots,x]$
        for $h = 1$ **to** y
          select T: $T = S_1[y-h+1,\ldots,y]$
          compute dupoption with U and T:
          $DupS_2 = S[y-h, x-k] + \gamma - \delta$
            $+dupcost(U,T)$
          $S[y][x] = \max\{S[y][x], DupS_2\}$
        endfor
      endfor
      for $k = 1$ **to** y
        select U: $U = S_1[y-k+1,\ldots,y]$
        for $h = 1$ **to** x
          select T: $T = S_2[x-h+1,\ldots,x]$
          compute dupoption with U and T:
          $DupS_1 = S[y-k, x-h] + \gamma - \delta$
            $+dupcost(U,T)$
          $S[y][x] = \max\{S[y][x], DupS_1\}$
        endfor
      endfor
  Find best score $S[y][x]$**.**
  Trace back and output alignment.
end program

3.2 Algorithm 1

In order to compute $dupcost(U,T)$, we use GWDP. In this particular case, the score we seek occurs in the last column of the computation array G. This imposes the restriction that we must use an integral number of copies of U. Additionally, we impose a penalty δ for each copy of U that is used. The penalty is added each time an alignment enters the final column, that is, when the score in the final column comes from the diagonal or from the left. Let $|U| = k$. We modify the GWDP recursion (Appendix B) as follows. In every equation for $E[i,j]$ or $G[i,j]$ for which j can equal k, we include a term $+I(j=k)*\delta$. The indicator function I equals one if the boolean expression is true and zero otherwise. (See Appendix C.)

Theorem 1 For two strings of length m, maximum alignment score under the DSI model and Assumptions 1 and 2 can be computed in time $O(m^5)$ and space $O(m^2)$.

Proof: Examining the program framework given above, we see that the time complexity is $O(m^4)$ times the complexity of the calculation $dupcost(U,T)$. (The SI model computations are performed in $O(m^2)$ time.) The *dupcost* computation for a single U and T can be done in $O(m^2)$ time. This gives a total of $O(m^6)$ time. But, observe that for a given (y,x) pair and a given U, the duplication and alignment score of U versus all the substrings $T = S_1[y-h+1,\ldots,y],\ h \in [1..y]$ can be computed in a *single* GWDP array by doing the computation backwards, starting at $S_1[y]$ and $U[k]$. The time complexity is therefore $O(m^3)$ times the complexity of the GWDP array for a given (U,y) pair or $O(m^5)$ time in total. The space complexity is $O(m^2)$ for the SI model computations and $O(m^2)$ for all GWDP computations since we can discard the computation for a given U after it is completed. Note that the space can be reduced to $O(m)$ if we use the divide and conquer method of [12]. ∎

3.3 Algorithm 2

The costly part of the algorithm above is clearly the *dupcost* calculation for a single (U,y) pair. We now show that using a modified form of $GWDP$, **we can, for a fixed U, and all y, find the optimal $T = T_y$ and maximum score $dupcost(U,T_y)$ in time $O(m^2)$ and space $O(m)$**. Note that this means that we are able to determine for each y the optimal h such that $T_y = S_1[y-h+1,\ldots,y]$.

Assume for a fixed x on S_2, U is fixed with $|U| = k$. We need to compute

$$\forall y, D[y] = \max_{0<h\le y}\{S[y-h,x-k] + dupcost(U,T) - \delta\}$$

where $D[y]$ is the best local alignment score for suffixes of $S_1[1,\ldots,y]$ and $S_2[1,\ldots,x]$ given that the alignment ends with a duplication of $U = S_2[x-k+1,\ldots,x]$ to match some $T = S_1[y-h+1,\ldots,y]$. Now, if $D[0] = S[0,x-k] + \gamma - \delta$ and for all $0 < h \le y, D[y-h]$ is correctly calculated, then we have

$$D[y] \quad = \quad \max_{0<h\le y} \begin{cases} D[y-h] + dupcost(U,T) \\ S[y-h,x-k] + \gamma + dupcost(U,T) - \delta \end{cases}$$

Note that the term $-\delta$ appears in the second case because *dupcost* overcounts the number of penalized copies of U, but $-\delta$ does not occur in the first case because $D[y-h]$ has already been adjusted for the overcount. Since *dupcost* is the same for both possibilities, we only need to keep track of the larger of $D[y-h]$ and $S[y-h,x-k] + \gamma - \delta$.

$D[y]$ is calculated using GWDP in array G. If $|U| = k$, then $D[y] = G[y,k]$. Selecting the max of $D[y-h] + dupcost(U,T)$ and $S[y-h,x-k] + \gamma + dupcost(U,T) - \delta$ for every h can be reduced to comparing two sets of values in each row $y-h$ of G (see figure 2). The first set is calculated in the two pass GWDP using the values from previous rows. The second set is calculated using the initial value $S[y-h,x-k] + \gamma - \delta$. (For this set, a single pass is sufficient because all the values come from deleting part of U. Two passes will not change any values set in the first pass.) After the two sets are computed, the maximum of the two values in each column is selected. Although this can be computed in a straightforward manner in three passes, the same result can be accomplished in two passes by setting $G[y-h,0]$ to the maximum of 1) the value it would receive from a previous row and 2) the value $S[y-h,x-k] + \gamma - \delta$ and then continuing with the two pass approach. The modifications to GWDP are summarized in Appendix C.

Theorem 2 For two strings of length m, maximum alignment score under the DSI model and Assumptions 1 and 2 can be computed in time $O(m^4)$ and space $O(m^3)$.

Proof: There are a total of m^2 substrings U from S_2. Each gets its values $dupcost(U,T)$ for all T from a single GWDP computation requiring $O(m^2)$ time and $O(m)$ space (by saving only the current row values). The total for all m^2 substrings U is $O(m^4)$ time. Since the calculations are dependent upon previously computed values $S[y-h, x-k]$, all the U from one of S_1 or S_2 can not complete their calculations at one time. Thus, we must maintain the current row of those computations. The space requirement is therefore $O(m^3)$. ∎

4 A heuristic alignment algorithm

To reduce the time of Algorithm 2, we can 1) reduce the number of (y,x) pairs that calculate *dupcost*, 2) reduce the number of substrings U that are examined and 3) reduce the size of the substrings. In this section, we examine several heuristics to accomplish these goals. It should be emphasized that **each of these heuristics is designed to speed the calculation without sacrificing relevant biological information**.

4.1 Limiting the (y,x) pairs that compute *dupcost*.

Typically, the strings that we seek to align contain at most a few regions that are clearly tandem repeats and these regions compose only a small percentage of the total length of the string. This being so, efficiency can be gained without sacrificing accuracy by computing *dupcost* only at indices (y,x) within tandem repeat regions.

To implement this restriction we must overcome two hurdles:

- **Tandem repeat regions within a string must be detected.** Exact algorithms for finding tandem repeats [27, 2, 24] have several deficiencies. First, they find the largest tandem repeat or a list of all tandem repeats from largest to smallest. Since smaller tandem repeats may be hidden in a largest repeat, finding smaller tandem repeats requires extra search. Second, the algorithms are not adaptable to an affine gap penalty. Rather they use a penalty which is a constant times the size of the gap. Third, the best of these algorithms has time $O(m^2 \log m)$. Two heuristic algorithms [4, 3] are designed specifically to rapidly find both large and small tandem repeats, can use any gap scoring scheme, run in near linear time and have the useful property of providing a consensus sequence for the tandem repeat unit. Although the former algorithm [4] has the weakness that the repeat unit size must be specified in advance, the latter algorithm [3] does not require *apriori* knowledge of pattern size. We use the latter algorithm to find tandem repeats.
- **Candidates for duplication must be detected.** If a tandem repeat region occurs in S_1, then we need to know what regions of S_2 contain a similar pattern that could be a candidate for duplication. Such regions may not be tandem repeats themselves. Rather, they may consist of only a single approximate copy or portion of a copy of the pattern in S_1. Starting with a **consensus pattern** from the tandem repeat in one sequence, we use LWDP on the other sequence to find duplication candidates.

4.2 Restricting the number and size of candidate substrings U.

Once a tandem repeat is found and a consensus sequence established, we know the basic unit size, k, of the repeat. If we assume that only units of size k are duplicated, then we can increase efficiency by choosing duplication candidate substrings U that are (or nearly are) of size k.

While it is not always correct to assume that duplication in a tandem repeat occurs in units of a single fixed size (instead of a mixture of that size and its multiples), it is nonetheless acceptable for the alignment problem. Duplication at mixed sizes again raises the question of producing an accurate *history* of the tandem repeat which we avoid here. Duplication at mixed sizes can be incorporated into the algorithm in many cases without a significant degradation in the time complexity.

In order to select the substrings U in S_2, we first start by finding a tandem repeat region in S_1 and getting its consensus c. The region in S_1 is detected by finding a candidate pattern P which is then aligned with surrounding sequence [3]. This establishes the tandem repeat region. To form the consensus, for each position i in P, we choose the majority base aligned with that position or an indel if the majority is a deletion. Additionally, between each pair of positions, we insert a base if the majority shows an insertion.

Using c, we find a *candidate region* R in S_2 that aligns with c using LWDP. We require that the candidate region align with at least some large percentage of c, say 90% so that an isolated region slightly smaller than c is allowed to be a candidate. Let c have length k and call c_i the k character string starting at position i in the concatenation cc. The collection $c_i, i \in [1..k]$ is just the k cyclic permutations of the consensus. A unit U is any substring of R that aligns with one of the c_i. We illustrate with an example. Let the consensus be

$$CGTTGA$$

and let R be

$$TTGCGTTTCGACGTGA$$

We first produce an alignment of the candidate R with the consensus:

```
3  4  5  6     1  2  3  4        5  6     1  2  3  4  5  6
T  T  G  A     C  G  T  T  -  -  G  A     C  G  T  T  G  A
T  T  G  -     C  G  T  T  T  C  G  A     C  G  T  -  G  A
1  2  3        4  5  6  7  8  9 10 11    12 13 14    15 16
```

Numbers along the top represent positions within c. Numbers along the bottom represent positions within R. Next, the alignment is padded with a small part of the consensus on both the left and the right (in order to account for regions smaller than c). For illustrative purposes we pad with one character on either end.

```
2  3  4  5  6     1  2  3  4        5  6   1  2  3  4  5  6     1
G  T  T  G  A     C  G  T  T  -  -  G  A   C  G  T  T  G  A     C
-  T  T  G  -     C  G  T  T  T  C  G  A   C  G  T  -  G  A     -
   1  2  3        4  5  6  7  8  9 10 11  12 13 14     15 16
```

Now, a unit U is any substring of R aligned with one of the cyclic permutations of c in the top row. For example, some of the units are:

$TTGC$	aligned with the first c_2
$CGTTTCG$	aligned with the first c_6
$GACGT$	aligned with the second c_5

Units are assigned to the position in R that corresponds to their rightmost character. Note that units have size $O(k)$ and that one position may be assigned several units. Nonetheless, under reasonable assumptions about the alignment parameters, the number of units is linear in the length of R.

4.3 Analysis

A complete outline of Algorithm 3 is given in Appendix D. For the analysis, we make the following assumptions:

> **Heuristic Assumption 3:** Tandem repeat regions and candidates R are the only substrings that need to be examined for the duplication option.
>
> **Heuristic Assumption 4:** Duplication occurs only for single units U selected from R as above.

Theorem 3 Let S_1 and S_2 each be of length m. Let S_1 contain a tandem repeat region of length M with consensus length k. Let S_2 contain candidates R with total combined length N. The maximum alignment score under the DSI model, Assumptions 1-2 and Heuristic Assumptions 3-4 can be computed in $O(m^2 + kNM)$ time and $O(m^2 + kN)$ space or $O(m + kN)$ space.

Proof: The total number of units U in S_2 is $O(N)$. Each unit has size $O(k)$. The time complexity of calculating all the duplication scores for a single unit using the modified GWDP of Algorithm 2 is $O(kM)$ and for all the units is $O(kNM)$. If we keep the entire GWDP array for each of the units, then the space is $O(kNM)$. If we keep only the final row of values for each array, then we need only $O(kN)$ space altogether. Local alignment in the SI model, detection of tandem repeats and candidates are all accomplished in time and space $O(m^2)$ or space $O(m)$. The total complexity is therefore $O(m^2 + kNM)$ time and $O(m^2 + kN)$ or $O(m + kN)$ space. ∎

5 Examples

We now present two biological examples to demonstrate the algorithm. In the alignments that follow, gaps due to duplication are indicated by lower case letters *in the sequence that contains the gap.* That is, these

letters are not really present in the sequence in which they are shown. The letters are an unmutated duplicate of an adjacent subsequence, corresponding to Assumption 2. The duplicated sequence is indicated by the symbols $< - - -- >$. We use the lower case letters to show the mutations detected by *dupcost*.

Example 1. As a first example, we show the alignment of two samples of cDNA from the *Plasmodium chabaudi* erythrocyte membrane antigen mRNA. One of the examples, S_1, (Accession # U80896, nucleotides 1-2483) is a complete cds of the mRNA. The other, S_2 (Acession # U58989, nucleotides 1-921) is a partial cds. Each sequence contains a 42 base pair pattern that is repeated tandemly. In S_1 the pattern occurs 20 times. In S_2, it occurs 19 times. Figure 3 is a schematic of the alignment. The left end of the actual alignment, shown in figure 4, includes the last 6 copies of the tandem repeat pattern.

Example 2. An intron region of the immunoglobulin ϵ chain genes in mouse and rat was previously described as having an abundance of $(CA)_n$ repeats [11]. It was suggested that these repeats may be candidates for recombination events. We examined the sequences from rat (Genbank Locus RATIGCA, using nucleotides 3149 - 3578) and mouse (Genbank Locus MUSIGECM, using nucleotides 949 - 1355). Each sequence contains a *different*, previously undiscovered 101 bp tandem repeat. (These repeats were found with the program described in [3].) Figure 5 is a schematic of the correct alignment of these sequences. Shown in the schematic is the remarkable fact that the duplicated subsequences have the same length and contain a common element of 20 characters. The alignment in [11], was incorrect, misaligning 81 nucleotides in each sequence because it did not include the duplication gaps. An alignment upon which the schematic is based is shown in figure 6.

The algorithms described in this paper are unable to produce the alignment in figures 5 and 6. This is because of the overlap of the duplicated patterns. Our algorithms, like other dynamic programming algorithms for alignment [21, 28, 25] are based on a left to right scan. Due to the principle of optimality, there must exist a cut point in the alignment such that the total score is the sum of the score to the left and to the right of the cut. Because of the overlap, though, such a cut point does not exist. Specifically, at point a in the schematic shown in figure 5, the calculations for the second duplication are started before the resulting score from the first duplication is known. This is due to the definition of *dupcost* which includes the duplicated copies *and the original copy*. By modifying the definition of *dupcost* to include only the duplicated copies, we obtained the alignment in figure 6. But, in general, this will violate the principle of optimality and can not be used for all sequences.

At this point, it is wise to stop and consider the quality of the data available in Genbank and other databases. Although it is *possible* that the 101 bp repeats are real, the fact that they are different yet still have the same length and the fact that the copies are unmutated, both suggest that the duplications are the result of a systematic sequencing error rather than a mutational process of nature.

6 Conclusion

We have demonstrated that wraparound dynamic programming can be efficiently incorporated into a local alignment algorithm in order to align sequences that contain tandem repeats. Given the abundance of tandem repeats, finding them and aligning sequences containing them will become an increasingly common objective.

7 Acknowledgement

The author wishes to thank Astrid Jervis for her help in finding the mouse and rat sequences used in example 2, and Richard Harlan for his help in finding the *Plasmodium* sequences used in example 1.

References

[1] J. Armour, T. Anttinen, C. May, E. Vega, A. Sajantila, J. Kidd, K. Kidd, J. Bertranpetit, S. Pääbo, and A. Jeffreys. Minisatellite diversity supports a recent African origin for modern humans. *Nature Genetics*, 13:154–160, 1996.

[2] G. Benson. A space efficient algorithm for finding the best nonoverlapping alignment score. *Theoretical Computer Science*, 145:357–369, 1995.

[3] G. Benson. An algorithm for finding tandem repeats of unspecified pattern size. Manuscript, 1996.

[4] G. Benson and M. Waterman. A method for fast database search for all k-nucleotide repeats. *Nucleic Acids Research*, 22:4828–4836, 1994.

[5] V. Campuzano, L. Montermini, M.D. Molto, L. Pianese, and M. Cossee. Friedreich's ataxia: Autosomal recessive disease caused by an intronic GAA triplet repeat expansion. *Science*, 271:1423–1427, 1996.

[6] A. Edwards, H. Hammond, L. Jin, C. Caskey, and R. Chakraborty. Genetic variation at five trimeric and tetrameric tandem repeat loci in four human population groups. *Genomics*, 12:241–253, 1992.

[7] V. Fischetti, G. Landau, J. Schmidt, and P. Sellers. Identifying periodic occurrences of a template with applications to a protein structure. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. 3rd annual Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science*, volume 644, pages 111–120. Springer-Verlag, 1992.

[8] Y.-H. Fu, A. Pizzuti, R.G. Fenwick Jr, J. King, S. Rajnarayan, P.W. Dunne, J. Dubel, G.A. Nasser, T. Ashizawa, P. DeJong, B. Wieringa, R. Korneluk, M.B. Perryman, H.F. Epstein, and C.T. Caskey. An unstable triplet repeat in a gene related to myotonic muscular dystrophy. *Science*, 255:1256–1258, 1992.

[9] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[10] H. Hamada, M. Seidman, B. Howard, and C. Gorman. Enhanced gene expression by the poly(dT-dG) poly(dC-dA) sequence. *Molecular and Cellular Biology*, 4:2622–2630, 1984.

[11] L. Hellman, M. Steen, M. Sundvall, and U. Pettersson. A rapidly evolving region in the immunoglobulin heavy chain loci of rat and mouse: postulated role of $(\text{dC-dA})_n$ $(\text{dG-dT})_n$ sequences. *Gene*, 68:93–100, 1988.

[12] D.S. Hirschberg. A linear space algorithm for computing longest common subsequences. *Communications of the ACM*, 18:341–343, 1975.

[13] Huntington's disease collaborative research group. A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease chromosomes. *Cell*, 72:971–983, 1993.

[14] G. Landau and J. Schmidt. An algorithm for approximate tandem repeats. In *Proc. 4th Annual Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science*, volume 648, pages 120–133. Springer-Verlag, 1993.

[15] M-Y. Leung, B.E.Blaisdell, C. Burge, and S. Karlin. An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *J. Mol. Biol*, 221:1367–1378, 1991.

[16] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Phys. Dokl.*, 10:707–710, 1966.

[17] Q. Lu, L. Wallrath, H. Granok, and S. Elgin. $(CT)_n$ $(GA)_n$ repeats and heat shock elements have distinct roles in chromatin structure and transcriptional activation of the *Drosophila hsp26* gene. *Molecular and Cellular Biology*, 13:2802–2814, 1993.

[18] W. Messier, S-H. Li, and C-B. Stewart. The birth of mircosatellites. *Nature*, 381:483, 1996.

[19] W. Miller and E. Myers. Approximate matching of regular expressions. *Bulletin of Mathematical Biology*, 51:5–37, 1989.

[20] A. Milosavljević and J. Jurka. Discovering simple DNA sequences by the algorithmic significance method. *CABIOS*, 9:407–411, 1993.

[21] S. Needleman and C. Wunch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[22] M. Pardue, K. Lowenhaupt, A. Rich, and A. Nordheim. $(dC\text{-}dA)_n$ $(dG\text{-}dT)_n$ sequences have evolutionarily conserved chromosomal locations in *Drosophila* with implications for roles in chromosome structure and function. *The EMBO Journal*, 6:1781–1789, 1987.

[23] R. Richards, K. Holman, S. Yu, and G. Southerland. Fragile X syndrome unstable element, $p(CCG)_n$, and other simple tandem repeat sequences are binding sites for specific nuclear proteins. *Hum. Mol. Genet.*, 2:1429–1435, 1993.

[24] J.P. Schmidt. All highest scoring paths in weighted grid graphs and its application to finding all approximate repeats in strings. In *Third Israel Symposium on Theory of Computing and Systems*, pages 67–77. IEEE Computer Society Press, 1995.

[25] M. Schoniger and M. Waterman. A local algorithm for DNA sequence alignment with inversions. *Bulletin of Mathematical Biology*, 54:521–536, 1992.

[26] P. Sellers. An algorithm for the distance between two sequences. *J. Comb. Theory*, 16:253–258, 1974.

[27] S.K.Kannan and E.W.Myers. An algorithm for locating regions of maximum alignment score. In *Proc. 4th Annual Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science*, volume 648, pages 74–86. Springer-Verlag, 1993.

[28] T. Smith and M. Waterman. Comparison of biosequences. *Advances in Applied Mathematics*, 2:482–489, 1981.

[29] S.A. Tishkoff, E. Dietzsch, W. Speed, A.J. Pakstis, and J.R. Kidd. Global patterns of linkage disequilibrium at the CD4 locus and modern human origins. *Science*, 271:1380–1387, 1996.

[30] A. Verkerk, M. Pieretti, J. Sutcliffe, Y. Fu, D. Kuhl, A. Pizzuti, O. Reiner, S. Richards, M. Victoria, F. Zhang, B. Eussen, G. van Ommen, A. Blonden, G. Riggins, J. Chastain, C. Kunst, H. Galjaard, C. Caskey, D. Nelson, B. Oostra, and S. Warren. Identification of a gene (FMR-1) containing a CGG repeat coincident with a breakpoint cluster region exhibiting length variation in fragile X syndrome. *Cell*, 65:905–914, 1991.

[31] J. Weber and P. May. Abundant class of human DNA polymorphisms which can be typed using the polymerase chain reaction. *Am. J. Hum. Genet.*, 44:388–396, 1989.

[32] H. Yee, A. Wong, J. van den Sande, and J. Rattner. Identification of novel single stranded d$(TC)_n$ binding proteins in several mamalian species. *Nucleic Acids Res.*, 19:949–953, 1991.

Appendix A: Wraparound Dynamic Programming

The ability to detect and align tandem repeats has been greatly facilitated by the development of Wraparound Dynamic Programming (WDP). This method was first introduced by Myers and Miller [19] and independently developed and simplified by Fischetti, Landau, Schmidt and Sellers [7]. The technique is applicable to both local (LWDP) and global (GWDP) sequence alignment.

Let A and B be two strings with lengths n and k respectively where k can be much smaller than n. The strength of WDP lies in its ability to align A with an unknown number of *tandem copies* of B using a matrix of size nk rather than n^2. The technique involves a normal similarity computation, but the cell values in each row are computed with *two* passes through the row. For LWDP, in the first pass, cell $S[i,1]$ (corresponding to A_i and B_1) is given the maximum of

1. a value derived from the cell $S[i-1,1]$, the first cell in the row above
2. a value derived from cell $S[i-1,k]$, the last cell in the row above and
3. zero

For GWDP, cell $S[i,1]$ is given the maximum of

1. a value derived from the cell $S[i-1,1]$,
2. a value derived from cell $S[i-1,k]$, above
3. a value derived from deleting the first $i-1$ characters of A and
4. a value derived from deleting the first i characters of A.

For the later two cases, we introduce a column zero in the GWDP array.

For both LWDP and GWDP, in the second pass, $S[i,1]$ receives the maximum of 1) its current value, and 2) a value derived from $S[i,k]$, the last cell in its row. Thus, in both the first and second passes, the computation wraps around.

WDP models the similarity computation of A with an unlimited number of copies of B in the sense that for any index j of B, WDP computes in $S[i,j]$ the highest score that would be obtained by aligning $A_1 \cdots A_i$ with $B^* B_1 \cdots B_j$, where B^* indicates zero or more tandem copies of B. The proof is not complicated and hinges on the observation that any maximum scoring alignment will not contain a single deletion of $h \geq k$ characters of B. This is so because otherwise, another alignment exists, identical except for having a deletion of only $h-k$ characters, and possesing a higher score. Since WDP examines all alignments with deletions in B of size $< k$, it produces the maximum scoring alignment.

Appendix B: Global Wraparound Dynamic Programming

Global wraparound dynamic programming (GWDP) is used to calculate the best global alignment score between a sequence T and tandem copies of a sequence U. In particular, let G be the computation array and let $|U| = k$. Then the complete recurrence is:

Initialize row zero:

$$\begin{aligned} G[0,0] &= 0 \\ G[0,j] &= G[0,0] + \alpha + j * \beta \end{aligned}$$

Initialize row i ($i \geq 1$):

$$G[i,0] = G[0,0] + \alpha + i * \beta$$

Pass 1 ($i \geq 1, j \geq 1$):

$$E[i,j] = \begin{cases} G[i,0] + \alpha + \beta & j = 1 \\ \max \begin{cases} E[i,j-1] + \beta \\ G[i,j-1] + \alpha + \beta \end{cases} & j > 1 \end{cases}$$

$$F[i,j] = \begin{cases} G[0,j] + \alpha + \beta & i = 1 \\ \max \begin{cases} F[i-1,j] + \beta \\ G[i-1,j] + \alpha + \beta \end{cases} & i > 1 \end{cases}$$

$$G[i,j] = \begin{cases} \max \begin{cases} G[i-1,0] + match(T_i, U_j) & \text{diagonal} \\ G[i-1,j] + match(T_i, U_j) & \text{wraparound diagonal} \\ F[i,1] & \text{above} \\ E[i,1] & \text{left} \end{cases} & j = 1 \\ max \begin{cases} G[i-1,j-1] + match(T_i, U_j) & \text{diagonal} \\ F[i,j] & \text{above} \\ E[i,j] & \text{left} \end{cases} & j > 1 \end{cases}$$

Pass 2 ($i \geq 1, j \geq 1$):

$$
E[i,j] \quad = \quad \begin{cases} \max \begin{cases} E[i,j] + \beta & j = 1 \\ & \text{wraparound} \\ G[i,j] + \alpha + \beta & \text{left} \end{cases} \\ \\ \max \begin{cases} E[i,j-1] + \beta & \\ & j > 1 \\ G[i,j-1] + \alpha + \beta & \end{cases} \end{cases}
$$

$$
G[i,j] \quad = \quad \max \begin{cases} G[i,j] \\ \\ E[i,j] \end{cases}
$$

Appendix C: GWDP for fixed U

The following recursion computes

$$\forall y, D[y] = \max_{0<h\leq y} \{S[y-h, x-k] + \gamma + dupcost(U,T) - \delta\}$$

where $U = S_2[x-k+1, \ldots, x]$, $|U| = k$ and $T = S_1[y-h+1, \ldots, y]$, $|T| = h$. $D[y]$ is the best local alignment score for suffixes of $S_1[1, \ldots, y]$ and $S_2[1, \ldots, x]$ given that the alignment ends with duplication of a substring U to match a substring T. The parameters are 1) gap initiation α, 2) gap extension β 3) duplication initiation γ and 4) duplication extension δ. The term $I(j=k)$ below is the indicator function for the boolean expression.

$$\text{Let } X_i \quad = \quad S[i, x-k] + \gamma - \delta$$

Initialize row zero:

$$G[0,0] \quad = \quad X_0$$

$$G[0,j] \quad = \quad G[0,0] + \alpha + j * \beta + I(j=k) * \delta$$

Initialize row i ($i \geq 1$):

$$F[i,0] \quad = \quad \begin{cases} G[0,0] + \alpha + \beta & i = 1 \\ \max \begin{cases} F[i-1,0] + \beta \\ G[i-1,0] + \alpha + \beta \end{cases} & i > 1 \end{cases}$$

$$G[i,0] \quad = \quad \max \begin{cases} X_i & \text{reinitialize} \\ F[i,0] & \text{above} \end{cases}$$

Pass 1 ($i \geq 1, j \geq 1$):

$$E[i,j] \quad = \quad \begin{cases} G[i,0] + \alpha + \beta + I(j=k) * \delta & j = 1 \\ \max \begin{cases} E[i,j-1] + \beta + I(j=k) * \delta \\ G[i,j-1] + \alpha + \beta + I(j=k) * \delta \end{cases} & j > 1 \end{cases}$$

$$F[i,j] \quad = \quad \begin{cases} G[0,j] + \alpha + \beta & i = 1 \\ \max \begin{cases} F[i-1,j] + \beta \\ G[i-1,j] + \alpha + \beta \end{cases} & i > 1 \end{cases}$$

$$
G[i,j] \quad = \quad \begin{cases}
\max \begin{cases}
G[i-1,0] & & \\
+match(T_i, U_j) & & \\
+I(j=k)*\delta & \text{diagonal} & \\
 & & \\
G[i-1,k] & & \\
+match(T_i, U_j) & \text{wraparound} & j=1 \\
+I(j=k)*\delta & \text{diagonal} & \\
 & & \\
F[i,1] & \text{above} & \\
 & & \\
E[i,1] & \text{left} &
\end{cases} \\
 & \\
max \begin{cases}
G[i-1,j-1] & & \\
+match(T_i, U_j) & & \\
+I(j=k)*\delta & \text{diagonal} & \\
 & & j>1 \\
F[i,j] & \text{above} & \\
 & & \\
E[i,j] & \text{left} &
\end{cases}
\end{cases}
$$

Pass 2 $(i \geq 1, j \geq 1)$:

$$E[i,j] = \begin{cases} \max \begin{cases} E[i,k] \\ +\beta + I(j=k) * \delta \end{cases} & j = 1 \text{ wraparound} \\ \max \begin{cases} G[i,k] \\ +\alpha + \beta + I(j=k) * \delta \end{cases} & \text{left} \\ \max \begin{cases} E[i,j-1] \\ +\beta + I(j=k) * \delta \end{cases} & j > 1 \\ \max \begin{cases} G[i,j-1] \\ +\alpha + \beta + I(j=k) * \delta \end{cases} & \end{cases}$$

$$G[i,j] = \max \begin{cases} G[i,j] \\ E[i,j] \end{cases}$$

Appendix D: Algorithm 3

INPUT: Strings $S_1[1 \ldots m]$ and $S_2[1 \ldots n]$

OUTPUT: Best scoring alignment of S_1 and S_2 under the DSI model with heuristic determination of tandem repeat regions and candidates for duplication.

begin program
Preprocess S_1 **and** S_2**.**
L_1 = **TRScan(**S_1**).** Returns a list of probable tandem repeats. Each list entry TR contains: a **substring** of S_1, the **substring indices**, and the **size** and a **consensus sequence** for the repeat unit.
for $TR \in L_1$
TR**.candidates = LWDP(**TR**.consensus,**S_2**).** Returns a list of candidate substrings. Each candidate C contains: a **substring** from S_2, the **substring indices**, and a **unit list** for each index of the substring.
end for
L_2 =**TRScan(**S_2**).**
for $TR \in L_2$
TR**.candidates=LWDP(**TR**.consensus,**S_1**).**
end for
end preprocess
for $y = 1$ **to** m
for $x = 1$ **to** n
$S[y][x] = \max\{\text{SI model}\}$
if (y is an index of some $TR \in L_1$**) AND (**x is an index of some $C \in TR$**.candidates)**
for each unit substring $U \in C[x]$**.unit-list**
$DupS_2$ =**Dupoption(**U, TR, y**)** Returns the optimal GWDP score of tandem copies of U versus subsequences of TR ending at y where the initial values for the computation are $S[TR$**.lower-index**$-1, U$**.lower-index**$-1] \ldots S[y, U$**.lower-index**$- 1]$.
$S[y][x] = \max\{S[y][x], DupS_2\}$
endfor
endif
if (x is an index of some $TR \in L_2$**) AND (**y is an index of some $C \in TR$**.candidates)**
for each unit substring $U \in C[y]$**.unit-list**
$DupS_1$ =**Dupoption(**U, TR, x**)** Returns the optimal GWDP score of tandem copies of U versus subsequences of TR ending at x where the initial values for the computation are $S[U$**.lower-index**$-1, T$**.lower-index**$-1] \ldots S[U$**.lower-index**$-1, x]$.
$S[y][x] = \max\{S[y][x], DupS_1\}$
endfor
endif
endfor
endfor
Find best score $S[y][x]$**, trace back and output alignment.**
end program

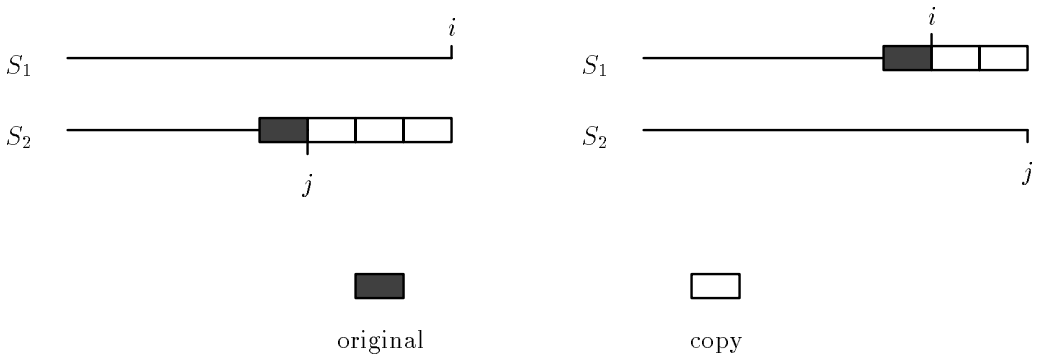

Figure 1:
The duplication option permits score $S[i, j]$ to represent an alignment ending with a duplication of the right end of one sequence aligned with the right end of the other.

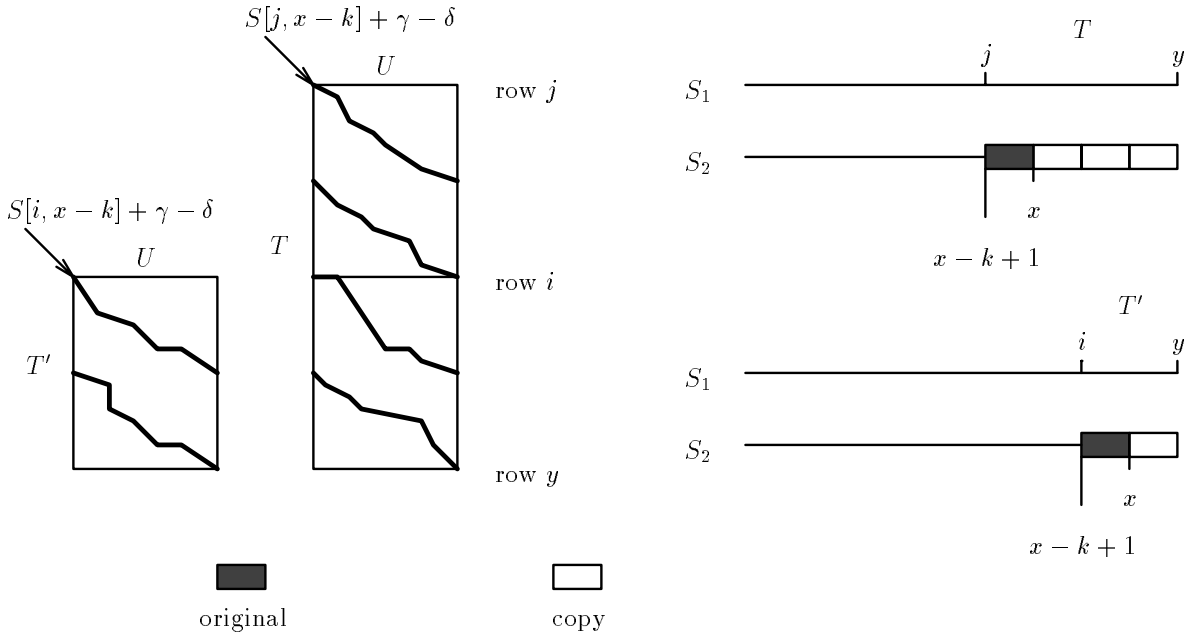

Figure 2:
Calculations for $Dup[y, x]$ for a fixed duplication unit U and two substrings T and T'. The heavy lines in the arrays represent the alignments which are illustrated at the right. The calculations can be merged so that only the best alternative is computed by setting, $\forall i$, $G[i][0]$ equal to the maximum of $S[i, x-k]+\gamma-\delta$ and the best score that $G[i][0]$ receives from the rows above.

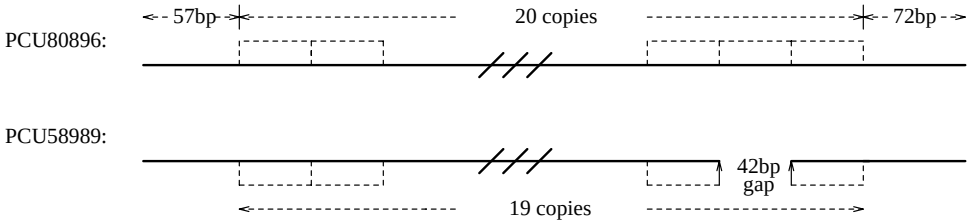

Figure 3:
A schematic of the alignment of two versions of cDNA from the *Plasmodium chabaudi* erythrocyte membrane antigen mRNA. The upper sequence contains 20 copies of a 42 nucleotide pattern and the lower sequence contains 19 copies. An SI model local alignment algorithm was not able to bridge the gap with a long indel.

```
Alignment: Top:     PCU80896      363 - 1325
           Bottom:  PCU58989        1 -  921

           *         * *                     *
     994 CAAATAGAAGAGAGCGATGCTCCTAAAGAAACTCAAGGCGAA    1035
     631 CAATTAGAAGAGAACAATGCTCCTAAAGAAACTCAAGCCGAA     672

          *                             *
    1036 GTAATAGTAGAGAACAATGCTCCTGAAGAAGCTCAAGCCGAA    1077
     673 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA     714



    1078 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA    1119
     715 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA     756

          *              *
    1120 GAAATAGTAGAGAACAACGCTCCTGAAGAAACTCAAGCCGAA    1161
     757 GTAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA     798
               <---------------------------------


    1162 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA    1203
     799 GAAATAGTAgagaacaatgctcctgaagaaactcaagccgaa     807
         -------->

                         *     *           *
    1204 GAAATAGTAGAGAACAATCCTCCTAAAGAAACTAAAAAAGAA    1245
     808 gaaatagtaGAGAACAATGCTCCTGAAGAAACTAAAGAAGAA     840


               * **         *
    1246 GAATTAGTAGAGTACTCTGAAGCTGACGTAAATGAAAGAGCA    1287
     841 GAATTAGTAAAACACTCTGAAGTTGACGTAAATGAAAGAGCA     882


         *            *
    1288 CATGAAATTATGAGCAAAGTTCTAGATCGCGTAAGACGC     1326
     883 GATGAAATTATGAACAAAGTTCTAGATCGCGTAAGACGC      921
```

Figure 4:
Alignment of *Plasmodium chabaudi* sequences. Only the left end of the alignment is shown, containing the final six copies of the tandem repeat.

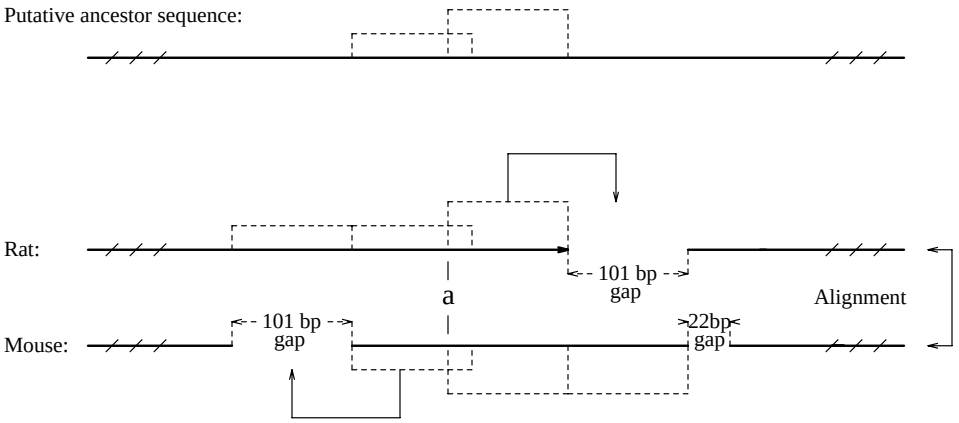

Figure 5:
A schematic of the alignment of an intron region in rat and mouse immunoglobulin ϵ chain genes. The rat contains two copies of one repeat and the mouse contains two copies of another. The two duplicated regions are the same length. The overlap is 20 nucleotides.

```
Alignment:    Top:     RATIGCA        3149 - 3578
              Bottom:  MUSIGECM        949 - 1355


                       *               **                 * *
3149 TCTGAGATACCTCTGAGGATCACCAATGGCAGAGTCGGCCAGCACCTCAGCCTCCAGGCC
 946 TCTGAGATACCTCTGAGGAACACCAATGGCAGAGTCAACCAGCACCTCAGCctccagact


            * *          **  *
3209 AA-TCCTTATACTTTGGCCCACTGCAGGCCATGAGAGATGGAGGAGG-TGGAGGCCTGAG
 997 aaatccttacattttggcccaccccaagccatgagagatggaggagggtggaggcctgag


       *  *   *                 *    *      * *         * *
3267 CTGTGGAAAACCAGAGACAGGAAGATGGTCTGTACTCCAGGCCAA-TCCTTATACTTTGG
 997 ctgcgggaaagcagagacaggaagatgggctgttCTCCAGACTAAATCCTTACATTTTGG
                                   <==========================

         **  *                                  *  *   *  <---
3326 CCCACTGCAGGCCATGAGAGATGGAGGAGG-TGGAGGCCTGAGCTGTGGAAAACCAGAGA
1023 CCCACCCCAAGCCATGAGAGATGGAGGAGGGTGGAGGCCTGAGCTGCGGGAAAGCAGAGA
     ============================================================

     -----------*----*---*-----------------**--*--------*--------
3385 CAGGAAGATGGTCTGTATGGAGAGAGTAGTAAACCAGATTATAGGGAGACTGAGGCAGGA
1083 CAGGAAGATGGGCTGTTTGGTGAGAGTAGTAAACCAGACAATGGGGAGACTAAGGCAGGA
     ================>

     -------*--------------------*-*------>              *    *
3445 GTAGAGCTCCTACAAGGCC-AGTAGTCTACCTTAGAGTgagacaggaagatggtctgtat
1143 GTAGAGCCCCTACAAGGCCCAG-AGTCTGCTTTAGAGTGAGACAGGAAGATGGGCTGTTT


      *                **  *        *              *
3504 ggagagagtagtaaaccagattatagggagactgaggcaggagtagagctcctacaaggc
1202 GGTGAGAGTAGTAAACCAGACAATGGGGAGACTAAGGCAGGAGTAGAGCCCCTACAAGGC


            * *                                           *
3482 c-agtagtctaccttagagtCCTATAAGTCTGGGCTGGGAGTCCATGTGTCCTGACTTGC
1262 CCAG-AGTCTGCTTTAGAGT----------------------CCATGTGTCCTGACCTGC


     *       **            *    **     **
3544 TCCTCAGATATCACAACCAAGATTCCTGGAGCCAGAGTGTGCATGCAGGCCCTAGAA
1299 CCCTCAGATGCCACAACCAAGATTTCTGGTTCCAGAGCATGCATGCAGGCCCTAGAA
```

Figure 6:
Alignment of an intron region in rat and mouse immunoglobulin ϵ chain genes.