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.

序列比对与串联重复

加里·本森

摘要

算法开发用于比较和校准生物序列，直到最近，一直基于突变事件的 SI 模型，该模型假定序列的修饰通过任何一种操作进行，即替换、插入或删除（后两种操作统称为 indels）。

尽管这个模型工作相当好，但长期以来很明显还有其他突变事件发生。

在这篇论文中，我们介绍了一个新的模型，即 DSI 模型，它包括另一个常见的突变事件，串联重复。串联重复产生串联重复序列，这在 DNA 中很常见，可能占人类基因组的 10%。它们是某些人类疾病的罪魁祸首，可能在 DNA 调控和进化中发挥多种功能。

使用 DSI 模型，我们为比较和比对含有串联重复的 DNA 序列开发了新的精确和启发式算法。

1 简介

大量研究致力于开发用于确定两个字符串相似性的高效算法。这项工作的一个明显动机来自分子生物学。发现相似的 DNA 序列通常具有共同的起源，并通过一系列进化事件而分化。识别相似序列可以揭示具有重要意义的结构和功能特性。

所使用的计算相似度的方法应反映序列通过突变事件发生分歧的集合。直到最近，最常见的突变事件模型仅允许三种操作将一个序列转换为另一个序列[16]。这些是替换（其中单个核苷酸碱基（DNA 的四个构建块之一 A、C、G、T）被不同的碱基所取代），以及插入和删除（统称为 indels，其中在两个现有碱基之间插入一个新的碱基序列，或者删除一个现有的子序列并关闭间隙）。我们将此称为 SI 模型，用于替换和 indels。SI 模型反映了频繁的突变事件，在此框架内，使用动态规划算法计算两个序列的最佳相似度得分，并生成它们的字符对齐[21, 26, 28, 9]。（对齐是将两个序列中的字符配对，以显示第一序列中的哪个字符与第二序列中的每个字符占据相同的“位置”。） 查看第 5 节以获取示例。）国际单位制模型工作相当好，然而，长期以来很明显，还有许多其他突变事件发生，包括 1）倒位（移除 DNA 的一个子序列并用其反转序列替换，2）易位（移除 DNA 的一个子序列并将其插入另一个位置，3）转座（将 DNA 的一个子序列移动或复制到另一个位置，通常由子序列内的结构促进，4）串联重复（复制）

fi

生物数学科学系，西奈山医学院，邮编 1023，一古斯塔夫·利维广场，纽约，纽约 10029-6574；（212）241-5777；benson@ecology.biomath.mssm.edu。部分资助由 NSF 资助项目 CCR-9623532 提供。

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.

将 DNA 子序列与原始副本相邻位置进行比对。随着越来越多的 DNA 序列数据可用于分析，准确比较明显经历了更复杂突变过程的序列的需求变得越来越关键。因此，开发出能够确定相似性并基于更完整的突变操作集产生对齐的算法变得非常重要。

在这篇论文中，我们讨论了串联重复。串联重复是一种突变过程，其中一段 DNA 被复制以产生一个或多个新副本，每个副本连续跟在先前的副本之后。例如：

AB C ! AB B B C

（这里每个字母代表一个固定但未指定的碱基数。）串联重复的结果是串联重复序列。随着时间的推移，串联重复序列会经历额外的突变，因此通常只有近似副本存在。串联重复序列频繁出现（可能占人类基因组的 10%或更多），并形成包括着丝粒和端粒在内的主要染色体结构。它们最近被怀疑至少与八种遗传性人类疾病有关，包括脆性 X 智力迟缓[30]、亨廷顿病[13]、肌强直性营养不良[8]和弗里德赖希共济失调[5]。串联重复序列可能在基因调控[17, 10, 22]、DNA–蛋白质结合[23, 32]和进化[11]中发挥重要作用。

除了在 DNA 功能和表达中的重要性外，串联重复序列在实验室工具中也很有用。在串联重复序列中，模式重复的拷贝数在个体之间通常是有变化的（多态性）。多态位点有助于将基因定位到染色体的特定区域（连锁研究）以及在 DNA 指纹识别[6, 31]中。最近，多态性串联重复序列已被用于支持“走出非洲”的人类进化和迁徙假说[29, 1]，并提出了灵长类动物中微卫星位点的进化历史[18]。

在这篇论文中，我们正式提出了一种新的突变模型，即 DSI 模型，用于串联重复、替换和插入/缺失。我们提出了当串联重复作为突变事件时计算序列相似性和比对的第一批算法。我们提供了精确和启发式算法。

早期对串联重复序列的研究主要集中在：1）生成具有已知模式的串联重复序列的对齐（环绕动态规划 – WDP），以及 2）检测未知串联重复序列。WDP [19, 7] 将未知数量的模式序列 B 的串联副本与序列 A 对齐。这不同于生成两个序列的局部对齐，其中任一序列可能包含串联重复序列作为子序列。WDP 在附录 A 中描述得更详细。检测串联重复序列的算法可以分为精确算法 [14, 27, 2, 24, 15] 和启发式算法 [20, 4, 3]。精确算法要么搜索串联重复序列，要么搜索最高得分的非重叠对齐区域（重复的更广泛类别）。评分方法包括汉明距离（仅替换）、单位成本编辑距离（替换和插入删除具有相同成本）和任意加权编辑操作。没有精确算法可以使用常见的成对间隙惩罚方案（第 2 节）。精确算法的最佳时间复杂度为 O(nlogn)，其中 n 为序列长度 [24]。

启发式算法要么使用匹配 k–元组搜索串联重复序列，要么使用数据压缩技术搜索“简单序列”（同样，这是一个更广泛的类别）。在我们的启发式方法中，我们利用了最新算法专门搜索串联重复序列[3]。

本文其余部分组织如下。在第 2 节中，我们形式化 DSI 模型并概述我们解决的问题。在第 3 节中，我们开发了一种针对串联重复的精确算法。最后，在第 4 节中，我们提出了一种启发式算法，该算法旨在以高效且具有生物学意义的方式对包含串联重复的 DNA 序列进行对齐。第 5 节中给出了两个示例。

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.

2 在 SI 和 DSI 模型下的局部对齐

以下我们将使用一元间隙惩罚来考察比对。在这个公式中，长度为 k 的插入或删除（indel）有一个关联的成本 c(k) = + k ，其中是打开间隙的成本，是扩展间隙一个字符的成本。（和是负值。）比对是基于评分函数从两个序列中字符的最优配对。在全局比对中，两个序列的全部都必须进行比对。在局部比对[28]中，任何两个子序列的最佳评分比对被确定。以下我们将关注局部比对。

全球对齐的修改很小。

对于序列 S 和 S，在 SI 模型下，具有单一间隙惩罚，局部比对得分 S[i; j]，当比对 S[1::i]和 S[1::j]的子串时，最优得分是四个值中的最大值：

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

哪里

E (i; j) 是在 S[1::j] 的右端以删除结束时的最高得分，F (i; j) 是在 S[1::i] 的右端以删除结束时的最高得分，M [i; j] 是在 S[i] 和 S[j] 之间匹配（或替换）结束时的最高得分，而零允许参与对齐的子串被最优选择。

对于 DSI 模型，我们添加了另一个选项，即重复选项。因此，D_up[i; j]是在对齐以 S[1::i]或 S[1::j]的右端重复结束的情况下给出的最高分数（见图 1）。我们在这里不将 D_up 细分为 E 和 F。其含义在 3 节中解释得更详细。因此，DSI 模型中的新递归公式为：

$$S[i;j] = \max \begin{cases} D_{up}[i;j] \\ E[i;j] \\ F[i;j] \\ M[i;j] \\ 0 \end{cases} \quad (1)$$

该论文的剩余部分解决了以下问题：

本地对齐与串联重复

字符串 S[1 : : : m]和 S[1 : : : n]各自包含零个、一个或多个串联重复序列，这些序列被非串联重复序列间隔。

最佳评分的 Sand Sunder DSI 模型局部对齐。

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.

在以下章节中，我们处理了高效计算 D 以获得具有生物学意义的比对的问题。我们的主要结果是关于 Wraparound 动态规划（WDP）[19, 7]的改进。WDP 在附录 A 中有更详细的解释。我们特别关注 GWDP，即使用 WDP 的全局比对，其完整递归过程在附录 B 中提供。

3 精确重复对齐算法

在本节中，我们研究了在 DSI 模型下产生对齐的两种精确算法的时间和空间复杂度。关于突变类型和顺序，我们做出以下假设：

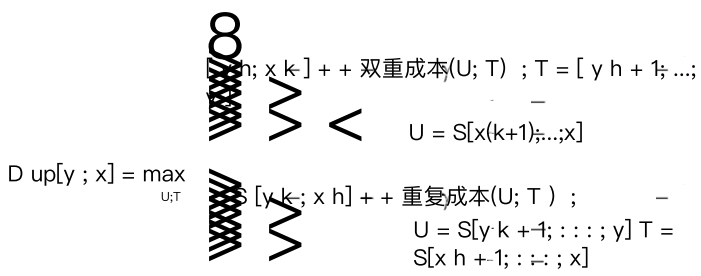
假设 1：没有切除（从串联重复区域移除副本）。

假设 2：复制发生在其他类型的突变（插入/缺失和替换）之前。

假设 1 和 2 简化了生物学，但它们对序列的一般对齐影响不大，更多的是影响对齐的得分。也就是说，我们正在寻找一种算法，它将大缺口识别为重复的结果，而不是插入或删除。即使这些假设导致串联重复模式的不正确副本在对齐中被重复，当动态规划算法面对大连续缺口时，也会优先选择所需的最小不匹配和插入/删除错误数量以进行重复和对齐，而不是选择大插入/删除。消除这些假设需要将对齐与追踪串联重复的进化历史这一更为复杂的问题混淆，特别是对于假设 2，需要对假设序列进行对齐。

3.1 包含复制选项

给定两个字符串 S 和 S，在每个索引对（y ; x）中，我们希望找到通过重复获得的最高分数。设 U 为一个重复的子串，T 为与 U 的重复副本对齐的子串（|U| = k，|T| = h）。我们感兴趣的是计算以下内容：



b oth 和 都是负值参数。为了避免冗余，在本论文的其余部分，我们将假设 U 来自 Sand T 来自 S。D up 是基于对齐以 U 与 T 对齐的串联重复为结束的对齐分数。在这里，S [i; j] 是 DSI 模型下的局部对齐分数，是重复起始成本，dupcost(U; T) 是复制和将子串 U 与子串 T 对齐的最大对齐分数。它由以下组成：

γ

1. 一个重复扩展成本为 ，对于在比对中使用的每个 U 的副本，以及 2. 将 U 的副本转换为 T 的 SI 模型成本。

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 = S2[x-k+1,...,x]
                for h = 1 to y
                    select T: T = S1[y-h+1,...,y]
                    compute dupoption with U and T:
                    DupS2 = S[y-h,x-k] + γ - δ
                            +dupcost(U,T)
                    S[y][x] = max{S[y][x], DupS2}
                endfor
            endfor
            for k = 1 to y
                select U: U = S1[y-k+1,...,y]
                for h = 1 to x
                    select T: T = S2[x-h+1,...,x]
                    compute dupoption with U and T:
                    DupS1 = S[y-k,x-h] + γ - δ
                            +dupcost(U,T)
                    S[y][x] = max{S[y][x], DupS1}
                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.)

该术语解释了这样一个事实：如果 U 的副本数量为 c，扩展成本应该是(c−1)乘以 b，因为一个副本不是重复的。由于我们将 dupcost 定义为包括 U 每个副本的修正项，因此我们的定义包括了修正项。

请注意，如果只有一个 U 与 T 对齐，则不应包含该术语。我们不会修改 D up 得分以考虑这种可能性，因为正确值将在单独的 SI 模型计算（方程 1）中反映。以下是构建算法 1 和 2 的程序框架概述。算法之间的区别在于我们计算 D up 的方式。

输入字符串 S[1 : : : m]和 S[1 : : : n] 输出：Sand Sunder DSI 模型的最佳得分局部对齐

```
开始程序
    for y = 1 到 m
        for x = 1 到 n
            S [y][x ] = 最大得分 fSI 模型选项
                        （替换和插入/缺失）
            for k = 1 到 x
                选择 U：U = S[xk+1;...;x]
                for h = 1 到 y
                    选择 T：T = S[y h + 1;...; y] 计算 D
                    upS 与 U 和 T 的重复选项：D upS =
                    S[y h; x k] +
                                +dupcost(U; T)
                    S [y ][x] = maxfS [y ][x]; D upSg
                endfor
            endfor
            for k = 1 到 y
                选择 U：U = S[y k + 1;...; y]
                for h = 1 到 x
                    选择 T：T = S[x h + 1;...; x] 计算 D
                    upS = S[y k; x h] +
                                +dupcost(U; T)
                    S [y ][x] = maxfS [y ][x]; D upSg
                endfor
    endfor 寻找最佳分数
    S [y][x].
回溯并输出对齐。结束程序
```

3.2 算法 1

为了计算 dupcost(U; T)，我们使用 GWDP。在这种情况下，我们寻求的分数出现在计算数组 G 的最后一列。这要求我们必须使用 U 的整数倍数。此外，我们为使用的每个 U 副本施加一个惩罚 。每次对齐进入最终列时，即最终列的分数来自对角线或左侧时，都会添加这个惩罚。令 jU j = k。我们按如下方式修改 GWDP 递归（附录 B）。对于每个 E[i; j]或 G[i; j]的方程，其中 j 可以等于 k，我们包括一个项+I (j = k) 。指示函数 I 等于布尔表达式为真时为 1，否则为 0。（参见附录 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.

定理 1 对于长度为 m 的两个字符串，在 DSI 模型和假设 1 和 2 下，最大对齐得分可以在时间 O(m)和空间 O(m)内计算。

程序框架的优点：考察上述给出的程序框架，我们发现时间复杂度是 O(m) 倍的 dupcost(U; T) 计算复杂度。(SI 模型计算在 O(m) 时间内完成。）对于单个 U 和 T 的 dupcost 计算可以在 O(m) 时间内完成。这总共给出了 O(m) 的时间。但是，观察到一个给定的 (y; x) 对和一个给定的 U，U 与所有子串 T = S[y h + 1; ...; y]; h ∈ [1::y] 的重复和比对分数可以通过单个 GWDP 数组计算，通过从 S[y] 和 U[k] 开始反向计算。因此，时间复杂度是 O(m) 倍给定 (U; y) 对的 GWDP 数组复杂度，或者总共 O(m) 的时间。空间复杂度对于 SI 模型计算是 O(m)，对于所有 GWDP 计算也是 O(m)，因为我们可以在完成给定 U 的计算后丢弃它。注意，如果我们使用 [12] 中的分治法，空间可以减少到 O(m)。

3.3 算法 2

算法中成本较高的部分是单个(U; y)对的 dupcost 计算。现在我们展示，通过使用修改后的 GW DP，对于固定的 U 和所有的 y，我们可以在时间 O(m)和空间 O(m)内找到最优的 T = T[h] 和最大分数 dupcost(U; T)。注意，这意味着我们能够为每个 y 确定最优的 h，使得 T = S[y h + 1; : : : ; y]。

假设对于 S 上的固定 x，U 是固定的且满足 jUj=k。我们需要计算

$$8y ; D[y] = \max_{0 \le h < y} \{ S[y\ h; x\ k] + dupcost(U; T) \}$$

D[y]是 S[1; : : : ; y]和 S[1; : : : ; x]的子串，在 U = S[x − k + 1; : : : ; x]与 T = S[y − h + 1; : : : ; y]匹配且以重复结束时的最佳局部对齐得分。现在，如果 D[0] = S[0; x − k] + ，并且对于所有 0 < h ⩽ y，D[y − h]都正确计算，那么我们有

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

请注意，在第二种情况下，术语“app ears”出现，因为 dupcost 高估了 U 受罚副本的数量，但在第一种情况下它不会出现，因为 D[y h]已经调整了高估。由于 dupcost 在这两种可能性中都是相同的，我们只需要跟踪 D[y h]和 S[y h; x k] + 中较大的一个。

D[y]使用 G 数组中的 GWDP 进行计算。如果 jUj=k，则 D[y]=G[y; k]。选择 D[y h] + dupcost(U; T)和 S[y h; x k] + + dupcost(U; T)的最大值对于每个 h 可以通过比较 G 中每行 y h 的两组值来简化（见图 2）。第一组值在两次遍历 GWDP 中使用前一行中的值计算得出。第二组值使用初始值 S[y h; x k] + 计算。(对于这组值，一次遍历就足够了，因为所有值都来自删除 U 的一部分。两次遍历不会改变第一次遍历中设置的任何值。）计算完这两组值后，选择每列中两个值的最大值。虽然这可以通过三次遍历直接计算，但可以通过将 G[y h; 0]设置为从上一行接收的值和 S[y h; x k] + 的最大值，然后继续使用两次遍历方法，在两次遍历中实现相同的结果。对 GWDP 的修改总结在附录 C 中。

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.

定理 2 对于长度为 m 的两个字符串，在 DSI 模型和假设 1 和 2 下，最大对齐得分可以在时间 O(m)和空间 O(m)内计算。

优点：从 S 中总共有 m 个子串 U。每个子串都从单个 GWDP 计算中获得其值 dupcost(U; T)，该计算需要 O(m)时间和 O(m)空间（仅保存当前行值）。所有 m 个子串 U 的总时间是 O(m)。由于计算依赖于先前计算过的值 S[y h; x k]，所以来自 S 或扫描的 U 不能一次性完成所有计算。因此，我们必须保持这些计算的当前行。因此，空间需求是 O(m)。

4 一种启发式对齐算法

为了减少算法 2 的时间，我们可以 1)减少计算 dupcost 的(y; x)对的数量，2)减少检查的子串 U 的数量，3)减少子串的大小。在本节中，我们考察了几个启发式方法来实现这些目标。应该强调的是，这些启发式方法都是为了加快计算速度，而不牺牲相关的生物信息。

4.1 限制计算 dupcost 的(y ; x)对。

通常，我们试图对齐的字符串最多只包含几个明显的串联重复区域，这些区域仅占字符串总长度的很小一部分。因此，可以在不牺牲准确性的情况下，仅计算串联重复区域内的索引（y；x）处的 dupcost，从而提高效率。

要实施这项限制，我们必须克服两个障碍：

串联重复区域必须在字符串中检测到。寻找串联重复区域的精确算法 rep eats [27, 2, 24] 存在几个缺陷。首先，它们找到最大的串联 rep eats 或从大到小排列的所有串联 rep eats 列表。由于较小的串联 rep eats 可能隐藏在最大的 rep eats 中，因此找到较小的串联 rep eats 需要额外的搜索。其次，算法不适应于线性间隙惩罚。相反，它们使用一个惩罚，这个惩罚是间隙大小的常数倍。第三，这些算法中最好的时间复杂度为 O(mlogm)。两个启发式算法[4, 3]专门设计用于快速找到大和小串联 rep eats，可以使用任何间隙评分方案，几乎以线性时间运行，并具有提供串联 rep eats 单元共识序列的有用特性。尽管前者算法[4]的弱点是必须预先指定重复单元的大小，但后者算法[3]不需要对模式大小有先验知识。我们使用后者算法来找到串联 rep eats。

候选重复项必须被检测。如果 S 中发生串联重复区域，那么我们需要知道 S 中哪些区域包含可能成为重复候选的类似模式。

这些区域可能不是串联重复。相反，它们可能只包含 S 中模式的单个近似副本或副本的一部分。从一个序列的串联重复中的共识模式开始，我们使用 LWDP 在另一个序列中找到重复候选者。

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.

4.2 限制候选子字符串 U 的数量和大小

一旦找到串联重复序列并建立共识序列，我们就知道了重复序列的基本单元大小，即 k。如果我们假设只有大小为 k 的单元被复制，那么通过选择大小为 k（或几乎为 k）的复制候选子串 U，我们可以提高效率。

虽然不能总是假设串联重复中的重复是以单个固定大小的单位发生的（而不是该大小及其倍数的混合），但对于对齐问题来说，这仍然是可接受的。在混合大小上的重复再次提出了产生串联重复准确历史记录的问题，我们在这里避免这个问题。在许多情况下，可以在不显著降低时间复杂度的情况下，将混合大小上的重复纳入算法中。

为了选择 S 中的子串 U，我们首先在 S 中找到一个串联重复区域，并获取其一致性序列 c。在 S 中检测到的区域是通过找到一个候选模式 P，然后将该模式与周围序列[3]对齐来建立的。这确定了串联重复区域。为了形成一致性序列，对于 P 中的每个位置 i，我们选择与该位置对齐的多数碱基，如果多数是插入，则选择一个插入。此外，在每对位置之间，如果多数显示插入，则插入一个碱基。

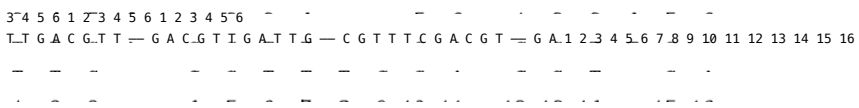
使用 C 语言，我们在 S 中找到一个候选区域 R，该区域使用 LWDP 与 c 对齐。我们要求候选区域至少与 c 的某个大百分比（例如 90%）对齐，这样略小于 c 的孤立区域也可以作为候选。设 c 的长度为 k，将 c 称为从位置 i 开始的 cc 连接的 k 个字符字符串。集合 c; i 2 [1::k]就是共识的 k 个循环排列。一个单元 U 是 R 的任何子串，它与 c 中的某个对齐。我们用一个例子来说明。设共识为

C GT T GA

并且让 R b e

T T GC GT T T C GAC GT GA

我们首先将候选 R 与共识进行对齐：



顶部数字表示 c 中的位置，底部数字表示 R 中的位置。接下来，对齐通过在左右两侧填充一小部分共识（为了考虑小于 c 的区域）来进行。为了说明目的，我们在两端各填充一个字符。

```
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.

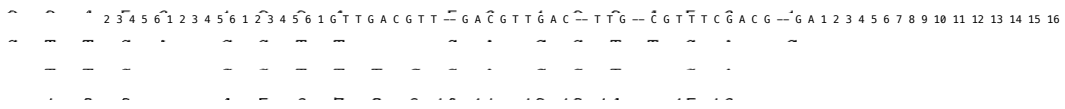
Heuristic Assumption 4: Duplication occurs only for single units U selected from R as above.

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



现在，一个单元 U 是 R 的任意子串，它与 c 在顶行的某个循环排列对齐。例如，一些单元包括：

T T GC 与第一个 c C GT T T C G 对齐；与第二个 c GAC GT 对齐

单元被分配到与它们最右侧字符对应的 R 中的位置。请注意，单元的大小为 O(k)，并且一个位置可以分配多个单元。尽管如此，在关于对齐参数的合理假设下，单元的数量与 R 的长度成线性关系。

4.3 分析

算法 3 的完整概述见附录 D。对于分析，我们做出以下假设：

启发式假设 3：串联重复区域和候选 R 是唯一需要检查重复选项的子字符串。

启发式假设 4：重复仅发生在从 R 中选出的单个单元 U，如上所述。

定理 3 设 S 和 Seach 为长度为 m 的序列。设 S 包含一个长度为 M 的串联重复区域，其一致性长度为 k。设 S 包含总长度为 N 的候选 R。在 DSI 模型、假设 1–2 和启发式假设 3–4 下，最大比对得分可以在 O(m+kN M)时间内和 O(m+kN)空间内或 O(m+kN)空间内计算。
总单元数 U 在 Sis O(N)。每个单元的大小为 O(k)。使用算法 2 的修改版 GWDP 计算单个单元的所有重复得分的复杂度为 O(kM)，对所有单元的复杂度为 O(kNM)。如果我们为每个单元保留整个 GWDP 数组，则空间复杂度为 O(kNM)。如果我们只为每个数组保留最后一行值，则总共只需要 O(kN)的空间。在 SI 模型中，局部对齐、串联重复和候选者的检测都在时间和空间 O(m)或空间 O(m)内完成。因此，总复杂度为 O(m+kNM)时间和 O(m+kN)或 O(m+kN)空间。

5 个示例

我们现在提供两个生物学示例来展示该算法。在以下对齐中，由于重复而产生的间隙由包含间隙的序列中的小写字母表示。也就是说，这些

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.

这些字母实际上并没有按照显示的顺序出现在序列中。这些字母是相邻子序列的一个未变异的副本，对应于假设 2。重复的序列由符号< >表示。我们使用小写字母来显示 dupcost 检测到的突变。

示例 1。作为第一个例子，我们展示了来自恶性疟原虫 chabaudi 红细胞膜抗原 mRNA 的两个 cDNA 样本的对齐。其中一个例子 S（登录号 U80896，核苷酸 1–2483）是 mRNA 的完整 cds。另一个，S（登录号 U58989，核苷酸 1–921）是部分 cds。每个序列都包含一个重复串联的 42 碱基模式。在 S 中，该模式出现 20 次。在 S 中，它出现 19 次。图 3 是对齐的示意图。实际对齐的左侧，如图 4 所示，包括串联重复模式的最后 6 个副本。

示例 2.小鼠和大鼠免疫球蛋白链基因的内含子区域之前被描述为富含(CA)重复[11]。有人提出，这些重复可能是重组事件的候选者。我们检查了大鼠（Genbank Loocus RATIGCA，使用核苷酸 3149–3578）和小鼠（Genbank Loocus MUSIGECM，使用核苷酸 949–1355）的序列。每个序列都包含一个不同的、之前未发现的 101 bp 串联重复。（这些重复是通过[3]中描述的程序发现的。）图 5 是这些序列正确对齐的示意图。示意图中显示了一个引人注目的事实，即重复的子序列长度相同，并包含一个 20 个字符的共同元素。[11]中的对齐是错误的，因为它没有包括重复间隙，导致每个序列错位 81 个核苷酸。图 6 显示了基于该示意图的对齐。

本文中描述的算法无法生成图 5 和图 6 中的对齐。这是因为重复模式的重叠。我们的算法，如其他用于对齐的动态规划算法[21, 28, 25]一样，基于从左到右的扫描。由于最优性原理，对齐中必须存在一个切割点，使得总得分是切割点左右得分的总和。然而，由于重叠，这样的切割点不存在。具体来说，在图 5 所示的示意图中的点 a，在第一个重复的结果得分已知之前就开始了第二个重复的计算。这是由于 dupcost 的定义包括了重复副本和原始副本。通过将 dupcost 的定义修改为仅包括重复副本，我们得到了图 6 中的对齐。但，一般来说，这将违反最优性原理，不能用于所有序列。

在这个阶段，明智的做法是停下来考虑 Genbank 和其他数据库中可用数据的质量。尽管 101 个碱基对的重复序列可能是真实的，但它们虽然不同却具有相同的长度，以及副本未发生突变的事实，都表明这些重复是系统测序错误的结果，而不是自然突变过程。

6 结论

我们已经证明，环绕动态规划可以有效地集成到局部比对算法中，以便比对包含串联重复序列。鉴于串联重复序列的丰富性，找到它们并比对包含它们的序列将变得越来越常见的目标。

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.

7 致谢

作者感谢 Astrid Jervis 在找到示例 2 中使用的鼠和鼠序列方面的帮助，以及 Richard Harlan 在找到示例 1 中使用的疟原虫序列方面的帮助。

参考文献

[1] J. Armour, T. Anttinen, C. May, E. Vega, A. Sa jantila, J. Kidd, K. Kidd, J. Bertranp etit, S. Paab o, and A. Je reys. 微卫星多样性支持现代人类最近起源于非洲。自然遗传学，13:154{160，1996。

[2] G. Benson. 一种寻找最佳非重叠比对得分的空间高效算法。理论计算机科学，145:357{369，1995。

[3] G. Benson. 一种寻找未指定模式大小的串联重复序列的算法。手稿，1996 年。

[4] G. Benson 和 M. Waterman. 一种快速数据库搜索所有 k 核苷酸重复序列的方法。核酸研究，22:4828{4836，1994。

[5] V. Campuzano, L. Montermini, M.D. Molto, L. Pianese 和 M. Cossee. 弗里德里希共济失调：由内含子 GAA 三联体重复扩张引起的常染色体隐性遗传病。科学，271:1423{1427，1996。

[6] A. Edwards, H. Hammond, L. Jin, C. Caskey, 和 R. Chakrab orty. 四个人群中五联体和四联体串联重复位点的遗传变异。基因组学，12:241{253，1992。

[7] V. Fischetti, G. Landau, J. Schmidt 和 P. Sellers. 识别模板的周期性出现及其在蛋白质结构中的应用。在 A. Apostolico, M. Crochemore, Z. Galil 和 U. Manber 编著的

tors，第 3 届组合模式匹配年度研讨会论文集，计算机科学讲座笔记
第 644 卷，第 111–120 页。Springer–Verlag，1992 年。

[8] Y.–H.傅，A.皮祖蒂，R.G.芬威克小，J.金，S.拉贾纳拉扬，P.W.邓恩，J.杜贝尔，G.A.纳塞尔，T.浅泽，P.德容，B.维林加，R.科内卢与肌强直性肌营养不良症相关基因中的不稳定三联重复。科学，255:1256{1258，1992。

[9] Gotoh O. 改进的生物序列匹配算法。分子生物学杂志，162(5):705–708，1982。

[10] H. Hamada, M. Seidman, B. Howard, 和 C. Gorman. 通过 p oly(dT–dG) p oly(dC–dA) 序列增强基因表达。分子和细胞生物学，4:2

[11] L.赫尔曼，M.斯蒂恩，M.桑德瓦，U.彼得松。免疫球蛋白中的一个快速进化区域

重链 Lo ci 鼠和鼠：提出(dC–dA)(dG–dT)序列的作用。基因，68:93{100，1988。

[12] D.S. Hirschberg. 用于计算最长公共子序列的线性空间算法。ACM 通讯，18:341{343，1975。

[13] 霍金森病协作研究组。一种包含三核苷酸重复序列的基因，在霍金森病染色体上扩张且不稳定。细胞，72:971{983，1993。

[14] G. 兰道和 J. 施密特。近似串联重复算法。在第四届年度研讨会论文集中

组合模式匹配，计算机科学讲座笔记，第 648 卷，第 120–133 页。
施普林格出版社，1993 年。

[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.

[15] 刘明耀，布莱斯德尔，伯奇，和卡尔林。一种用于识别多个长分子序列中错误匹配的高效算法。分子生物学杂志，221：1367–1378，1
[16] V. I. Levenshtein. 能够纠正删除、插入和反转的二进制码。苏联物理

文档，10:707{710，1966.

[17] Q. Lu, L. Wallrath, H. Granok 和 S. Elgin. (CT)(GA)rep 消耗和热冲击元件具有不同的
染色质结构中果蝇 hsp26 基因的转录激活作用。分子与细胞生物学，13:2802{2814，1993。

[18] W.梅西耶，李 S–H，和 C–B.斯图尔特。微卫星的诞生。自然，381:483，1996。

[19] W. Miller 和 E. Myers. 正则表达式的近似匹配。数学生物学通报，51:5{37，1989。
[20] A. 米洛萨夫列维奇和 J. 尤尔卡。通过算法意义方法发现简单的 DNA 序列。

CABIOS，9:407{411，1993 年。

[21] S. Needleman 和 C. Wunch. 适用于寻找两种蛋白质氨基酸序列相似性的通用方法。J. Mol. Biol., 48:443{453, 1970.
[22] M. Pardue, K. Lowenhaupt, A. Rich, 和 A. Nordheim. (dC–dA)(dG–dT)序列具有进化

arily 保守的果蝇染色体位置，对染色体结构和功能的作用有影响。《欧洲分子生物学组织杂志》，6:1781{1789，
1987。

[23] R. Richards, K. Holman, S. Yu 和 G. Southerland. fragile X 综合征不稳定元件，p(CCG)，以及其他简单串联重
复序列是特定核蛋白的结合位点。Hum. Mol.

遗传学，2:1429{1435, 1993.

[24] J.P. 施密特。加权网格图中所有最高得分路径及其在找到所有中的应用
字符串中的近似重复。在第三届以色列计算与系统理论研讨会，第几页
67{77. IEEE 计算机学会出版社，1995 年。

[25] M. Schoniger 和 M. Waterman. 一种用于 DNA 序列比对中倒置的局部算法。公报
数学生物学，54:521{536，1992。

[26] P. Sellers. 一种计算两个序列之间距离的算法。组合论杂志，16:253{258，1974。

[27] S.K.Kannan 和 E.W.Myers. 一种用于定位最大对齐得分区域的算法。在《进程》
第四届组合模式匹配年会，计算机科学讲座笔记，第卷
648，第 74–86 页。Springer–Verlag，1993 年。

[28] T. Smith 和 M. Waterman. 生物序列比较。应用数学进展，2:482{489，
1981.

[29] S.A. Tishko, E. Dietzsch, W. Speed, A.J. Pakstis 和 J.R. Kidd. CD4 位点的连锁不平衡的全球模式与现代人类起源。科学，271:138(
[30] A. Verkerk, M. Pieretti, J. Sutclie, Y. Fu, D. Kuhl, A. Pizzuti, O. Reiner, S. Richards, M. Victoria,

张 F，尤森 B，范奥门 G，布洛登 A，里吉斯 G，查斯坦 J，库恩特 C，加利亚德 H，卡斯凯 C，尼尔森 D，乌
斯特拉 B，沃伦 S。鉴定了一个包含 CGG 重复的基因（FMR–1），该基因与一个在脆性 X 综合征中表现出长度变
异的断裂点簇区域一致。
细胞，65:905{914, 1991.

[31] J. Web er 和 P. May. 可通过聚合酶链反应进行分型的丰富的人类 DNA 多态性类。美国人类遗传学杂志，
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.

[32] 叶浩，黄安，范登桑德，和拉特纳。在几种哺乳动物中鉴定新型单链 id(TC)结合蛋白。核酸研究，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.

附录 A：环绕动态规划

检测和校准串联重复序列的能力因 Wraparound 动态规划（WDP）的发展而大大简化。该方法首先由 Myers 和 Miller [19] 提出，并由 Fischetti、Landau、Schmidt 和 Sellers [7] 独立发展和简化。该技术适用于局部（LWDP）和全局（GWDP）序列比对。

A 和 B 是两个字符串，分别具有长度 n 和 k，其中 k 可以远小于 n。WDP 的强大之处在于其能够使用大小为 nk 的矩阵而不是 n 来将 A 与未知数量的 B 串联副本对齐。该技术涉及正常的相似度计算，但每个行的单元格值是通过两次遍历该行来计算的。对于 LWDP，在第一次遍历中，单元格 S[i; 1]（对应于 A 和 B）被赋予最大值

1. 由单元格 S[i−1;1]导出的值，即上方行的第一个单元格
2. 由单元格 S[i−1;k]导出的值，即上方行的最后一个单元格
3. 零

对于 GWDP，单元格 S[i; 1]被赋予最大值

1. 由单元格 S[i 1; 1]导出的值
2. 由单元格 S[i 1; k]导出的值，以上
3. 从 A 中删除前 i−1 个字符得到的值
4. 从 A 中删除前 i 个字符得到的值。

对于后两种情况，我们在 GWDP 数组中引入一列零。

对于 LWDP 和 GWDP，在第二次遍历中，S[i; 1]接收其当前值和从 S[i; k]（其行中的最后一个单元格）派生出的值的最大值。因此，在第一次和第二次遍历中，计算都会循环。

WDP 模型计算 A 与 B 无限副本之间的相似度，即在 B 的任何索引 j 处，WDP 在 S[i; j]中计算将 A 与 B B B 对齐的最高得分，其中 B 表示 B 的零个或多个串联副本。证明并不复杂，关键在于观察任何最大得分对齐都不会包含 B 中 h k 个字符的单个删除。这是因为否则，另一个对齐存在，除了只有一个 h k 个字符的删除外，其他都相同，并且具有更高的得分。由于 WDP 检查所有在 B 中删除大小小于 k 的对齐，因此它产生最大得分对齐。

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

附录 B：全局环绕动态规划

全局环绕动态规划（GWDP）用于计算序列 T 和序列 U 串联副本之间的最佳全局比对得分。特别是，设 G 为计算数组，且 jUj = k。那么完整的递推关系是：初始化第 0 行：

$$G[0;0] = 0 \quad G[0;j] = G[0;0] + + j$$

初始化行 i（i≥1）：

$$G[i;0] = G[0;0] + + i$$

第 1 次迭代（i≥1；j≥1）：

$$E[i;j] = \begin{cases} G[i;j-1] + + \beta & j > 1 \\ \max \begin{cases} [i;j] \quad 1] + \beta \\ G[i;j-1] + + \chi \quad \beta \end{cases} & j \text{ 大于 } 1 \end{cases}$$

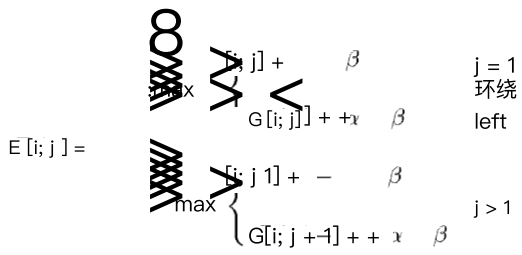
$$F[i;j] = \begin{cases} G[i-1;0] + + i = 1 \beta & i > 1 \\ \max \begin{cases} [i\ 1;j] + \beta \\ G[i-1;j] + + \chi \quad \beta \end{cases} & i \text{ 大于 } 1 \end{cases}$$

$$G[i;j] = \begin{cases} G[i-1;0] + \text{match}(T;U) \text{ 对角} & i = 1;\ j = 1 \\ G[i\ 1;j] \\ +\text{match}(T;U) \text{循环回绕} \\ \text{对角线} \\ \max \quad E[i;1] \text{ 上述} \\ E[i;1] \text{ 左} \\ G[i-1;j\ 1] + \text{match} \\ (T;U) \text{对角线} \\ \max \quad F[i;j] \text{ 上述} \\ E[i;j] \text{ 左} \end{cases} \quad \begin{array}{l} i > 1;\ j = 1 \\ \\ \\ \\ j > 1 \\ \\ j \text{ 大于 } 1 \end{array}$$

第 2 次迭代（i≥1；j≥1）：

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

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



G[i; j] = 最大 [i; j]
E [i; j]]

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

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

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

应用附录 C：针对 Xed U 的 GWDP

以下递归计算

8y ; D[y] = 最大 − − γ −

U = S[x k + 1; : : : ; x]，|U| = k，T = S[y h + 1; : : : ; y]，|T| = h。D[y]是在 S[1; : : : ; y]和 S[1; : : : ; x]的子串 U 和 T 匹配且以子串 U 的重复结束的情况下，S[1; : : : ; y]和 S[1; : : : ; x]后缀的最佳局部比对得分。参数包括 1）间隙起始，2）间隙扩展，3）重复起始和 4）重复扩展。下面的 I(j = k)是布尔表达式的指示函数。 α β γ

X = S [i; x k] + − γ −

初始化行零：

G[0; 0] = X

G[0; j] = G[0; 0] + + j +I (j = k)β

初始化行 i（i≥1）：

$$\begin{cases} G[0;0] + + i \cdot \beta \end{cases}$$

$$F[i;0] = \max \begin{cases} [i\ 1;0] + \beta & i > 1 \\ G[i\ 1;0] + + \alpha\ \beta & i \text{ 大于 } 1 \end{cases}$$

(i 重新初始化)

$$G[i;0] = \text{最大} \begin{cases} F[i;0] \text{ 上述} \end{cases}$$

第 1 次（i） 1; j 1):

$$\begin{cases} G[i;0] + + + (j = k) & j = 1 \end{cases}$$

$$E[i;j] = \max \begin{cases} ([i;j\ 1] + + I(j = k)) & j > 1 \\ G[i;j+1] + + + 1\ (j=k) & j \text{ 大于 } 1 \end{cases}$$

$$\begin{cases} G[i;j\ 1] + + i \cdot \beta \end{cases}$$

$$F[i;j] = \max \begin{cases} [i\ 1;j] + \beta & i > 1 \\ G[i\ 1;j] + + \alpha\ \beta & i \text{ 大于 } 1 \end{cases}$$

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

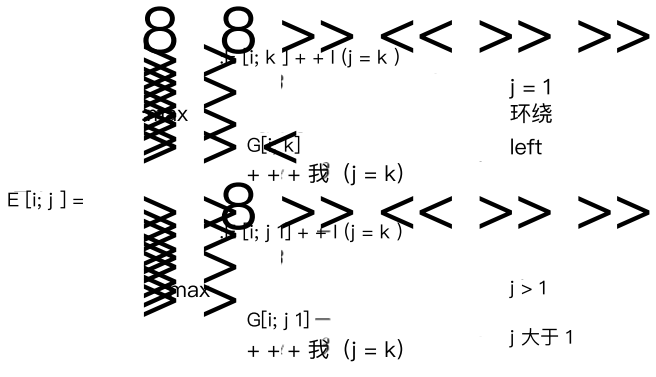
$$
G[i;j] = \begin{cases} G[i;0] + \text{匹配}(T;U) \\ \quad + I\,(j=k)\ \text{对角线} \\ \max \begin{cases} G[i\,1;k] \\ \quad + \text{match}(T;U)\ \text{wraparound} + I\,(j=k)\ \text{对角线} \\ E\,[i;1]\ \text{上述} \\ E\,[i;1]\ \text{左} \end{cases} & j=1 \\ G[i;j\,1] + \text{匹配}(T;U) \\ \quad + I\,(j=k)\ \text{对角线} \\ \max \begin{cases} E\,[i;j\,1]\ \text{上述} \\ E\,[i;j\,1]\ \text{左} \end{cases} & j>1\ \text{j 大于 1} \end{cases}
$$

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

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

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

第 2 步（i） 1; j 1):



G[i; j] = 最大 [i; j]
E [i; j]

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

附录 D：算法 3

输入字符串 S[1 : : : m]和 S[1 : : : n]的输出：与 DSI 模型的最佳得分对齐以及通过启发式确定串联重复区域和复制候选者的方法。

开始程序
预处理 沙 S.
L= TRScan(S). 返回一个可能的串联重复序列列表。每个列表条目 TR 包含：
S 的子串、子串索引
该重复单元的大小和一致性序列。
for T R 2 L
T R.candidates = LWDP(T R.共识,S).
返回候选子字符串列表。每个候选字符串 C 包含：
从 S 中提取子串，子串的索引以及每个子串索引的单元列表。
结束 for 循环
L=TRScan(S)。
for T R 2 L
T R.candidates=LWDP(T R.consensus,S) 结束 for 结束
preprocess for y = 1 到 m
for x = 1 到 n
S [y][x] = maxfSI 模型 g if (y 是某些 T R 2 L 的索引) AND (x 是某些 C 2 T R：候选人 的索引)
对于每个单元子串 U 2 C [x].unit−list
D upS=Dup 选项(U; T R; y)
返回 U 串联副本相对于 T R 子序列（以 y 结尾）的最佳 GWDP 分数，其中计算初始值为 S[T R.lower−index1; U .lower−index1]...S[y; U .lower−index 1]。S[y][x] = max{S[y][x]; D upSg}
endfor
endif
如果(x 是某个 T R 2 L 的索引)并且(y 是某个 C 2 T R：候选人)的索引
对于每个单元子串 U 2 C [y] .unit−list
D upS=Dup 选项(U; T R; x)
返回 U 串联副本相对于 T R 子序列（以 x 结尾）的最佳 GWDP 分数，其中计算初始值为 S [U .lower−index1; T .lower−index1] :: S [U .lower−index1; x]。S [y][x] = max{S [y][x]; D upSg}
endfor
endif
endfor
endfor 寻找最佳得分 S [y][x]，回溯并输出对齐。end program

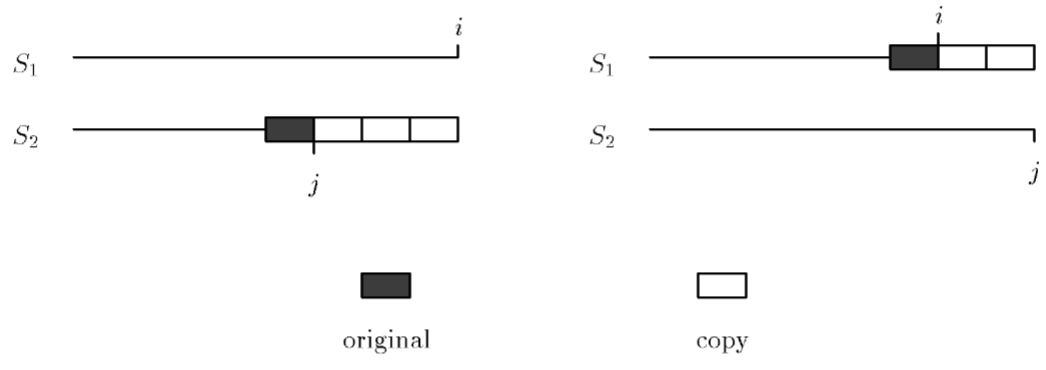


Figure 1:
The duplication option permits score $S[i, j]$ to represent an alignment ending with a duplication of the right end of one sequence aligned with the right end of the other.

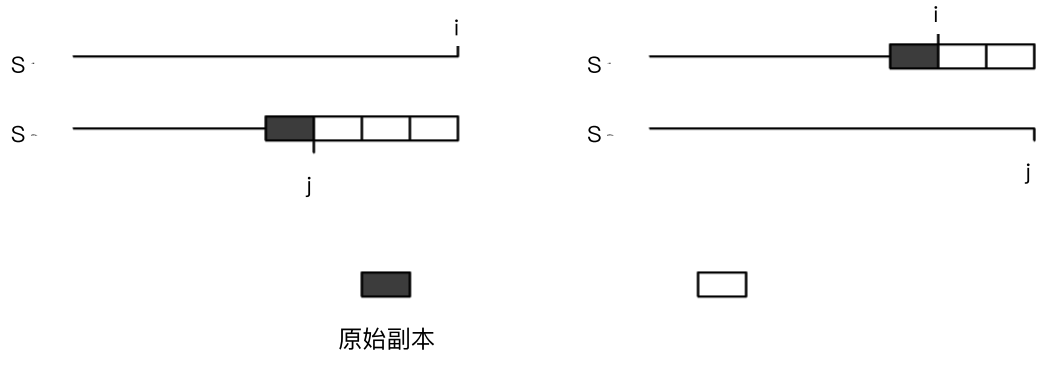


图 1：复制选项允许分数 S[i; j]表示一个以复制一个序列的右端与另一个序列的右端对齐的结束的对齐。

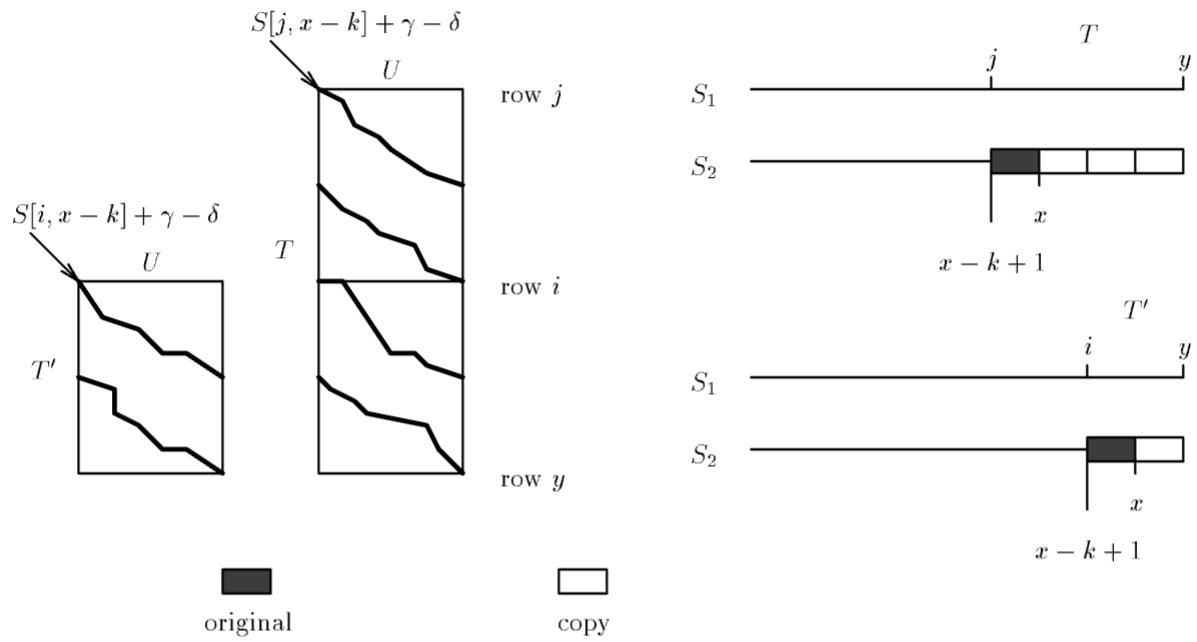


Figure 2:
Calculations for $Dup[y, x]$ for a fixed duplication unit U and two substrings T and T'. The heavy lines in the arrays represent the alignments which are illustrated at the right. The calculations can be merged so that only the best alternative is computed by setting, $\forall i$, $G[i][0]$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

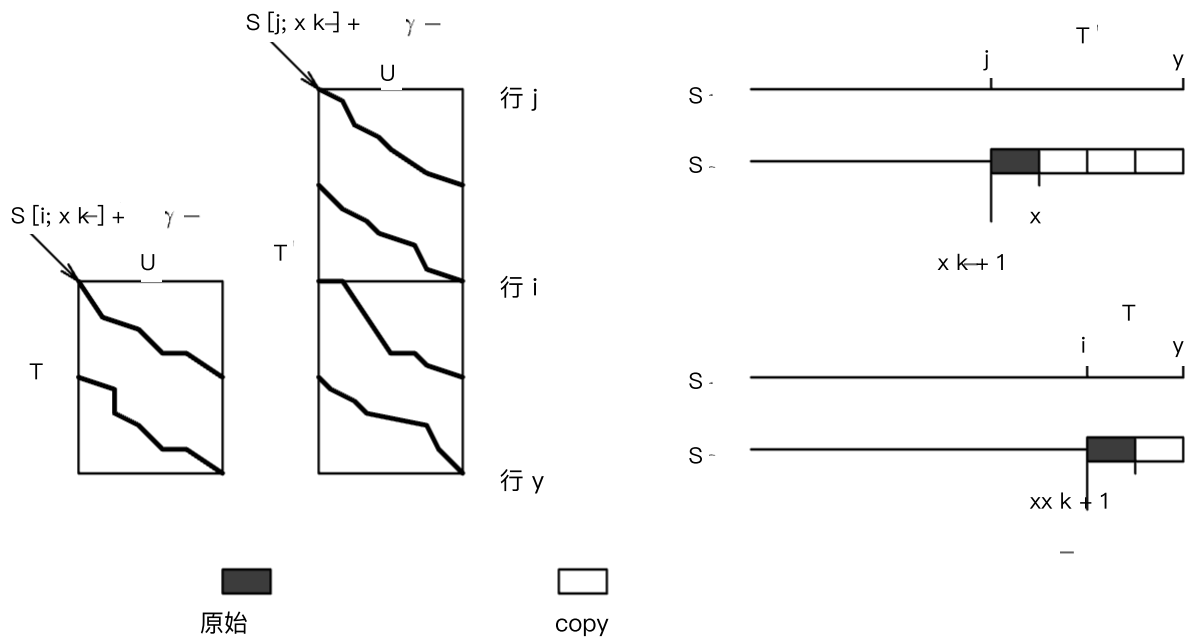


图 2：对于固定复制单元 U 和两个子串 T 和 T 的计算。数组中的粗线表示右侧所示的对齐。这些计算可以合并。

仅通过将 8i，G[i][0]设置为 S[i; x k] + 和 G[i][0]从上方行收到的最佳得分的最大值来计算最佳替代方案。 – γ –

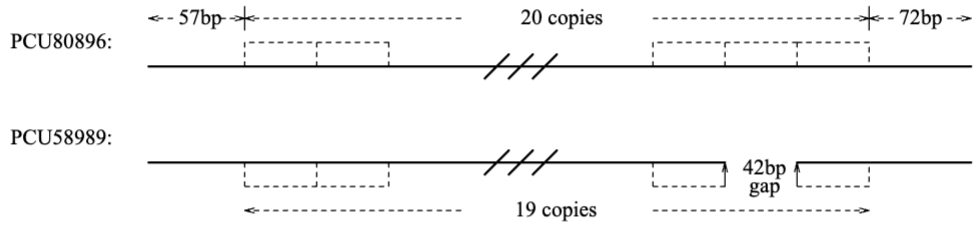


Figure 3:
A schematic of the alignment of two versions of cDNA from the *Plasmodium chabaudi* erythrocyte membrane antigen mRNA. The upper sequence contains 20 copies of a 42 nucleotide pattern and the lower sequence contains 19 copies. An SI model local alignment algorithm was not able to bridge the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

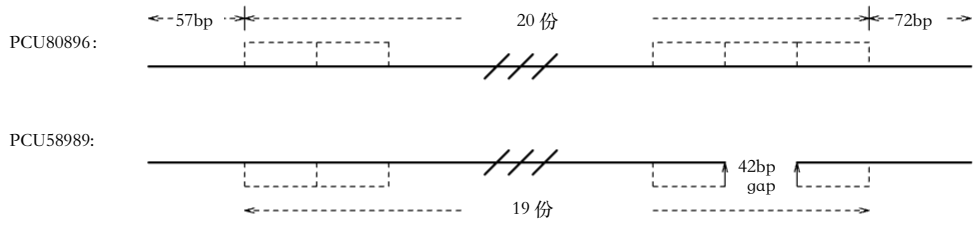


图 3：显示来自恶性疟原虫 chabaudi 红细胞膜抗原 mRNA 的两个 cDNA 版本的排列示意图。上游序列包含 20 个 42 个核苷酸模式的副本，下游序列包含 19 个副本。一个 SI 模型局部排列算法无法通过长插入/删除来填补间隙。

```
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.

对齐：顶部：PCU80896 363 – 1325
底部：PCU58989 1 – 921

∗ ∗ ∗ ∗ 994 CAAATAGAAGAGAGCGATGCTCCTAAAGAAACTCAAGGCGAA 1035 631
CAATTAGAAGAGAACAATGCTCCTAAAGAAACTCAAGCCGAA 672 ∗ ∗

1036 GTAATAGTAGAGAACAATGCTCCTGAAGAAGCTCAAGCCGAA 1077
673 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA 714
1036 GTAATAGTAGAGAACAATGCTCCTGAAGAAGCTCAAGCCGAA 1077
673 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA 714
（注：输入文本为序列号和序列，无需翻译。）

1078 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA 1119
1078 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA 1119
715 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA 756 ∗ ∗ 1120
GAAATAGTAGAGAACAACGCTCCTGAAGAAACTCAAGCCGAA 1161 757 GTAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA 798
<–––––––––––––––––––––––––––––––––

1162 GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA 1203 799
GAAATAGTAGAGAACAATGCTCCTGAAGAAACTCAAGCCGAA
图 4:

疟原虫查伯迪序列对齐。仅显示对齐的左侧，包含最后六个串联重复序列。
fi

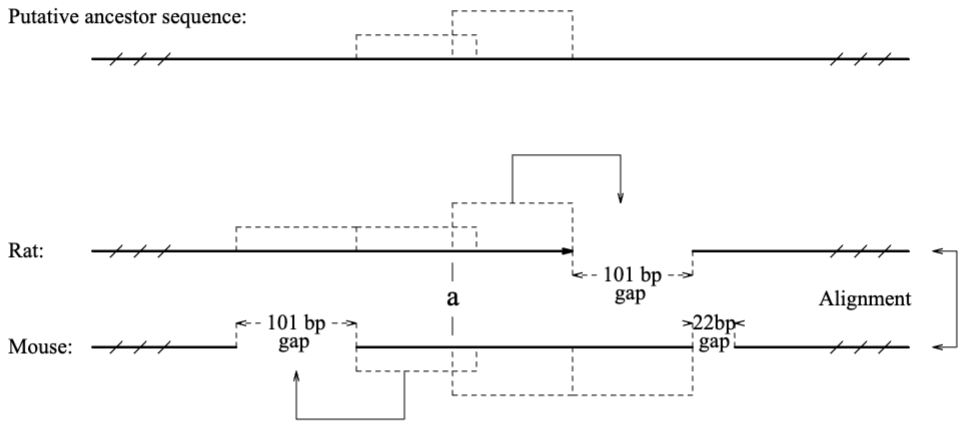


Figure 5:
A schematic of the alignment of an intron region in rat and mouse immunoglobulin ϵ chain genes. The rat contains two copies of one repeat and the mouse contains two copies of another. The two duplicated regions are the same length. The overlap is 20 nucleotides.

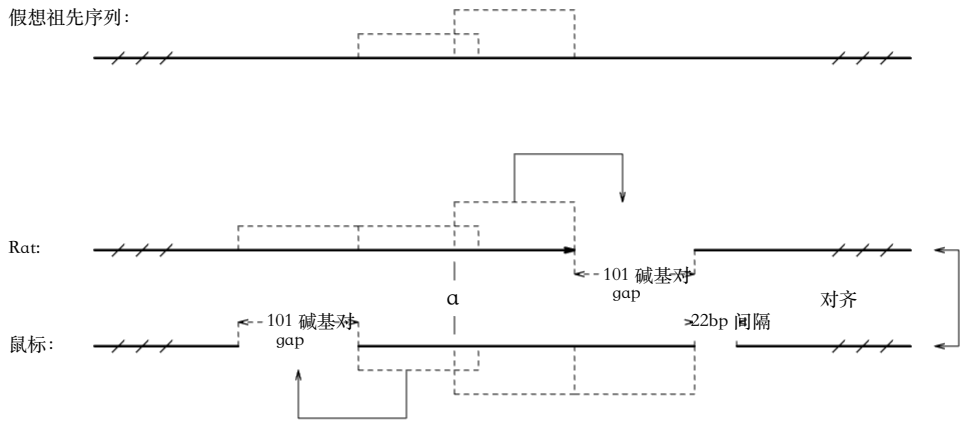


图 5：大鼠和小鼠免疫球蛋白链基因内含子区域对齐示意图。大鼠含有两个重复序列副本，小鼠含有两个不同的重复序列副本。两个重复区域长度相同。重叠区域为 20 个核苷酸。

```
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.

```
对齐：顶部：RATIGCA 3149 – 3578 底部：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
===============>

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

图 6：

大鼠和小鼠免疫球蛋白链基因内含子区域的对齐。