Inverse Parametric Sequence Alignment *

Fangting Sun * Fernández-Baca * Wei Yu **
Abstract

We consider the inverse parametric sequence alignment problem, where a sequence alignment is given and the task is to determine parameter values such that the given alignment is optimal at that parameter setting. We describe a $O(mn \log n)$-time algorithm for inverse global alignment without gap penalty and a $O(mn \log m)$ time algorithm for global alignment with gap penalty, where $m, n \ (n \leq m)$ are the lengths of input strings. We then discuss algorithms for local alignment and multiple sequence alignment.

Key words: Parametric analysis, inverse parametric analysis, global alignment, local alignment, sum-of-pairs multiple alignment, phylogenetic alignment, star alignment

* Research partially supported by grant CCR-9988348 from the National Science Foundation.
* Department of Computer Science, Iowa State University, Ames, IA 50010
**Department of Computer Science, Washington University, St. Louis, MO 63130
Email addresses: ftsun@cs.iastate.edu (Fangting Sun), fernande@cs.iastate.edu (Fernández-Baca), weiyu@ccrc.wustl.edu (Wei Yu).
1 Introduction

High similarity between biomolecular sequences (DNA, RNA, or amino acid sequences), usually implies significant functional or structural similarity. Thus sequence alignment, which can be used for determining similarity between biological sequences, has become essential in modern molecular biology. There are hundreds of papers written on this topic and its applications to biology. The review [1] gives relevant references.

Given two sequences $S$ and $T$ of lengths $n$ and $m$, $n \leq m$, an alignment is a pair $\mathcal{A} = (S', T')$, where $S'$ and $T'$ are equal-length strings, and is obtained by inserting special space characters into $S$ and $T$ respectively. A match is a position where $S'$ and $T'$ have the same characters. A mismatch is a position in which $S'$ and $T'$ have different characters, neither of which is a space. An indel is a position in which one of $S'$ and $T'$ has a space. A gap is a sequence of one or more consecutive spaces in $S'$ or $T'$.

An alignment $\mathcal{A}$ can be characterized by its number of matches, mismatches, indels and gaps, denoted $w, x, y, z$, respectively. In scoring an alignment matches are rewarded, while mismatches, indels and gaps are penalized. Let $\alpha, \beta$ and $\gamma$ denote the mismatch, indel and gap penalties. Then the score of $\mathcal{A}$ is

$$\text{score}_\mathcal{A} = w - \alpha x - \beta y - \gamma z$$
The cost of $\mathcal{A}$ is

$$cost_{\mathcal{A}} = \alpha x + \beta y + \gamma z$$

The case where the weight of the matches is a parameter is ignored since we can divide all the parameters by this value and reduce it to the above case. The optimal alignment problem is to find a maximum-score or minimum-cost alignment $\mathcal{A}$ between two strings. For fixed weights, this problem can be solved in $O(mn)$ time [19]. The problem we have just defined is often referred to as global alignment, so as to distinguish it from its local version, which is defined later (see Section 2).

When using sequence alignment methods to study sequences, there is often considerable disagreement about how to weigh matches, mismatches, indels and gaps. Indeed, the biological significance of the resulting alignment can be greatly affected by the choice of parameter settings. Parametric sequence alignment is a tool that efficiently explores such penalty variation. It avoids the problem of choosing fixed parameter settings by computing the optimal alignment as a function of variable parameters for penalties. The parametric sequence alignment problem is to compute optimal alignments for fixed sequences as a function of varying penalties. The value of an alignment is a linear function of the parameters; thus the parameter space can be partitioned into optimal regions such that in every region one alignment is optimal throughout and the regions are maximal for this property. This approach was proposed by Fitch and Smith [7] at first. Later, both mathematical formu-
lations and algorithms for parametric sequence alignment were obtained by Gusfield et al. [11,12]. Additional work is found in [5,6,22,24,25].

In inverse parametric optimization [4] one is given a parametric optimization problem and a desired optimal solution and the task is to determine parameter settings such that the given solution is optimal for those values. The inverse parametric sequence alignment problem is to find parameter values such that reference alignment is optimal for those values or, if no such settings exist, find a parameter setting minimizing the numerical difference between the score of the optimal alignment and the score of the reference alignment. These parameter values define inverse optimal points on the parameter space. Further, if the reference alignment do be optimal at those points, we call them exact inverse optimal points. Inverse parametric computation is useful for deducing parameter settings where the optimal alignment is likely to reconstruct correct alignments that have been determined by other methods.

One way to locate the “correct” parameter settings is to first construct the entire decomposition of the parameter space and then choose the correct values. Alternatively, one can try to find the parameter settings directly. This can be done by gradient descent [12], although it is not clear how to obtain bounds on the worst-case performance of this method. Megiddo’s method of parametric search [15,16] can be used instead, leading to a $O(m^2n^2)$ method for the case where only one parameter is varied. While powerful, Megiddo’s method has the drawback that it leads to complex algorithms. Improvements
in the running time are possible by relying on the existence of a parallel algorithm for the problem, but this only complicates the results further. Here we give an approach that is much simpler than Megiddo’s method and exploits the integer nature of the scoring of sequence alignments.

Since the optimal regions are bounded by the intersection of hyperplanes, all regions are convex polygons \([7,11,12,6]\). Hence, the inverse optimal parameter setting(s) must occur at a single vertex (intersection point of three or more optimal regions), at a single edge (intersection line between two optimal regions), or at a single complete polygon of the polygonal decomposition of the parameter space. The main idea of our algorithms is to find the inverse-optimal point in the parameter space using binary search. Our main contribution is a proof that this simple algorithm converges quickly.

After considering the pairwise sequence alignment, we study the alignment for three or more sequences. Multiple sequences alignment is just a generalization of pairwise sequence alignment. From a practical point of view, multiple sequence alignment can be significantly more informative than pairwise alignment. However, when three or more sequences are involved, the definitions become more complex and the associated problems become harder to solve \([21]\). For multiple sequences, many objective functions have been studied, including the sum-of-pairs objective \([2,9]\), the maximum-trace objective \([14]\), and objectives defined in terms of an evolutionary tree \([18,26,23]\). Phylogenetic alignment is a problem defined in term of an evolutionary tree, in which
the input is a tree whose leaves are labeled by sequences and the objective is to find a labeling of the internal nodes that minimizes the total length of the tree [18]. Generalized phylogenetic alignment problem can be defined in similar way: the input is a set $S$ of sequences and one must find a sequence-labeled tree of minimum length wherein the elements of $S$ are the labels of the leaves of the tree [13]. We will study sum-of-pairs multiple alignment, phylogenetic alignment and generalized phylogenetic alignment.

The rest of this paper is organized as follows. Section 2 reviews the definitions for different types of alignment and the related parametric properties. Section 3 gives a $O(mn \log n)$ algorithm for global alignment without gap penalty. A $O(mn \log m)$ algorithm for global alignment with gap penalty is described in Section 4. Further results are illustrated in Section 5. Section 6 discusses open problems.

2 Preliminaries

We now give formal definitions for the different alignments we will study and some basic parametric properties related to them.
2.1 Pairwise alignments

In a pairwise alignment, which is defined in Section 1, two sequences are padded by gaps, to achieve same length, and to display maximum similarity on a character-by-character basis. The score of a pairwise alignment $\mathcal{A}$ is decided by the number of matches, mismatches, indels, gaps and scoring scheme. Global scoring scheme and local scoring scheme are two main schemes. In this paper, we only consider alphabet-independent scoring schemes.

2.1.1 Global alignment

In global scoring scheme, the score of an alignment is determined by entire input strings, as defined in Section 1.

Given two sequences $S$ and $T$ of lengths $n$ and $m$, $n \leq m$, let $\mathcal{A}$ be an alignment with $w_A$ matches, $x_A$ mismatches, $y_A$ indels and $z_A$ gaps. Then the score and cost of $\mathcal{A}$ are:

\[
\text{score}_A = w_A - \alpha x_A - \beta y_A - \gamma z_A, \quad \text{cost}_A = \alpha x_A + \beta y_A + \gamma z_A
\]

The global similarity and distance between $S$ and $T$ are defined as:

\[
sim(S, T) = \max\{\text{score}_A : \mathcal{A} \text{ is an alignment of } S \text{ and } T\} \quad (1)
\]

\[
dist(S, T) = \min\{\text{cost}_A : \mathcal{A} \text{ is an alignment of } S \text{ and } T\} \quad (2)
\]
The optimal alignment problem is to compute the similarity or distance between two strings. For fixed weights, this problem can be solved in $O(mn)$ time [19].

The following equation holds [5]:

$$score_A(\alpha, \beta, \gamma) = \frac{n + m}{2} - cost_A(\alpha + 1, \beta + 1/2, \gamma),$$

where $n, m$ are the lengths of the input strings. Therefore,

$$sim(S, T) = \frac{n + m}{2} - dist(S, T).$$

That is, the global similarity and distance between two strings are equivalent. Thus in this paper, we only consider global similarity between strings.

2.1.2 Local alignment

In many applications, two strings may not be highly similar in their entirety but may contain regions that are highly similar. Thus, local similarity is far more meaningful than global similarity. In this case we need to find and extract a pair of regions, one from each of the two given strings, that exhibit high similarity. This is called local alignment.

Given two sequences $S$ and $T$, a local alignment is obtained by finding sub-strings $S'$ and $T'$ of $S$ and $T$, respectively, whose optimal global alignment
score is maximum over all pairs of substrings from $S$ and $T$ [10].

If the lengths of $S$ and $T$ are $n$ and $m$, then the local alignment problem can be solved in $O(mn)$ time for fixed weights of matches, mismatches, indels and gaps [20].

2.1.3 Parametric and inverse-parametric global alignment

For any global alignment $A$ with $w$ matches, $x$ mismatches and $y$ indels, we have

$$2w + 2x + y = n + m$$

(3)

where $n, m$ are the lengths of given sequences. According to equation (3), we have following theorem for global alignment:

**Theorem 2.1 (Gusfield et al. [11])** Any line forming a boundary between two regions on the $(\alpha, \beta)$ plane is of the form $\beta = c + (c + 0.5)\alpha$, for some $c > -1/2$.

In this case, $\gamma = 0$, that is, the gap penalty is zero. Figure 1 illustrates the decomposition on $(\alpha, \beta)$ plane. When the gap penalty is positive, we have:

**Theorem 2.2 (Gusfield et al. [11])** Any line forming a boundary between three or more regions in $(\alpha, \beta, \gamma)$ space is of the form $\beta = c + (c + 1/2)\alpha, \gamma = d + d\alpha$.  

10
Fig. 1. Decomposition of the \((\alpha, \beta)\) plane induced by \(S = tgaactcact, T = ccatgcag\). The corresponding optimum alignments are \(A_1 = \{tgaactcact, ccatgcag\}\), \(A_2 = \{tga - cact, cc - a - tgcag\}\), \(A_3 = \{- - tga - cact, ccatg - c - - ag - -\}\)

From the properties of optimal regions of the decomposition in the search space (see Section 1), the reference alignment can only occur on a single vertex, or single boundary line, or a complete optimal region. An simple example is showed in Figure 2 in which only one parameter \(\alpha\) is counted; that is, the score of alignment is a function of \(\alpha\): \(score = w - \alpha x\). The straight line represents the score of the reference alignment, which goes down while the value of \(\alpha\) increases. The continuous line segments represent the score of the optimal alignments for different values of \(\alpha\). The score of the optimal alignments decreases when the value of \(\alpha\) goes up. Since for different values of \(\alpha\) there are different optimal alignments, the slopes of those line segments are different.

One line segment corresponds to one complete optimal region in the decomposition of parameter space. There are three different cases for this situation. Figure 2(a) shows the situation where the inverse optimal parameter settings occur at a single complete optimal region, Figure 2(b) shows the situation
where the inverse optimal parameter setting occurs at a single vertex, Figure 2(c) shows the situation where the reference alignment cannot be an optimal alignment.

![Graphs showing score vs. reference](image)

Fig. 2. The relationship between optimal alignments and reference alignment

### 2.2 Multiple alignments

Given sequences $S_1, \ldots, S_k$, where $S_i$ has length $n_i$, a **multiple alignment** $\mathcal{A}$ is obtained by inserting spaces in each sequence to get strings with the same length. The result is a matrix with exactly $k$ rows, such that each character of each sequence appears in exactly one column and each column contains at least one character different from space. The **induced pairwise alignment** of $S_i$ and $S_j$ from $\mathcal{A}$ is obtained by removing all rows of $\mathcal{A}$ except those corresponding to $S_i$ and $S_j$ and deleting any columns containing two spaces.

#### 2.2.1 Sum-of-pairs alignment

Given a set of sequences $\mathcal{S} = \{S_1, \ldots, S_k\}$, the object is to find a multiple alignment $\mathcal{A}$ for $\mathcal{S}$ such that $\sum_{i<j} score_{\mathcal{A}(i,j)}$ is maximized, where $score_{\mathcal{A}(i,j)}$ is the
score of the pairwise alignment between \( S_i \) and \( S_j \) induced by \( \mathcal{A} \). Here we use global alphabet-independent scoring scheme. Sum-of-pairs alignment is known to be NP-hard, and it can be solved in \( O(n^k) \) time [3].

Equation (3) holds for each pairwise alignment between \( S_i \) and \( S_j \). Thus, for every multiple alignment \( \mathcal{A} \) we have

\[
\sum_{i<j} (2w_{ij} + 2x_{ij} + y_{ij}) = \sum_{i<j} (n_i + n_j) \tag{4}
\]

Therefore, Theorem 2.1 and 2.2 both hold for sum-of-pairs alignment.

2.2.2 Phylogenetic and Generalized phylogenetic Alignment

A tree \( T \) with a distinct string label (from a set of strings \( \mathcal{S} \)) assigned to each leaf is called a phylogeny. Given a phylogeny \( T \) for a set of sequences \( \mathcal{S} \), a phylogenetic alignment is an assignment of one string label to each internal node of \( T \) which minimizes the total length of the resulting tree. The length of an internally-labeled phylogeny \( T \) is the sum of the pairwise distances between the labels of adjacent nodes. Generalized phylogenetic alignment is to find a minimum-length internally-labeled phylogeny tree for a set of sequences phylogenetic alignment \( \mathcal{S} \). These two problems are NP-hard [18,23].

From [5], we have bounds on the number of matches, mismatches, indels and gaps:
Lemma 2.1 (Fernández-Baca et al. [5])  For any alignment $A$ of a set $S$ of $k$ sequences of length $n$ to a phylogeny with $r$ internal nodes, $w_A + x_A \leq nk r$ and $z_A \leq y_A \leq nk(k + 2r - 1)$.

*Star alignment* is a special case in phylogenetic alignment, where the phylogenetic tree has only one internal node. The bounds for star alignment can be improved as following:

Lemma 2.2 (Fernández-Baca et al. [5])  Let $A$ be an optimal star alignment for a set of $k$ sequences of length $n$ under alphabet-independent distance-based global scoring. Then $y_A, z_A \leq nk$.

3 Global Alignment without Gap Penalty

In this section we consider global alignment where the gap penalties are ignored ($\gamma = 0$). Then, the score function is $score = w - \alpha x - \beta y$. Given a reference alignment $A_0$ with $w_0$ matches, $x_0$ mismatches and $y_0$ indels, we must find an inverse-optimal point for $A_0$ in the $\alpha$,$\beta$ plane. From Theorem 2.1, we have:

**Corollary 3.1** Suppose $(\alpha_0, \beta_0)$ is an inverse-optimal point for reference alignment $A_0$ in the $\alpha, \beta$ space. Then all points on the line that goes through $(-1, -1/2)$ and $(\alpha_0, \beta_0)$ are inverse-optimal for $A_0$.

We need some additional preliminary results:
Lemma 3.1 ([11]) The positive $\beta$-axis intersects all the region boundaries.

Let $A_1, A_2, \ldots, A_k$ be the optimal alignments encountered by $\beta$ axis in order of increasing $\beta$-value. Then $y_{i+1} < y_i$ for all $A_i (i < k)$.

Let $A_i, A_j$ be the optimal alignments in two neighboring optimal regions encountered by the $\beta$ axis, with score $w_i - \alpha x_i - \beta y_i$ and $w_j - \alpha x_j - \beta y_j$. Since the scores of $A_i, A_j$ are equal along the boundary line between the regions, the equation of this line between the regions is:

$$
\beta = \frac{w_i - w_j}{y_i - y_j} + \frac{x_j - x_i}{y_i - y_j} \alpha
$$

(5)

A breakpoint along any given line is a point where the line moves between two adjacent optimal regions. Notice that the coordination of breakpoint along $\beta$-axis is $(w_i - w_j)/(y_i - y_j)$, where $-n \leq w_i - w_j, y_i - y_j \leq n, (w_i - w_j)(y_i - y_j) > 0$ and they are all integers; thus, the largest coordination of them is $n$.

Lemma 3.2 The length of the interval between any two successive breakpoints along the $\beta$-axis is greater than $1/n^2$.

Proof According to Equation (5), the boundary line of two neighboring optimal regions where $A_i, A_j$ are optimal respectively intersects the $\beta$ axis at $(0, \frac{w_i - w_j}{y_i - y_j})$.

Let $A_i, A_j, A_k (i < j < k)$ be the optimal alignments in three consecutive optimal regions when going along the $\beta$ axis and let $\Delta w_1 = w_j - w_k, \Delta w_2 = \cdots$
\[ w_i - w_j, \Delta y_1 = y_j - y_k, \Delta y_2 = y_i - y_j. \] Then the length \( \Delta \beta \) of the interval between two breakpoints on the \( \beta \) axis is:

\[
\Delta \beta = \frac{w_j - w_k}{y_j - y_k} - \frac{w_i - w_j}{y_i - y_j} = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta y_1 \Delta y_2}
\]

Since \( \Delta w_1, \Delta w_2, \Delta y_1, \Delta y_2 \) are all integers and \( \Delta \beta > 0 \), we have \( \Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1 \geq 1 \). Notice that \( m - n \leq y \leq m + n \), then \( \Delta y_1 + \Delta y_2 = y_i - y_k \leq (m + n) - (m - n) \leq 2n \). Therefore \( \Delta y_1 \Delta y_2 \leq ((\Delta y_1 + \Delta y_2)/2)^2 \leq n^2 \). It follows that \( \Delta \beta > 1/n^2 \). \( \square \)

Our algorithm use binary search on the \( \beta \) axis. The details are given in Algorithm 1.

**Theorem 3.1** Algorithm 1 correctly solves the inverse parametric alignment problem for global alignment without gaps in \( O(mn \log n) \) time.

**Proof** From equation (5), all breakpoints on the \( \beta \) axis lie below \((0, n)\), so it is correct to restrict the search space to the portion on the \( \beta \)-axis between \((0, 0)\) to \((0, n)\). Lemma 3.1 guarantees that binary search works for this problem, since the algorithm can decide to go up or down along the \( \beta \) axis according to the number of indels.

If the algorithm finds a point \((0, \beta_0)\) such that the optimal alignment \( \mathcal{A} \) for that point has the same number of indels as the reference alignment, then
Algorithm 1 Global alignment without gap penalties

1:  high = n
2:  low = 0
3:  compute the optimal global alignment \( A_{\text{high}} \) and \( A_{\text{low}} \) at points \((0, \text{high})\) and \((0, \text{low})\) respectively
4:  while \(((\text{high} - \text{low}) > 1/n^2 \) or \( A_{\text{high}} \) is same as \( A_{\text{low}} \)) do
5:      mid = low + (high − low)/2
6:      compute the optimal global alignment \( A_{\text{mid}}(w_{\text{mid}}, x_{\text{mid}}, y_{\text{mid}}) \) at point
7:          \((0, \text{mid})\)
8:      if \( y_{\text{mid}} = y_0 \) then
9:          return the line passing through \((-1, -\frac{1}{2})\) and \((0, \text{mid})\)
10:     else
11:       if \( y > y_0 \) then
12:           low = mid
13:           \( A_{\text{low}} = A_{\text{mid}} \)
14:       else
15:           high = mid
16:           \( A_{\text{high}} = A_{\text{mid}} \)
17:     end if
18: end while
19: mid = low + (high − low)/2
20: if \( A_{\text{high}} \) is the same as \( A_{\text{low}} \) then
21: return the line passing through \((-1, -\frac{1}{2})\) and \((0, \text{mid})\)
22: else
23: compute \( \beta_0 \) such that \( w_{\text{high}} - \beta_0 y_{\text{high}} = w_{\text{low}} - \beta_0 y_{\text{low}} \)
24: return the line passing through \((-1, -\frac{1}{2})\) and \((0, \beta_0)\)
25: end if

\((0, \beta_0)\) is either inverse-optimal \((w = w_0)\) or it is approximately inverse-optimal \((w \neq w_0)\). Following Corollary 3.1, the algorithm returns a line.

Lemma 3.2 shows that when the length of the search interval is smaller than \(1/n^2\), it cannot contain a complete optimal region. It includes either part of one optimal region (\( A_{\text{high}} \) equals \( A_{\text{low}} \)) or one breakpoint. In the first case, all points in the remaining search space are approximately inverse-optimal. In the second case, the breakpoint is inverse-optimal or all points in the remaining search space are approximately inverse-optimal. Thus Algorithm 1 gives the
correct answer.

Steps 1, 3, 5-9, 12 need $O(1)$ time; step 11 and 13-22 need $O(mn)$ time; step 4 needs $O(mn)$ time, and the while statement can loop at most $3 \log n$ times. Therefore, the total time is $3 \log n \cdot O(mn) = O(mn \log n)$. \hfill \qed

4 Global Alignment with Gap Penalty

In this section we solve the inverse global alignment problem with gap penalty. There are now three parameters to consider, $\alpha$, $\beta$ and $\gamma$. Given a reference alignment $A_0$ with $w_0$ matches, $x_0$ mismatches, $y_0$ indels and $z_0$ gaps, we need to find a point on the $\alpha$, $\beta$, $\gamma$ space where $A_0$ is optimal or approximately optimal.

First, let us describe the boundary lines of the optimal regions in the $\alpha$, $\beta$, $\gamma$. From Theorem 2.2, we have

**Corollary 4.1** All region boundaries intersect with either the positive $\beta, \gamma$ coordinate plane or with the positive $\alpha, \gamma$ coordinate plane.

As in Equation (5), the boundary line between optimal regions on the $\beta, \gamma$ plane associated with alignments $A_i$ and $A_j$ has the form

$$\beta = \frac{w_i - w_j}{y_i - y_j} - \frac{z_i - z_j}{y_i - y_j} \gamma$$

(6)
A vertex is the intersection point of three or more optimal regions. Suppose vertex $v$ is intersection point of optimal regions whose optimal alignments are $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$. Let $\Delta w_1 = w_1 - w_2, \Delta w_2 = w_2 - w_3$, etc. Then $v = (\beta_v, \gamma_v)$, where

$$
\beta_v = \frac{\Delta w_1 \Delta z_2 - \Delta w_2 \Delta z_1}{\Delta y_1 \Delta z_2 - \Delta y_2 \Delta z_1} \quad \text{and} \quad \gamma_v = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta z_1 \Delta y_2 - \Delta z_2 \Delta y_1} \quad (7)
$$

Finally, we introduce an important property about the centroid of the convex polyhedra

**Theorem 4.1 (Grunbaum [8])** Let $S$ be a convex body of volume 1 in $\mathbb{R}^d$. Let $v_1$ be the larger of the two volumes in a division of $S$ by a hyperplane through its centroid. Then $v_1 \leq 1 - (\frac{d}{d+1})^d$.

According to Corollary 4.1, if there is an inverse-optimal point on the search space, then there must be an inverse-optimal point on either the positive $\beta, \gamma$ coordinate plane or the positive $\alpha, \gamma$ coordinate plane. Thus we can search on the $\beta, \gamma$ ($\alpha = 0$) coordinate plane first. If an inverse-optimal point is found on the $\beta, \gamma$ coordinate plane, algorithm terminates; otherwise, continue to search on the $\alpha, \gamma$ coordinate plane. If an inverse-optimal point is found on the $\alpha, \gamma$ coordinate plane, then return it. If there is no inverse-optimal point on the $\alpha, \gamma$ coordinate plane either, then return an approximate inverse-optimal point.

The algorithm uses the following idea to reduce the search space. Let $v = (0, \beta_v, \gamma_v)$ be a point in the remaining search space on the $\beta, \gamma$ plane and let
$A_v$ be the optimal alignment at $v$, with $w_v$ matches, $x_v$ mismatches, $y_v$ indels and $z_v$ gaps. If reference alignment $A_0$ is optimal at $v$, then $v$ is an inverse-optimal point. Otherwise, suppose $A_0$ is optimal at point $(0, \beta, \gamma)$. By the optimality of $A_0$ and $A_v$, it follows that:

\[
 w_0 - \beta y_0 - \gamma z_0 \geq w_v - \beta y_v - \gamma z_v \quad \text{and} \quad w_v - \beta y_v - \gamma z_v \geq w_0 - \beta y_0 - \gamma z_0.
\]

Therefore,

\[
 (y_v - y_0)\beta + (z_v - z_0)\gamma \geq (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v \tag{8}
\]

If $A_0$ is not an optimal alignment at any point on the $\beta, \gamma$ plane, then suppose $(0, \beta, \gamma)$ is an approximately inverse-optimal point that minimizes the numerical difference between the score of the optimal alignment and the score of $A_0$, and that $\hat{A}$ is optimal at that point. Thus, we have:

\[
 (w_v - \beta y_v - \gamma z_v) - (w_0 - \beta y_0 - \gamma z_0) \geq (\hat{w} - \beta \hat{y} - \gamma \hat{z}) - (w_0 - \beta y_0 - \gamma z_0)
\]

and

\[
 (\hat{w} - \beta \hat{y} - \gamma \hat{z}) \geq (w_v - \beta y_v - \gamma z_v)
\]

Therefore

\[
 (y_v - y_0)\beta + (z_v - z_0)\gamma \geq (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v,
\]
which is the same as inequality (8).

The boundary line \( l \) of the halfplane defined by (8) passes through \( v \), so \( l \) divides the remaining search space into two regions. The region whose points do not satisfy inequality (8) can be discarded, thereby reducing the search space. Line \( l \) becomes a new boundary line for the remaining search space; we say that this boundary line is \textit{defined by} point \( v \) and alignment \( A_v \). According to the Theorem 4.1, if the \textit{centroid} of the present search space is selected as point \( v \), the area of search space will reduce by a factor of at least \( 4/9 \).

Using the centroid as described above, we repeatedly reduce the search region on the \( \beta, \gamma \) plane until its area is smaller than \( 1/2m^\gamma \). By Theorem 4.2, which is proved later, when the region is reduced to this size, it cannot include a complete optimal region.

After the binary search terminates, if there exists an inverse-optimal point, there must exist an inverse-optimal \textit{vertex} in the remaining search space. From Lemma 4.2, proved later, the inverse-optimal vertex cannot occur on the boundary line of the remaining search space. From Theorem 4.3, also proved below, if there exists an inverse-optimal vertex \((\beta^*, \gamma^*)\) in the remaining search space, there exist two distinct boundary lines \( l_1 \) and \( l_2 \), defined by \((\beta_1, \gamma_1)\) and \( A_1 \) and by \((\beta_2, \gamma_2)\) and \( A_2 \), such that \( A_1, A_2 \) that are optimal at \((\beta^*, \gamma^*)\); that is, \((\beta^*, \gamma^*)\) is the intersection of the scores of \( A_0, A_1, \) and \( A_2 \), and we say \((\beta^*, \gamma^*)\) is \textit{located (determined)} by \( l_1 \) and \( l_2 \) or \( A_1 \) and \( A_2 \). From
Lemma 4.4, if there exists an inverse-optimal vertex in the remaining search space, when $m$ is big, we can use the two longest boundary lines to locate that vertex; when $m$ is small enough, we need check different pairs of boundary lines to locate that inverse-optimal vertex.

The details of the algorithm are shown in Algorithm 2.

To show that Algorithm 2 is correct, we first need to prove some results.

**Lemma 4.1** The distance between any two vertices on the $\beta, \gamma$ plane is greater than $1/m^3$ and the distance between a vertex and a boundary line of optimal regions is greater than $1/m^3$.

**Proof** According to Equation (6) and (7), select a boundary line $l$ of optimal regions and a vertex $v = (\beta_v, \gamma_v)$ as:

$$l : \beta = \frac{\Delta w_i}{\Delta y_i} - \frac{\Delta z_i}{\Delta y_i} \gamma$$

$$\beta_v = \frac{\Delta w_1 \Delta z_2 - \Delta w_2 \Delta z_1}{\Delta y_1 \Delta z_2 - \Delta y_2 \Delta z_1}, \gamma_v = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta z_1 \Delta y_2 - \Delta z_2 \Delta y_1}$$

Then the distance between $v$ and $l$ is:

$$d = \left| \frac{\Delta y_i}{\Delta z_i} \gamma_v - \frac{\Delta w_i}{\Delta y_i} \right| \frac{|\Delta y_i|}{\sqrt{\Delta y_i^2 + \Delta z_i^2}}$$

Since $-m < \Delta w_i, \Delta y_i, \Delta z_i < m$, when $d > 0$, $d > 1/m^3$
Algorithm 2 Global alignment with gap penalties

1: set the search space \( R = \{(0, \beta, \gamma) \mid 0 \leq \beta \leq m^2, 0 \leq \gamma \leq m^2\} \)
2: \( \textbf{while} \ (\text{Area}(R) > 1/2 m^7) \ \textbf{do} \)
3: \( \ \text{compute the centroid } v(\beta_v, \gamma_v) \text{ of search space } R \)
4: \( \ \text{compute the optimal global alignment } \mathcal{A}_v \text{ at point } v \text{ on } \beta, \gamma \text{ space} \)
5: \( \textbf{if} \ (\mathcal{A}_v \text{ is the same as } \mathcal{A}_0) \text{ then} \)
6: \( \ \text{return the line passing through } (-1, -\frac{1}{2}, 0) \text{ and } (0, \beta_v, \gamma_v) \)
7: \( \textbf{else} \)
8: \( \ R \leftarrow R \cap \text{the halfplane defined by Equation (8)} \)
9: \( \textbf{end if} \)
10: \( \textbf{end while} \)
11: \( d = \max \{|w| \mid u, v \text{ are points on the boundary of } R\} \)
12: \( l_1 = \text{the length of the longest boundary line of } R \)
13: \( l_2 = \text{the length of the second longest boundary line of } R \)
14: \( \textbf{if} \ (d < 1/m^3 \text{ or } l_2 \geq 1/m^4 \text{ or } m > 80) \ \textbf{then} \)
15: \( \ \text{alignments } \mathcal{A}_1, \mathcal{A}_2 \text{ define the boundary line } l_1, l_2 \)
16: \( \text{checkvertex}(\mathcal{A}_1, \mathcal{A}_2, R) \)
17: \( \textbf{else} \)
18: \( \textbf{if} \ (l_1 \geq 1/m^4) \ \textbf{then} \)
19: \( \ \text{store all boundary lines of } R \text{ except } l_1 \text{ into stack } S \)
20: \( \ \text{alignment } \mathcal{A}_1 \text{ defines the boundary line } l_1 \)
21: \( \textbf{while} \ (S \text{ is not empty}) \ \textbf{do} \)
22: \( \ \text{pop a boundary line defined by alignment } \mathcal{A}_2 \text{ from } S \)
23: \( \ \text{checkvertex}(\mathcal{A}_1, \mathcal{A}_2, R) \)
24: \( \textbf{end while} \)
25: \( \textbf{else} \)
26: \( \ \text{store all pairs of different boundary lines of } R \text{ into stack } S \)
27: \( \textbf{while} \ (S \text{ is not empty}) \ \textbf{do} \)
28: \( \ \text{pop a pair of boundary lines defined by alignments } \mathcal{A}_1, \mathcal{A}_2 \text{ from } S \)
29: \( \ \text{checkvertex}(\mathcal{A}_1, \mathcal{A}_2, R) \)
30: \( \textbf{end while} \)
31: \( \textbf{end if} \)
32: \( \textbf{end if} \)
33: \( \text{continue search on } \alpha, \gamma \text{ plane} \)

Since the distance between any two vertices should be greater than the distance between a vertex and a boundary line of an optimal region, the distance between any two vertices is greater than \(1/m^3\). \( \Box \)

Theorem 4.2 The area of any complete optimal region on the \( \beta, \gamma \) plane is
**Function 1 checkvertex**($A_1, A_2, R$)

1: compute ($\hat{\beta}, \hat{\gamma}$) so $A_0, A_1, A_2$ have same score at ($\hat{\beta}, \hat{\gamma}$)  
2: if (($\hat{\beta}, \hat{\gamma}$) is in $R$) then  
3: compute the optimal global alignment $\hat{A}$ at point ($\hat{\beta}, \hat{\gamma}$)  
4: if ($\hat{A}$ and $A_0$ have same score at ($\hat{\beta}, \hat{\gamma}$)) then  
5: return the line passing through ($-1, -\frac{1}{2}, 0$) and ($0, \hat{\beta}, \hat{\gamma}$)  
6: end if  
7: end if

greater than $1/2m^6$

**Proof** A complete optimal region is composed of at least three vertices. Suppose the minimal complete optimal region is made with 3 vertices, then it is a triangle. By Lemmas 4.1, the base of this triangle has length greater than $1/m^3$ and the height of this triangle is greater than $1/m^3$. Therefore, the area is greater than $\frac{1}{2} \cdot \frac{1}{m^3} \cdot \frac{1}{m^3} = 1/2m^6$. \hfill $\square$

The precondition for the following results is that the area of the remaining search space is smaller than $\frac{1}{2m^7}$.

**Lemma 4.2** The inverse-optimal vertex cannot be on the boundary of the remaining search space.

**Proof** Assume that the reference alignment is $A_0(w_0, y_0, z_0)$, that the inverse-optimal vertex ($\beta^*, \gamma^*$) is on a boundary line $l$ of the remaining search space, and that the boundary line $l$ is defined by ($\beta_v, \gamma_v$) and alignment $A_v(w_v, y_v, z_v)$. Notice that $A_0$ is not optimal at ($\beta_v, \gamma_v$), then ($\beta^*, \gamma^*$) $\neq$ ($\beta_v, \gamma_v$). Since ($\beta^*, \gamma^*$)
is on $l$, according to Equation (8), we have:

$$(y_v - y_0)\beta^* + (z_v - z_0)\gamma^* = (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v$$  \hfill (*)

By the optimality of $A_0$ and $A_v$, we have:

$$w_0 - \beta^* y_0 - \gamma^* z_0 \geq w_v - \beta^* y_v - \gamma^* z_v \quad \text{and} \quad w_v - \beta_v y_v - \gamma_v z_v > w_0 - \beta_v y_0 - \gamma_v z_0.$$  

Therefore,

$$(y_v - y_0)\beta^* + (z_v - z_0)\gamma^* > (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v$$  \hfill (**)  

Equations (*), (**) are contradictory, so the assumption is wrong. Thus, the inverse-optimal vertex cannot be on any boundary line. \hfill \square

**Lemma 4.3** Suppose reference alignment $A_0(w_0, y_0, z_0)$ is optimal at vertex $(\beta^*, \gamma^*)$ in the remaining search space. Let $l$ be a boundary line of the remaining search space defined by $(\beta_v, \gamma_v)$ and alignment $A_v$. If $A_v$ is not optimal at $(\beta^*, \gamma^*)$, then the distance between $(\beta^*, \gamma^*)$ and $l$ is greater than $1/m^3$.

**Proof** According to inequality (8), the boundary line $l$ is

$$(y_v - y_0)\beta + (z_v - z_0)\gamma = (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v$$ \hfill (*)
Suppose $\mathcal{A}_v$ is not optimal at $(\beta^*, \gamma^*)$. From the optimality of $\mathcal{A}_0$ and $\mathcal{A}_v$, we have:

$$w_0 - \beta^* y_0 - \gamma^* z_0 = w_v - \beta^* y_v - \gamma^* z_v + \Delta c_1, \quad \Delta c_1 > 0 \quad (***)$$

$$w_v - \beta_v y_v - \gamma_v z_v = w_0 - \beta_v y_0 - \gamma_v z_0 + \Delta c_2, \quad \Delta c_2 > 0 \quad (((**))$$

Adding (***) and (((**)) we obtain

$$(y_v - y_0)\beta^* + (z_v - z_0)\gamma^* = (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v + \Delta c_1 + \Delta c_2 \quad (***)$$

Equations (*) and (((**))) define two parallel lines, and (((**))) passes through $(\beta^*, \gamma^*)$. Thus the distance between $(\beta^*, \gamma^*)$ and line $l$ equals the distance between line (((**))) and line $l$. Hence the distance is:

$$d = \frac{|\Delta c_1 + \Delta c_2|}{\sqrt{(y_0 - y_v)^2 + (z_0 - z_v)^2}} = \frac{\Delta c_1 + \Delta c_2}{\sqrt{(y_0 - y_v)^2 + (z_0 - z_v)^2}}$$

According to (**), $\Delta c_1 = w_0 - w_v - (y_0 - y_v)\beta^* - (z_0 - z_v)\gamma^* > 0$

Since $(\beta^*, \gamma^*)$ is a vertex, according to Equation (7) it is given by an expression of the form:

$$\beta^* = \frac{\Delta w_1 \Delta z_2 - \Delta w_2 \Delta z_1}{\Delta y_1 \Delta z_2 - \Delta y_2 \Delta z_1} \quad \text{and} \quad \gamma^* = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta z_1 \Delta y_2 - \Delta z_2 \Delta y_1}$$
It is clear that \( \Delta c_1 > 1/m^2 \), then

\[
d > \left| \frac{1}{m^2} \frac{1}{\sqrt{(y_0 - y_v)^2 + (z_0 - z_v)^2}} \right| > \frac{1}{m^3}
\]

\( \square \)

**Theorem 4.3** If there exists an inverse-optimal vertex \((\beta^*, \gamma^*)\) in the remaining search space, then there exist two boundary lines that are defined by \((\beta_1, \gamma_1)\), where alignment \(A_1\) is optimal, and \((\beta_2, \gamma_2)\), where alignment \(A_2\) is optimal, and alignments \(A_1, A_2\) are optimal at \((\beta^*, \gamma^*)\).

**Proof** Suppose reference alignment \(A_0\) is optimal at vertex \((\beta^*, \gamma^*)\) in the remaining search space.

Assume that no boundary line that is defined by \((\beta_i, \gamma_i)\), alignment \(A_i\) and \(A_i\) is optimal at \((\beta^*, \gamma^*)\). Then from Lemma 4.3, the distance between \((\beta^*, \gamma^*)\) and any point on the boundary of the remaining search space is greater than \(1/m^3\). Hence the area of the remaining search space is greater than \(1/m^6\). But the area of the remaining search space is smaller than \(1/2m^7\), a contradiction. So there exists at least one boundary line that is defined by \((\beta_1, \gamma_1)\), alignment \(A_1\) and \(A_1\) is optimal at \((\beta^*, \gamma^*)\).

If there is only one boundary line that satisfies the above requirements, we can find a contradiction from similar reasoning. Thus there exist at least two
boundary lines that satisfy above requirements.

**Lemma 4.4** If there exists an inverse-optimal vertex in the remaining search space, then when $m > 80$, it can be located using the two longest boundary lines.

**Proof** Suppose that there exists an inverse-optimal vertex in the remaining search space. Notice that if the distance of the two farthest points on the boundary is smaller than $1/m^3$, then any two boundary lines can be used to locate the inverse-optimal vertex (according to Lemma 4.3).

Since the search space can be initially restricted to $0 \leq \beta \leq m^2, 0 \leq \gamma \leq m^2$, the area of remaining search space is smaller than $1/2m^7$, and every iteration reduces the search by at least $4/9$ and increases the number of boundary lines by at most one, there are at most $\log_2 \frac{2}{1/3} + 11 \log_2 m$ boundary lines of the remaining search space.

If the length of a boundary line $l$ is more than $1/m^4$, then $l$ can be used to locate that inverse-optimal vertex. Thus if the lengths of the two longest boundary lines are greater than $1/m^4$, then the inverse-optimal vertex can be located by them. If there are no two boundary lines whose lengths are greater than $1/m^4$, then the distance of the two farthest points on the boundary is smaller than $d = (\log_2 \frac{2}{1/3} + 11 \log_2 m) \cdot \frac{1}{m^3}$. When $m > 80$, $d < \frac{1}{m^3}$, thus the two longest boundary lines can be used to locate the inverse-optimal vertex.
Now we prove the correctness of Algorithm 2 and analyze its running time.

**Theorem 4.4** If there exists an inverse-optimal point on the $\beta, \gamma$ coordinate plane, then Algorithm 2 can find it in $O(mn \log m)$ time.

**Proof** According to Equation 7, the maximum coordinate for a vertex is $(m^2, m^2)$. Thus, we can restrict the search space to $0 \leq \beta \leq m^2, 0 \leq \gamma \leq m^2$. If the algorithm finds an inverse-optimal point $(\beta_v, \gamma_v)$ during the binary search, it returns a line. Lemma 4.3 shows when the area of the remaining search space is smaller than $1/2m^7$, it cannot include a complete optimal region. So binary search terminates and the algorithm begins to check the vertices in the remaining region. If there exists an inverse-optimal vertex in the remaining search space, from Lemma 4.3 and Lemma 4.4, step 11-33 can find it. So if there exists inverse-optimal point on $\beta, \gamma$ coordinate plane, Algorithm 2 can find that point.

The initial area of search space is $m^4$; the binary search terminates when the area is less than $1/2m^7$. Every iteration reduces the area of search space by at least $4/9$ and increases the number of boundary lines by at most 1, so there are $O(\log m)$ iterations and $O(\log m)$ boundary lines. In every iteration, the algorithm computes one optimal alignment which takes $O(mn)$ time. Thus steps 1-10 need $O(mn \log m)$ time. Step 11 needs $O(\log m)$ time, since there
are \(O(\log m)\) intersection points on the boundary and the two farthest points must both be intersection points. Step 12-20 need \(O(mn)\) time. In step 22-32, since \(m < 80\), we can consider the number of different pairs of boundary lines as constant, then the time that step 22-32 need is \(O(mn)\). Thus the total time need by Algorithm 2 is \(O(mn \log m)\). \(\square\)

The above algorithm also works for searching on the \(\alpha, \gamma\) plane, only that, in step 34, when we have found that the reference alignment cannot be optimal on the \(\alpha, \gamma\) coordinate plane, we need to return the centroid of the remaining search space as an approximately inverse-optimal point.

5 Local Alignment and Multiple Alignment

In this section, we apply our algorithms to the inverse parametric local alignment and multiple alignment.

The inverse parametric local alignment without gap penalty \((\gamma = 0)\) problem asks to find an inverse-optimal or approximately inverse-optimal point on the \(\alpha, \beta\) coordinate plane. This problem can be solved by slightly modifying Algorithm 2 of Section 4.

Note that in this case, we search over the \(\alpha, \beta\) coordinate plane. Similar to equations (6),(7),(8), the boundary line between optimal regions on the \(\alpha, \beta\)
plane has the form

\[ \alpha = \frac{w_i - w_j}{x_i - x_j} - \frac{y_i - y_j}{x_i - x_j} \beta, \]  

(9)

a vertex \( v = (\alpha_v, \beta_v) \) on the \( \alpha, \beta \) plane has the form

\[ \alpha_v = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta x_1 \Delta y_2 - \Delta x_2 \Delta y_1} \quad \text{and} \quad \beta_v = \frac{\Delta w_1 \Delta x_2 - \Delta w_2 \Delta x_1}{\Delta y_1 \Delta x_2 - \Delta y_2 \Delta x_1}, \]  

(10)

and the inequality used to reduce the search space has the form

\[ (x_v - x_0)\alpha + (y_v - y_0)\beta \geq (x_v - x_0)\alpha_v + (y_v - y_0)\beta_v \]  

(11)

Since \(-m \leq \Delta w, \Delta x, \Delta y \leq m\), \( \Delta w, \Delta x, \Delta y \) are all integers, Lemma 4.1, 4.2, 4.3, 4.4 and Theorem 4.2, 4.3 all hold for the searching on \( \alpha, \beta \) plane. Thus we can modify Algorithm 2 as follows: search on the \( \alpha, \beta \) plane instead of the \( \beta, \gamma \) plane; computing local alignments instead of global ones. The algorithm analysis in Section 4 implies that this algorithm runs in \( O(mn \log m) \) time.

We now consider alphabet-independent global inverse parametric weighted sum-of-pair multiple alignment. Since Theorem 2.1 and 2.2 hold for this case, we can modify Algorithm 1 and Algorithm 2 to solve this problem. The equation for boundary lines, the coordinate forms for breakpoints and vertices and the inequality used to reduce the search space are all same as the pairwise global alignment case. The difference is that \(-nk \leq \Delta w, \Delta x, \Delta y \leq nk\), where \( k \) is the number of sequences and \( n \) is maximum length of those sequences and
\( \Delta w, \Delta x, \Delta y \) are all integers. Thus the initial search area and termination condition need slight change, therefore the iteration number becomes \( O(\log(nk)) \) instead of \( O(\log n) \). On the other hand, every iteration needs \( O(n^k) \) time. Thus for sum-of-pairs multiple alignment with alphabet-independent global scoring scheme, our algorithm needs \( O(n^k \log(nk)) \) time. Similarly to pairwise local alignment, the local version without gap penalty for sum-of-pair multiple alignment can be solved using our strategy.

The same idea can be applied to phylogenetic alignment. While Theorem 2.1 and 2.2 do not hold for phylogenetic alignment, we can only apply our algorithm to the cases where gap penalties are ignored within alphabet-independent scoring scheme. According to Lemma 2.1, we have 
\[-nkr \leq w_\mathcal{A}, x_\mathcal{A} \leq nkr, -nk(k + 2r - 1) \leq y_\mathcal{A} \leq nk(k + 2r - 1) \] where \( k \) is the number of sequences, \( r \) is the number of internal nodes and \( n \) is the maximum length of those sequences.

Through the same reasoning as for sum-of-pairs alignment, this problem can be solved in \( O(\log(nk^2 + 2nkr)) \) iterations and every iteration needs \( O(n^k) \) time. Then totally, the algorithm needs \( O(n^k \log(nk^2 + 2nkr)) \) time. For the star alignment case, we have 
\[-nk \leq w_\mathcal{A}, x_\mathcal{A}, y_\mathcal{A} \leq nk \] (Lemma 2.2), which implies that \( O(\log(nk)) \) iterations are needed, for a total of \( O(n^k \log(nk)) \) time.
6 Conclusion

We have proposed two efficient algorithms in which binary search strategies are used in one-dimension and two-dimension search space respectively, and we apply the algorithms to different cases.

Many other problems, such as inverse parametric local alignment with gap penalty, inverse parametric alphabet-dependent global alignment, need search in parameter spaces of dimension three or higher. Thus, extending our binary search strategy into fixed-dimensional space becomes an interesting open problem.

Our analyses depend on the integrality of the number of matches, mismatches, indels and gaps. However, most optimization problems, including weighted sum-of-pair multiple alignment, do not have this property. It is an open problem whether our search strategy can be extended to continuous situations.

References


*Proceedings of the 4th Symposium on Combinatorial Pattern Matching*,


[16] N. Megiddo. Applying parallel computation algorithms in the design of serial

alignment under a fixed evolutionary tree. *Discrete Applied Mathematics*,


[19] D. Sankoff and E. J. Kruskal. Time warps, string edits, and macromolecules:
the theory and practice of sequence comparison. *Addison-Wesley*, 1983.


