Two-Piece Gap-Affine Penalties for Partial Order Alignment on Graphs

Mathematical Framework and Algorithmic Design

July 2, 2025

1 Introduction

Partial order alignment (POA) extends traditional sequence alignment to align sequences against directed acyclic graphs (DAGs) representing multiple sequences. The current POASTA implementation uses standard affine gap penalties, which may over-penalize long insertions and deletions. This document presents a mathematical framework for implementing two-piece gap-affine penalties that better model biological gap length distributions.

2 Mathematical Framework

2.1 Standard Affine Gap Model

The current gap penalty function is defined as:

$$g_{\text{affine}}(k) = \begin{cases} 0 & \text{if } k = 0 \\ \alpha + \beta \cdot (k-1) & \text{if } k > 0 \end{cases} \tag{1}$$

where α is the gap opening penalty, β is the gap extension penalty, and k is the gap length.

2.2 Two-Piece Gap-Affine Model

The two-piece gap-affine model introduces a breakpoint k_0 to distinguish between short and long gaps:

$$g_{\text{two-piece}}(k) = \begin{cases} 0 & \text{if } k = 0 \\ \alpha + \beta_1 \cdot (k-1) & \text{if } 1 \leq k \leq k_0 \\ \alpha + \beta_1 \cdot (k_0 - 1) + \beta_2 \cdot (k - k_0) & \text{if } k > k_0 \end{cases} \tag{2}$$

where:

$$\alpha = \text{gap opening penalty} \tag{3}$$
$$\beta_1 = \text{gap extension penalty for short gaps} \tag{4}$$
$$\beta_2 = \text{gap extension penalty for long gaps} \tag{5}$$
$$k_0 = \text{breakpoint length} \tag{6}$$

Typically, $\beta_2 < \beta_1$ to reduce over-penalization of long gaps.

2.3 Biological Justification

Empirical studies of protein structural alignments show that gap length distributions exhibit bilinear behavior in log-space:

$$\log P(\text{gap length} = k) = \begin{cases} a_1 + b_1 \cdot k & \text{if } k \leq 3 \\ a_2 + b_2 \cdot k & \text{if } k > 3 \end{cases} \tag{7}$$

where $|b_2| < |b_1|$, indicating that long gaps ($k > 3$) are relatively less penalized than expected from a single exponential distribution.

3 Graph Alignment Formulation

3.1 Problem Definition

Let $G = (V, E)$ be a directed acyclic graph representing a partial order alignment, and let $q = q_1 q_2 ... q_n$ be a query sequence. The goal is to find an optimal alignment path through G that aligns q to the graph.

3.2 State Space

Define the state space as:

$$\mathcal{S} = \{(i, v, s, l) : i \in [0, n], v \in V, s \in \Sigma, l \in \mathbb{N}\} \tag{8}$$

where:

$$i = \text{position in query sequence} \tag{9}$$
$$v = \text{current node in graph} \tag{10}$$
$$s = \text{alignment state} \in \{\text{M}, \text{I}_1, \text{I}_2, \text{D}_1, \text{D}_2\} \tag{11}$$
$$l = \text{current gap length} \tag{12}$$

The alignment states are:

$$\text{M} = \text{match/mismatch state} \tag{13}$$
$$\text{I}_1 = \text{insertion state, short gap} \tag{14}$$
$$\text{I}_2 = \text{insertion state, long gap} \tag{15}$$
$$\text{D}_1 = \text{deletion state, short gap} \tag{16}$$
$$\text{D}_2 = \text{deletion state, long gap} \tag{17}$$

3.3 Dynamic Programming Recurrence Relations

Let $\text{OPT}(i, v, s, l)$ denote the optimal alignment score for state (i, v, s, l).

3.3.1 Match State

$$\begin{aligned}\mathrm{OPT}(i, v, \mathrm{M}, 0) = \min_{u \in \mathrm{pred}(v)} \{&\mathrm{OPT}(i-1, u, \mathrm{M}, 0) + \sigma(q_i, v), && (18)\\ &\min_{l'} \mathrm{OPT}(i-1, u, \mathrm{I}_1, l') + \sigma(q_i, v), && (19)\\ &\min_{l'} \mathrm{OPT}(i-1, u, \mathrm{I}_2, l') + \sigma(q_i, v), && (20)\\ &\min_{l'} \mathrm{OPT}(i-1, u, \mathrm{D}_1, l') + \sigma(q_i, v), && (21)\\ &\min_{l'} \mathrm{OPT}(i-1, u, \mathrm{D}_2, l') + \sigma(q_i, v)\} && (22)\end{aligned}$$

where $\sigma(q_i, v)$ is the substitution score for aligning query character q_i with graph node v.

3.3.2 Short Insertion States

$$\mathrm{OPT}(i, v, \mathrm{I}_1, 1) = \mathrm{OPT}(i-1, v, \mathrm{M}, 0) + \alpha + \beta_1 \cdot 0 \quad (23)$$

$$\mathrm{OPT}(i, v, \mathrm{I}_1, l) = \mathrm{OPT}(i-1, v, \mathrm{I}_1, l-1) + \beta_1 \quad \text{for } 1 < l \leq k_0 \quad (24)$$

3.3.3 Long Insertion States

$$\mathrm{OPT}(i, v, \mathrm{I}_2, k_0+1) = \mathrm{OPT}(i-1, v, \mathrm{I}_1, k_0) + \beta_2 \quad (25)$$

$$\mathrm{OPT}(i, v, \mathrm{I}_2, l) = \mathrm{OPT}(i-1, v, \mathrm{I}_2, l-1) + \beta_2 \quad \text{for } l > k_0 + 1 \quad (26)$$

3.3.4 Short Deletion States

$$\mathrm{OPT}(i, v, \mathrm{D}_1, 1) = \min_{u \in \mathrm{pred}(v)} \mathrm{OPT}(i, u, \mathrm{M}, 0) + \alpha + \beta_1 \cdot 0 \quad (27)$$

$$\mathrm{OPT}(i, v, \mathrm{D}_1, l) = \min_{u \in \mathrm{pred}(v)} \mathrm{OPT}(i, u, \mathrm{D}_1, l-1) + \beta_1 \quad \text{for } 1 < l \leq k_0 \quad (28)$$

3.3.5 Long Deletion States

$$\mathrm{OPT}(i, v, \mathrm{D}_2, k_0+1) = \min_{u \in \mathrm{pred}(v)} \mathrm{OPT}(i, u, \mathrm{D}_1, k_0) + \beta_2 \quad (29)$$

$$\mathrm{OPT}(i, v, \mathrm{D}_2, l) = \min_{u \in \mathrm{pred}(v)} \mathrm{OPT}(i, u, \mathrm{D}_2, l-1) + \beta_2 \quad \text{for } l > k_0 + 1 \quad (30)$$

4 A* Algorithm Adaptation

4.1 Heuristic Function

The admissible heuristic function for the remaining alignment cost is:

$$h(i, v, s, l) = h_{\mathrm{gap}}(n - i) + h_{\mathrm{sub}}(i, v) \quad (31)$$

where $h_{\mathrm{gap}}(r)$ is the minimum gap cost for r remaining characters:

$$h_{\mathrm{gap}}(r) = \begin{cases} 0 & \text{if } r = 0 \\ \alpha + \beta_1 \cdot (r-1) & \text{if } r \leq k_0 \\ \alpha + \beta_1 \cdot (k_0 - 1) + \beta_2 \cdot (r - k_0) & \text{if } r > k_0 \end{cases} \quad (32)$$

and $h_{\mathrm{sub}}(i, v)$ is the minimum substitution cost from node v to the end of the graph.

4.2 Priority Function

The A* priority function is:

$$f(i, v, s, l) = g(i, v, s, l) + h(i, v, s, l) \tag{33}$$

where $g(i, v, s, l)$ is the actual cost to reach state (i, v, s, l).

5 Complexity Analysis

5.1 Time Complexity

The time complexity of the two-piece gap-affine algorithm is:

$$O(|V| \cdot |E| \cdot n \cdot k_{\max}) \tag{34}$$

where $k_{\max}$ is the maximum gap length considered. This represents a factor of $k_{\max}$ increase over the standard affine model.

5.2 Space Complexity

The space complexity is:

$$O(|V| \cdot n \cdot k_{\max}) \tag{35}$$

The additional space requirement can be managed through:

- Limiting $k_{\max}$ to biologically reasonable values (e.g., 20-50)
- Using sparse representations for gap length states
- Implementing state pruning strategies

6 Implementation Considerations

6.1 Parameter Selection

Based on empirical studies, recommended parameters are:

$$k_0 = 3 \quad \text{(breakpoint at length 3)} \tag{36}$$
$$\beta_1 = 2 \quad \text{(standard extension penalty)} \tag{37}$$
$$\beta_2 = 1 \quad \text{(reduced penalty for long gaps)} \tag{38}$$
$$\alpha = 6 \quad \text{(gap opening penalty)} \tag{39}$$

6.2 Memory Optimization

To manage memory requirements:

1. Use blocked storage with separate blocks for different gap length ranges
2. Implement lazy state creation
3. Employ bit-packing for small gap lengths
4. Use hash maps for sparse long gap states

7 Conclusion

The two-piece gap-affine penalty model provides a more biologically realistic approach to gap scoring in partial order alignment. While it increases computational complexity by a factor of $k_{\max}$, careful implementation with appropriate parameter selection and optimization strategies can make it practical for real-world applications.

The mathematical framework presented here provides a solid foundation for implementing this model in the POASTA system, with clear algorithmic specifications and complexity analysis to guide the implementation process.