Language-Agnostic Subtitle Synchronization

Kaegi

October 1, 2019

Contents

Introduction

Motivation

- understanding quiet, fast speech
- foreign movies
- language learning

Introduction

Motivation

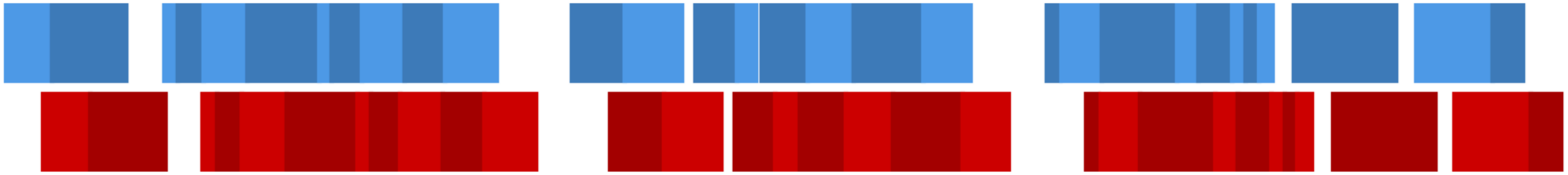
- understanding quiet, fast speech
- foreign movies
- language learning

Subtitles online often badly synchronized!

Error patterns in subtitle synchronization

Error patterns in subtitle synchronization

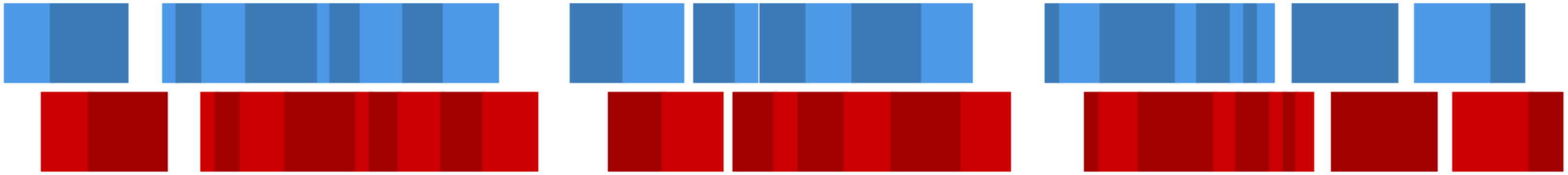
- Constant Shift



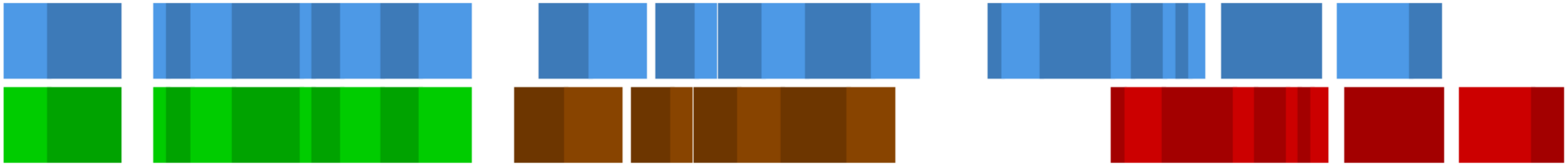
Introduction

Error patterns in subtitle synchronization

- Constant Shift

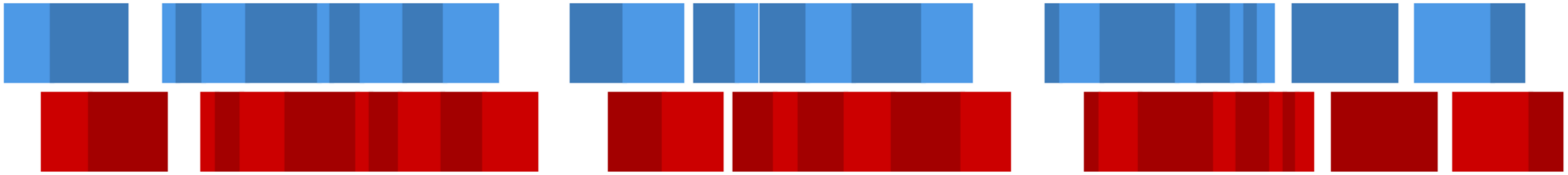


- Splits

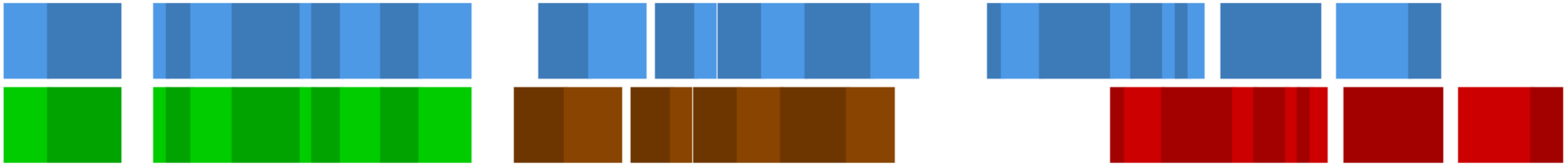


Error patterns in subtitle synchronization

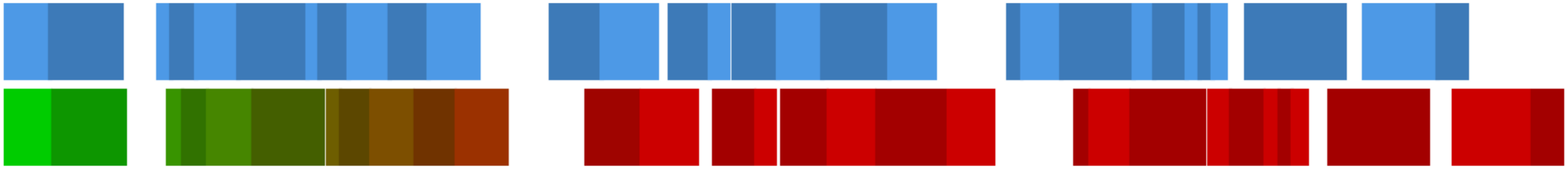
- Constant Shift



- Splits



- Framerate Differences



Introduction

Process overview

1. Extract audio from video.

Process overview

1. Extract audio from video.
2. Perform voice-activity-detection on audio.

Introduction

Process overview

1. Extract audio from video.
2. Perform voice-activity-detection on audio.
3. Extract intervals from input subtitle.

Process overview

1. Extract audio from video.
2. Perform voice-activity-detection on audio.
3. Extract intervals from input subtitle.
4. Align subtitle intervals to speech intervals.

Voice-Activity-Detection

WebRTC voice-activity-detection:

1. Calculate energies on 6 sub-bands 80Hz-250Hz, ..., 3000Hz-4000Hz for 10ms

Voice-Activity-Detection

WebRTC voice-activity-detection:

1. Calculate energies on 6 sub-bands 80Hz-250Hz, ..., 3000Hz-4000Hz for 10ms
2. Calculate probabilities of speech and non-speech using Gaussian distributions of energies

Voice-Activity-Detection

WebRTC voice-activity-detection:

1. Calculate energies on 6 sub-bands 80Hz-250Hz, ..., 3000Hz-4000Hz for 10ms
2. Calculate probabilities of speech and non-speech using Gaussian distributions of energies
3. Weight probabilities on sub-bands

Voice-Activity-Detection

WebRTC voice-activity-detection:

1. Calculate energies on 6 sub-bands 80Hz-250Hz, ..., 3000Hz-4000Hz for 10ms
2. Calculate probabilities of speech and non-speech using Gaussian distributions of energies
3. Weight probabilities on sub-bands
4. Compare with threshold

Optimal no-split alignment

Basic definitions

Given two time spans $a = [a_1, a_2)$ and $b = [b_1, b_2)$:

Optimal no-split alignment

Basic definitions

Given two time spans $a = [a_1, a_2)$ and $b = [b_1, b_2)$:

$$\begin{aligned} \texttt{start}(a) &= a_1 \\ \texttt{end}(a) &= a_2 \end{aligned}$$

Optimal no-split alignment

Basic definitions

Given two time spans $a = [a_1, a_2)$ and $b = [b_1, b_2)$:

$$\begin{aligned} \texttt{start}(a) &= a_1 \\ \texttt{end}(a) &= a_2 \\ \texttt{length}(a) &= a_2 - a_1 \end{aligned}$$

Optimal no-split alignment

Basic definitions

Given two time spans $a = [a_1, a_2)$ and $b = [b_1, b_2)$:

$$\begin{aligned} \texttt{start}(a) &= a_1 \\ \texttt{end}(a) &= a_2 \\ \texttt{length}(a) &= a_2 - a_1 \\ \texttt{overlap}(a, b) &= \texttt{length}(a \cap b) \end{aligned}$$

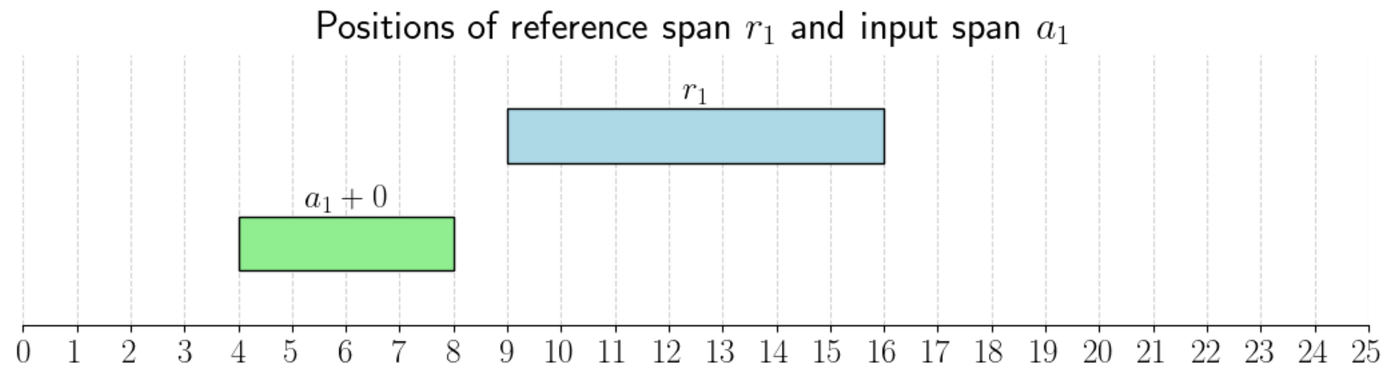
Optimal no-split alignment

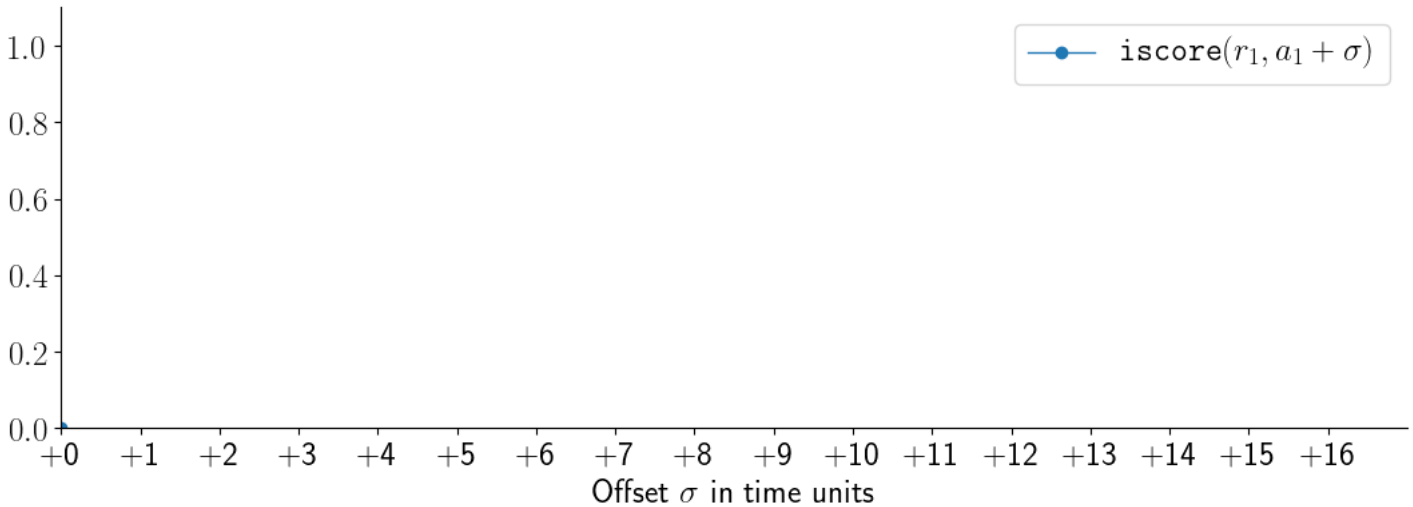
Basic definitions

Given two time spans $a = [a_1, a_2)$ and $b = [b_1, b_2)$:

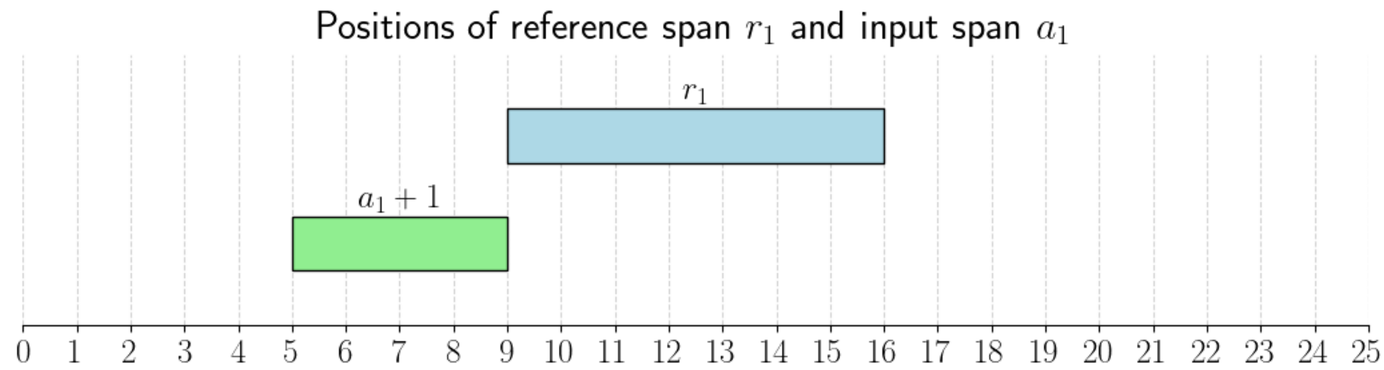
$$\begin{aligned}
\texttt{start}(a) &= a_1 \\
\texttt{end}(a) &= a_2 \\
\texttt{length}(a) &= a_2 - a_1 \\
\texttt{overlap}(a, b) &= \texttt{length}(a \cap b) \\
\texttt{iscore}(a, b) &= \frac{\texttt{overlap}(a, b)}{\texttt{min}(\texttt{length}(a), \texttt{length}(b))}
\end{aligned}$$

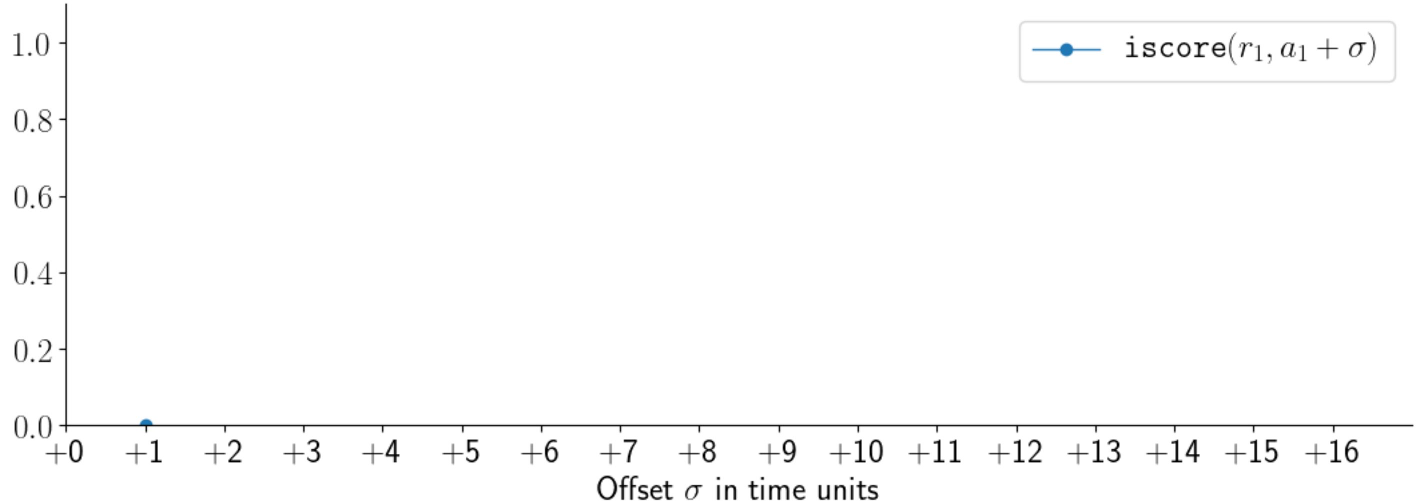
Optimal no-split alignment



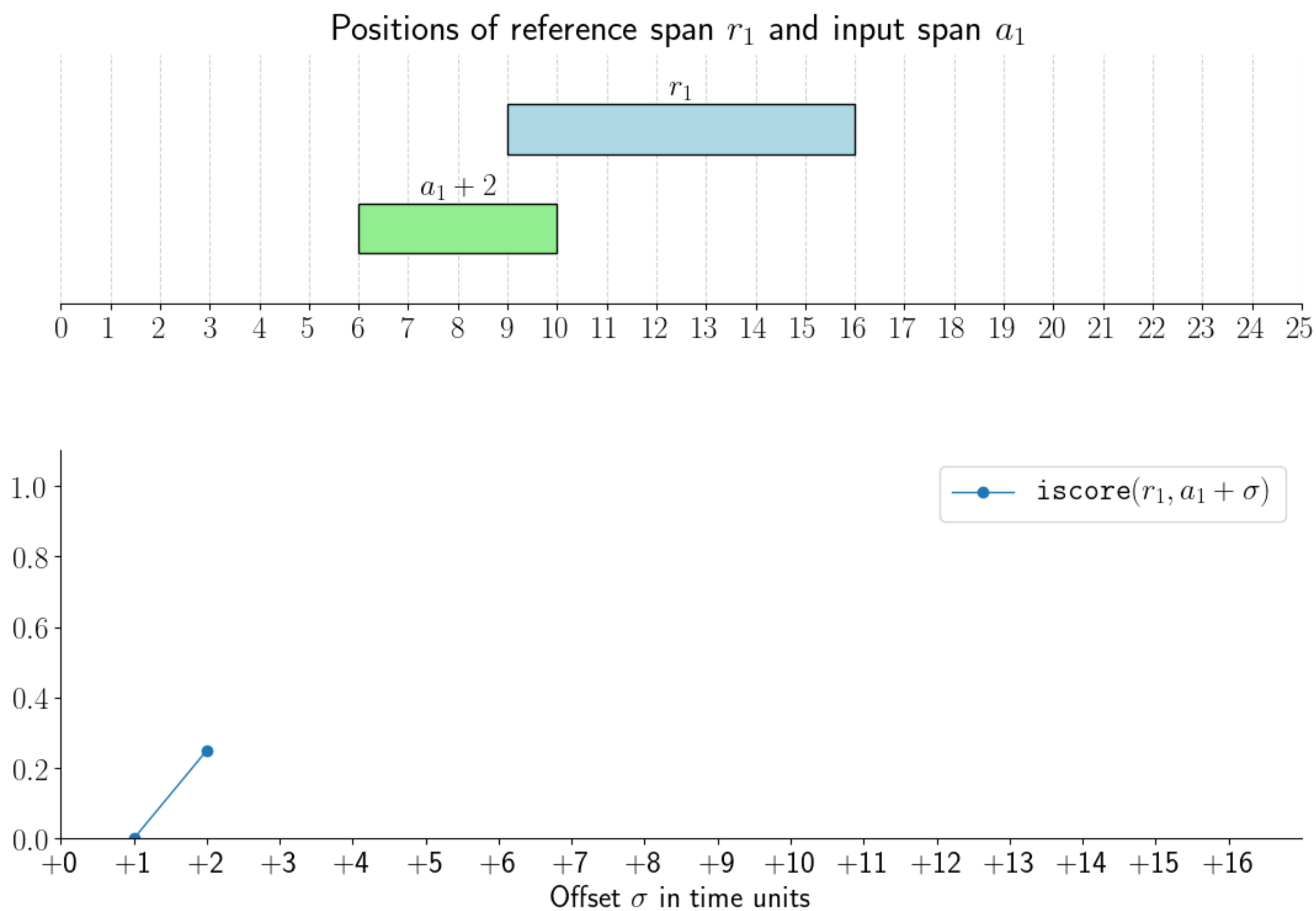


Optimal no-split alignment

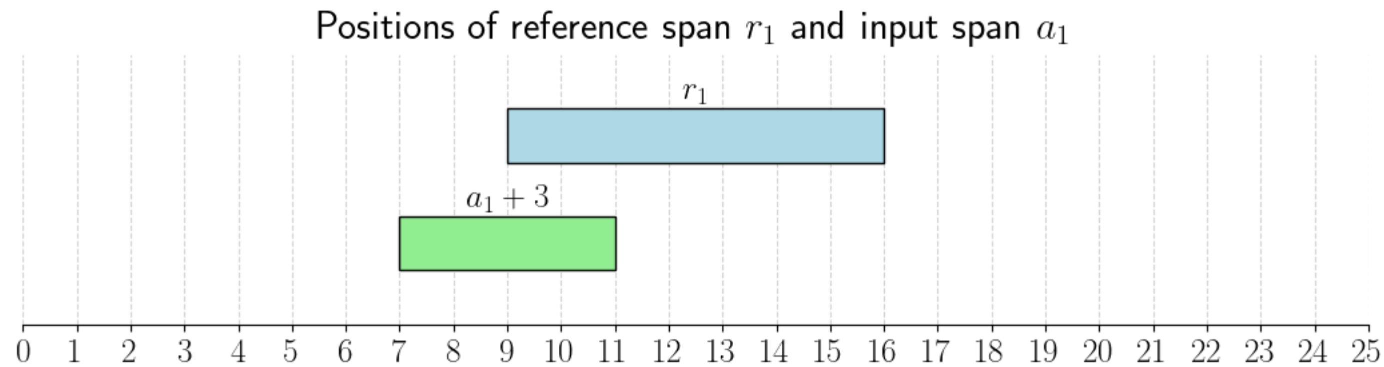


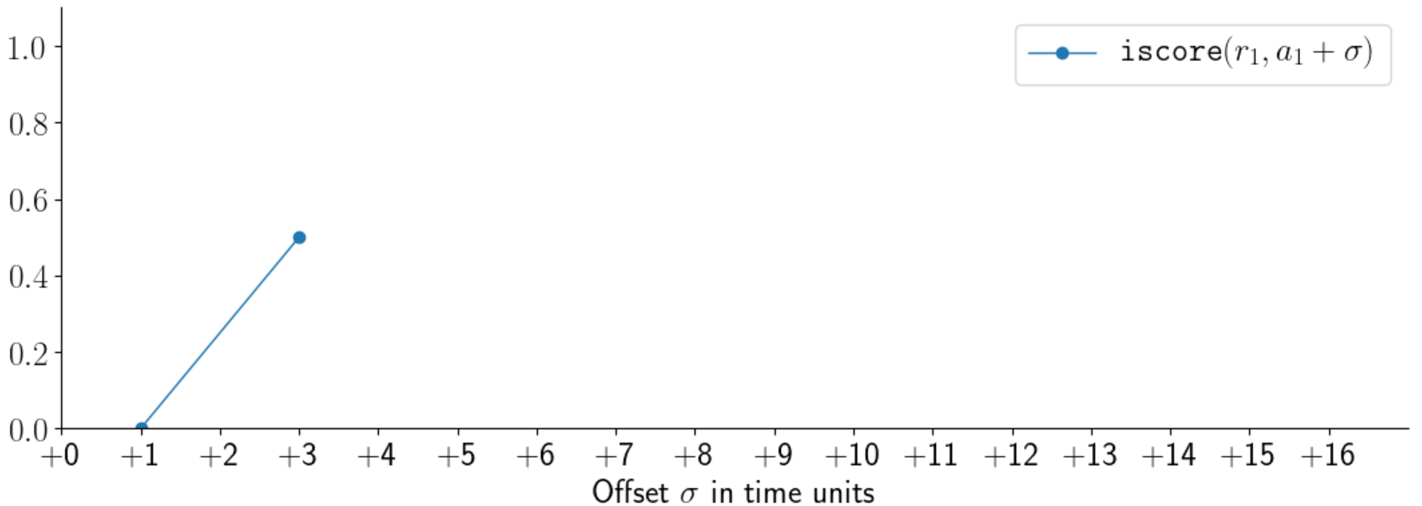


Optimal no-split alignment

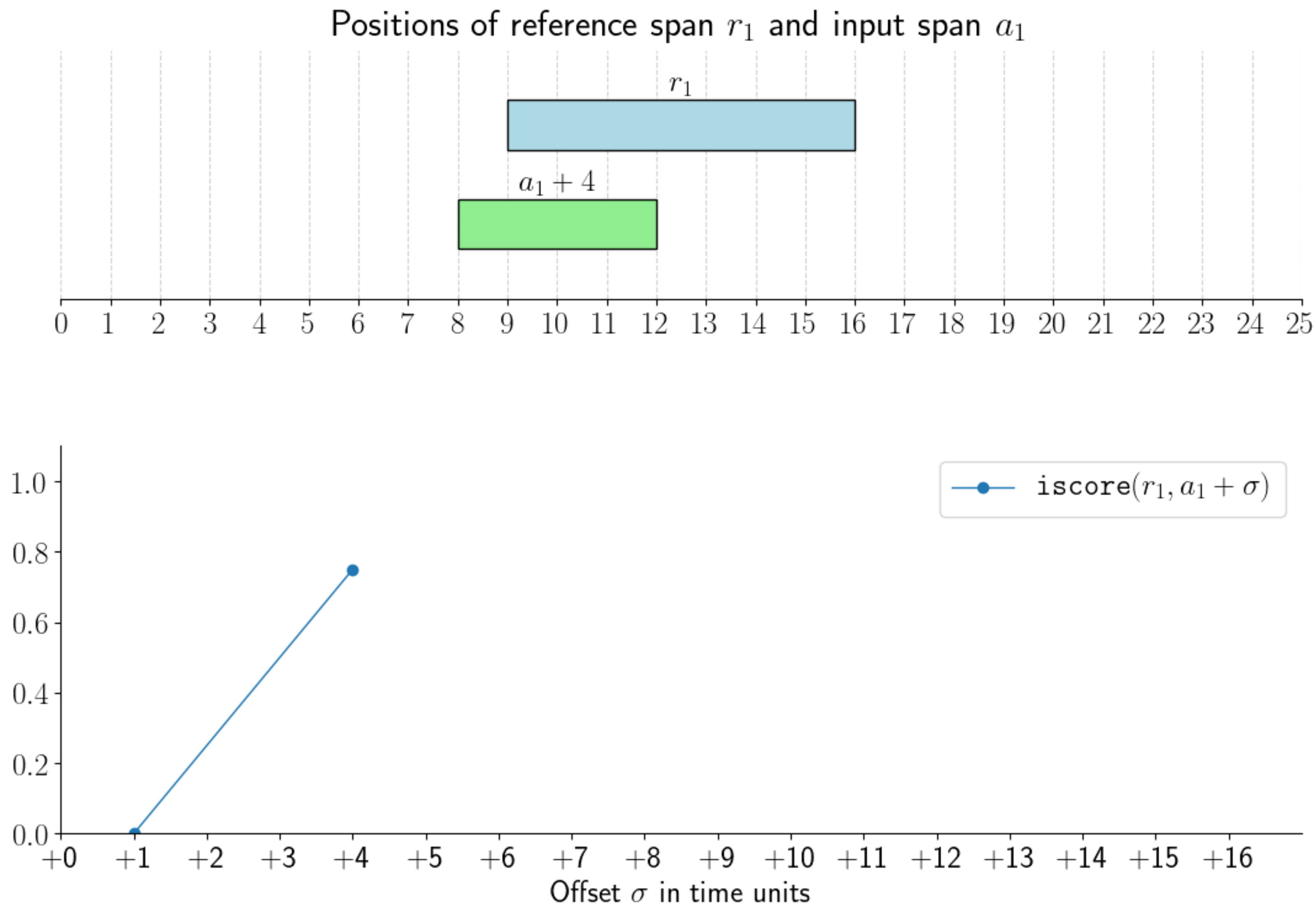


Optimal no-split alignment

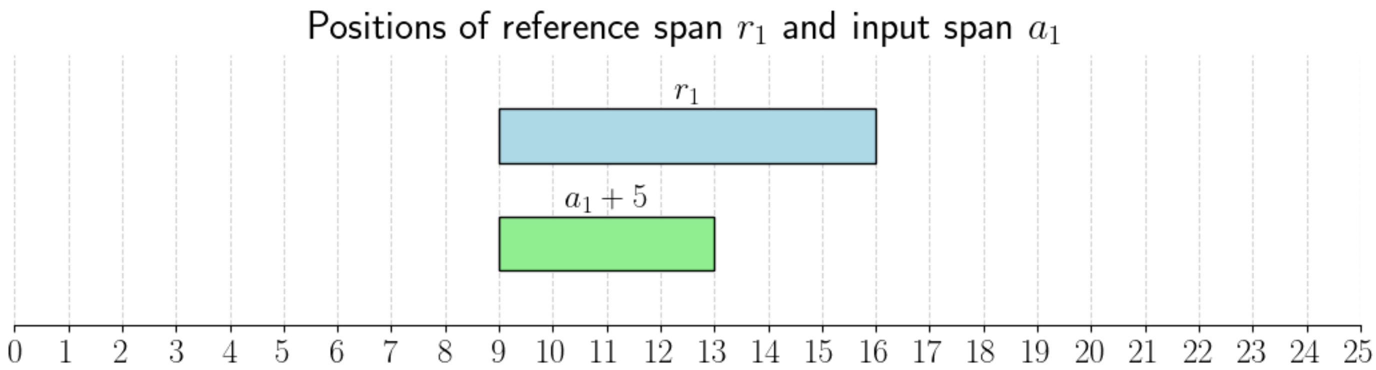


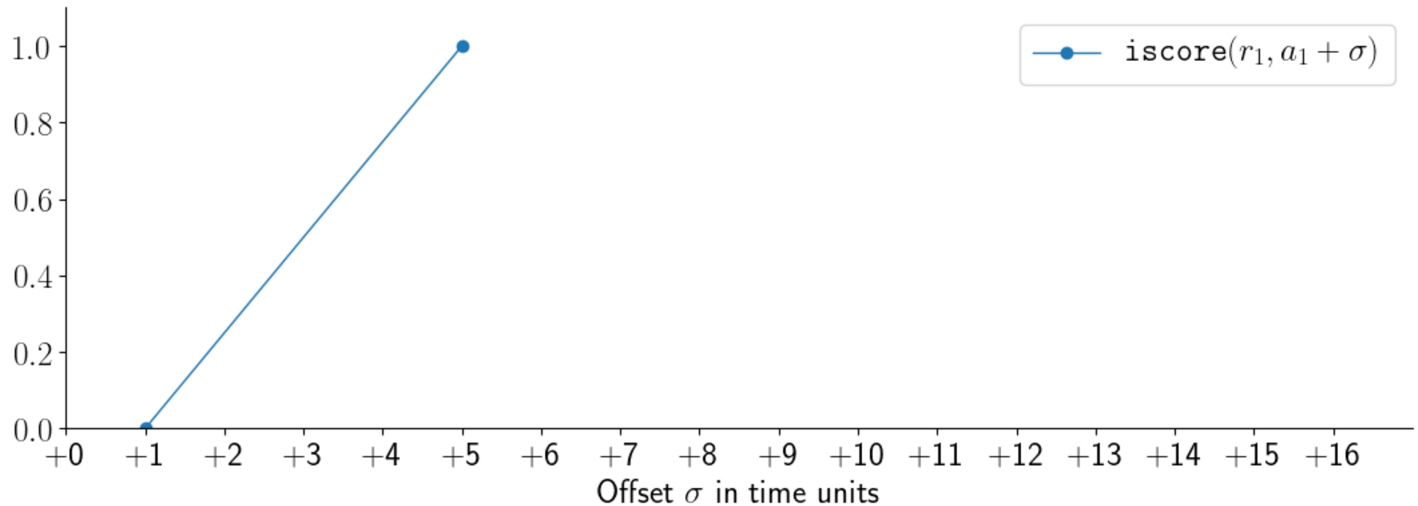


Optimal no-split alignment

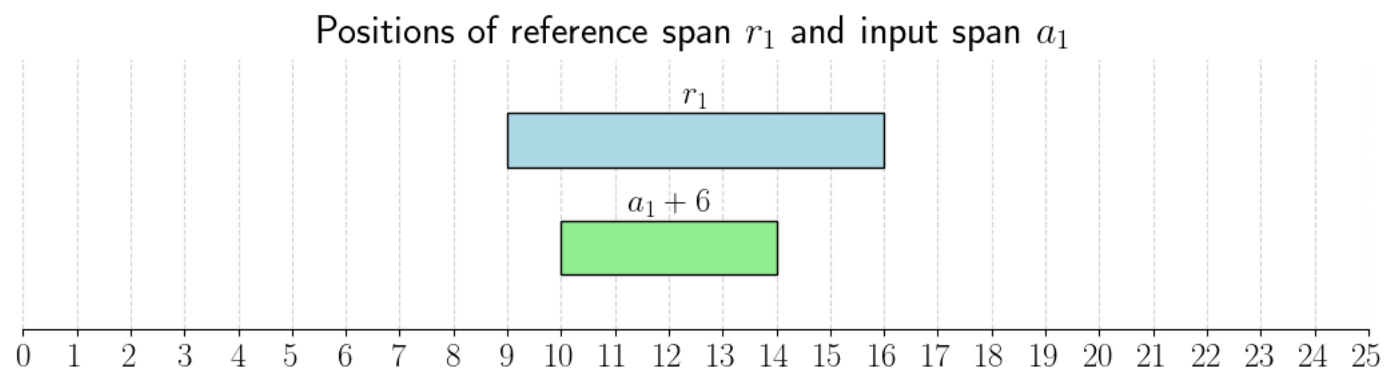


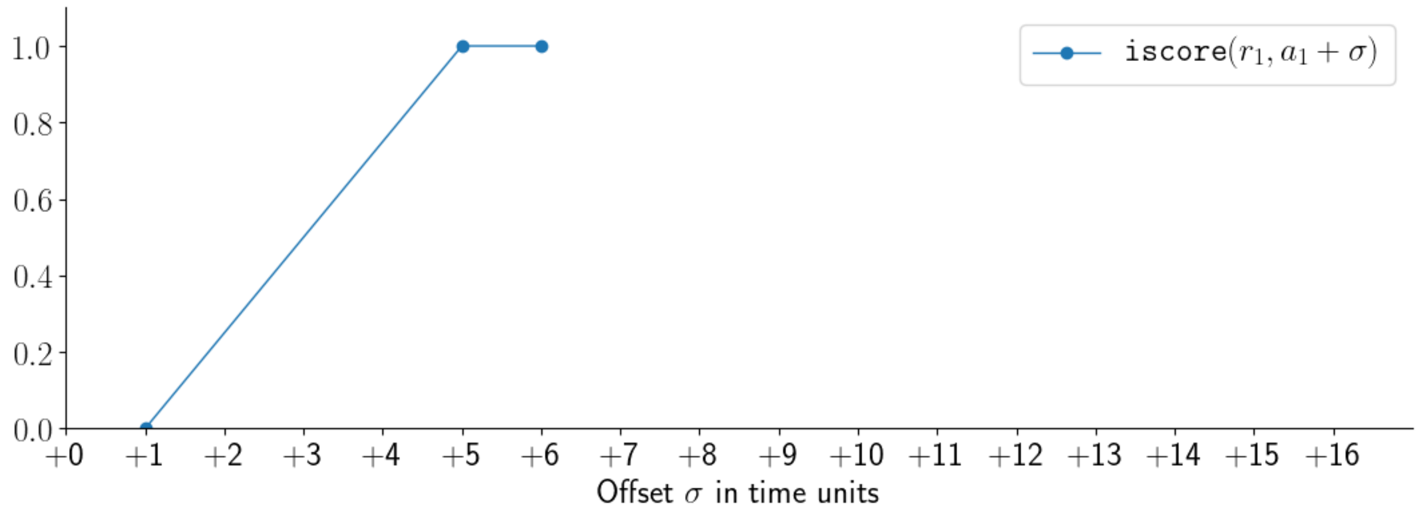
Optimal no-split alignment



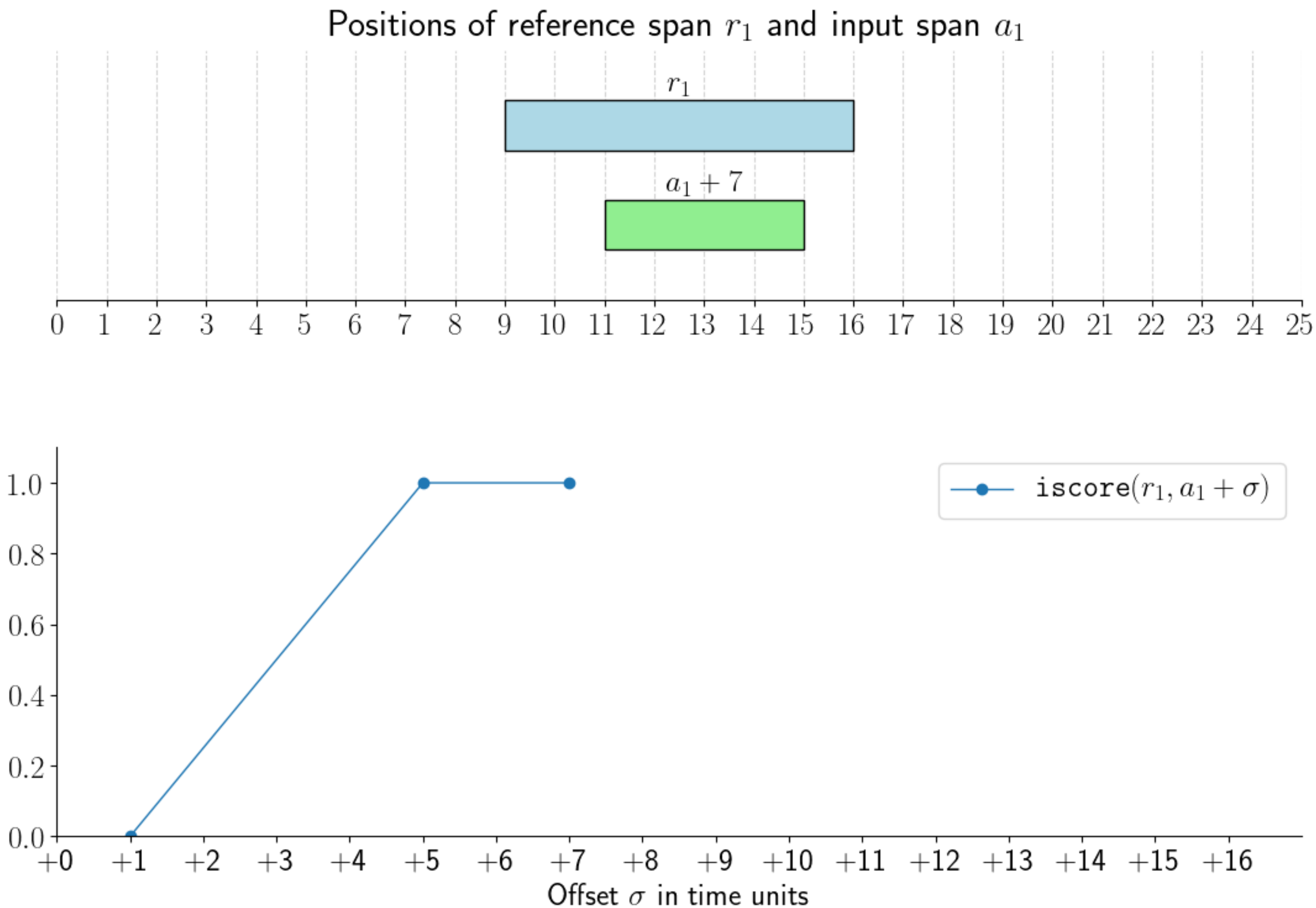


Optimal no-split alignment

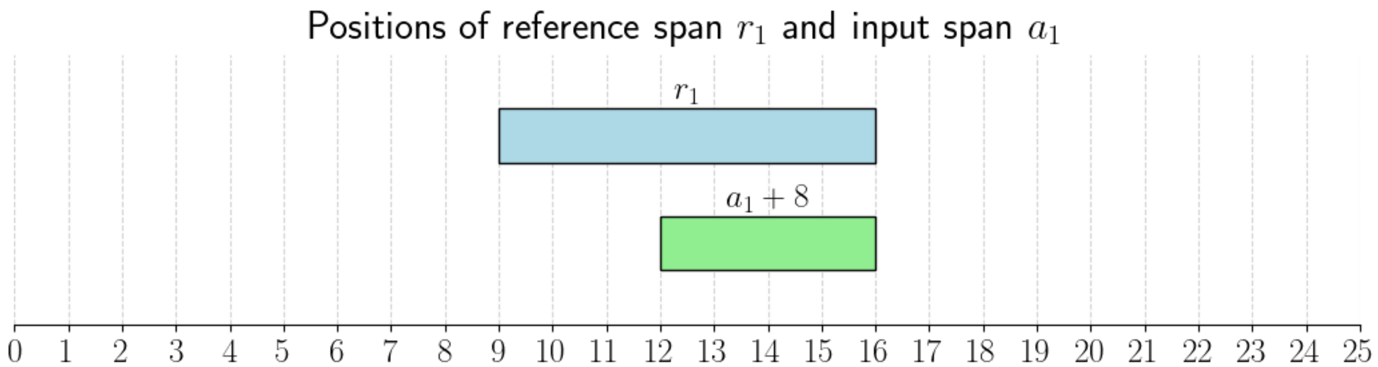


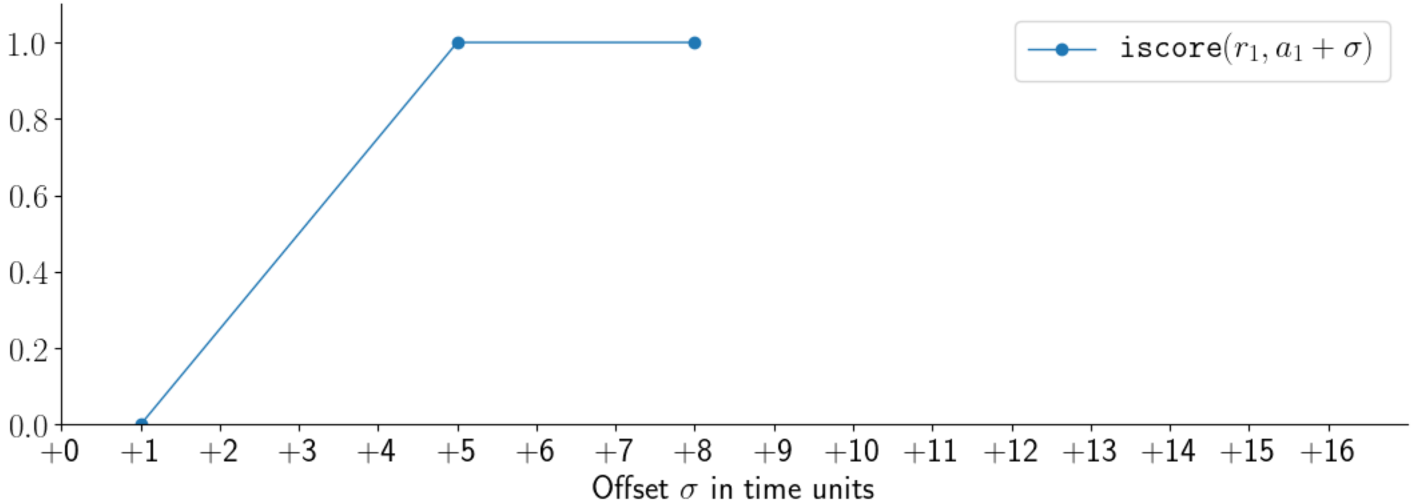


Optimal no-split alignment

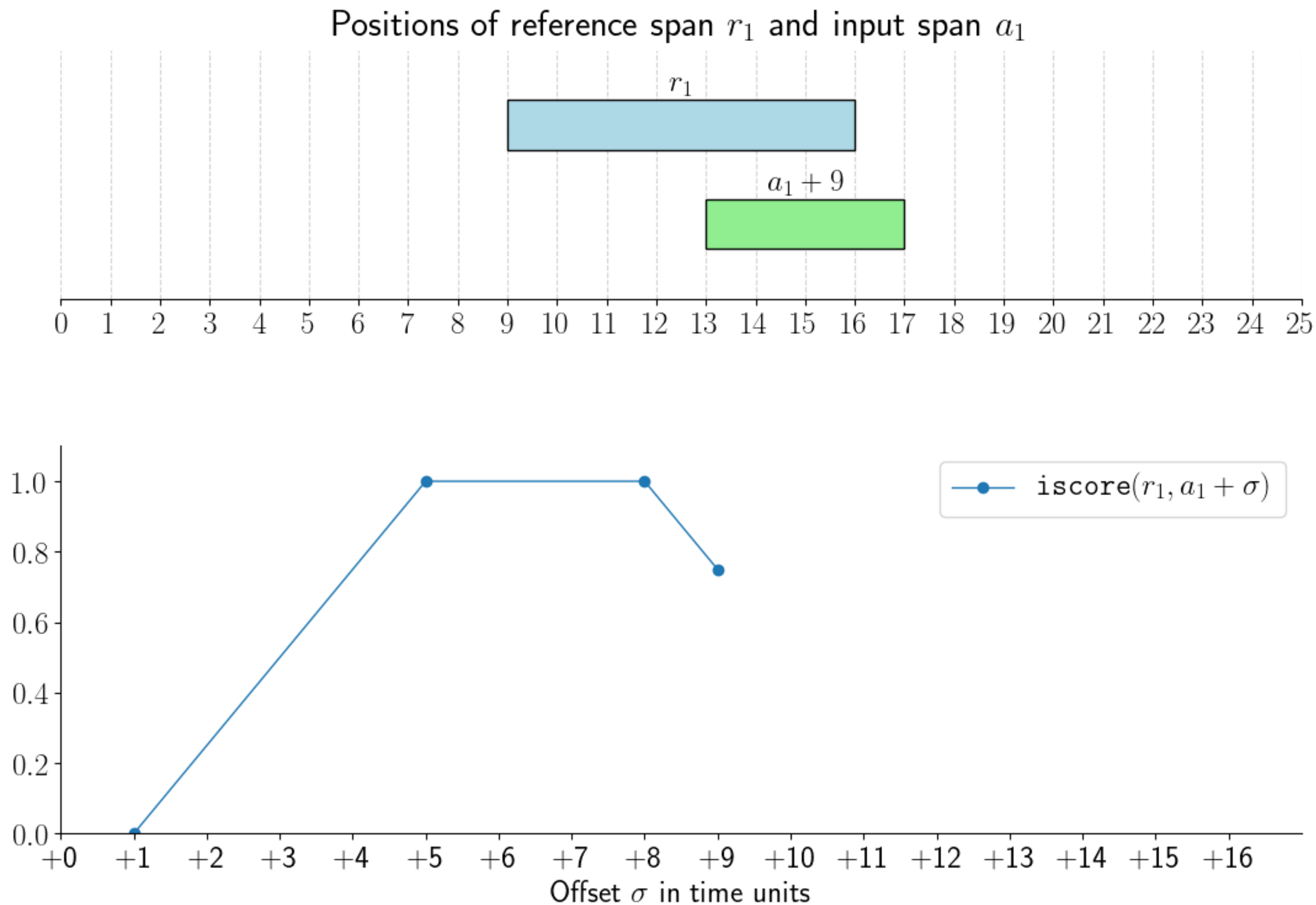


Optimal no-split alignment

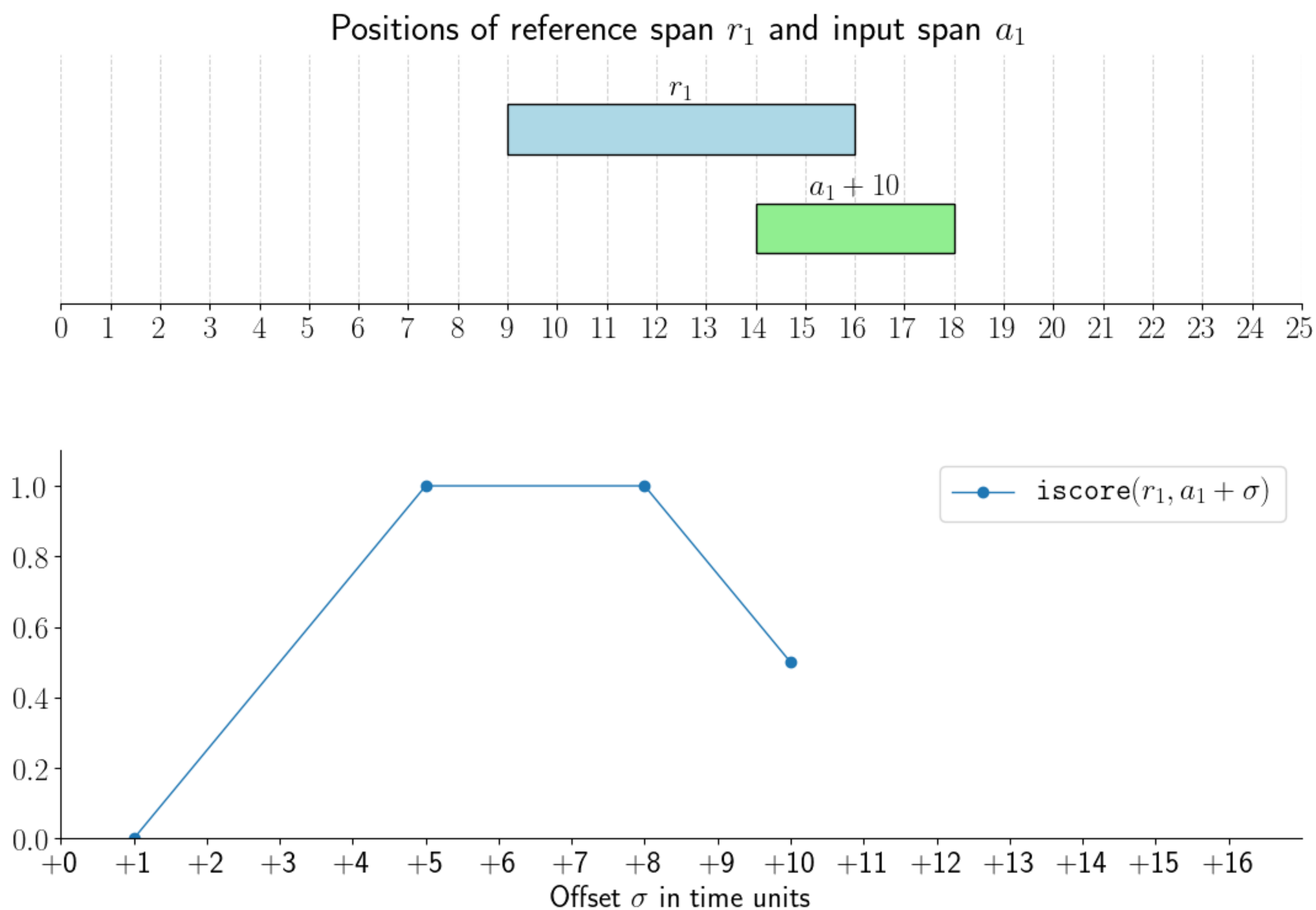




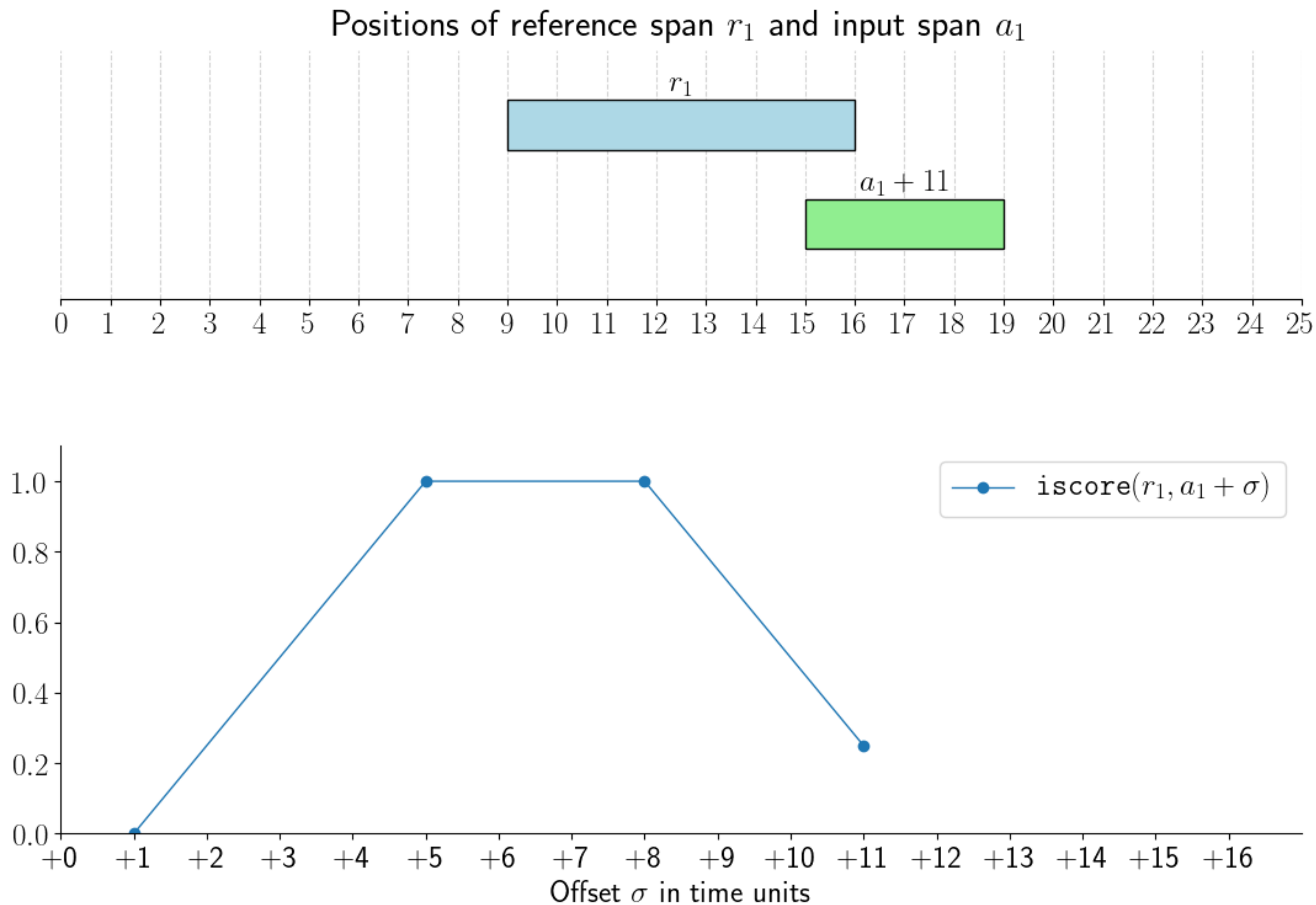
Optimal no-split alignment



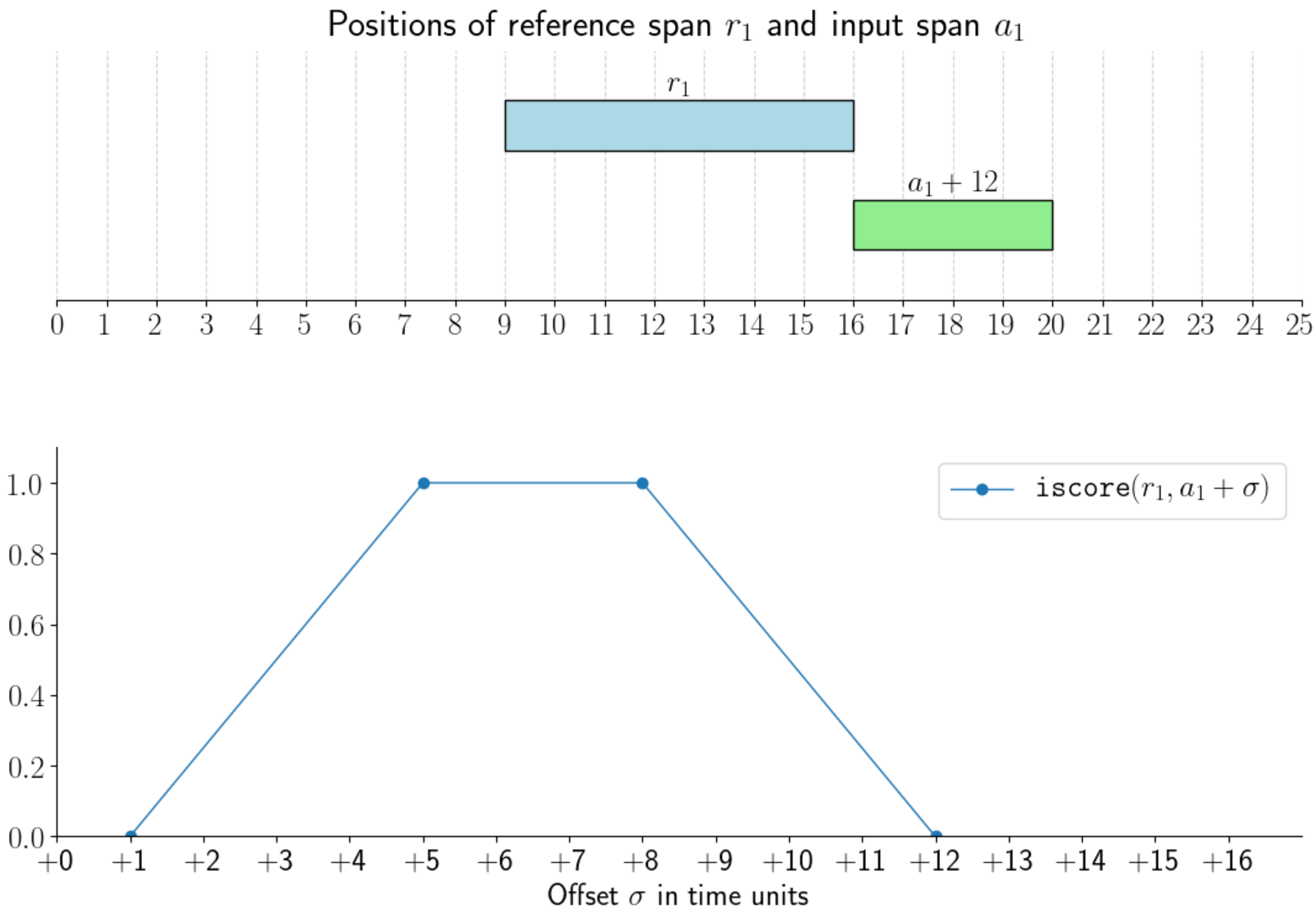
Optimal no-split alignment



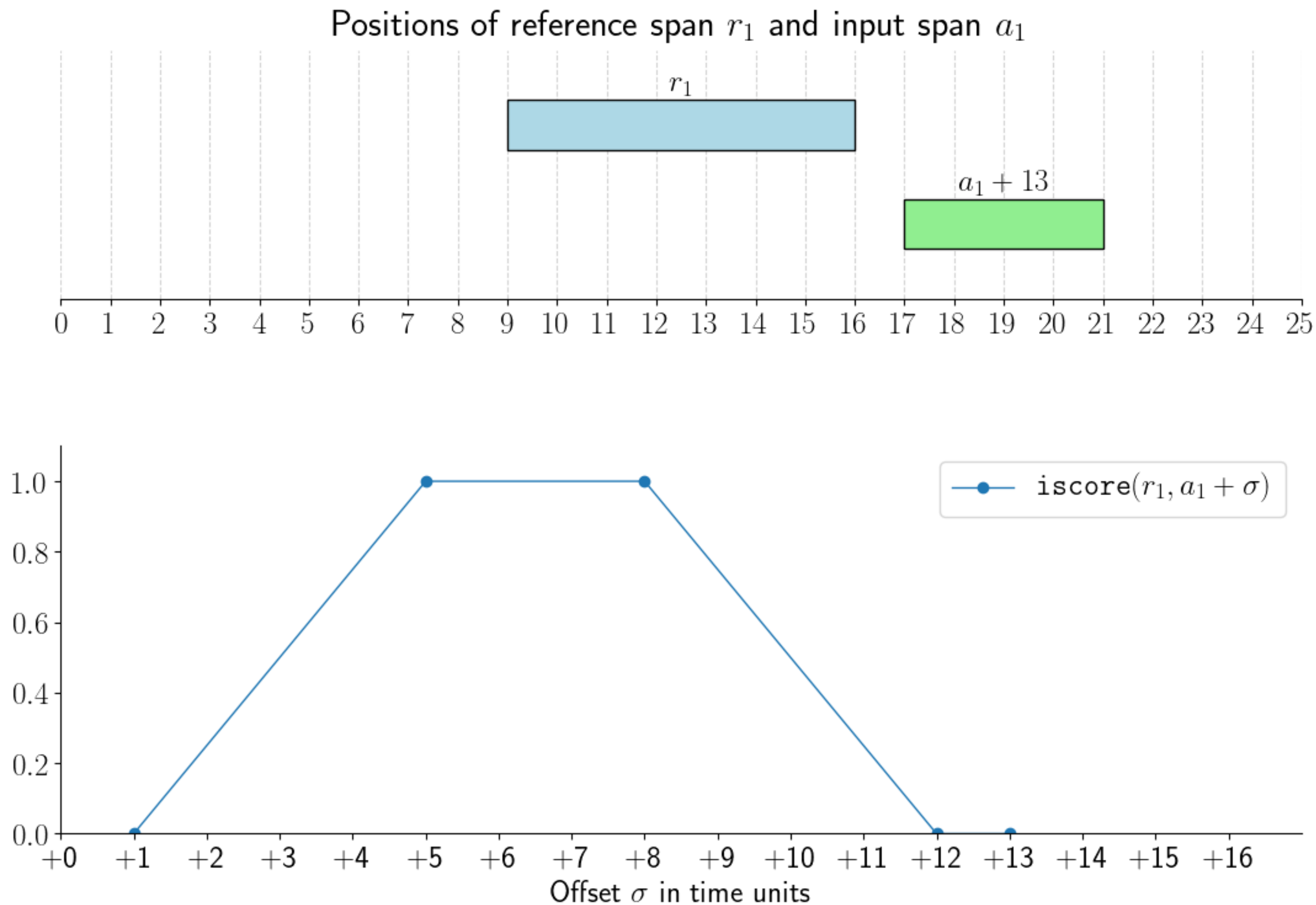
Optimal no-split alignment



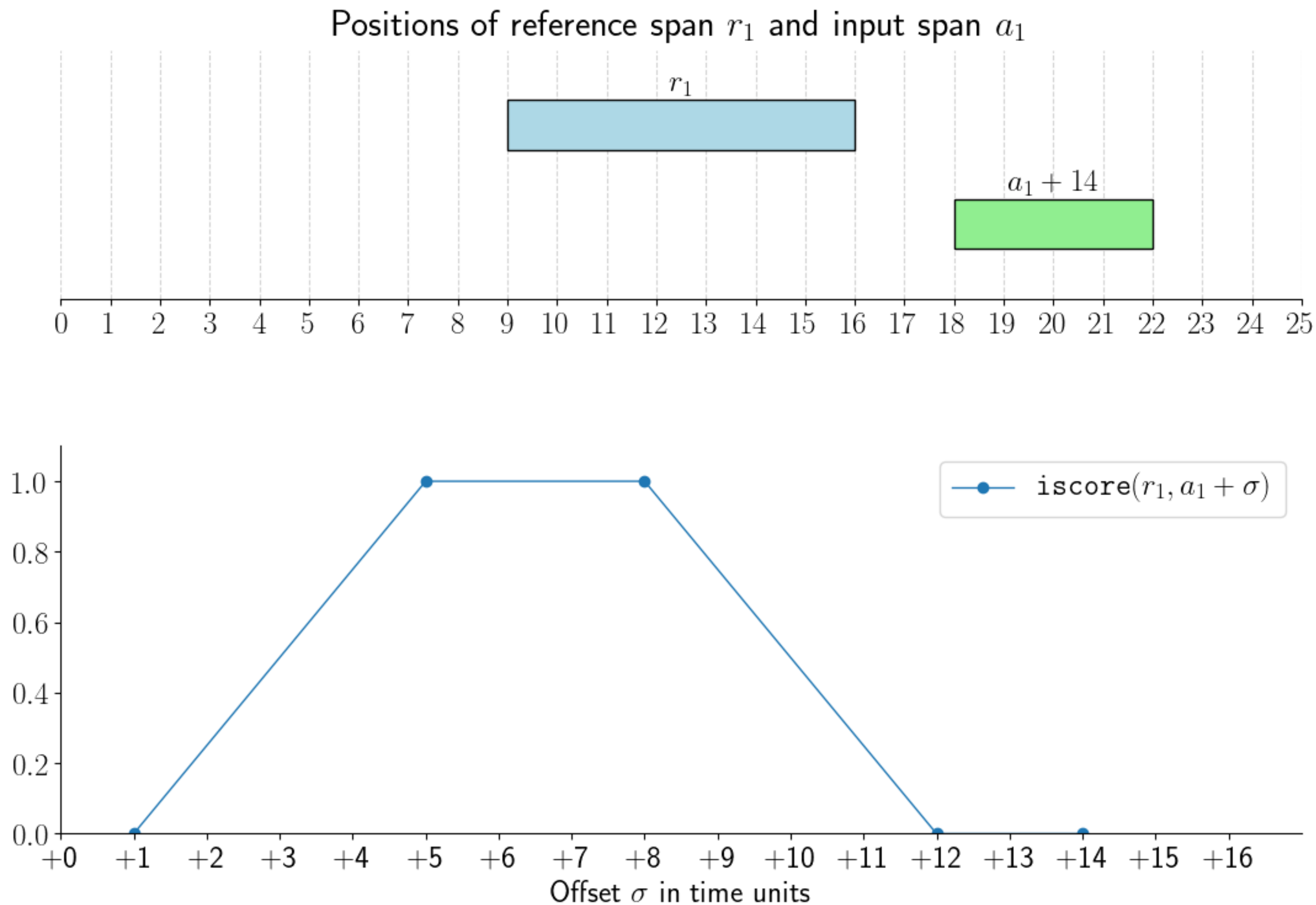
Optimal no-split alignment



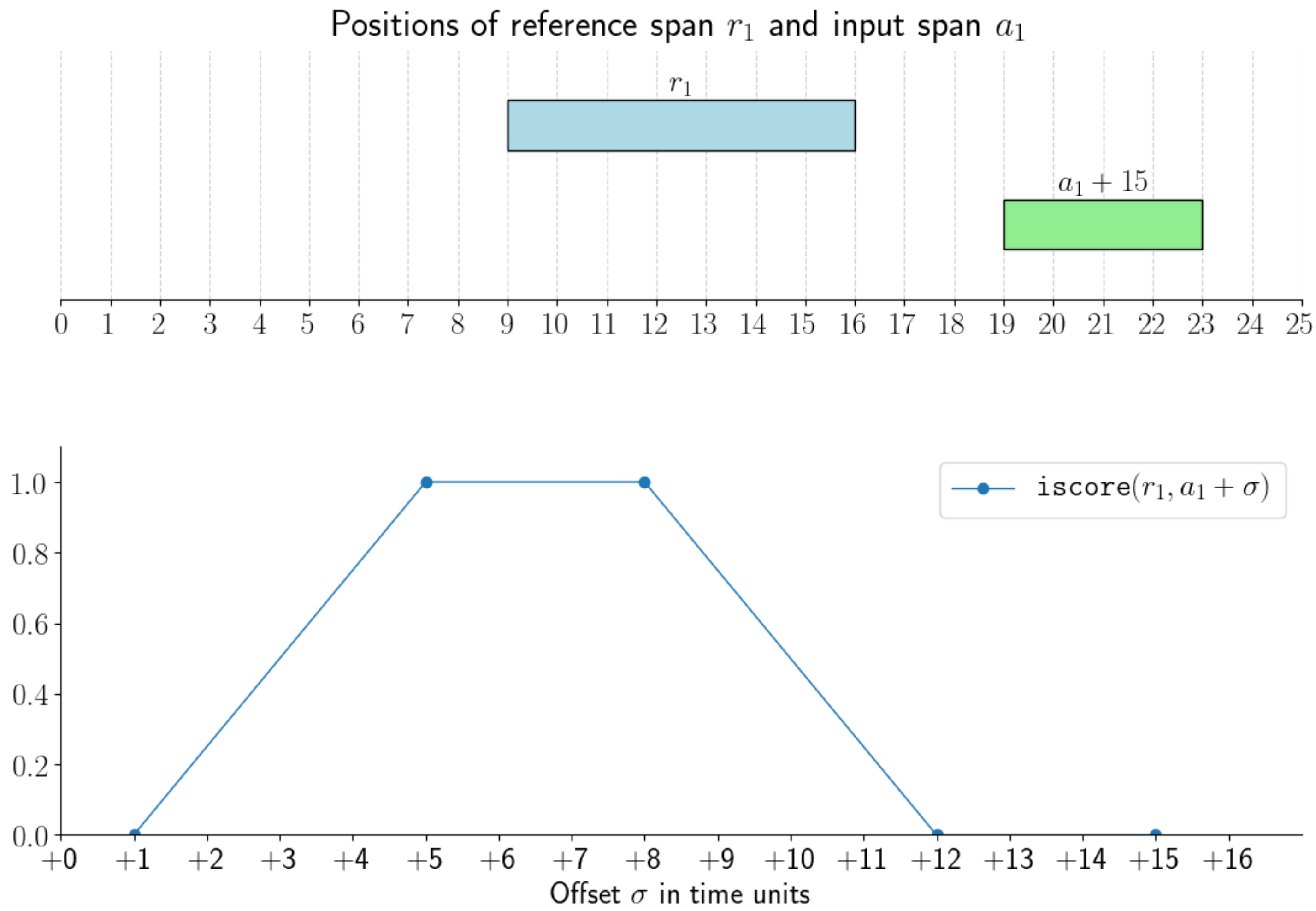
Optimal no-split alignment



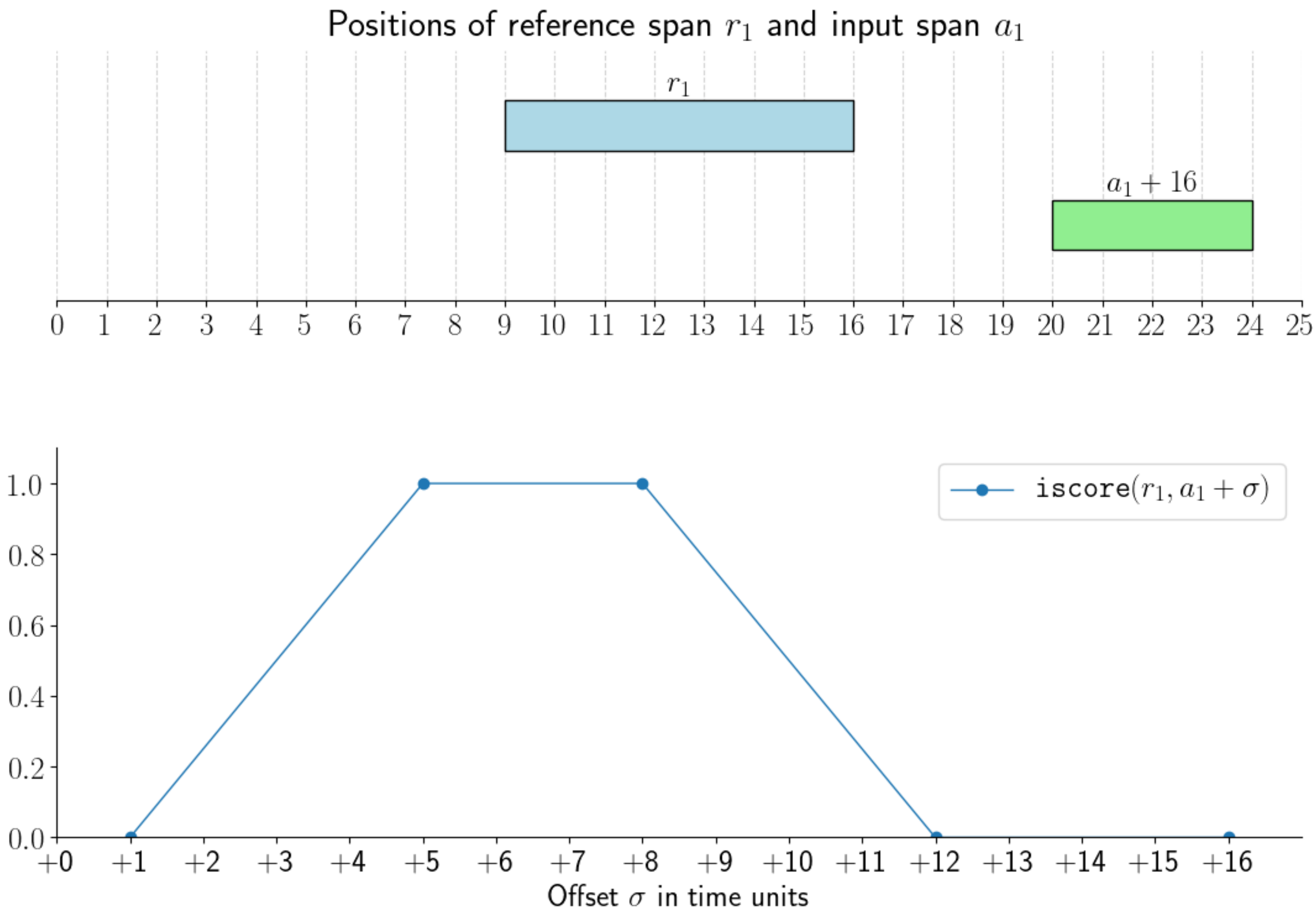
Optimal no-split alignment



Optimal no-split alignment



Optimal no-split alignment



Optimal no-split alignment

Scoring for no-split alignments

Given two sequences of spans $r = (r_1, r_2, \ldots, r_K)$ and $a = (a_1, a_2, \ldots, a_N)$, and a weighting function $w : \{1, \ldots, K\} \times \{1, \ldots, N\} \to \mathbb{R}_{>0}$ the `nosplit_score` is defined as

Optimal no-split alignment

Scoring for no-split alignments

Given two sequences of spans $r = (r_1, r_2, \ldots, r_K)$ and $a = (a_1, a_2, \ldots, a_N)$, and a weighting function $w : \{1, \ldots, K\} \times \{1, \ldots, N\} \to \mathbb{R}_{>0}$ the `nosplit_score` is defined as

$$\texttt{nosplit\_score}(r, a, \sigma, w) = \sum_{n=1}^{N} \sum_{k=1}^{K} \texttt{iscore}(r_k, a_n + \sigma) \cdot w(k, n)$$

Optimal no-split alignment

Scoring for no-split alignments

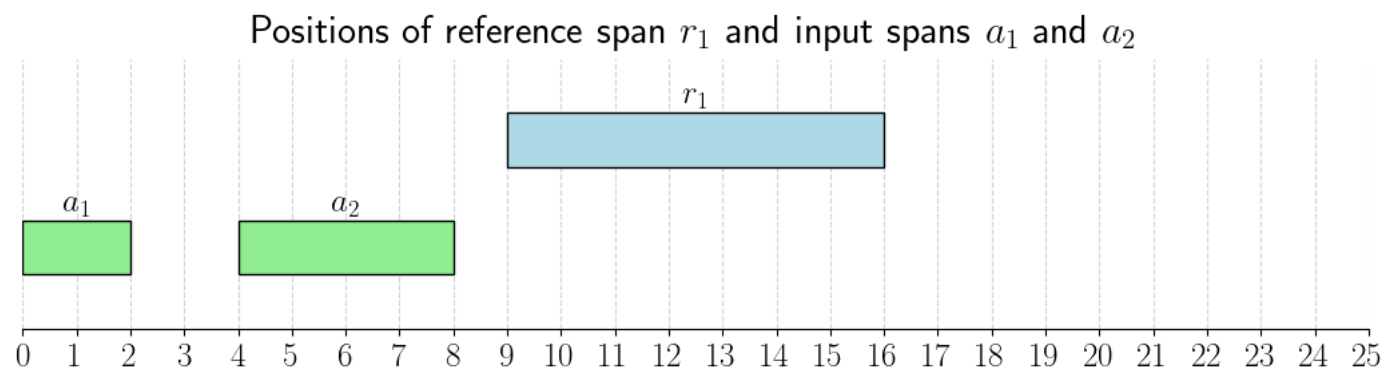
Given two sequences of spans $r = (r_1, r_2, \dots, r_K)$ and $a = (a_1, a_2, \dots, a_N)$, and a weighting function $w : \{1, \dots, K\} \times \{1, \dots, N\} \to \mathbb{R}_{>0}$ the `nosplit_score` is defined as

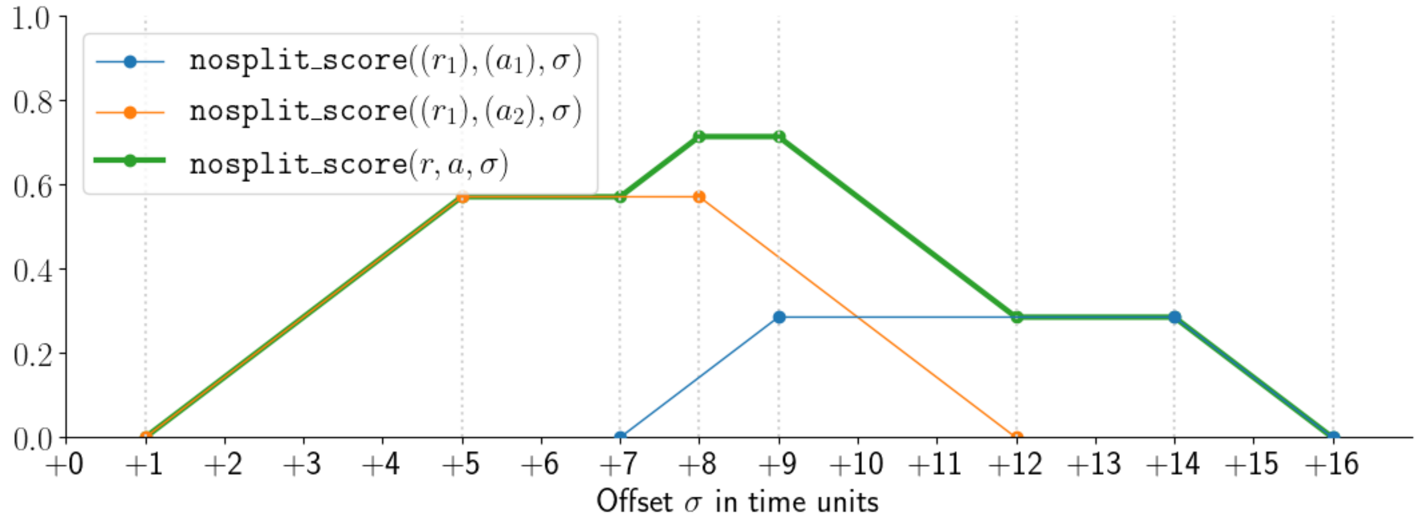
$$\texttt{nosplit\_score}(r, a, \sigma, w) = \sum_{n=1}^{N} \sum_{k=1}^{K} \texttt{iscore}(r_k, a_n + \sigma) \cdot w(k, n)$$

Exemplary weighting function

$$w(k, n) = \frac{\min(\texttt{length}(r_k), \texttt{length}(a_n))}{\max(\texttt{length}(r_k), \texttt{length}(a_n))}$$

Optimal no-split alignment





Optimal no-split alignment

Finding the optimal no-split offset σ

$$\begin{aligned} K &\approx 1300 \\ N &\approx 1300 \\ T_r = \mathtt{end}(r_K) - \mathtt{start}(r_1) &\approx 8'000'000 \\ T_a = \mathtt{end}(a_N) - \mathtt{start}(a_1) &\approx 8'000'000 \end{aligned}$$

Optimal no-split alignment

Finding the optimal no-split offset σ

$$
\begin{aligned}
K &\approx 1300 \\
N &\approx 1300 \\
T_r = \mathtt{end}(r_K) - \mathtt{start}(r_1) &\approx 8'000'000 \\
T_a = \mathtt{end}(a_N) - \mathtt{start}(a_1) &\approx 8'000'000
\end{aligned}
$$

- Brute-Force: $O(KN(T_r + T_a))$

Optimal no-split alignment

Finding the optimal no-split offset σ

$$\begin{aligned} K &\approx 1300 \\ N &\approx 1300 \\ T_r = \texttt{end}(r_K) - \texttt{start}(r_1) &\approx 8'000'000 \\ T_a = \texttt{end}(a_N) - \texttt{start}(a_1) &\approx 8'000'000 \end{aligned}$$

- Brute-Force: $O(KN(T_r + T_a))$ Days!

Finding the optimal no-split offset σ

$$\begin{aligned} K &\approx 1300 \\ N &\approx 1300 \\ T_r = \texttt{end}(r_K) - \texttt{start}(r_1) &\approx 8'000'000 \\ T_a = \texttt{end}(a_N) - \texttt{start}(a_1) &\approx 8'000'000 \end{aligned}$$

- Brute-Force: $O(KN(T_r + T_a))$ Days!
- Efficient Brute-Force: $O((K + N) \cdot (T_r + T_a))$

Finding the optimal no-split offset σ

$$
\begin{aligned}
K &\approx 1300 \\
N &\approx 1300 \\
T_r = \texttt{end}(r_K) - \texttt{start}(r_1) &\approx 8'000'000 \\
T_a = \texttt{end}(a_N) - \texttt{start}(a_1) &\approx 8'000'000
\end{aligned}
$$

- Brute-Force: $O(KN(T_r + T_a))$ Days!
- Efficient Brute-Force: $O((K+N) \cdot (T_r + T_a))$ Minutes!

Optimal no-split alignment

Finding the optimal no-split offset σ

$$
\begin{aligned}
K &\approx 1300 \\
N &\approx 1300 \\
T_r = \texttt{end}(r_K) - \texttt{start}(r_1) &\approx 8'000'000 \\
T_a = \texttt{end}(a_N) - \texttt{start}(a_1) &\approx 8'000'000
\end{aligned}
$$

- Brute-Force: $O(KN(T_r + T_a))$ Days!
- Efficient Brute-Force: $O((K + N) \cdot (T_r + T_a))$ Minutes!
- Tracking slope changes: ?

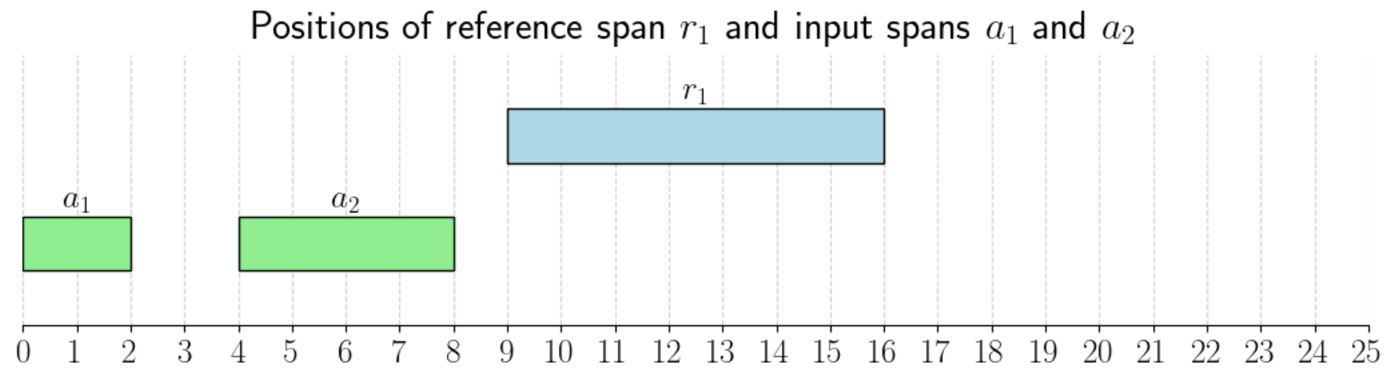
Optimal no-split alignment

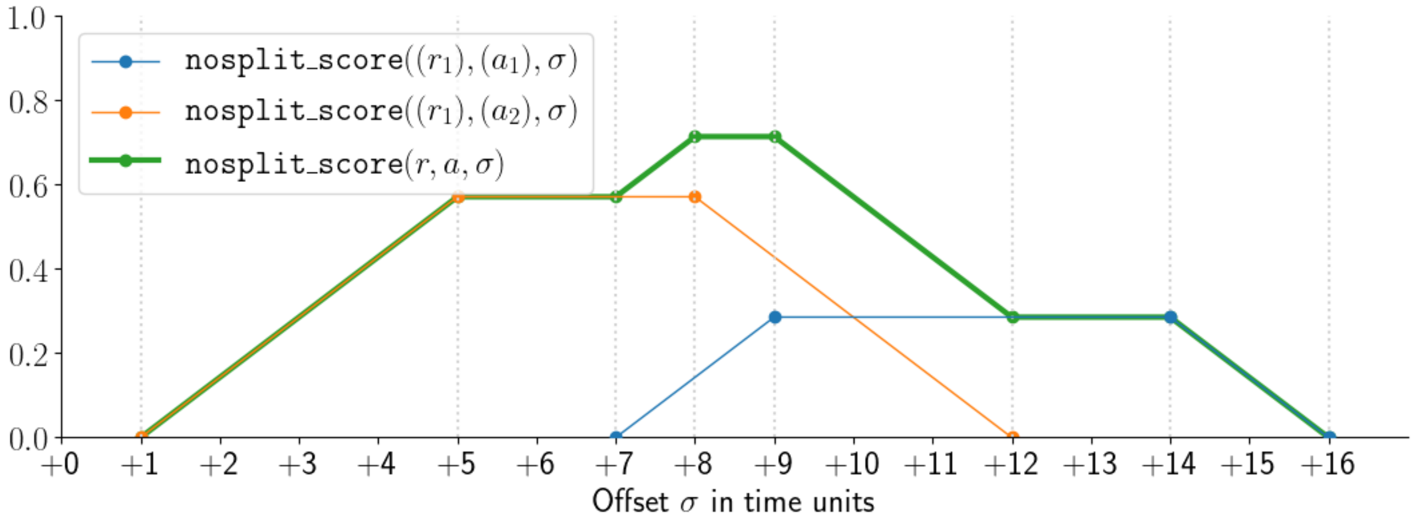
Finding the optimal no-split offset σ

$$
\begin{aligned}
K &\approx 1300 \\
N &\approx 1300 \\
T_r = \texttt{end}(r_K) - \texttt{start}(r_1) &\approx 8'000'000 \\
T_a = \texttt{end}(a_N) - \texttt{start}(a_1) &\approx 8'000'000
\end{aligned}
$$

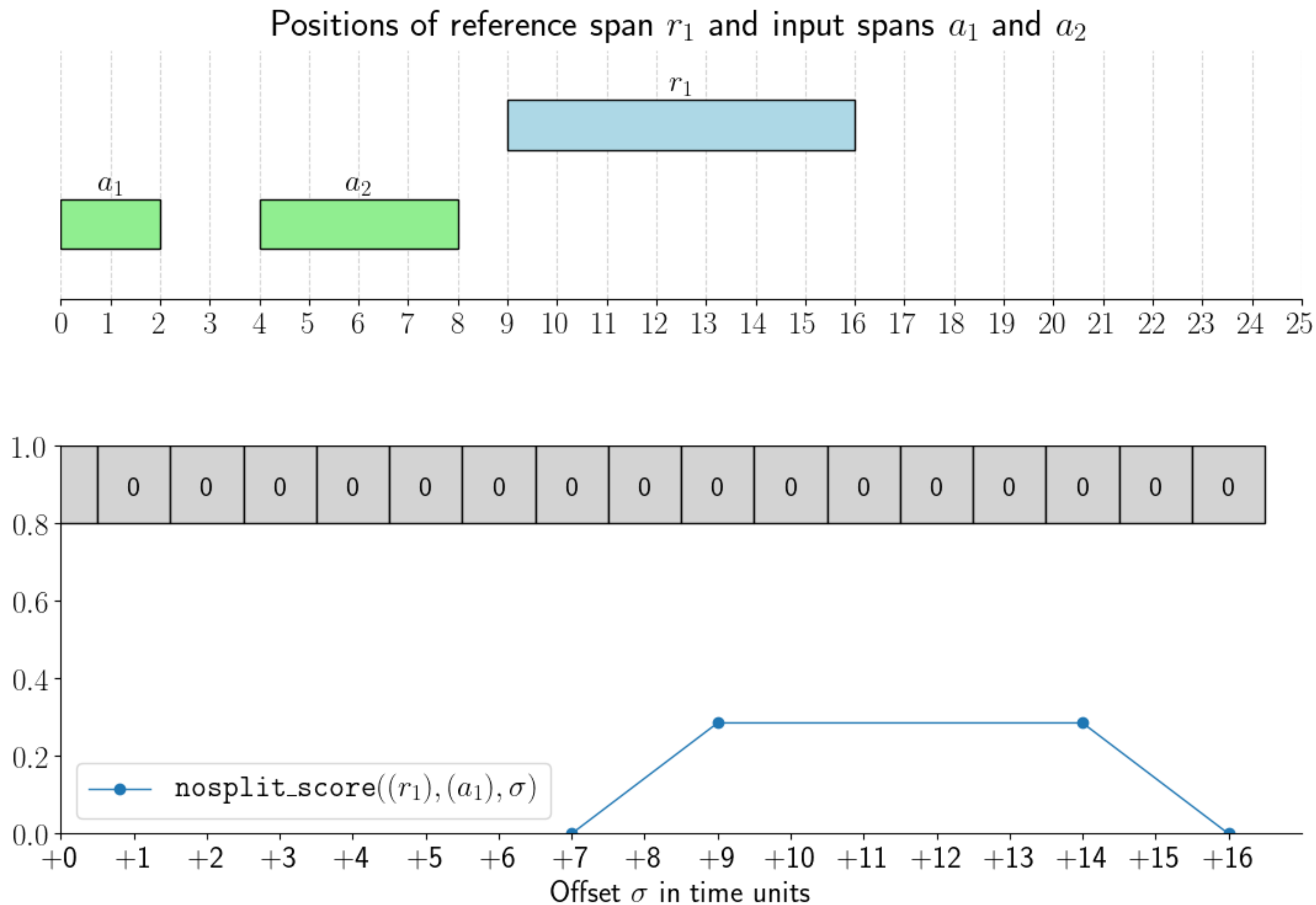
- Brute-Force: $O(KN(T_r + T_a))$ Days!
- Efficient Brute-Force: $O((K+N) \cdot (T_r + T_a))$ Minutes!
- Tracking slope changes: ? Milliseconds!

Optimal no-split alignment

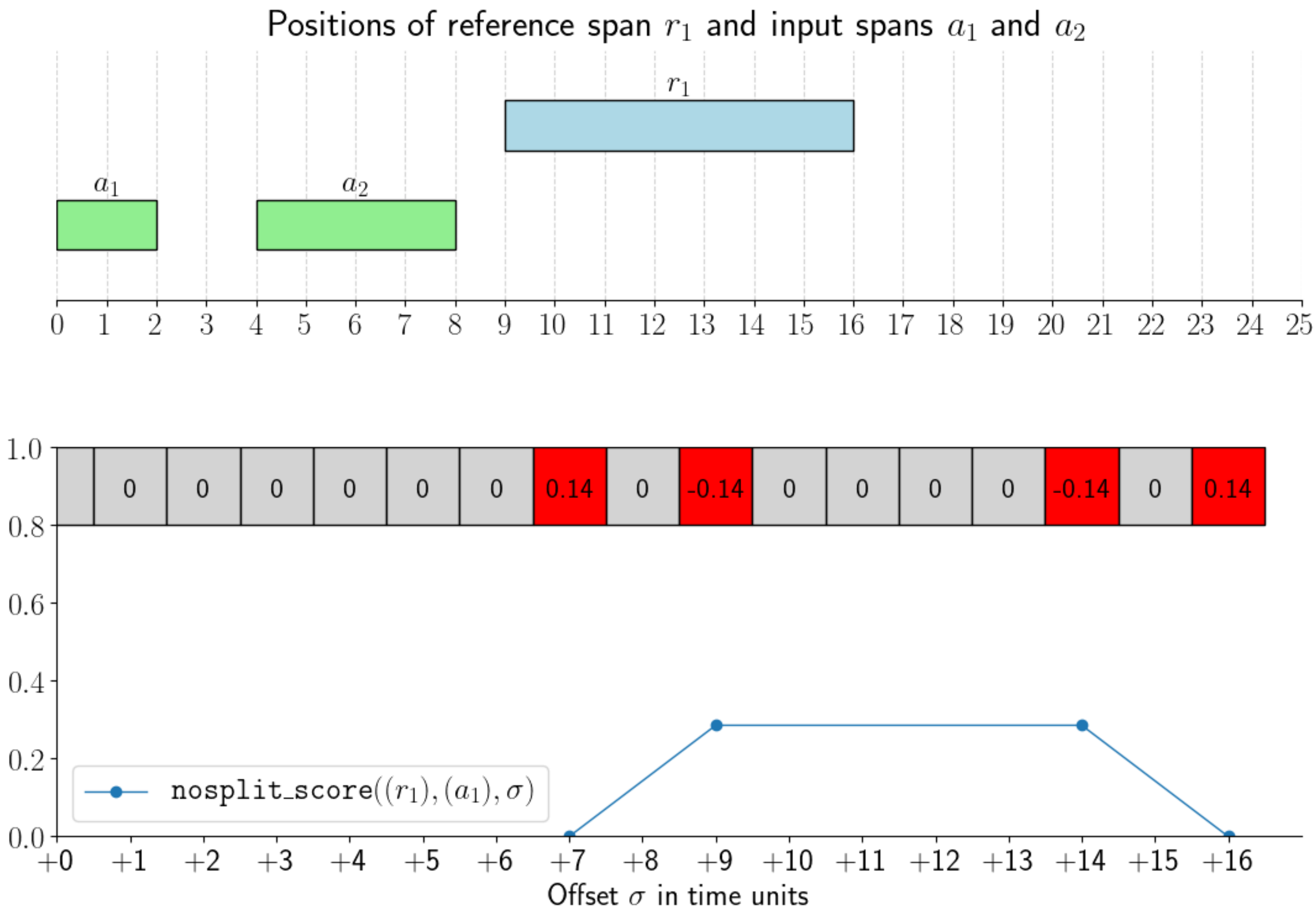




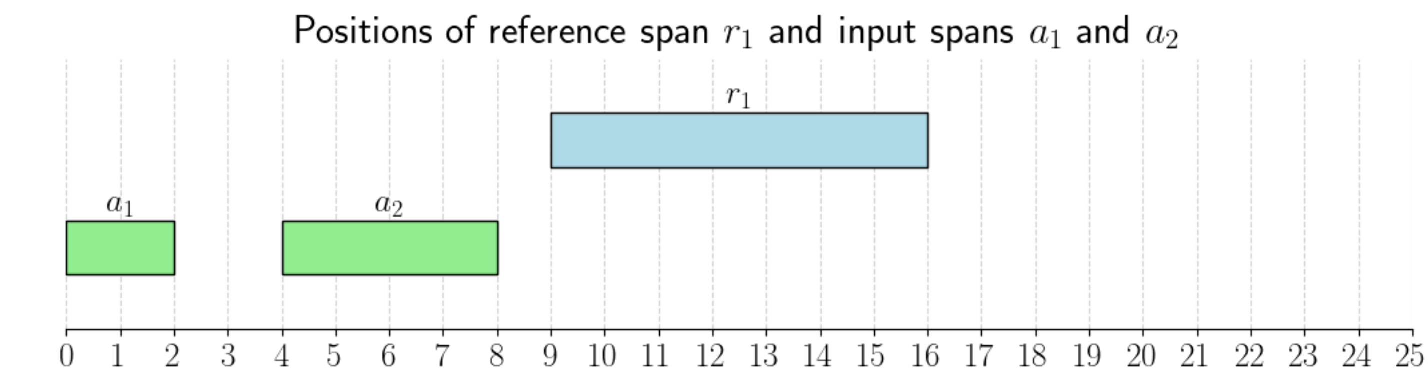
Optimal no-split alignment

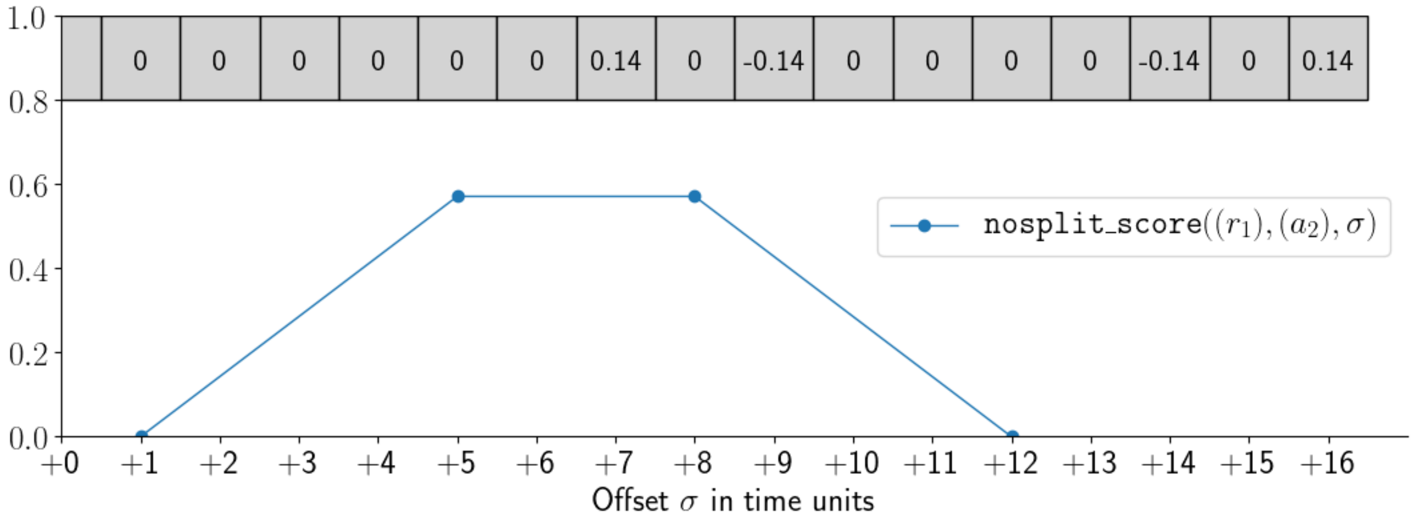


Optimal no-split alignment

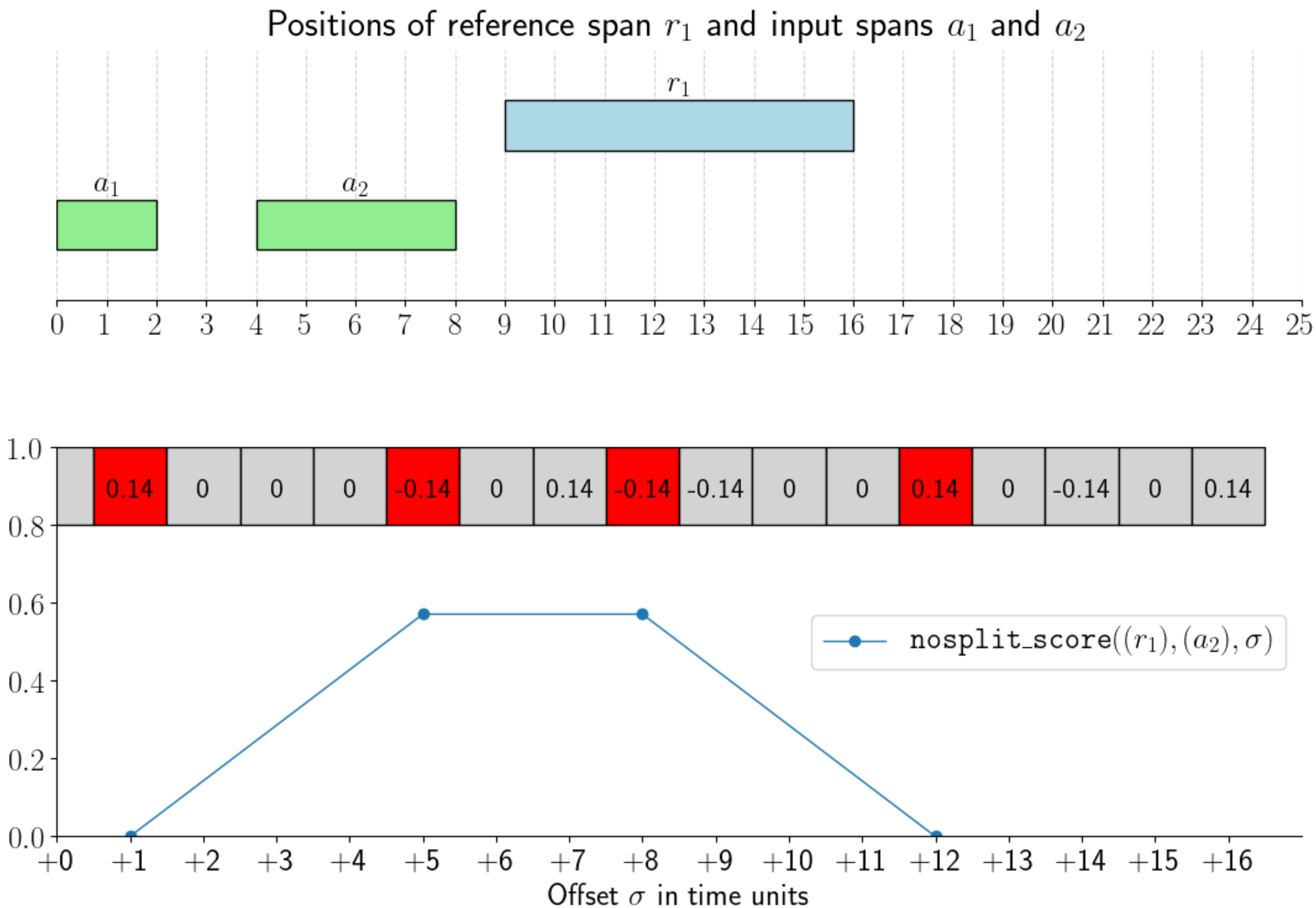


Optimal no-split alignment

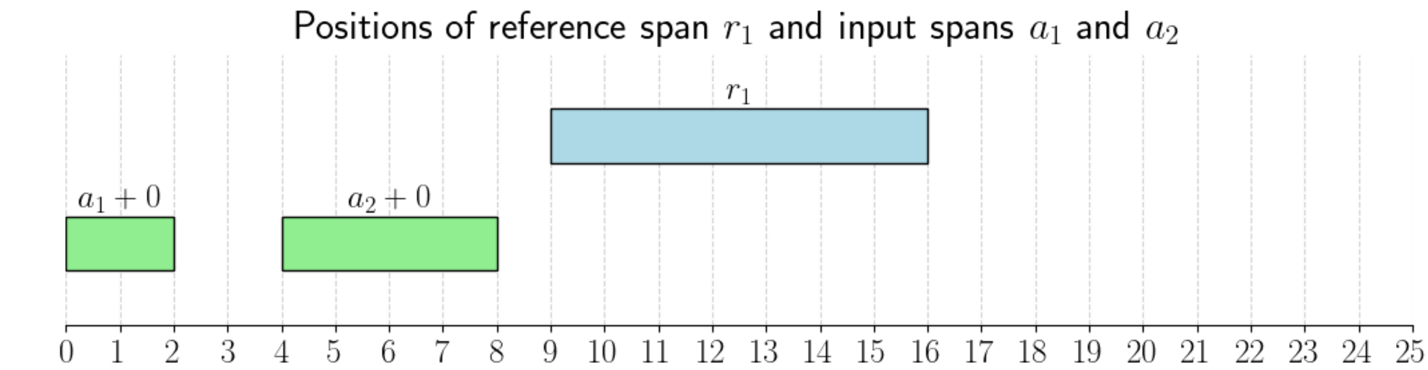


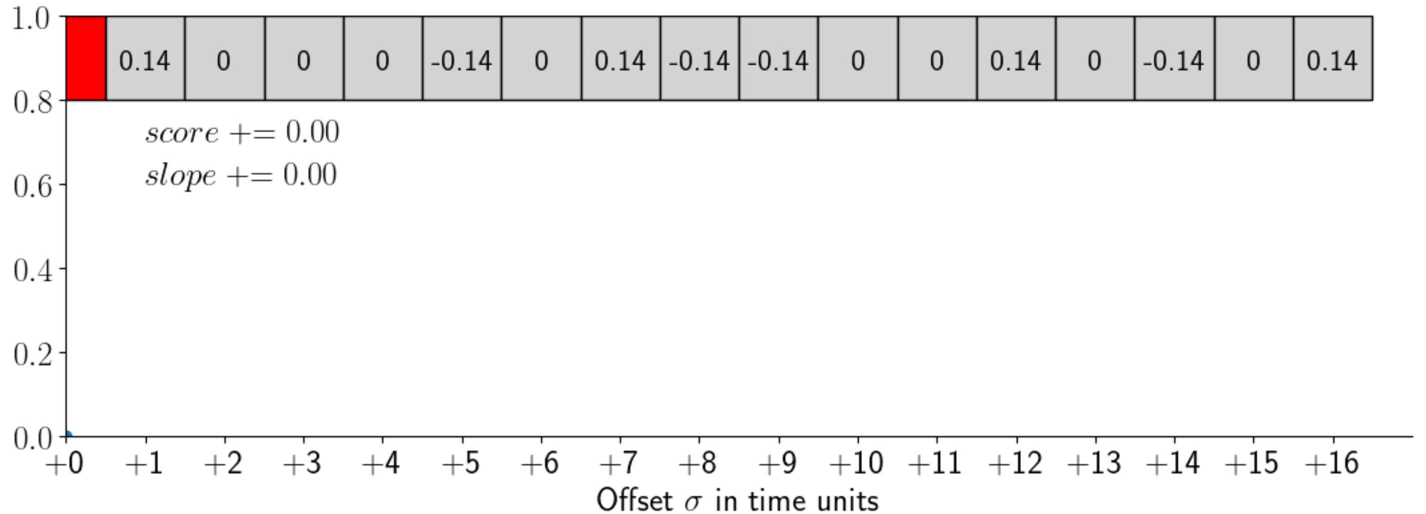


Optimal no-split alignment

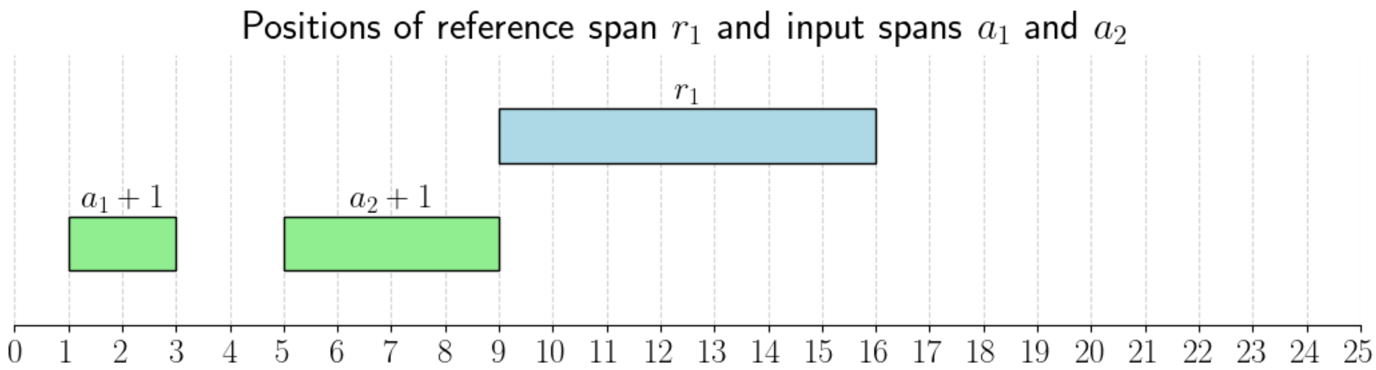


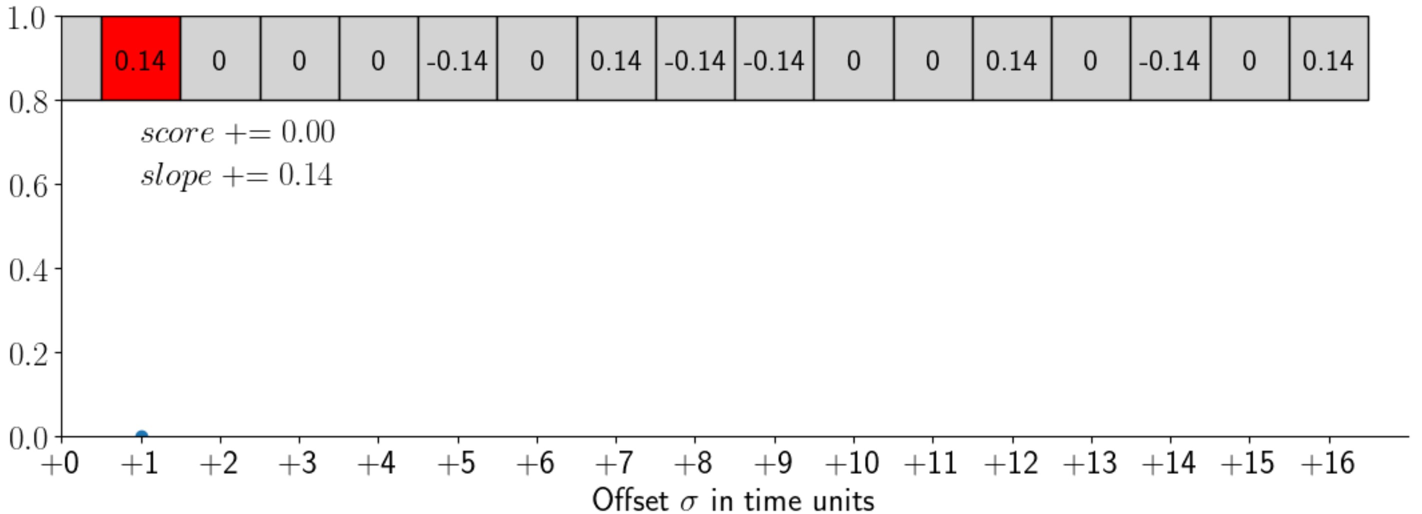
Optimal no-split alignment



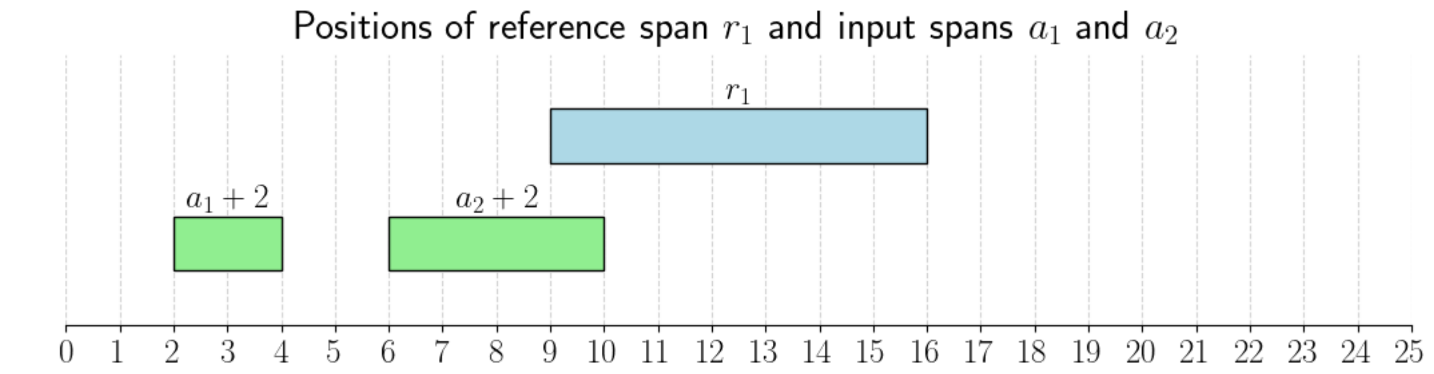


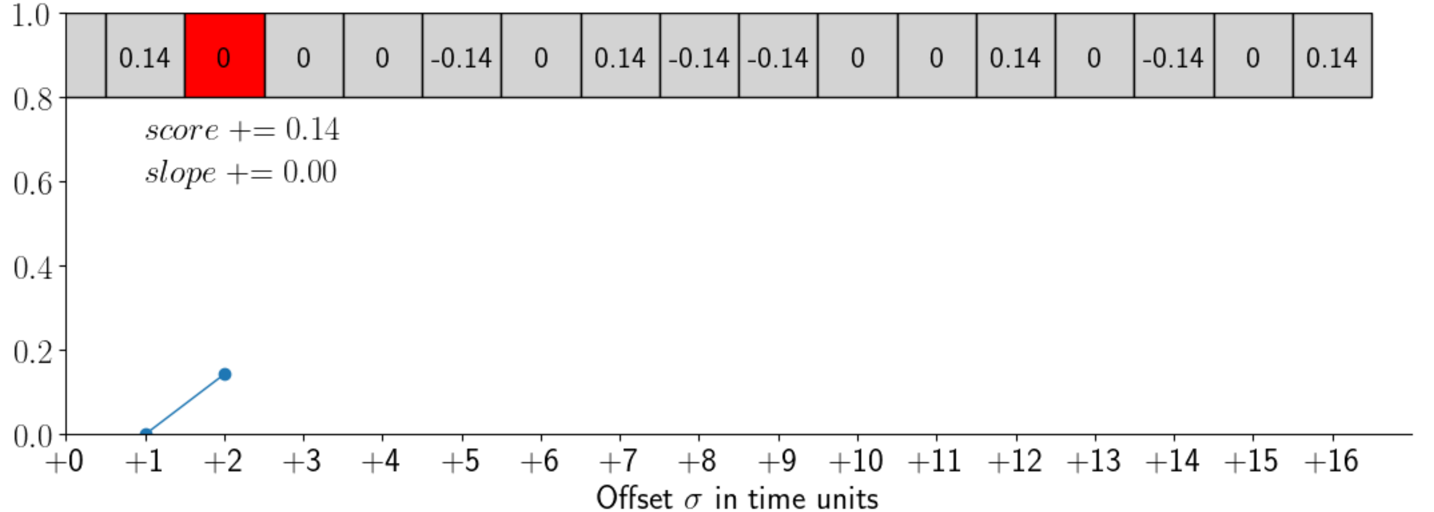
Optimal no-split alignment



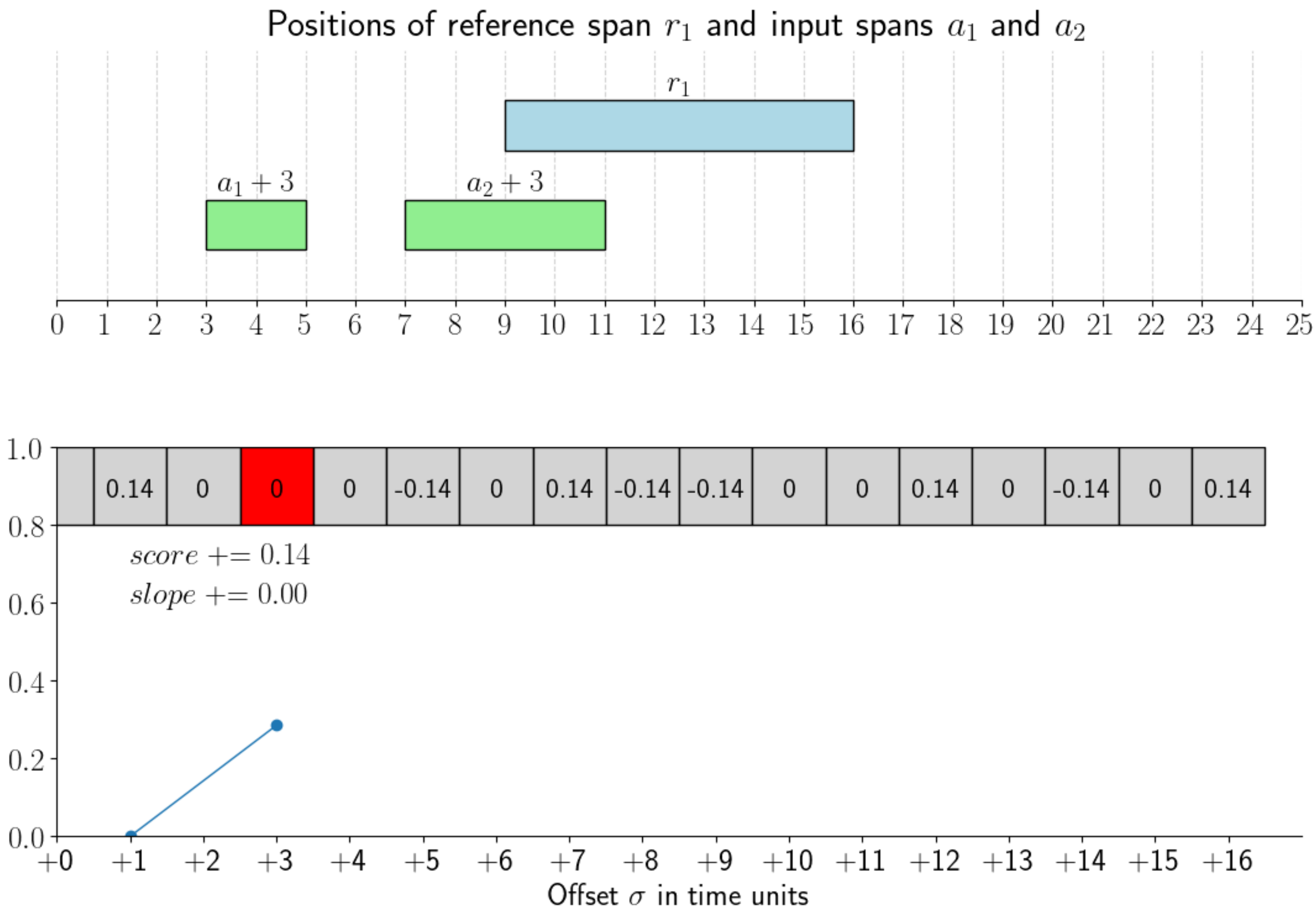


Optimal no-split alignment

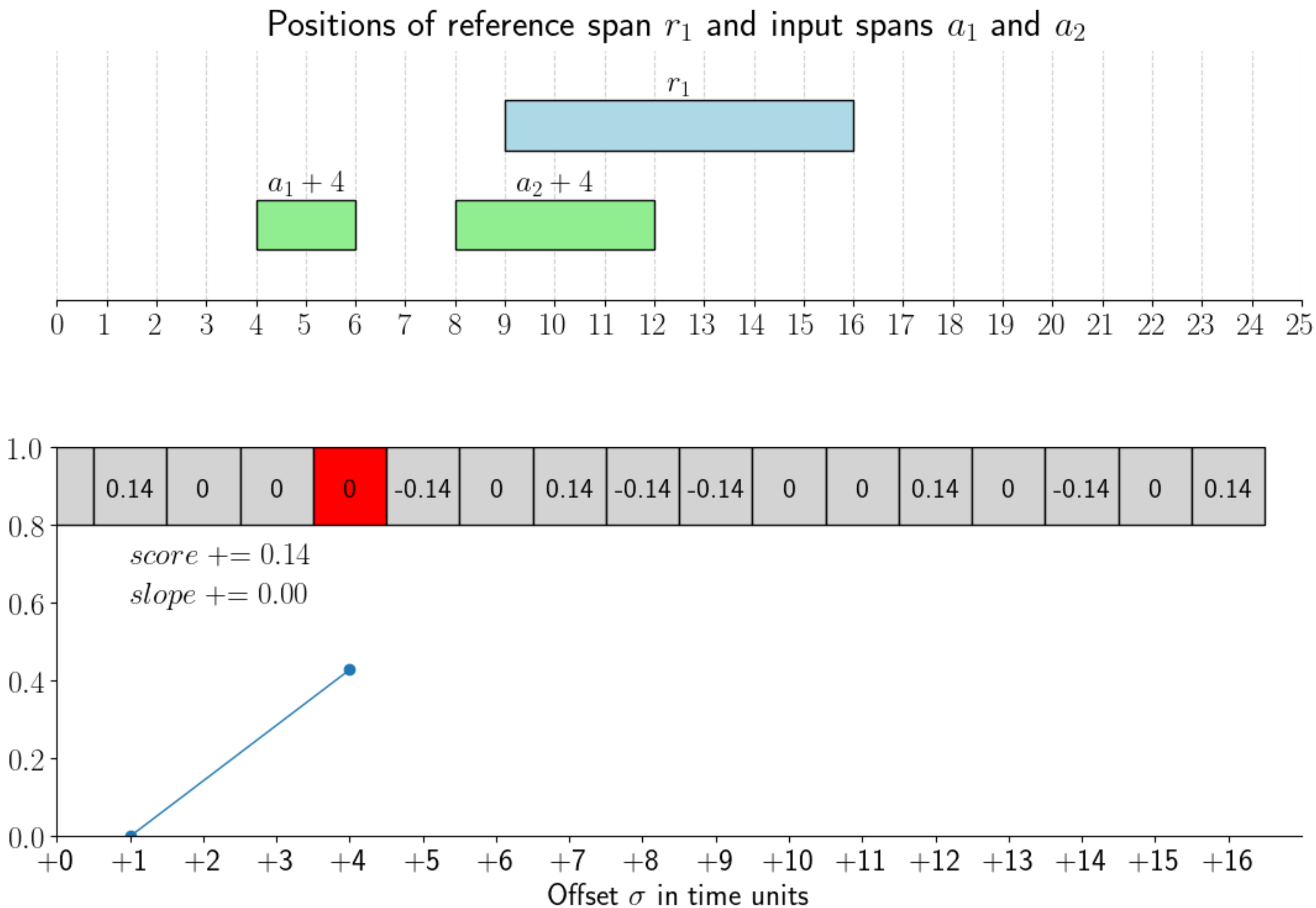




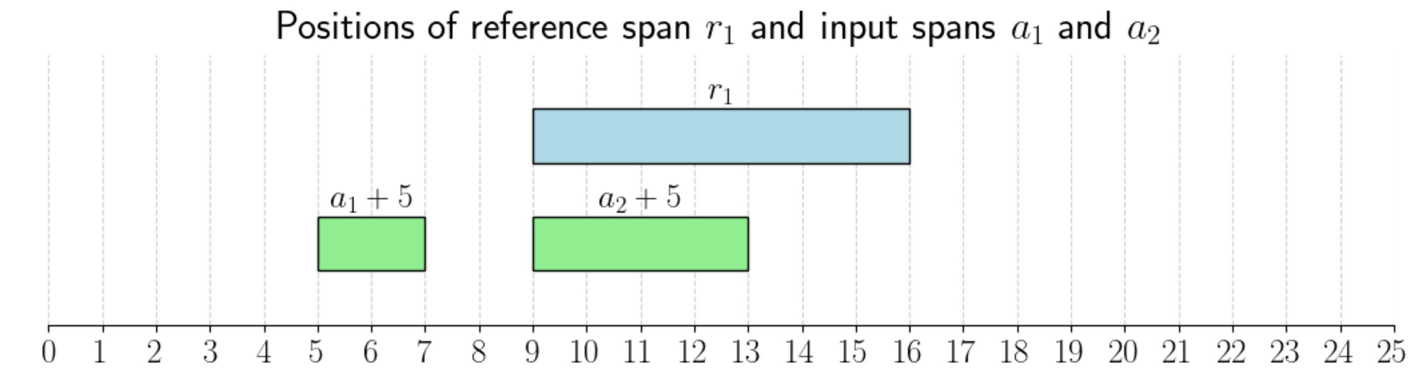
Optimal no-split alignment

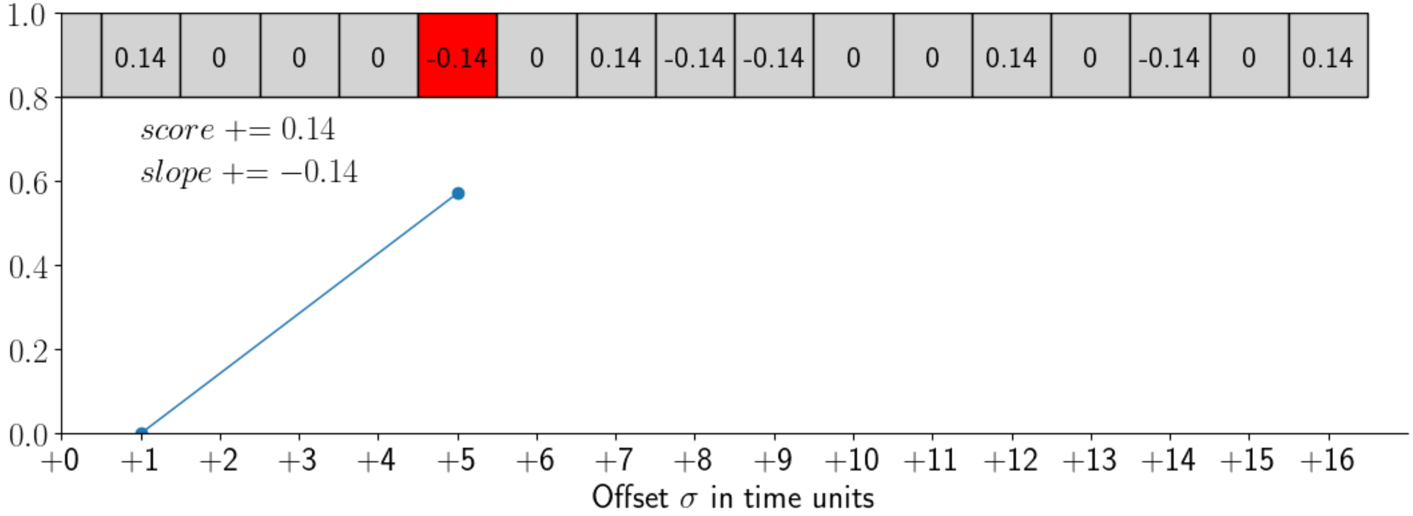


Optimal no-split alignment

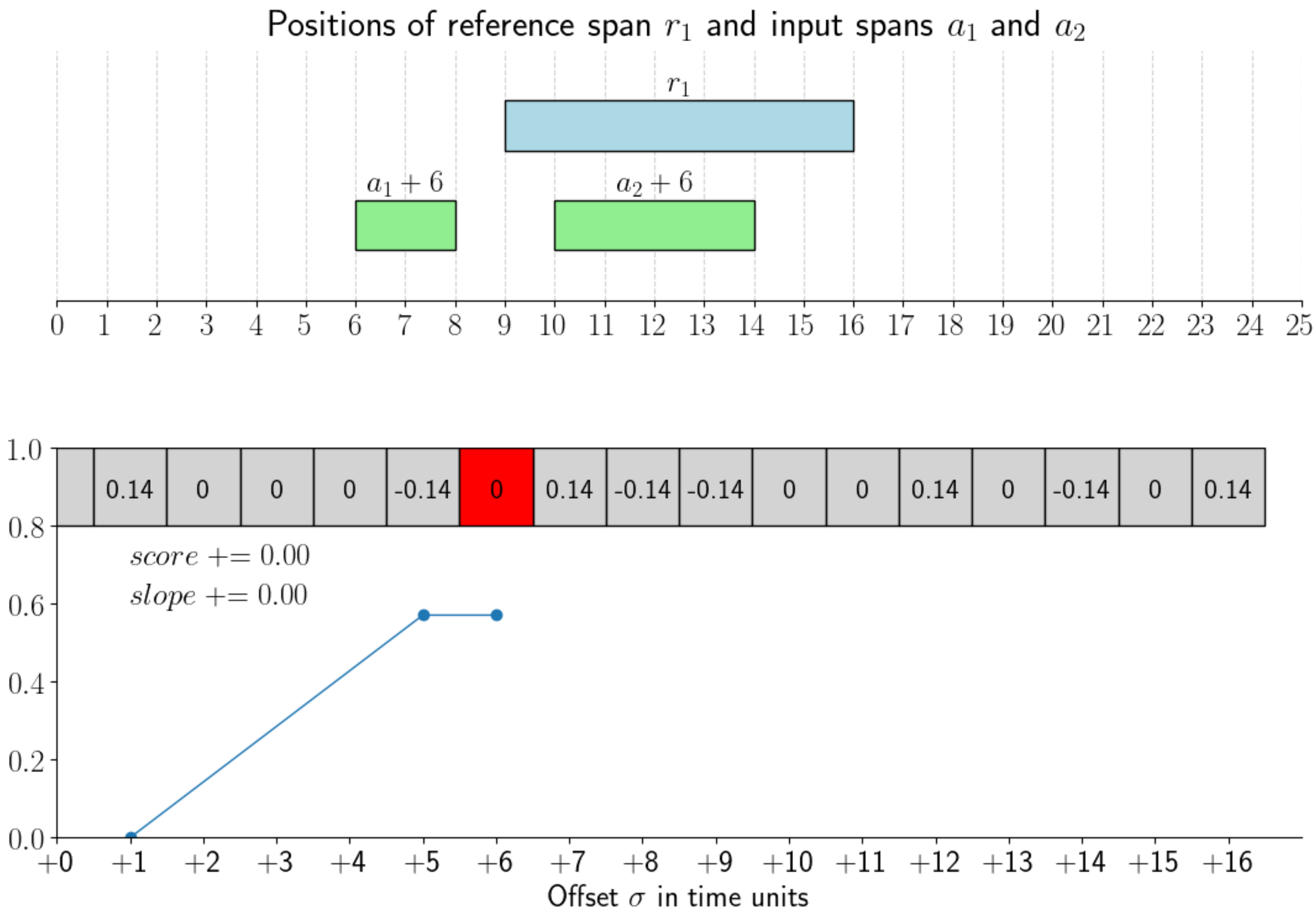


Optimal no-split alignment

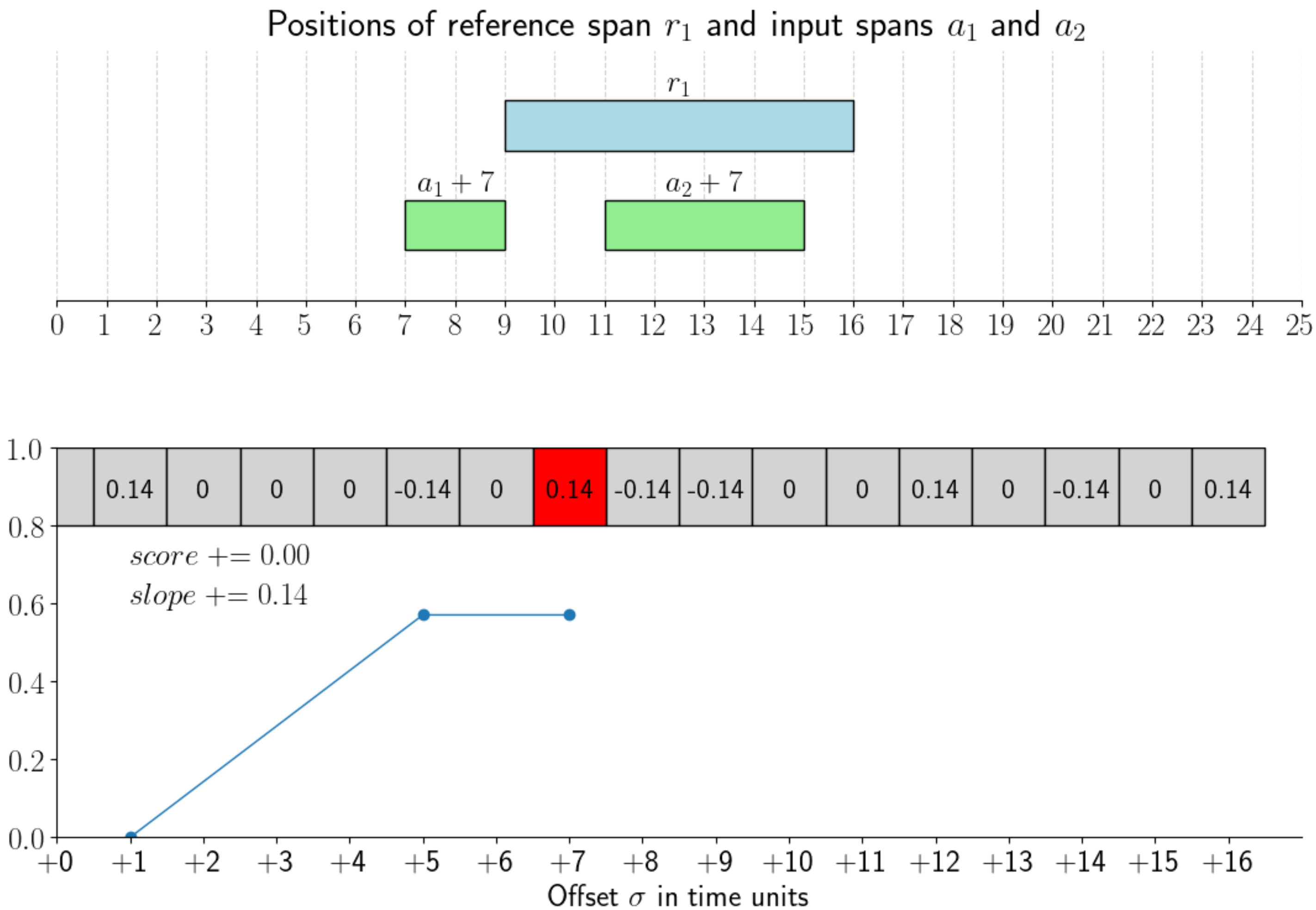




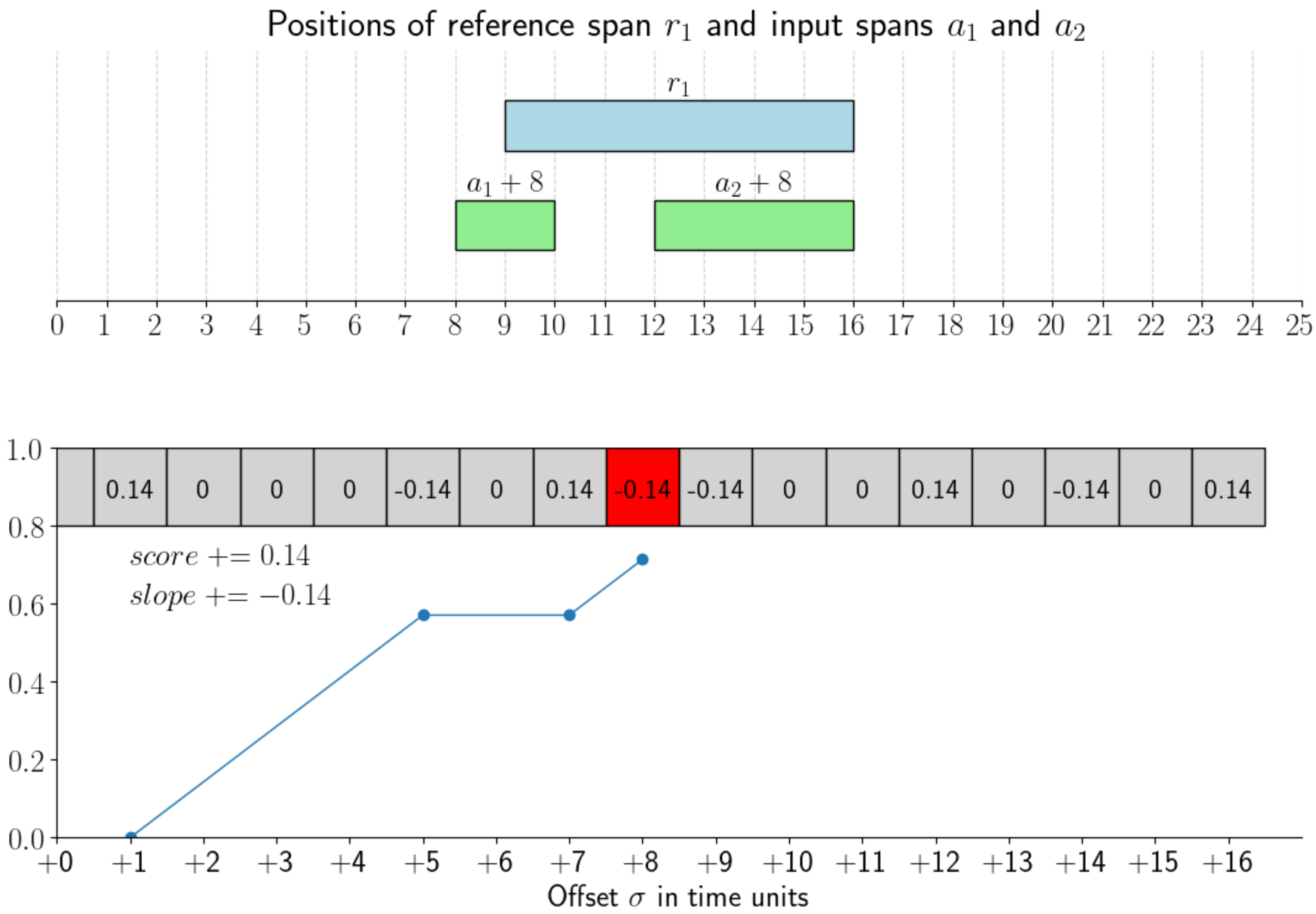
Optimal no-split alignment



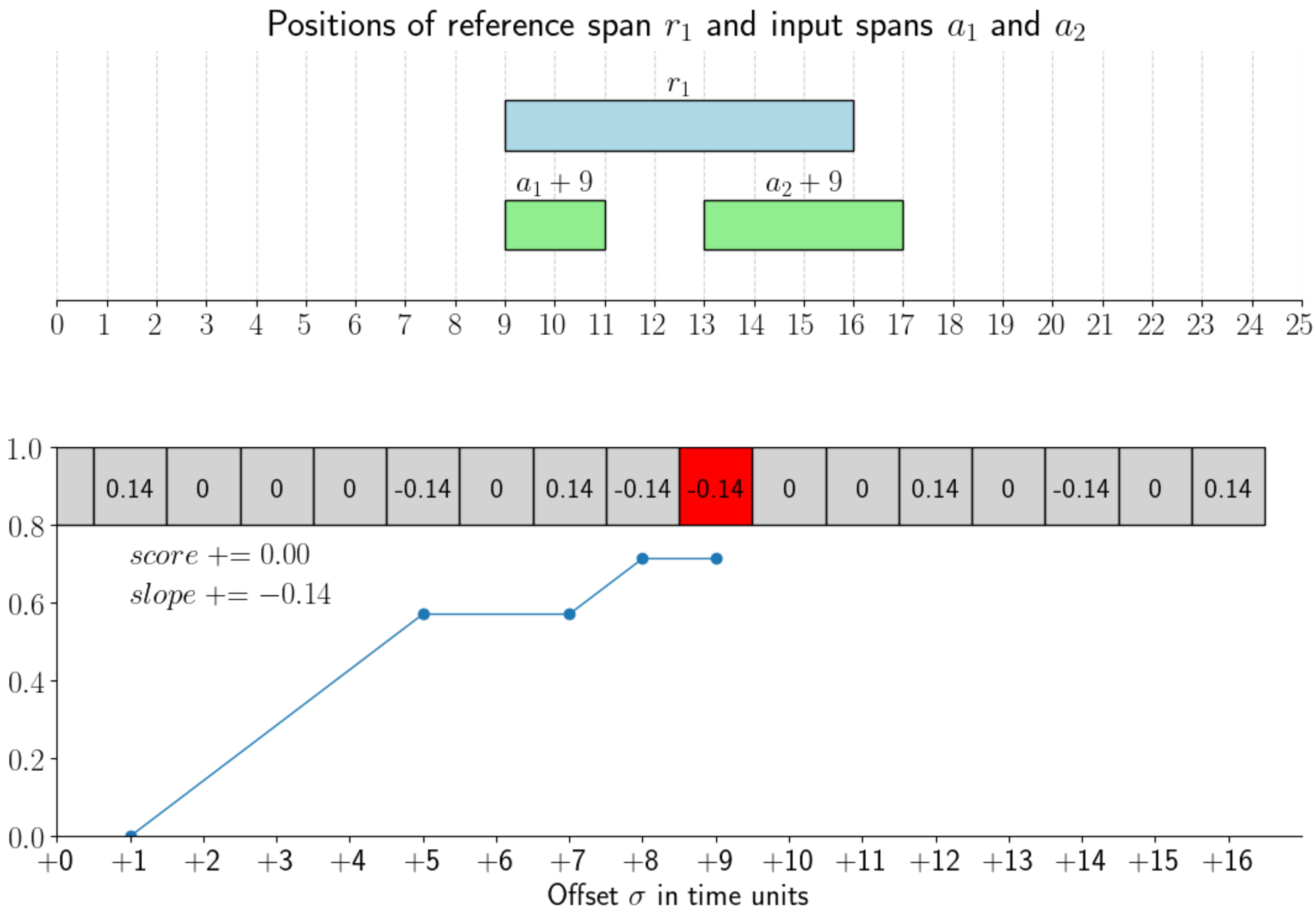
Optimal no-split alignment



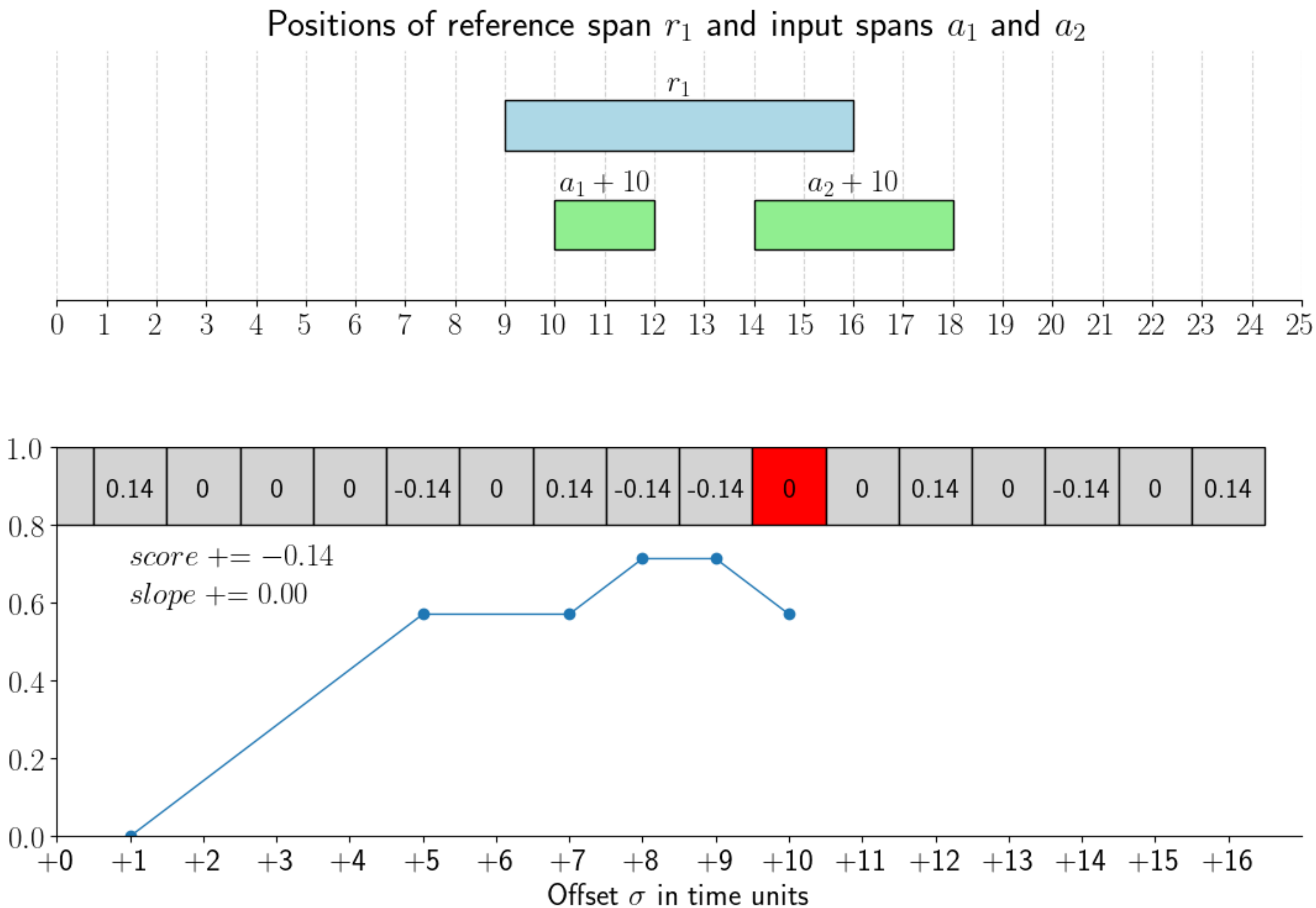
Optimal no-split alignment



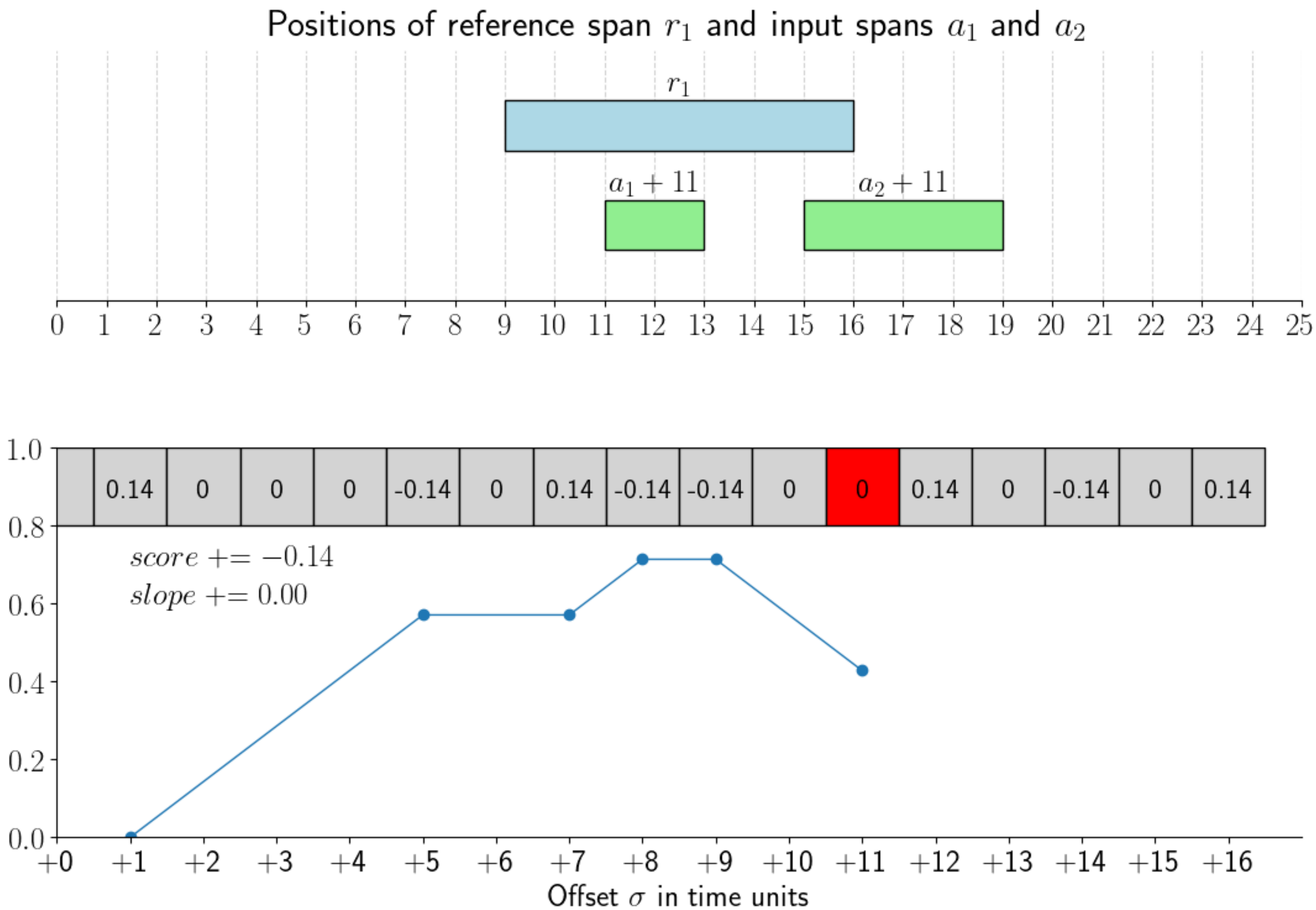
Optimal no-split alignment



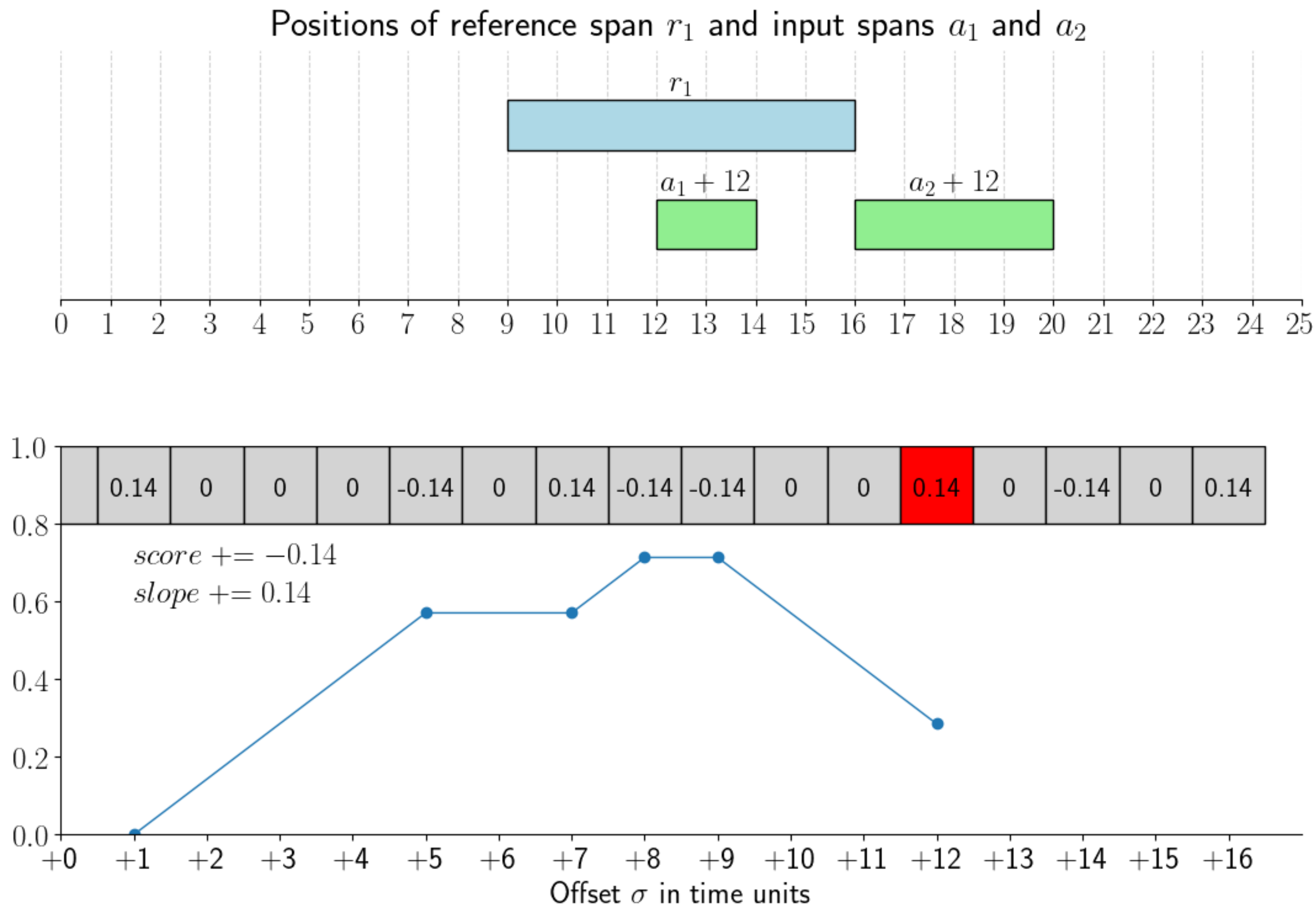
Optimal no-split alignment



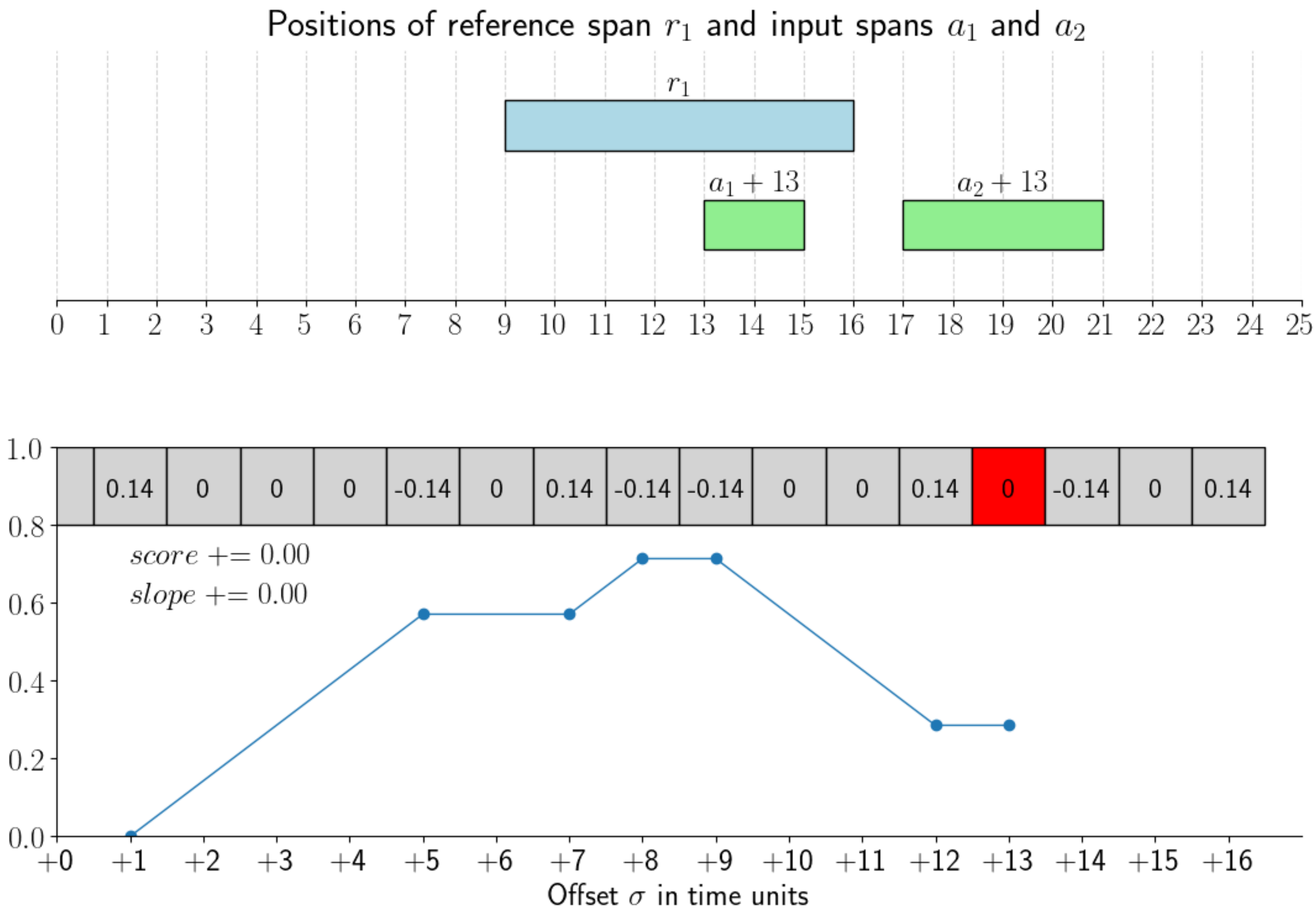
Optimal no-split alignment



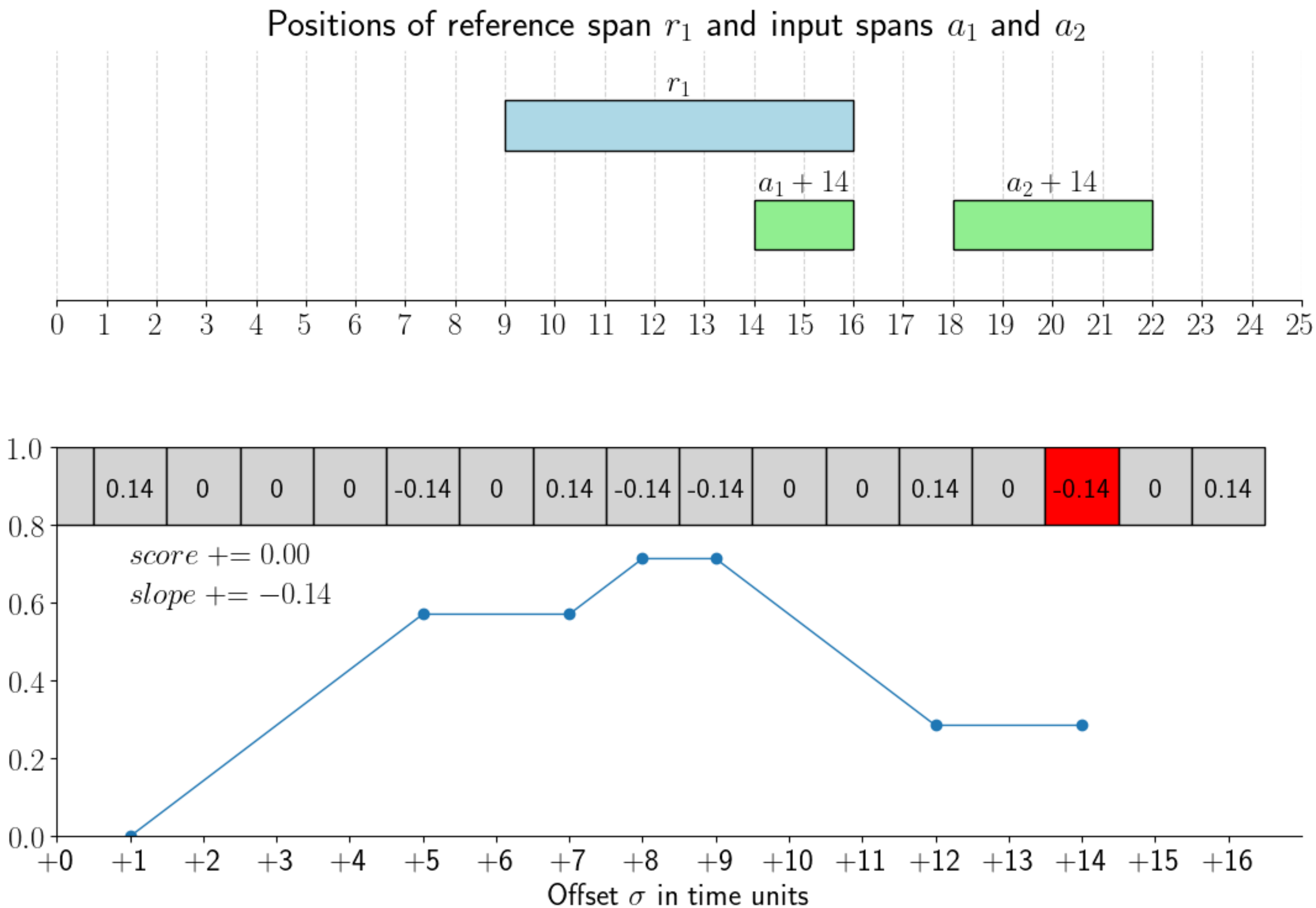
Optimal no-split alignment



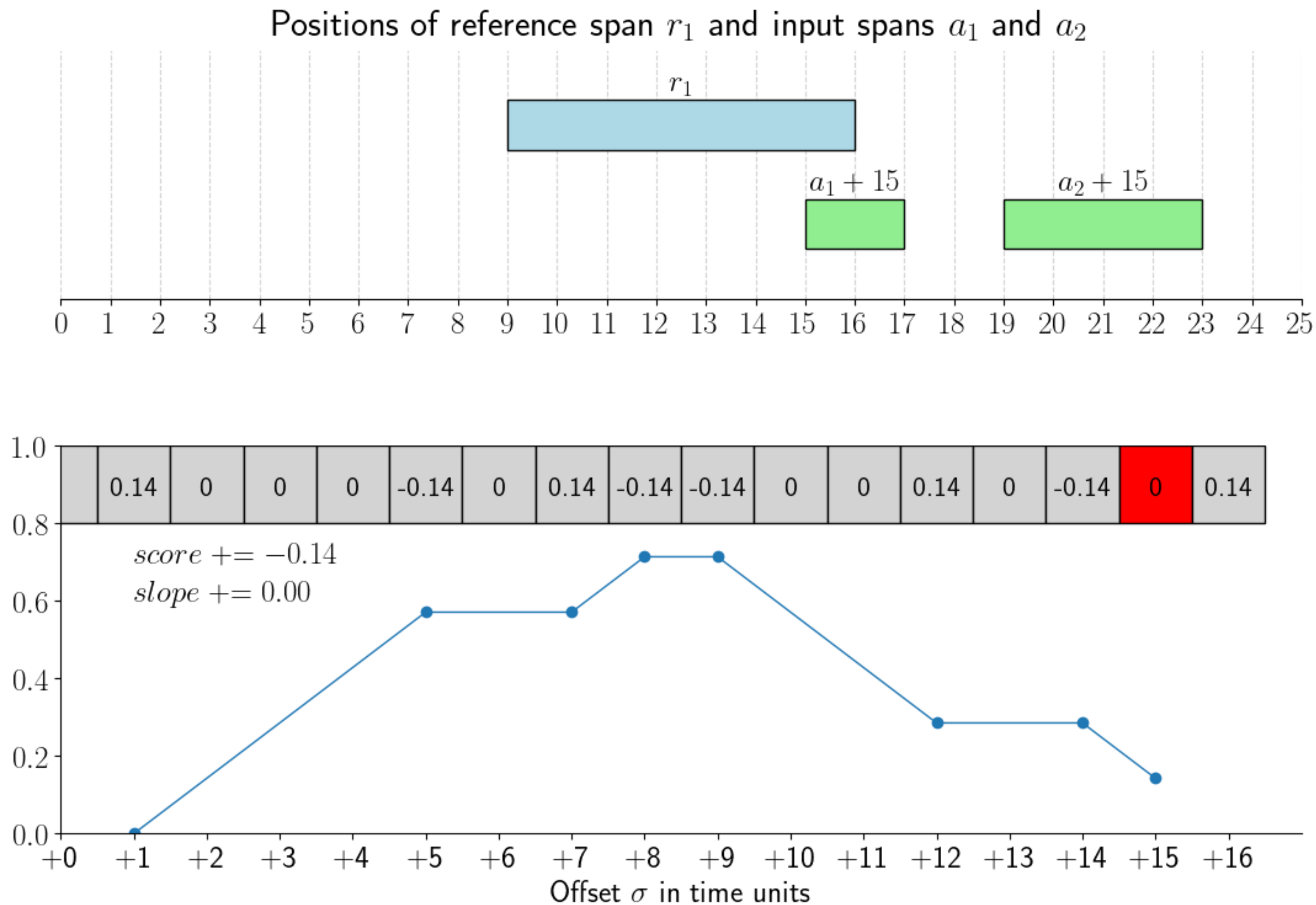
Optimal no-split alignment



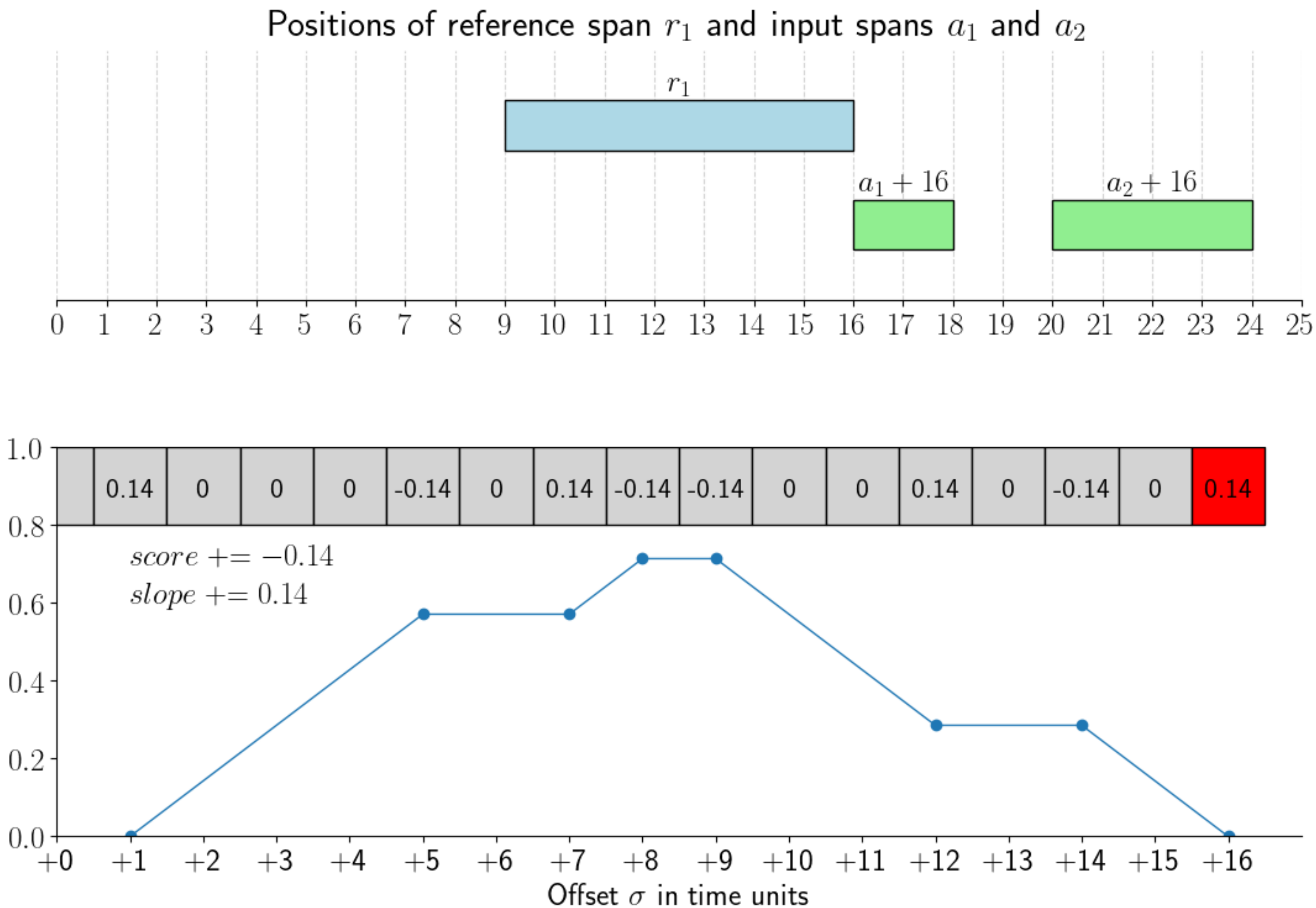
Optimal no-split alignment



Optimal no-split alignment



Optimal no-split alignment



Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations

Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations
- runtime complexity: $O(KN + T_r + T_a)$
- space complexity: $O(T_r + T_a)$

Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations
- runtime complexity: $O(KN + T_r + T_a)$
- space complexity: $O(T_r + T_a)$

Variation of the algorithm which sorts $4KN$ slope changes with merge sort:

Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations
- runtime complexity: $O(KN + T_r + T_a)$
- space complexity: $O(T_r + T_a)$

Variation of the algorithm which sorts $4KN$ slope changes with merge sort:

- worst-case complexity $O(KN \log \min(K, N))$

Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations
- runtime complexity: $O(KN + T_r + T_a)$
- space complexity: $O(T_r + T_a)$

Variation of the algorithm which sorts $4KN$ slope changes with merge sort:

- worst-case complexity $O(KN \log \min(K, N)) \rightarrow$ 3 times slower on real data

Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations
- runtime complexity: $O(KN + T_r + T_a)$
- space complexity: $O(T_r + T_a)$

Variation of the algorithm which sorts $4KN$ slope changes with merge sort:

- worst-case complexity $O(KN \log \min(K, N)) \rightarrow$ 3 times slower on real data
- space complexity $O(KN)$

Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations
- runtime complexity: $O(KN + T_r + T_a)$
- space complexity: $O(T_r + T_a)$

Variation of the algorithm which sorts $4KN$ slope changes with merge sort:

- worst-case complexity $O(KN \log \min(K, N)) \rightarrow$ 3 times slower on real data
- space complexity $O(KN) \rightarrow$ independent of timestamps

Optimal no-split alignment

Analysis of the slope tracking algorithm

- $4KN$ "insertions"
- $T_r + T_a$ iterations
- runtime complexity: $O(KN + T_r + T_a)$
- space complexity: $O(T_r + T_a)$

Variation of the algorithm which sorts $4KN$ slope changes with merge sort:

- worst-case complexity $O(KN \log \min(K, N)) \rightarrow$ 3 times slower on real data
- space complexity $O(KN) \rightarrow$ independent of timestamps

Hybrid: Switch depending on the ratio $\frac{4KN}{T_r + T_a}$!

Optimal split alignment

Split alignment σ

Given the input span sequence $a = (a_1, \ldots, a_N)$, a *split alignment* is a sequence of offsets $\sigma = (\sigma_1, \ldots, \sigma_N)$ which does not reorder the input sequence:

Optimal split alignment

Split alignment σ

Given the input span sequence $a = (a_1, \ldots, a_N)$, a *split alignment* is a sequence of offsets $\sigma = (\sigma_1, \ldots, \sigma_N)$ which does not reorder the input sequence:

$$\begin{gathered}
\mathtt{end}(a_1 + \sigma_1) \leq \mathtt{start}(a_2 + \sigma_2) \\
\mathtt{end}(a_2 + \sigma_2) \leq \mathtt{start}(a_3 + \sigma_3) \\
\vdots \\
\mathtt{end}(a_{N-1} + \sigma_{N-1}) \leq \mathtt{start}(a_N + \sigma_N)
\end{gathered}$$

Optimal split alignment

Number of splits in σ

The *number of splits* of the alignment, $\texttt{splits}(\sigma)$ is defined as

$$\texttt{splits}(\sigma) = \sum_{n=1}^{N-1} \begin{cases} 1 & \text{if } \sigma_n \neq \sigma_{n+1} \\ 0 & \text{if } \sigma_n = \sigma_{n+1} \end{cases}$$

Optimal split alignment

Number of splits in σ

The *number of splits* of the alignment, $\texttt{splits}(\sigma)$ is defined as

$$\texttt{splits}(\sigma) = \sum_{n=1}^{N-1} \begin{cases} 1 & \text{if } \sigma_n \neq \sigma_{n+1} \\ 0 & \text{if } \sigma_n = \sigma_{n+1} \end{cases}$$

Scoring for split alignments

Given two sequences of spans $r = (r_1, r_2, \ldots, r_K)$ and $a = (a_1, a_2, \ldots, a_N)$, a weighting function w and the split penalty p, the $\texttt{score}$ of an alignment $\sigma = (\sigma_1, \ldots, \sigma_N)$ is defined as

$$\texttt{score}(r, a, \sigma, p, w) = \sum_{n=1}^{N} \sum_{k=1}^{K} \texttt{iscore}(r_k, a_n + \sigma_n) \cdot w(k, n) - \texttt{splits}(\sigma) \cdot p$$

Finding the optimal split alignment

Optimal split alignment

Finding the optimal split alignment

Very difficult problem!

Finding the optimal split alignment

Very difficult problem!

- Enumerating all alignments: $O(\binom{T_r+T_a}{N})$

Finding the optimal split alignment

Very difficult problem!

- Enumerating all alignments: $O(\binom{T_r+T_a}{N}) \approx 10^{5700}$ years

Optimal split alignment

Finding the optimal split alignment

Very difficult problem!

- Enumerating all alignments: $O(\binom{T_r + T_a}{N}) \approx 10^{5700}$ years
- Enumerating all splits: $O(2^N)$

Optimal split alignment

Finding the optimal split alignment

Very difficult problem!

- Enumerating all alignments: $O(\binom{T_r+T_a}{N}) \approx 10^{5700}$ years
- Enumerating all splits: $O(2^N) \approx 10^{370}$ years

Finding the optimal split alignment

Very difficult problem!

- Enumerating all alignments: $O(\binom{T_r+T_a}{N}) \approx 10^{5700}$ years
- Enumerating all splits: $O(2^N) \approx 10^{370}$ years
- Can we do it in under 5 minutes?

Finding the optimal split alignment

Very difficult problem!

- Enumerating all alignments: $O(\binom{T_r+T_a}{N}) \approx 10^{5700}$ years
- Enumerating all splits: $O(2^N) \approx 10^{370}$ years
- Can we do it in under 5 minutes? Yes!

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \mathtt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \texttt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} + \texttt{end}(a_{n-1}) \leq \sigma'_n + \texttt{start}(a_n)}} \texttt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \texttt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \leq \sigma'_n + \texttt{start}(a_n) - \texttt{end}(a_{n-1})}} \texttt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \texttt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \leq \sigma'_n + \texttt{extra}(n)}} \texttt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n=\sigma'_n}} \texttt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1}\leq\sigma'_n+\texttt{extra}(n)}} \texttt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \texttt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \leq \sigma'_n + \texttt{extra}(n)}} \texttt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Recursion formula for t_n

$$t_n(\sigma'_n) = \overbrace{\texttt{score}(r, (a_n), \sigma'_n)}^{\text{detach } a_n} +$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \texttt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \leq \sigma'_n + \texttt{extra}(n)}} \texttt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Recursion formula for t_n

$$t_n(\sigma'_n) = \overbrace{\texttt{score}(r, (a_n), \sigma'_n)}^{\text{detach } a_n} + \max \begin{cases} t_{n-1}(\sigma'_n) & \text{no-split...} \\ \end{cases}$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \mathtt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \leq \sigma'_n + \mathtt{extra}(n)}} \mathtt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Recursion formula for t_n

$$t_n(\sigma'_n) = \overbrace{\mathtt{score}(r, (a_n), \sigma'_n)}^{\text{detach } a_n} + \max \begin{cases} t_{n-1}(\sigma'_n) & \text{no-split...} \\ s_n(\sigma'_n) - p & \text{...or split?} \end{cases}$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \texttt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \leq \sigma'_n + \texttt{extra}(n)}} \texttt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Recursion formula for s_n

$$s_n(\sigma'_n) = \max \begin{cases} & \sigma_{n-1} = \sigma'_n + \texttt{extra}(n) \\ & \sigma_{n-1} < \sigma'_n + \texttt{extra}(n) \end{cases}$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n=\sigma'_n}} \mathtt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \le \sigma'_n + \mathtt{extra}(n)}} \mathtt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Recursion formula for s_n

$$s_n(\sigma'_n) = \max \begin{cases} t_{n-1}(\sigma'_n + \mathtt{extra}(n)) & \sigma_{n-1} = \sigma'_n + \mathtt{extra}(n) \\ & \sigma_{n-1} < \sigma'_n + \mathtt{extra}(n) \end{cases}$$

Optimal split alignment

Recursion form

$$t_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_n) \\ \text{where } \sigma_n = \sigma'_n}} \mathtt{score}(r, (a_1, \ldots, a_n), (\sigma_1, \ldots, \sigma_n))$$

$$s_n(\sigma'_n) = \max_{\substack{(\sigma_1,\ldots,\sigma_{n-1}) \text{ where} \\ \sigma_{n-1} \leq \sigma'_n + \mathtt{extra}(n)}} \mathtt{score}(r, (a_1, \ldots, a_{n-1}), (\sigma_1, \ldots, \sigma_{n-1}))$$

Recursion formula for s_n

$$s_n(\sigma'_n) = \max \begin{cases} t_{n-1}(\sigma'_n + \mathtt{extra}(n)) & \sigma_{n-1} = \sigma'_n + \mathtt{extra}(n) \\ s_n(\sigma'_n - 1) & \sigma_{n-1} < \sigma'_n + \mathtt{extra}(n) \end{cases}$$

Optimal split alignment

Recursion formula

$$t_n(\sigma'_n) = \texttt{score}(r, (a_n), \sigma'_n) + \max \begin{cases} t_{n-1}(\sigma'_n) \\ s_n(\sigma'_n) - p \end{cases}$$

$$s_n(\sigma'_n) = \max \begin{cases} t_{n-1}(\sigma'_n + \texttt{extra}(n)) \\ s_n(\sigma'_n - 1) \end{cases}$$

Optimal split alignment

Recursion formula

$$t_n(\sigma'_n) = \texttt{score}(r, (a_n), \sigma'_n) + \max \begin{cases} t_{n-1}(\sigma'_n) \\ s_n(\sigma'_n) - p \end{cases}$$

$$s_n(\sigma'_n) = \max \begin{cases} t_{n-1}(\sigma'_n + \texttt{extra}(n)) \\ s_n(\sigma'_n - 1) \end{cases}$$

Culmulative recursion formula

$$t_n = \texttt{score}(r, (a_n), \_) + \max(t_{n-1}, s_n - p)$$

Optimal split alignment

Recursion formula

$$t_n(\sigma'_n) = \texttt{score}(r, (a_n), \sigma'_n) + \max \begin{cases} t_{n-1}(\sigma'_n) \\ s_n(\sigma'_n) - p \end{cases}$$

$$s_n(\sigma'_n) = \max \begin{cases} t_{n-1}(\sigma'_n + \texttt{extra}(n)) \\ s_n(\sigma'_n - 1) \end{cases}$$

Culmulative recursion formula

$$\begin{aligned} t_n &= \texttt{score}(r, (a_n), \_) + \max(t_{n-1}, s_n - p) \\ s_n &= \qquad\qquad \texttt{shift}(t_{n-1}, -\texttt{extra}(n)) \end{aligned}$$

Optimal split alignment

Recursion formula

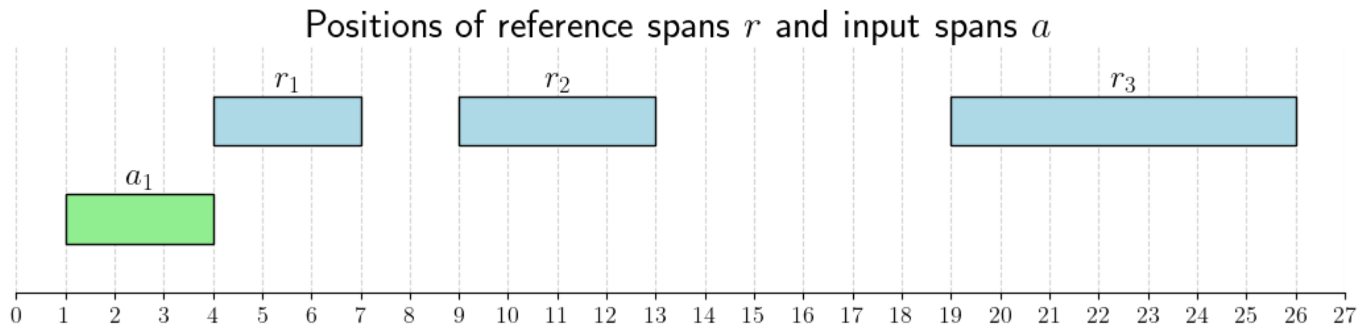
$$t_n(\sigma'_n) = \texttt{score}(r, (a_n), \sigma'_n) + \max \begin{cases} t_{n-1}(\sigma'_n) \\ s_n(\sigma'_n) - p \end{cases}$$

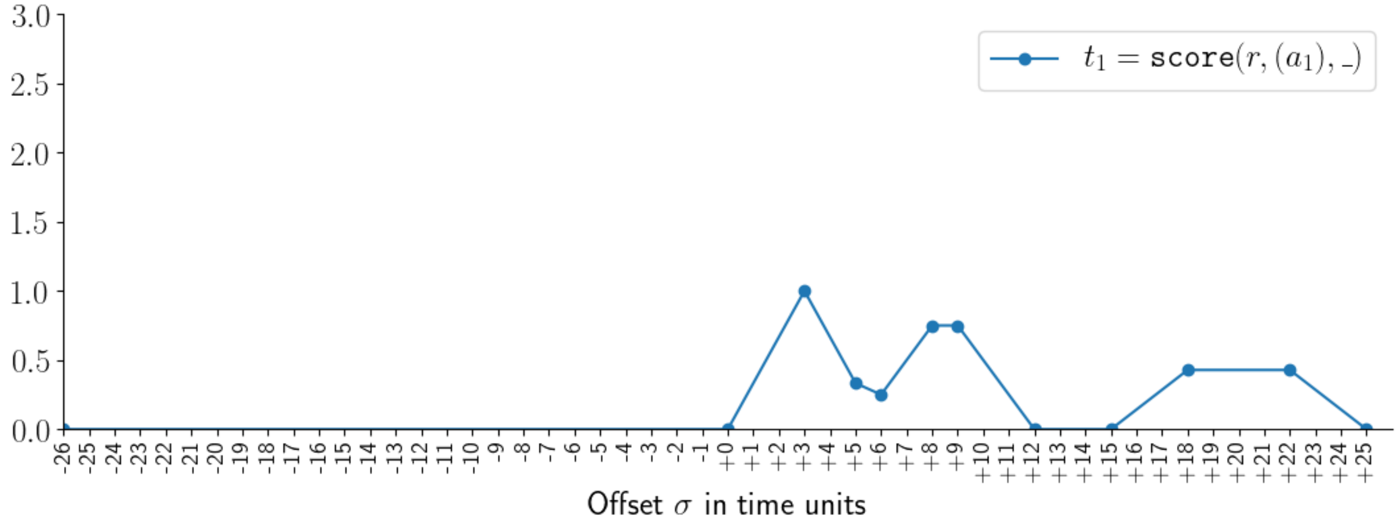
$$s_n(\sigma'_n) = \max \begin{cases} t_{n-1}(\sigma'_n + \texttt{extra}(n)) \\ s_n(\sigma'_n - 1) \end{cases}$$

Culmulative recursion formula

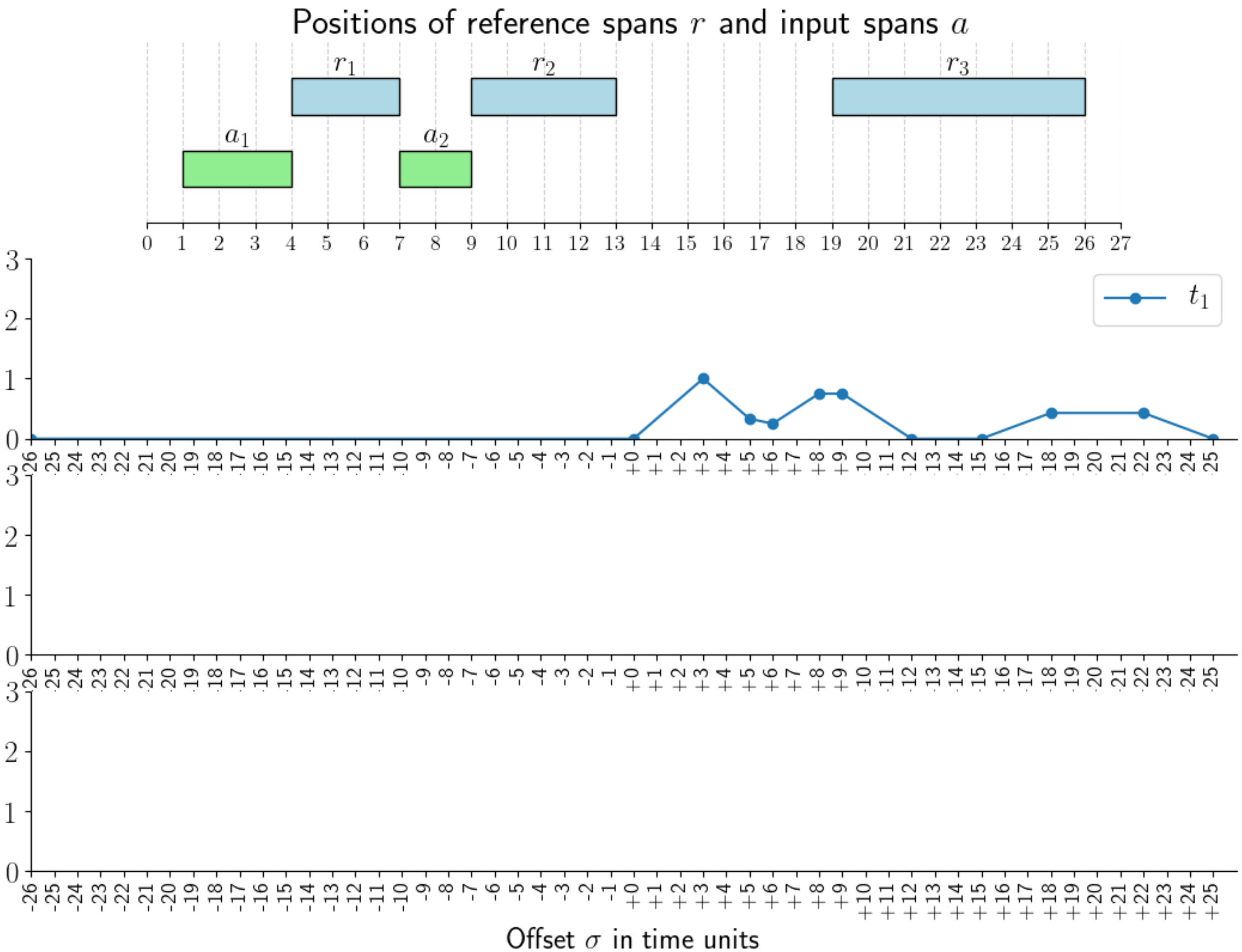
$$\begin{aligned} t_n &= \texttt{score}(r, (a_n), \_) + \max(t_{n-1}, s_n - p) \\ s_n &= \texttt{left\_to\_right\_max}(\texttt{shift}(t_{n-1}, -\texttt{extra}(n))) \end{aligned}$$

Optimal split alignment

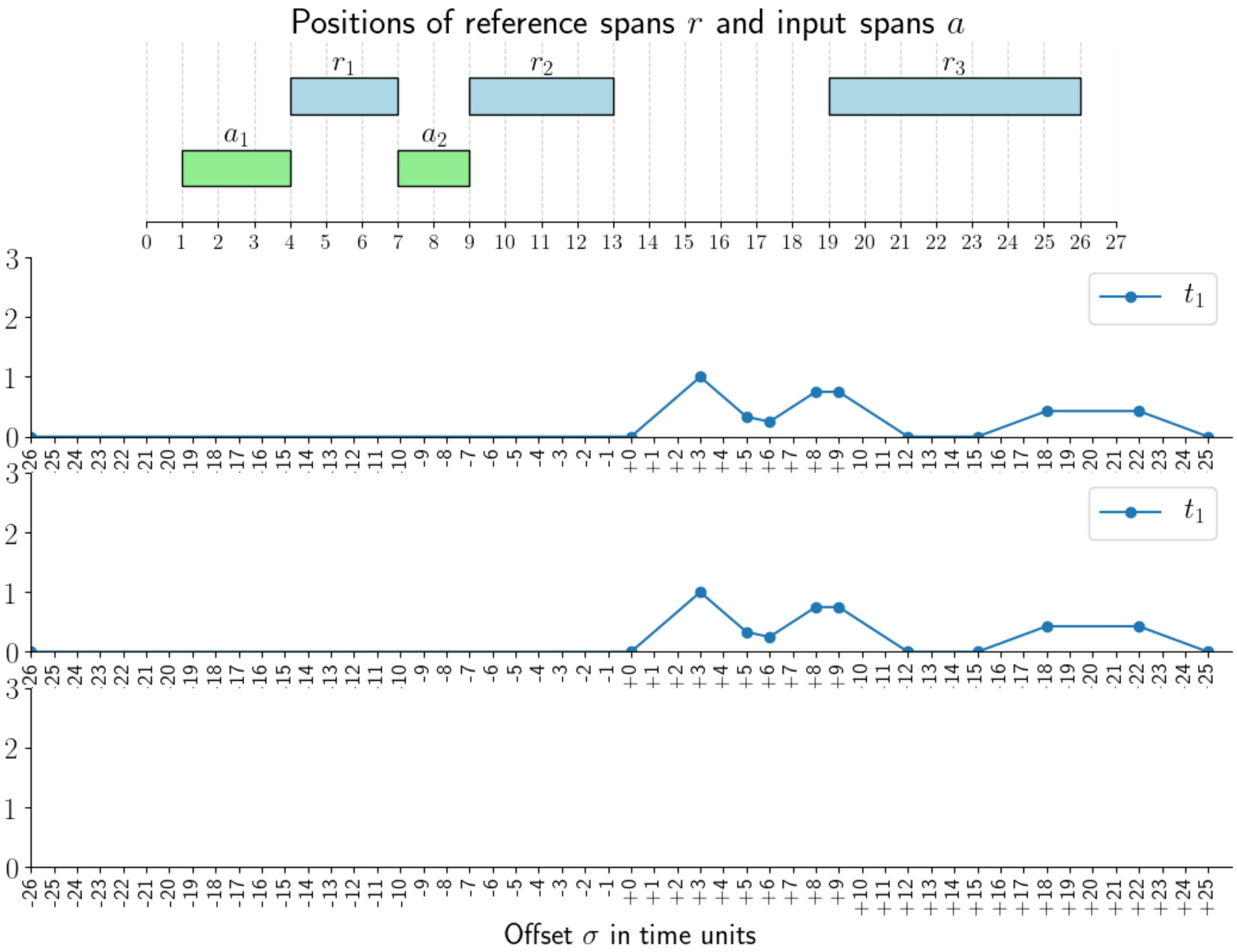




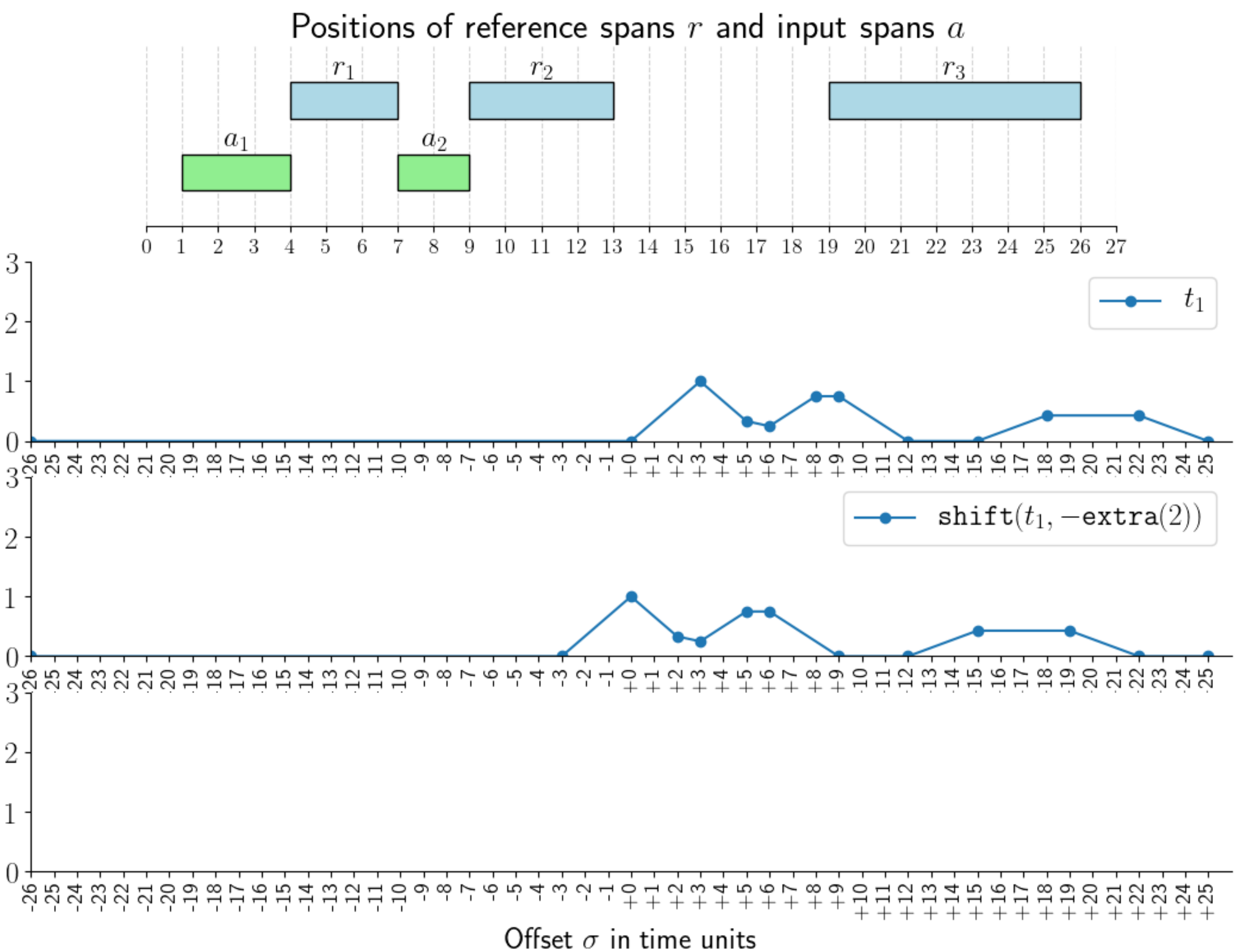
Optimal split alignment



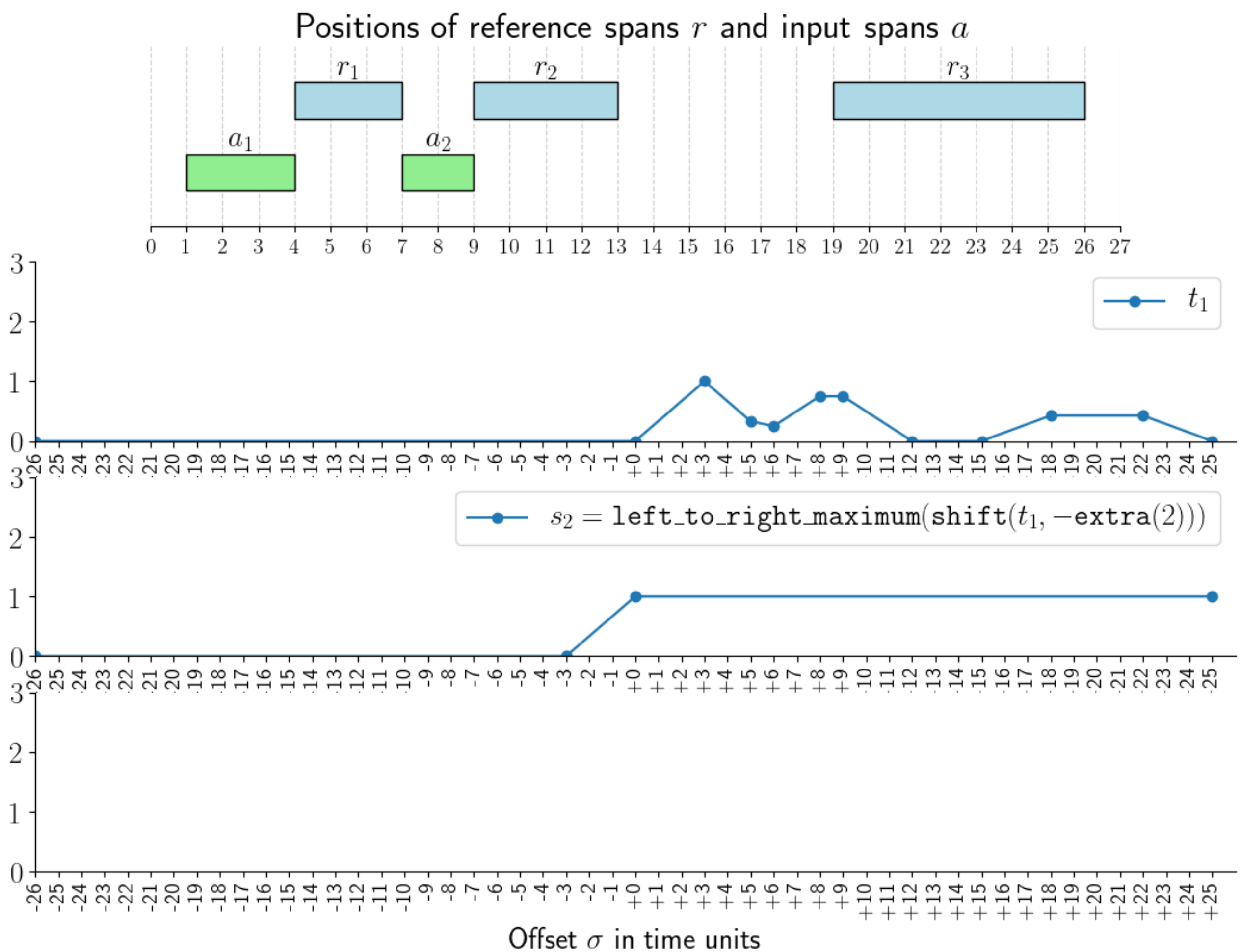
Optimal split alignment



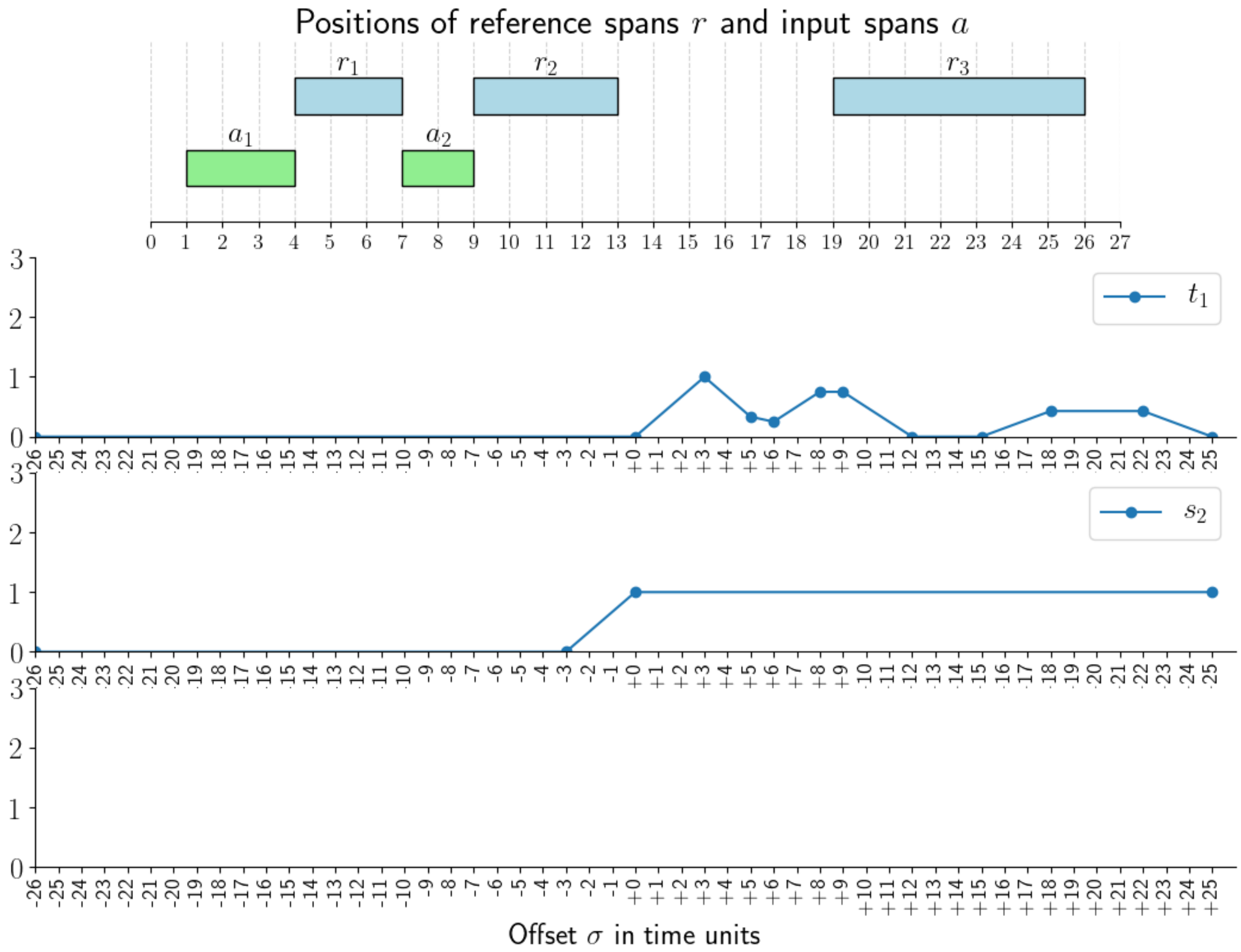
Optimal split alignment



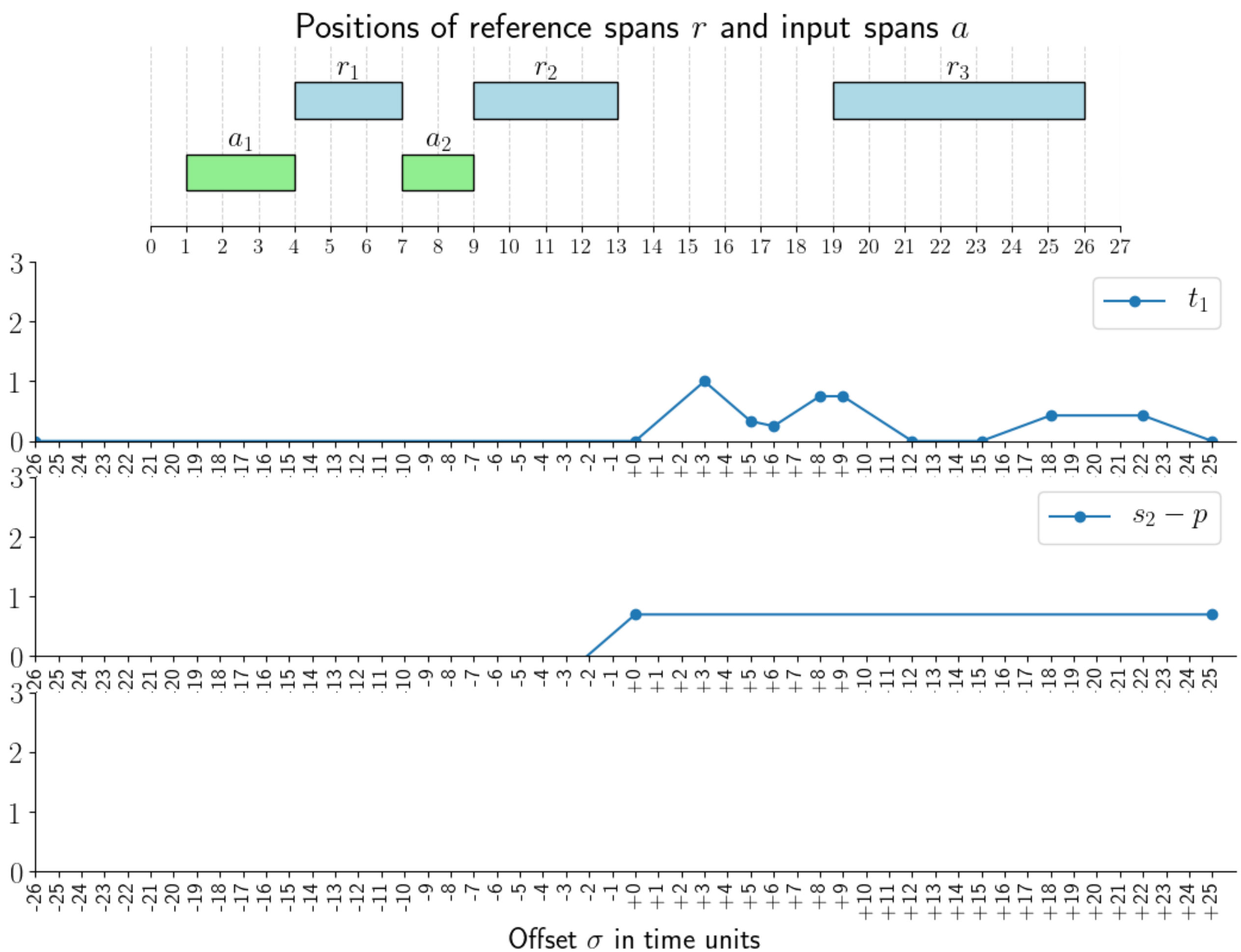
Optimal split alignment



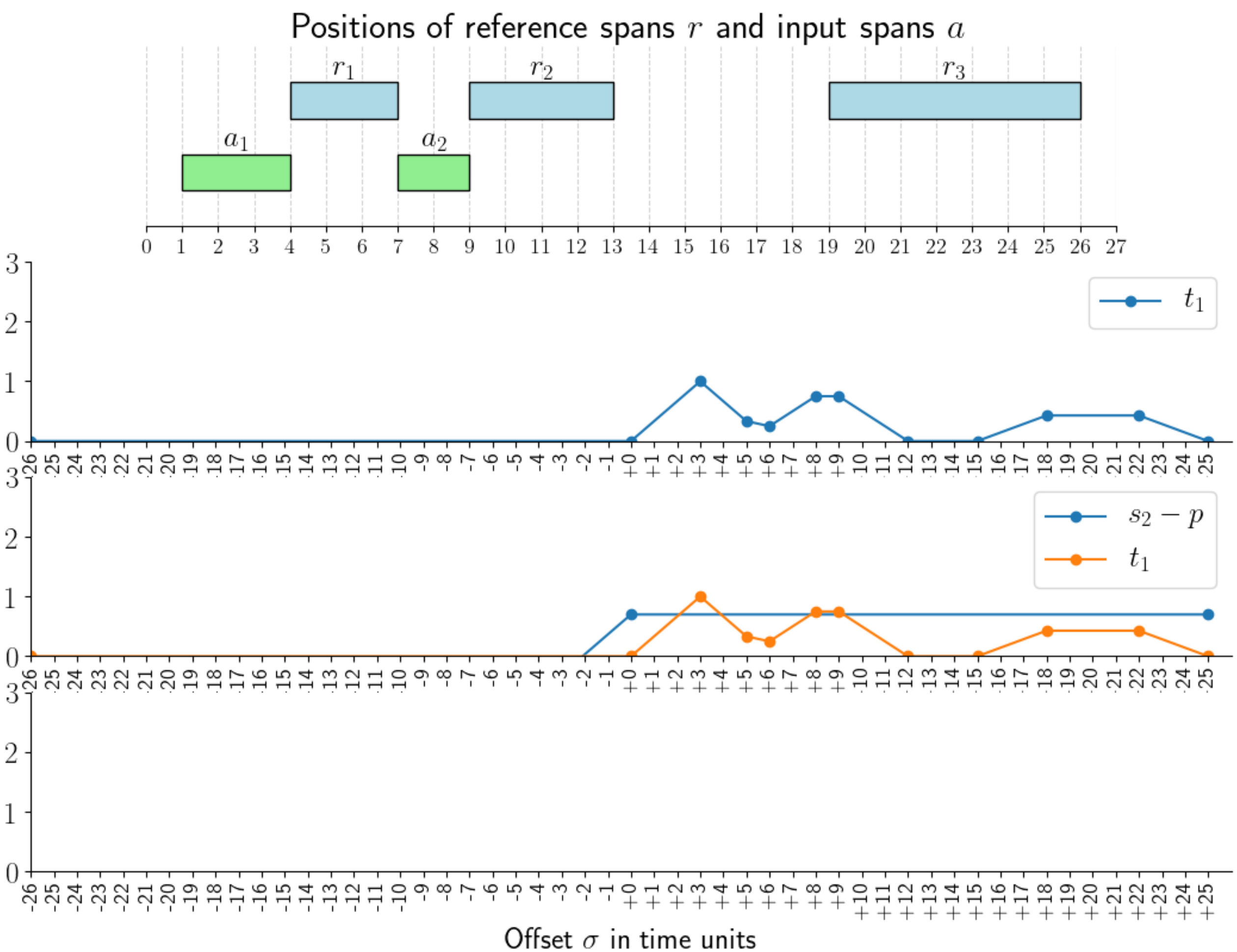
Optimal split alignment



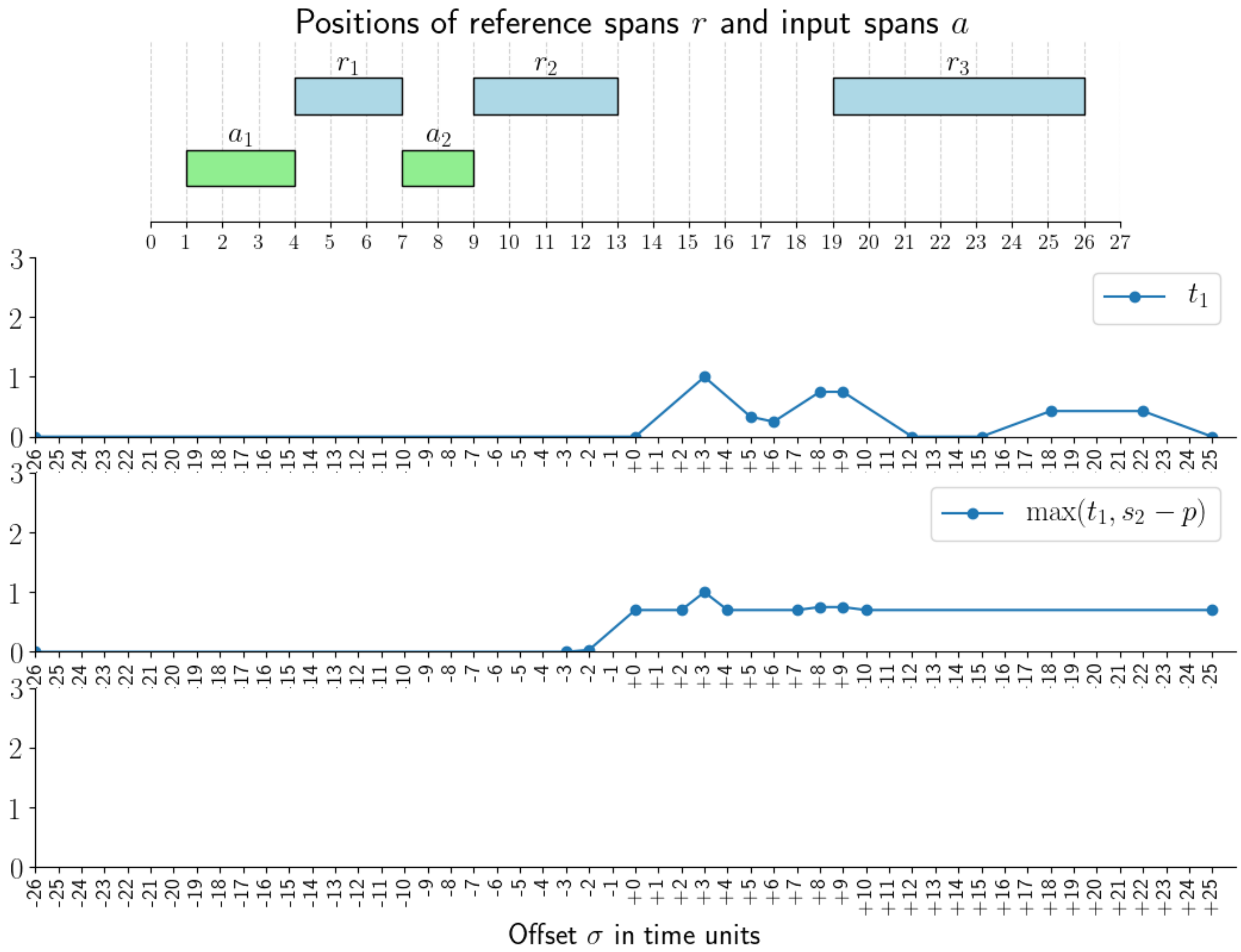
Optimal split alignment



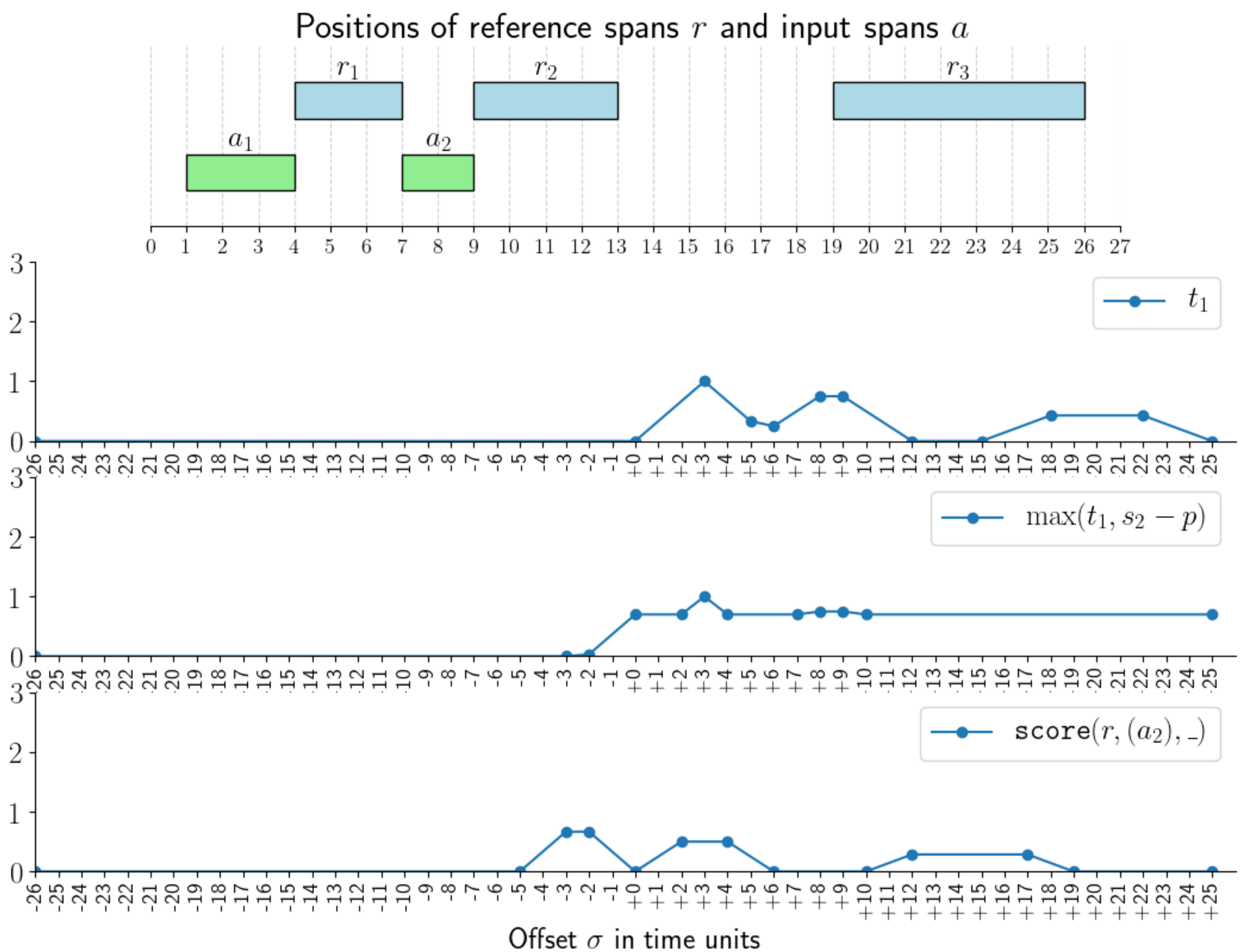
Optimal split alignment



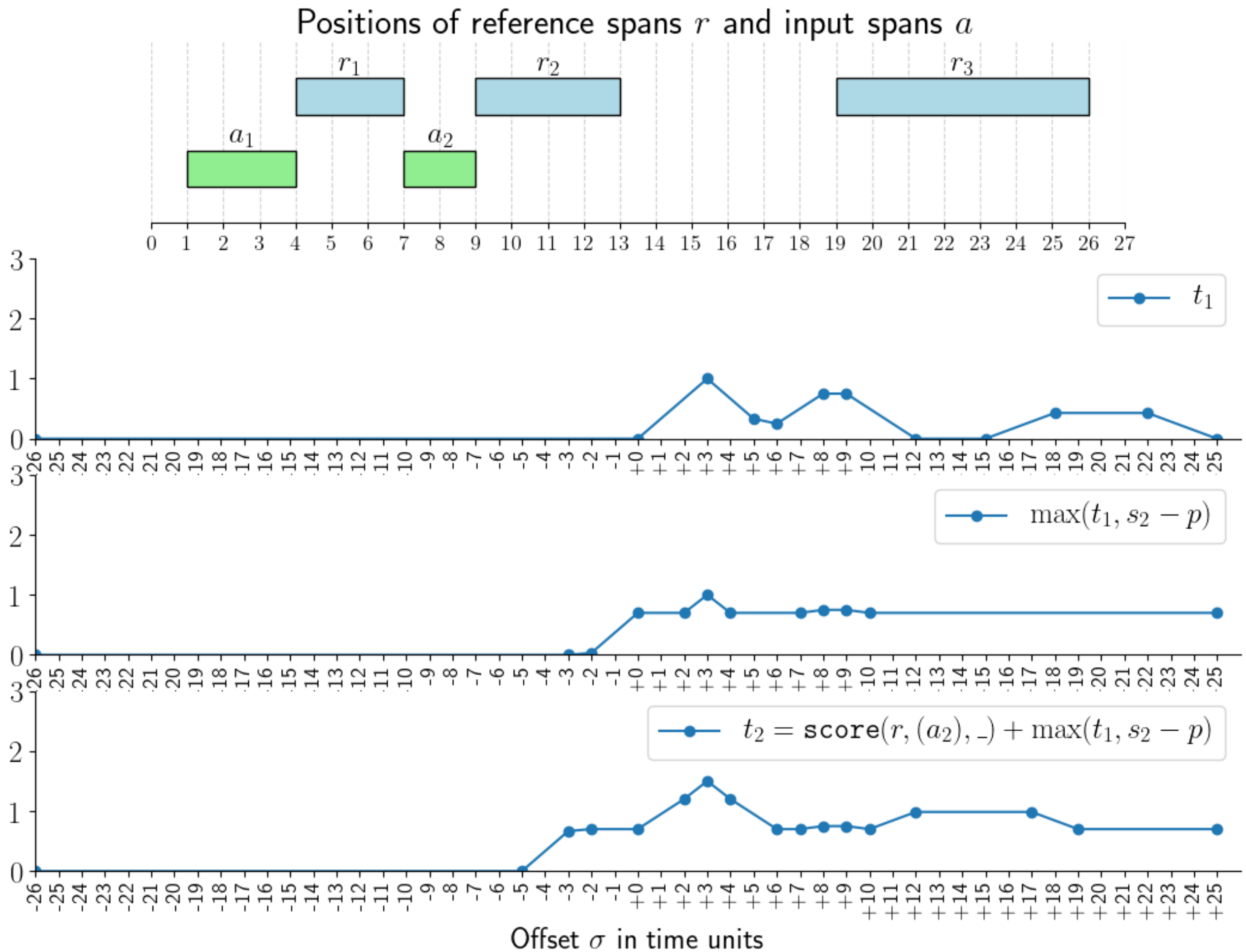
Optimal split alignment



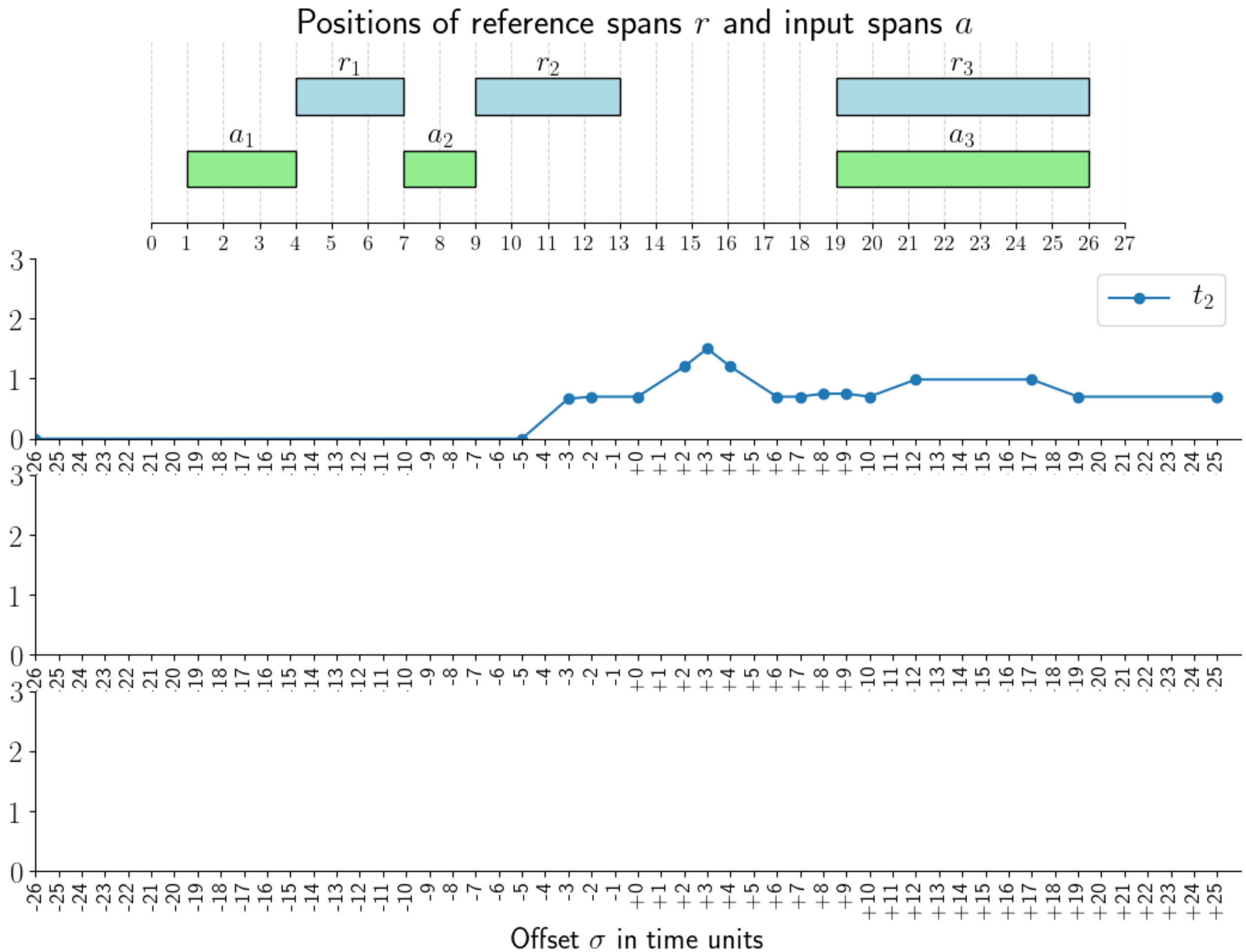
Optimal split alignment



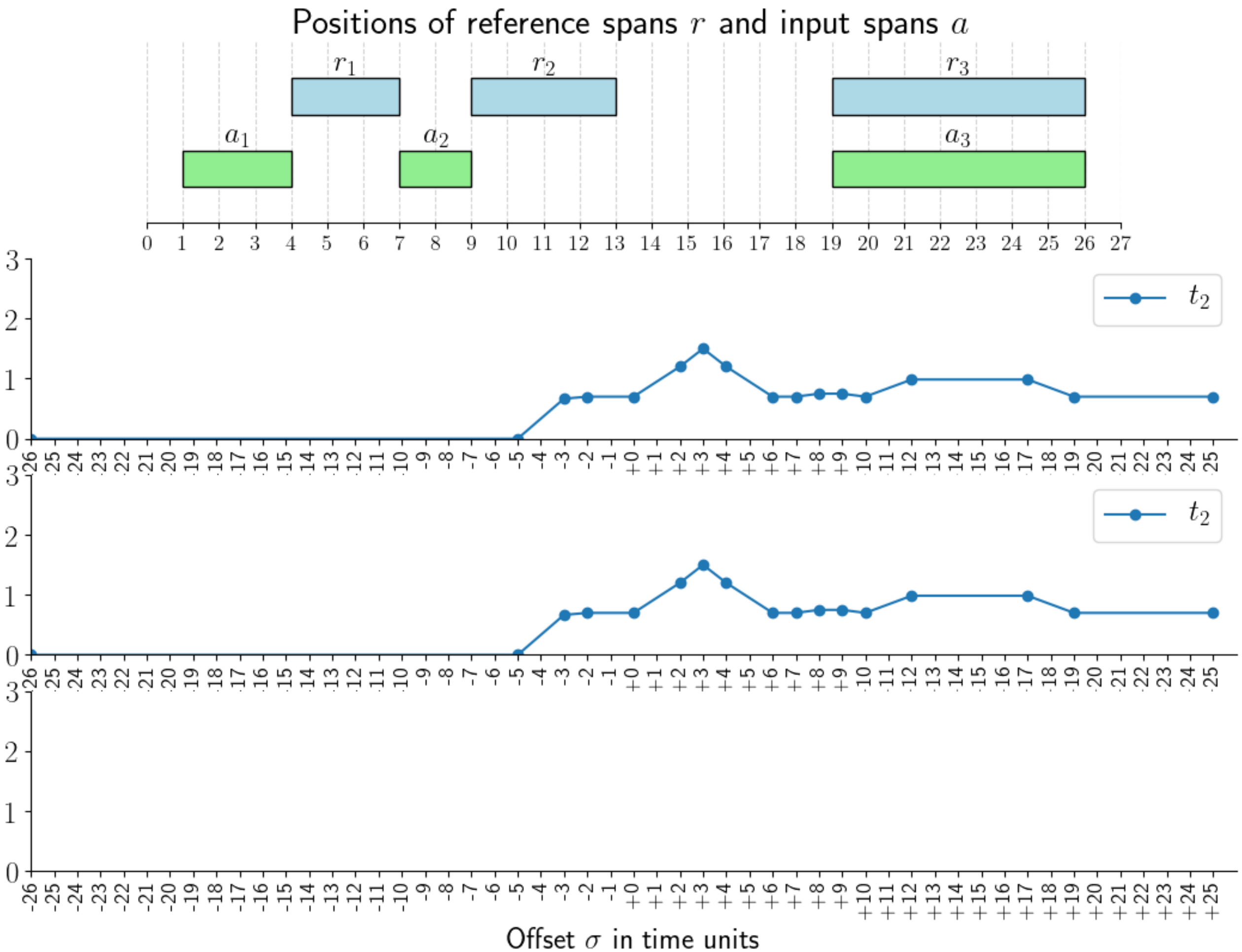
Optimal split alignment



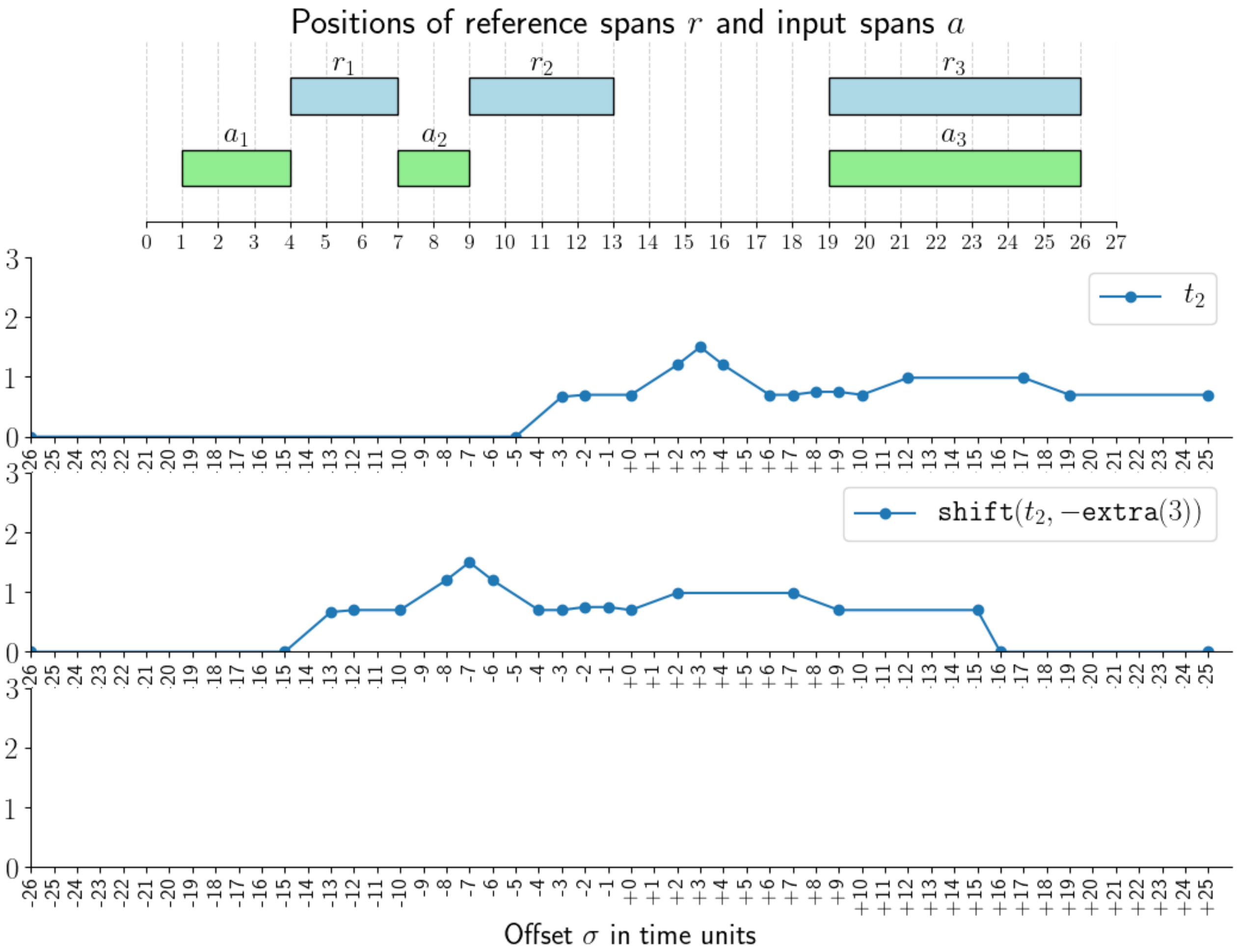
Optimal split alignment



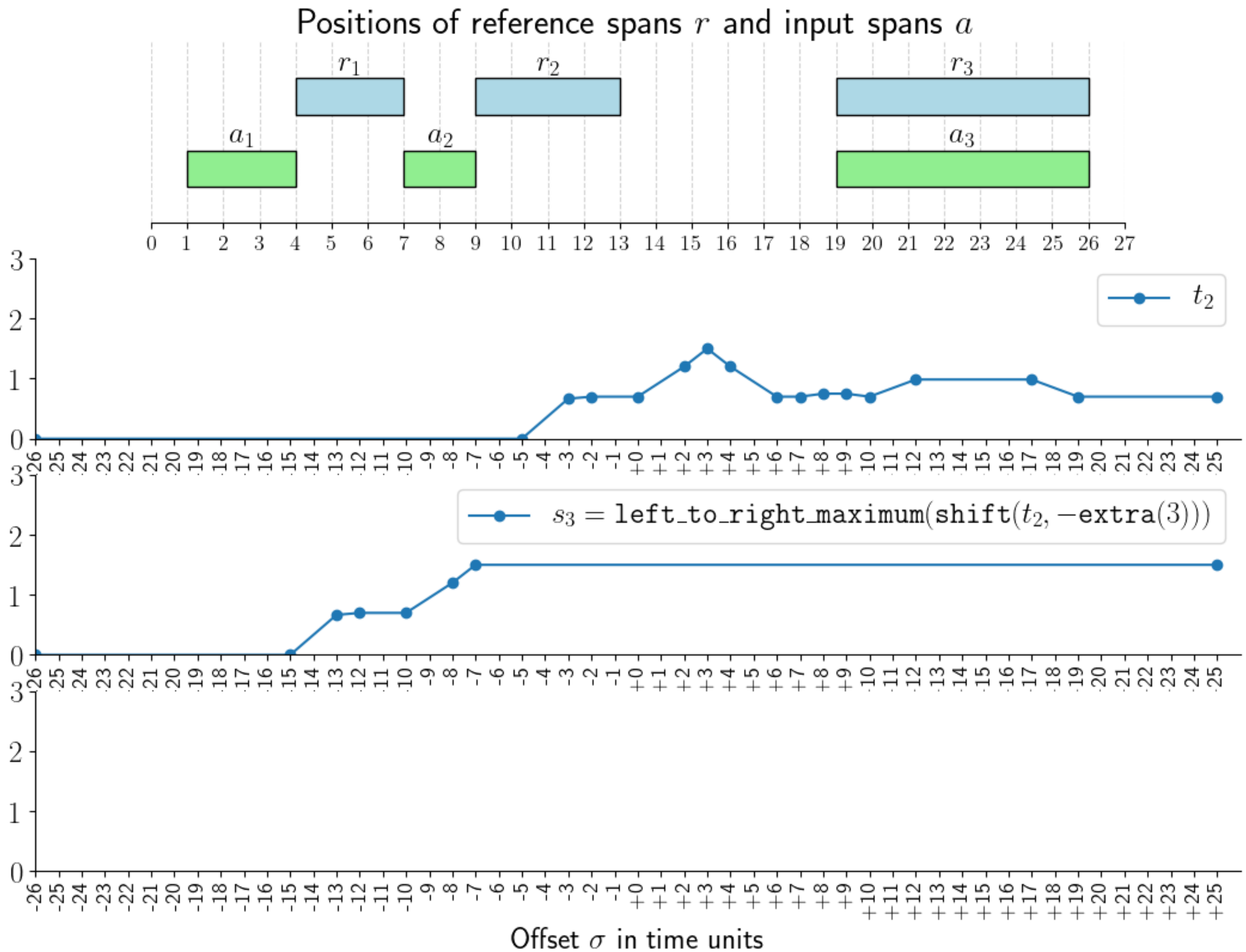
Optimal split alignment



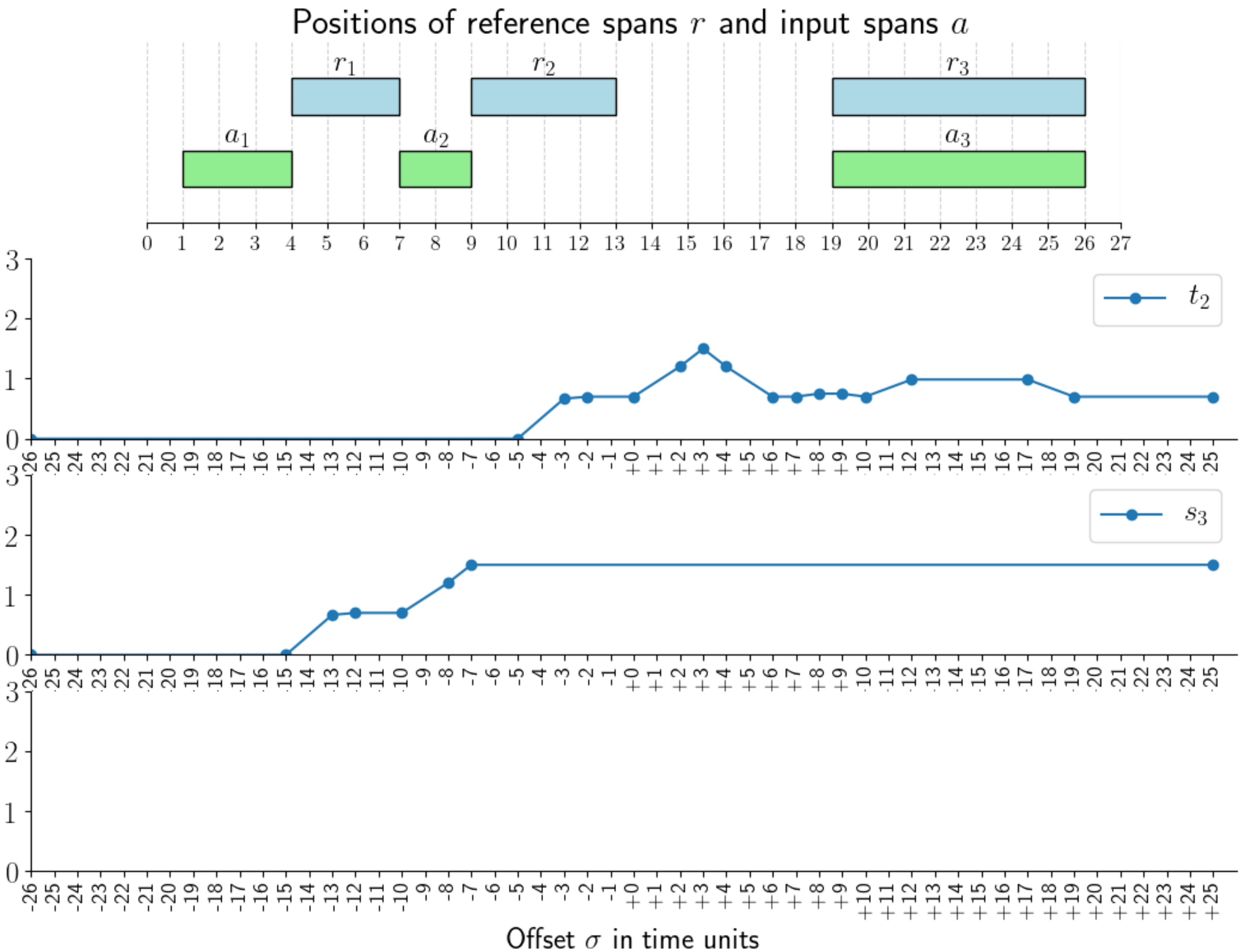
Optimal split alignment



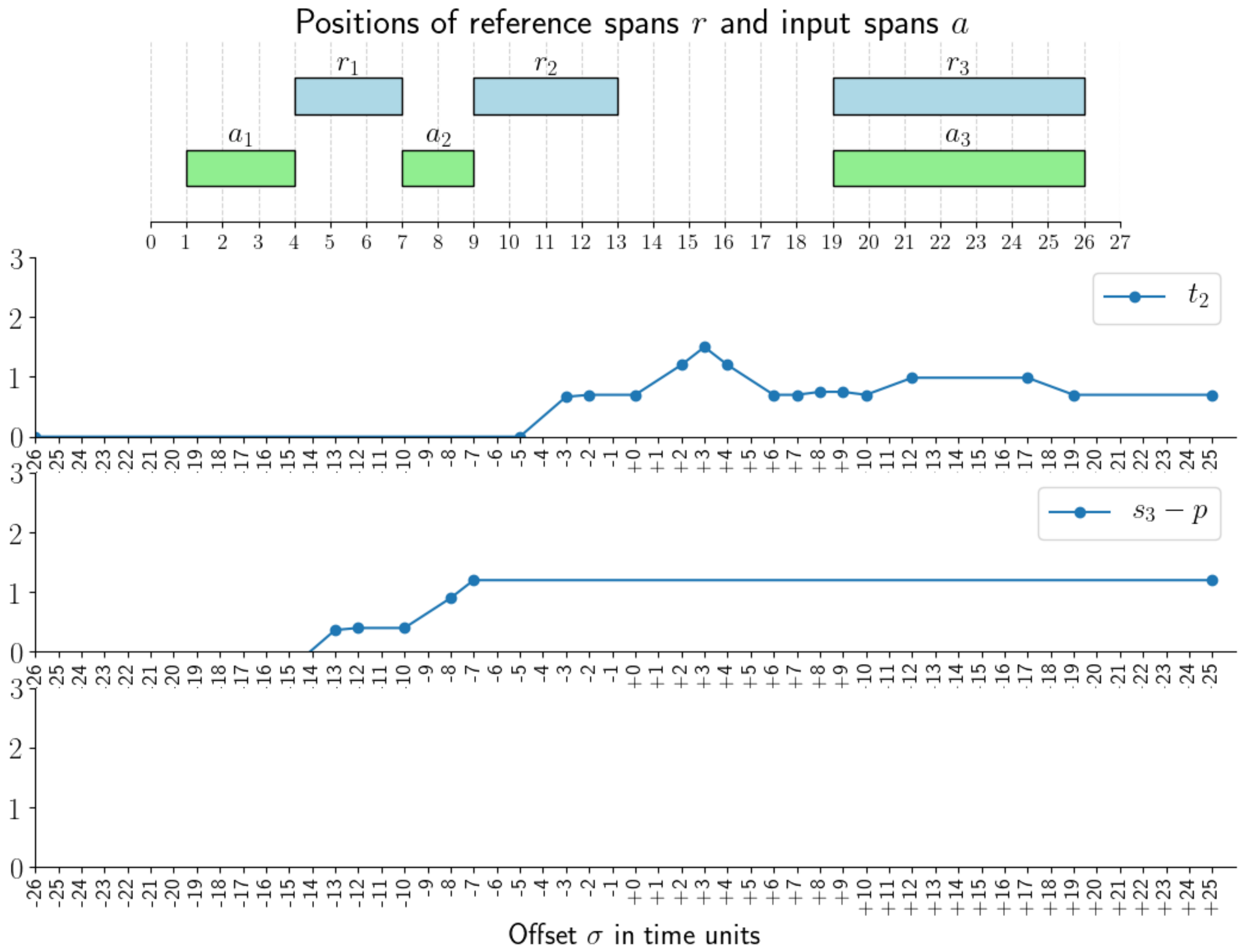
Optimal split alignment



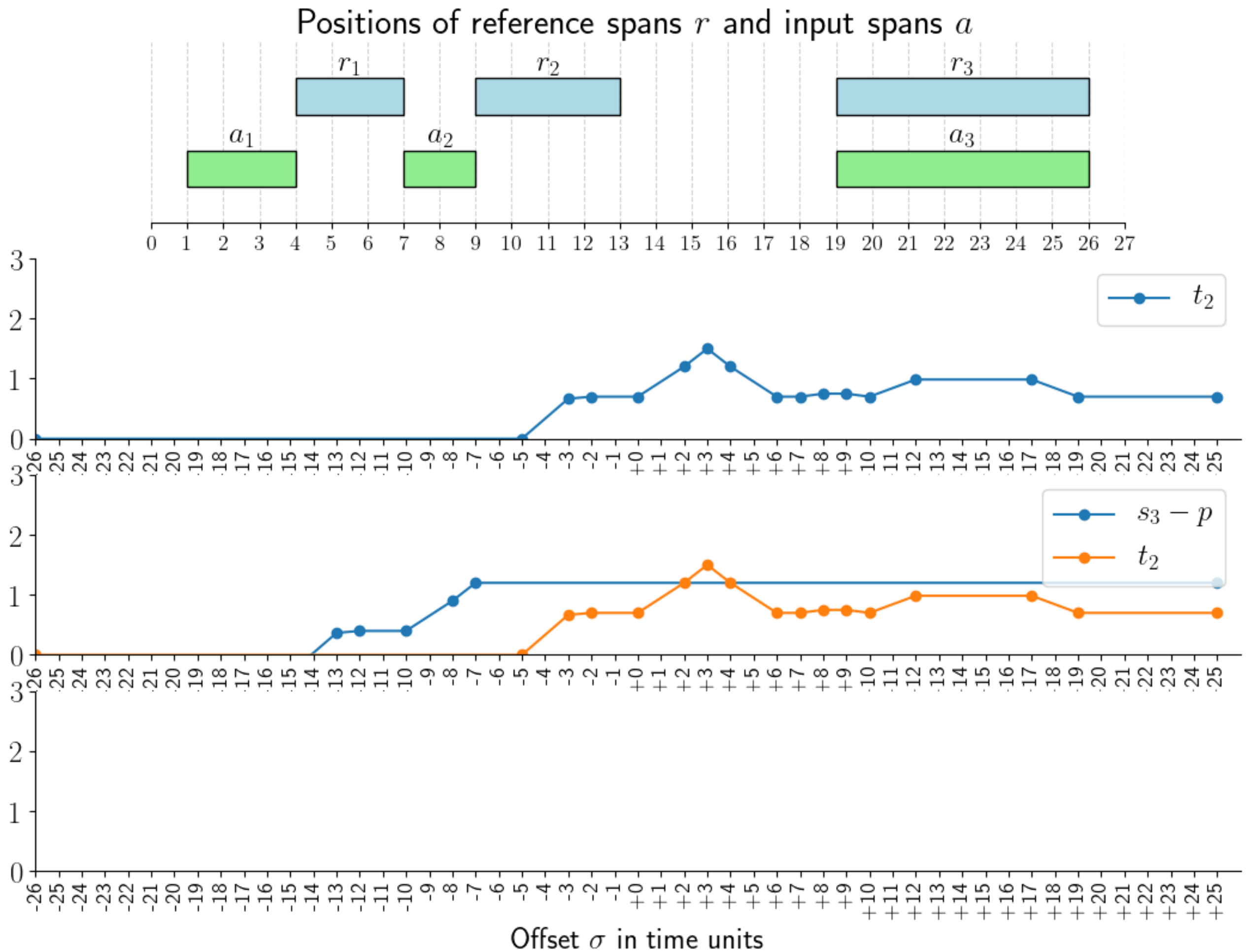
Optimal split alignment



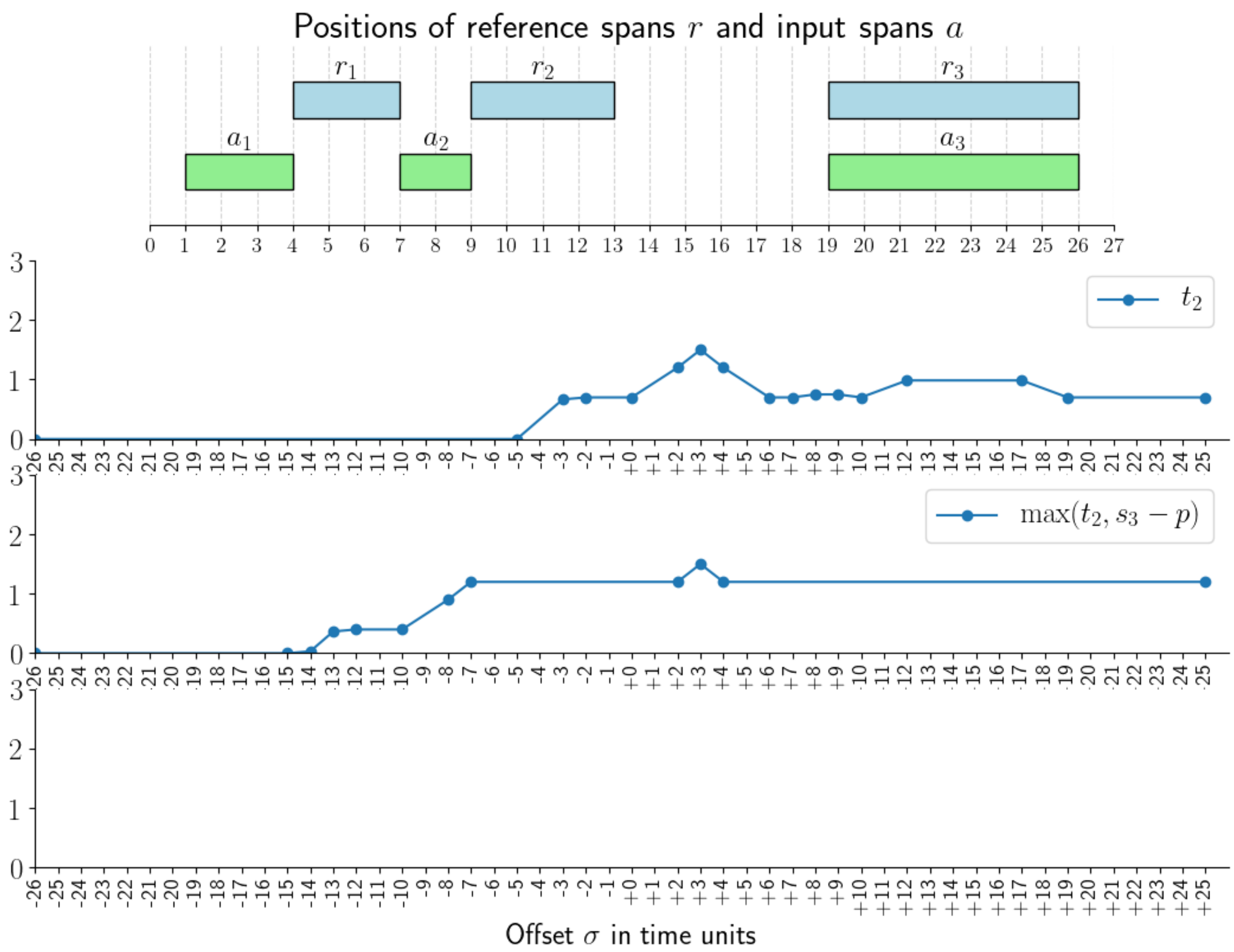
Optimal split alignment



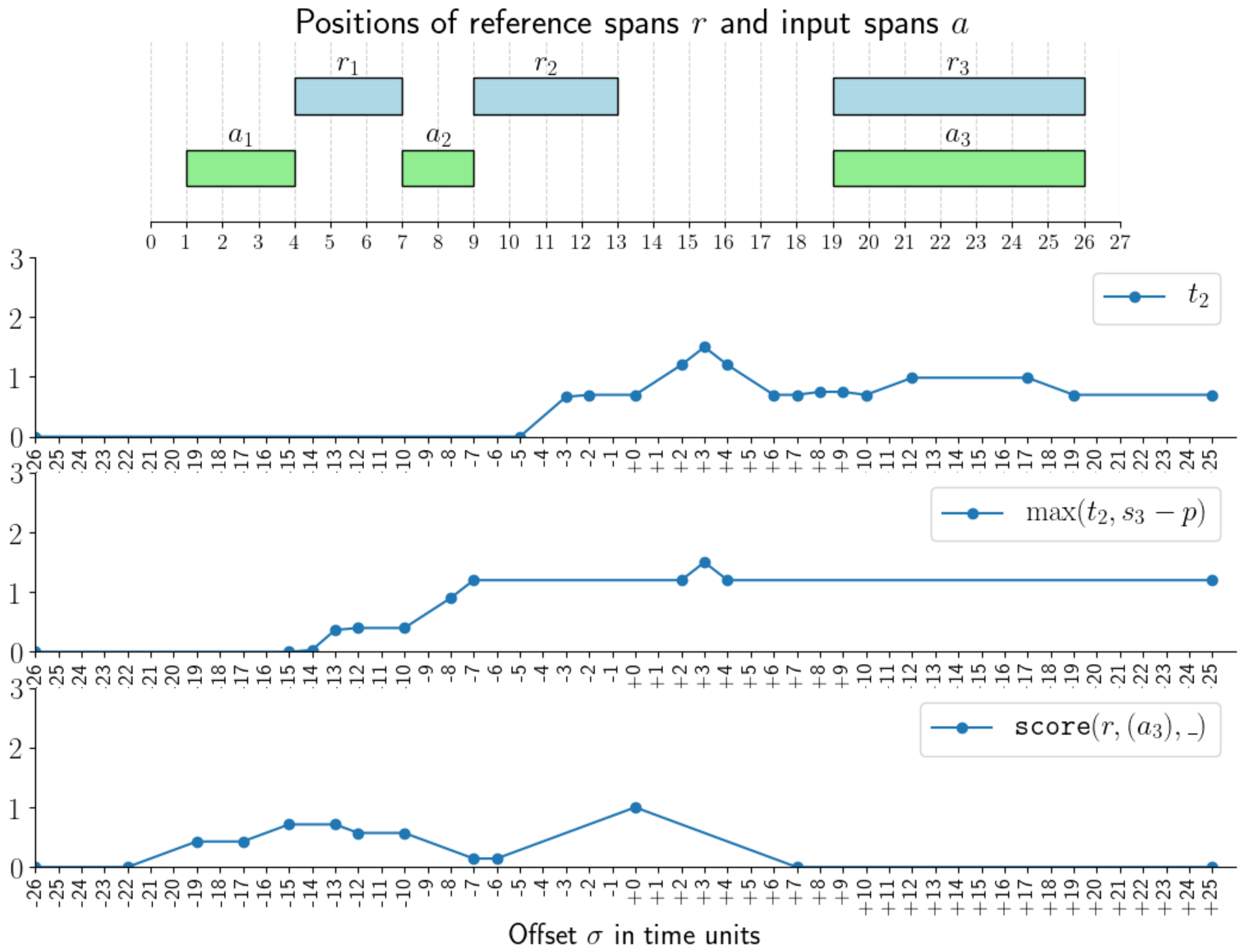
Optimal split alignment



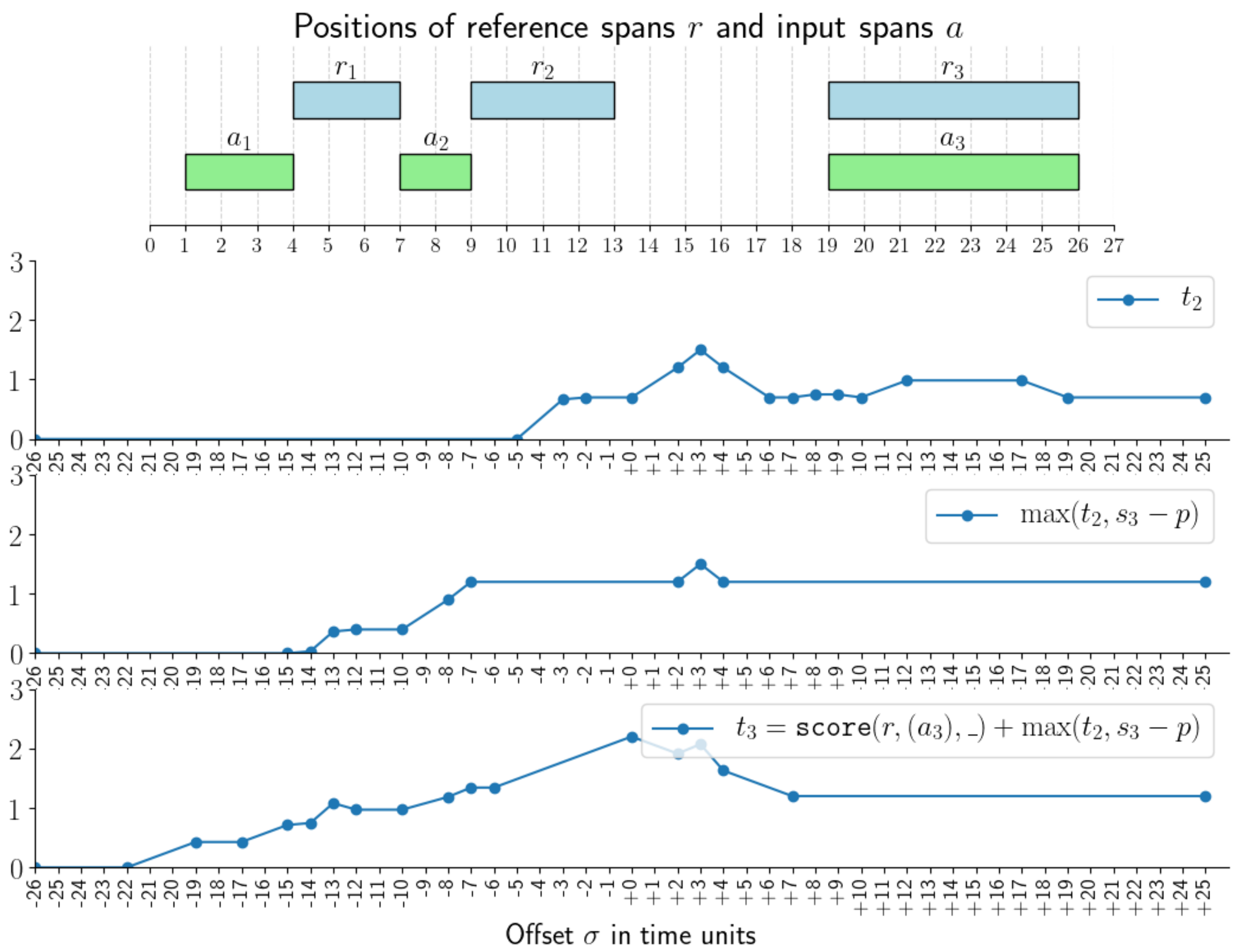
Optimal split alignment



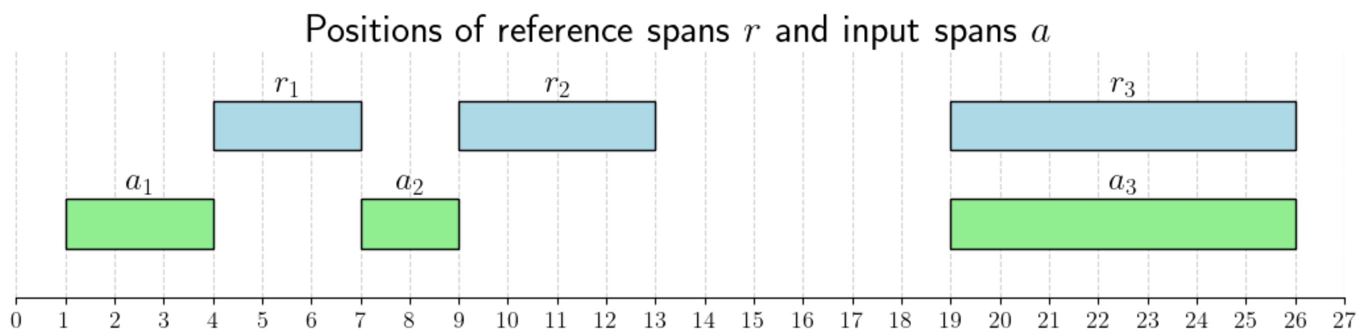
Optimal split alignment

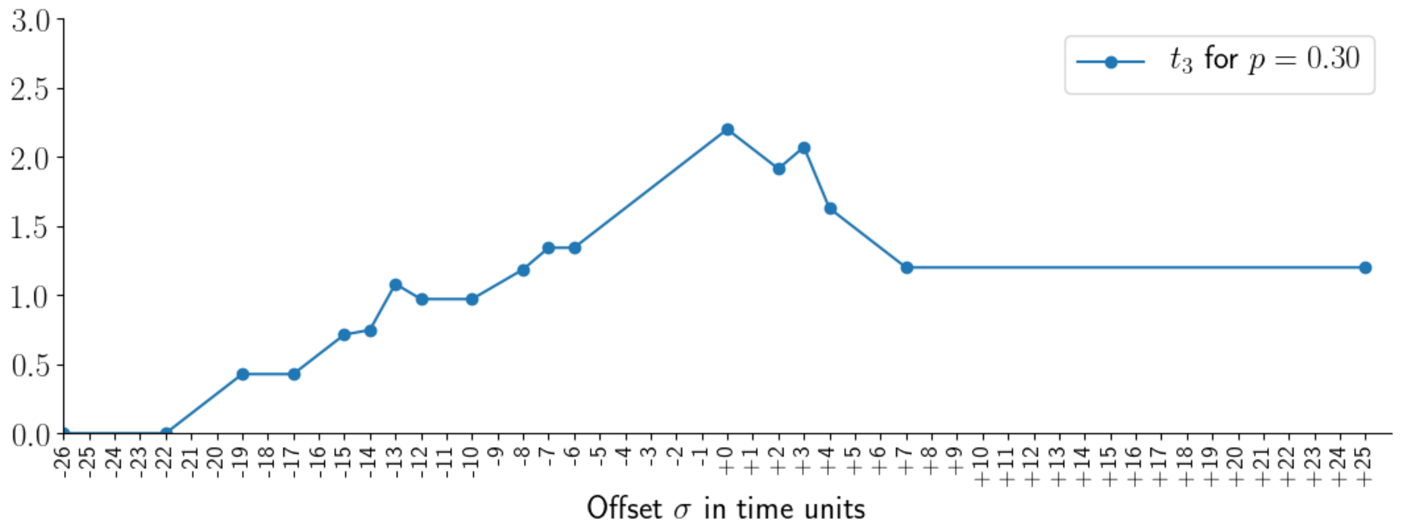


Optimal split alignment

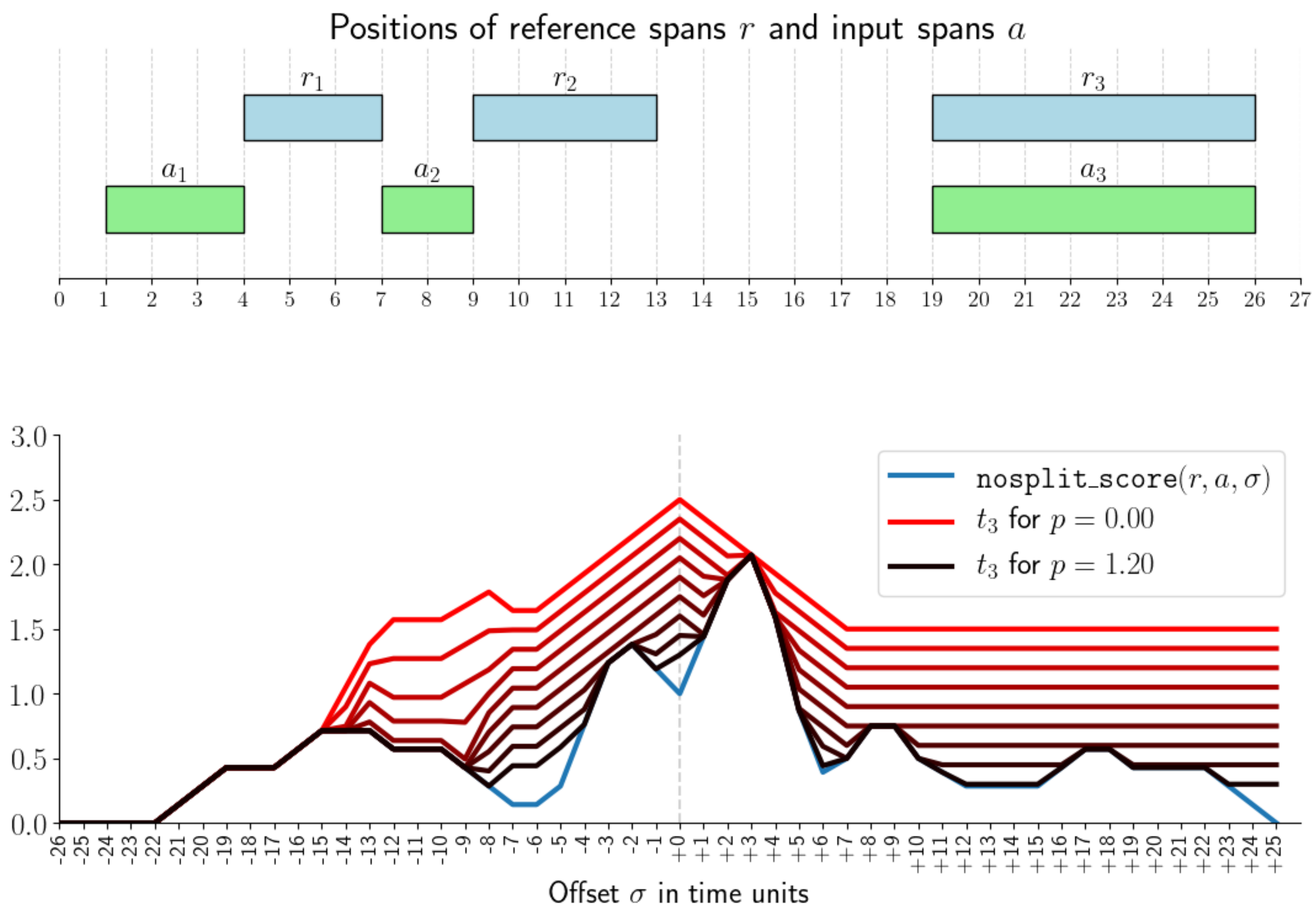


Optimal split alignment





Optimal split alignment



Optimal split alignment

Extracting optimal split alignment $\sigma^* = (\sigma_1^*, \ldots, \sigma_N^*)$

- maximum of t_N occurs for σ_N^*
- recursion formula selects σ_{n-1}^* depending on σ_n^*

Optimal split alignment

Analysis

- runtime complexity: $O(N \cdot (T_r + T_a))$

Optimal split alignment

Analysis

- runtime complexity: $O(N \cdot (T_r + T_a)) \approx 3$ min

Optimal split alignment

Analysis

- runtime complexity: $O(N \cdot (T_r + T_a)) \approx 3$ min
- space complexity: $O(N \cdot (T_r + T_a))$

Optimal split alignment

Analysis

- runtime complexity: $O(N \cdot (T_r + T_a)) \approx 3$ min
- space complexity: $O(N \cdot (T_r + T_a)) \approx 53.6$GB for $\sigma_n \to \sigma_{n-1}$ table

Optimal split alignment

Analysis

- runtime complexity: $O(N \cdot (T_r + T_a)) \approx 3\ \text{min}$
- space complexity: $O(N \cdot (T_r + T_a)) \approx 53.6\text{GB for } \sigma_n \rightarrow \sigma_{n-1} \text{ table}$

Algorithm useless?

Optimal split alignment

Analysis

- runtime complexity: $O(N \cdot (T_r + T_a)) \approx 3$ min
- space complexity: $O(N \cdot (T_r + T_a)) \approx 53.6$GB for $\sigma_n \rightarrow \sigma_{n-1}$ table

Algorithm useless?

- Merge segments with maximum error $\epsilon \approx 3$ seconds

Optimal split alignment

Analysis

- runtime complexity: $O(N \cdot (T_r + T_a)) \approx 3$ min
- space complexity: $O(N \cdot (T_r + T_a)) \approx 53.6\text{GB}$ for $\sigma_n \to \sigma_{n-1}$ table

Algorithm useless?

- Merge segments with maximum error $\epsilon \approx 3$ seconds
- save $\sigma_n \to \sigma_{n-1}$ in linear segments: $< 150\text{MB}$ for 118 subtitles

Framerate correction

Assumption: Framerates can only differ by a few common fractions.

Framerate correction

Assumption: Framerates can only differ by a few common fractions.

- $25/23.976$
- $25/24$
- $24/23.976 = 30/29.97 = 60/59.94 = 1001/1000$

Framerate correction

Assumption: Framerates can only differ by a few common fractions.

- $25/23.976$
- $25/24$
- $24/23.976 = 30/29.97 = 60/59.94 = 1001/1000$

Compare no-split score of all subtitles scaled with the 7 ratios.

Framerate correction

Assumption: Framerates can only differ by a few common fractions.

- $25/23.976$
- $25/24$
- $24/23.976 = 30/29.97 = 60/59.94 = 1001/1000$

Compare no-split score of all subtitles scaled with the 7 ratios.

- 27 out of 118 subtitles: framerate difference All corrected!

Framerate correction

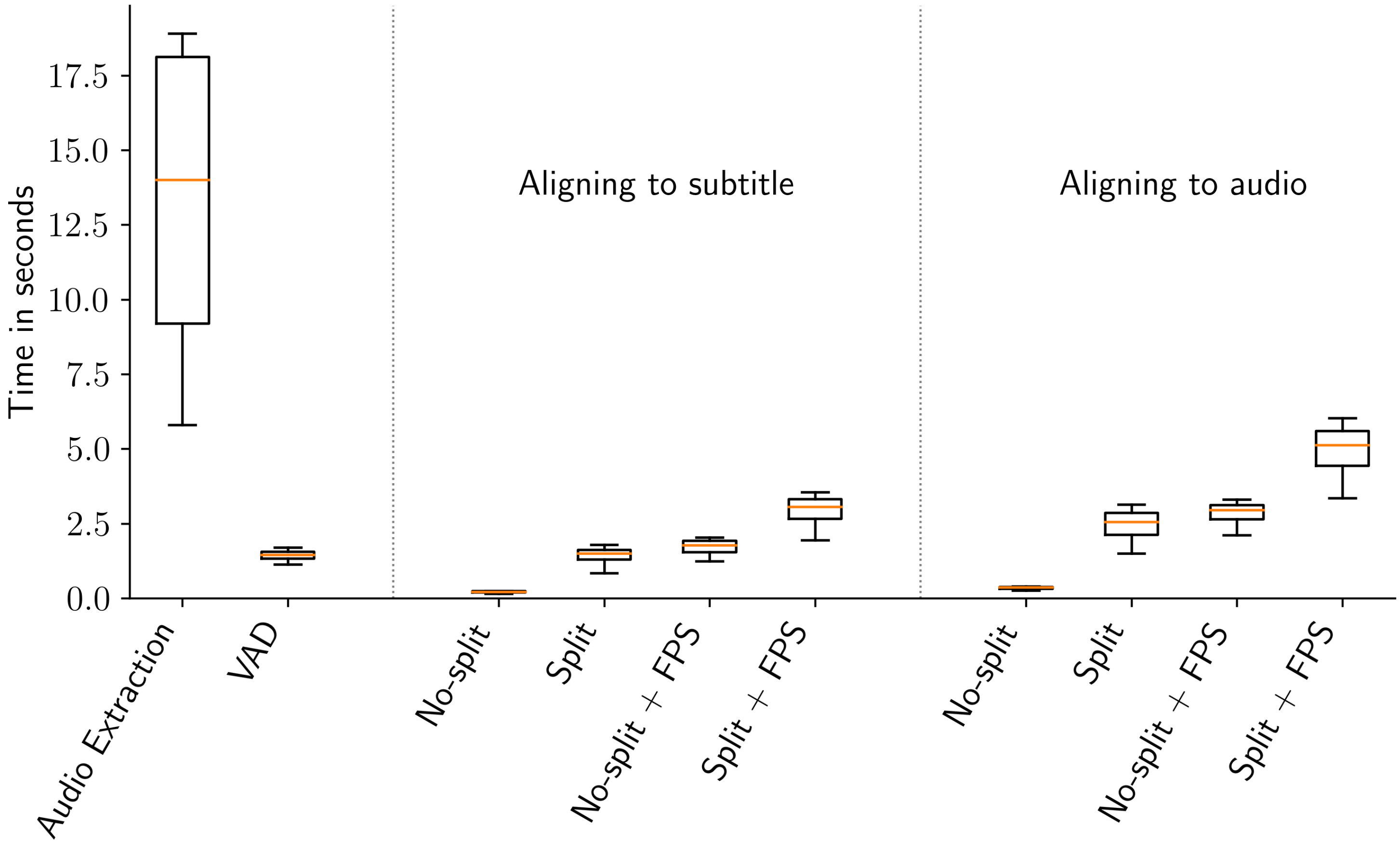
Assumption: Framerates can only differ by a few common fractions.

- $25/23.976$
- $25/24$
- $24/23.976 = 30/29.97 = 60/59.94 = 1001/1000$

Compare no-split score of all subtitles scaled with the 7 ratios.

- 27 out of 118 subtitles: framerate difference All corrected!
- 91 out of 118 subtitle: no framerate difference 3 wrong guesses!

Performance comparison

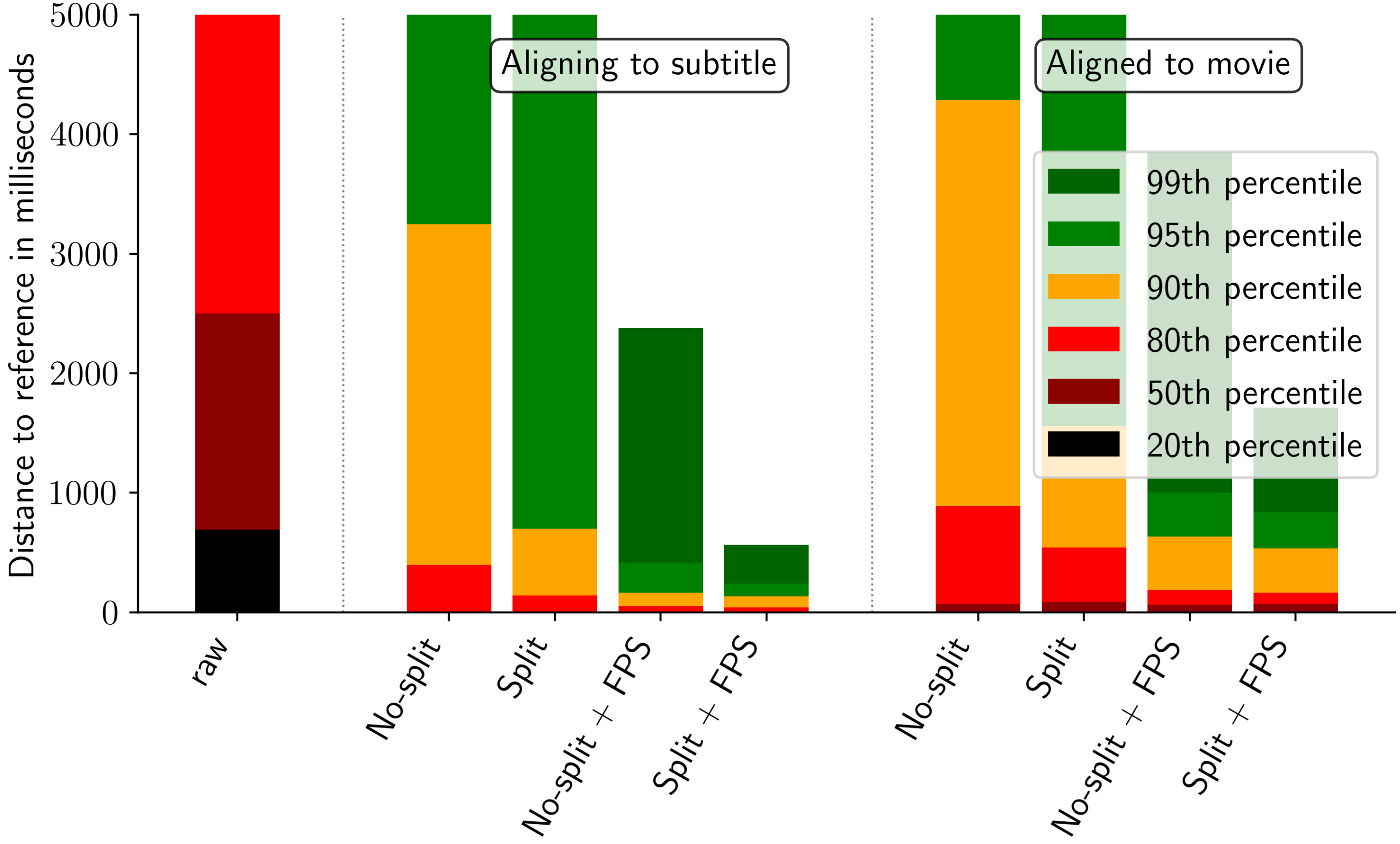


Results

Test Database

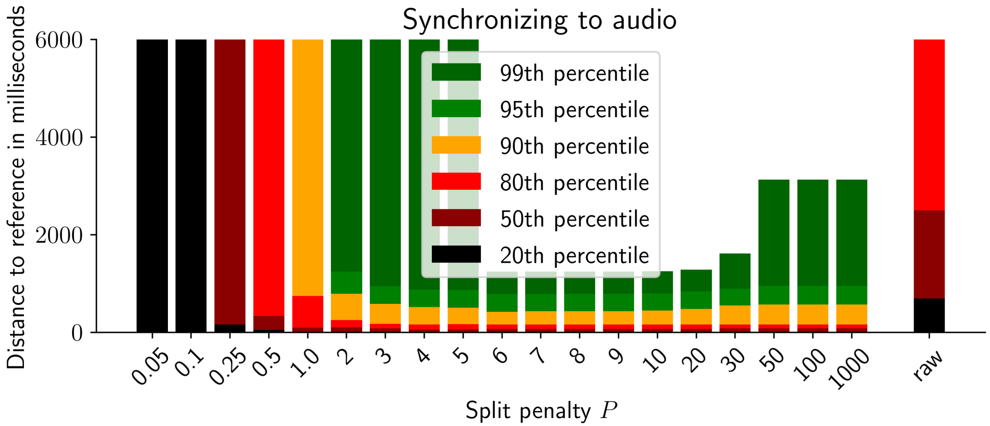
- 29 movies + 29 "reference subtitles"
- 118 input subtitles
- compare alignment against reference subtitle

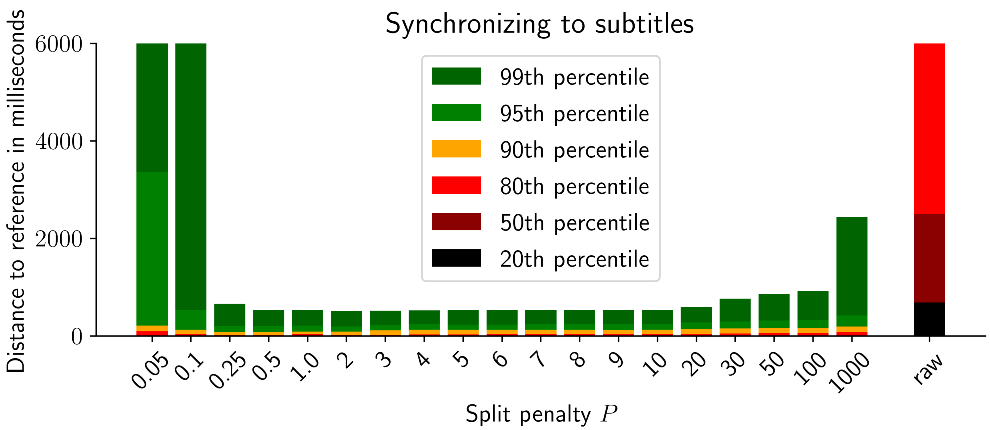
Alignment algorithms comparison



Results

Split penalties





Results

Alignment Classification

A "good subtitle" is defined here as

- less than 25% of lines having a distance of at most 300ms
- less than 70% of lines having a distance of at most 500ms
- less than 95% of lines having a distance of at most 1000ms
- less than 99% of lines having a distance of at most 1300ms

Results

Alignment Classification