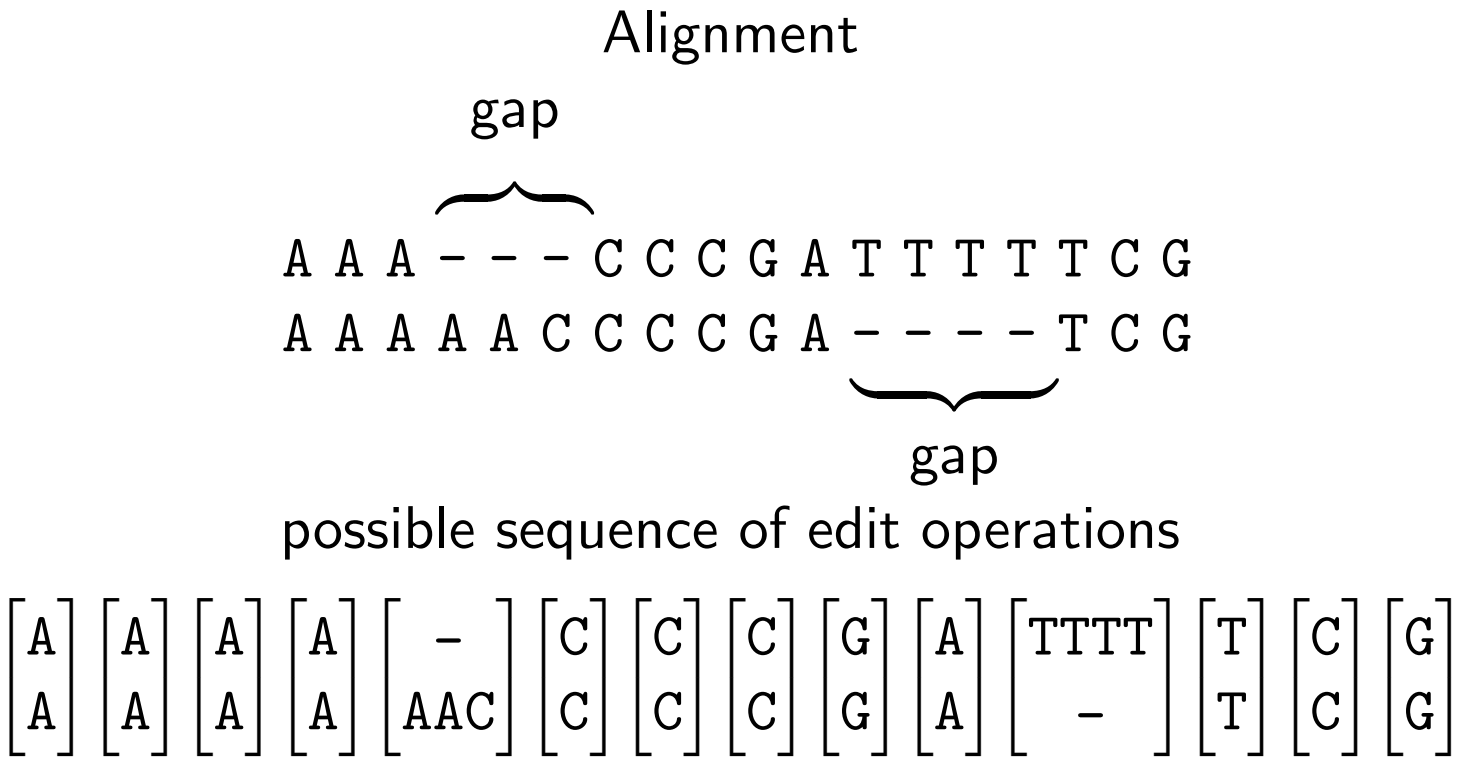
Variations of the cost and score model

- The usual evolutionary progression of biosequences involves individual point mutations, i.e. the insertion, deletion, or replacement of a single nucleotide or residue.
- Accordingly, the usual cost and score models assign costs and scores to individual edit operations.
- However, sometimes it is necessary to consider the deletion or insertion of entire segments as single units and to assign costs or scores to these.
- Thus we have to extend the alignment model by intervals of unaligned (i.e. inserted or deleted) characters, called *gaps* in the following.
- The notion of edit operations is generalized such that

$$(\alpha, \beta) \in (\mathcal{A} \times \mathcal{A}) \cup \underbrace{(\mathcal{A}^+ \times \{\varepsilon\})}_{|\text{gap}| \geq 1} \cup \underbrace{(\{\varepsilon\} \times \mathcal{A}^+)}_{|\text{gap}| \geq 1}$$

for each edit operation $\alpha \to \beta$.

Figure 1: An alignment and the possible sequence of edit operations according to the extended alignment model.

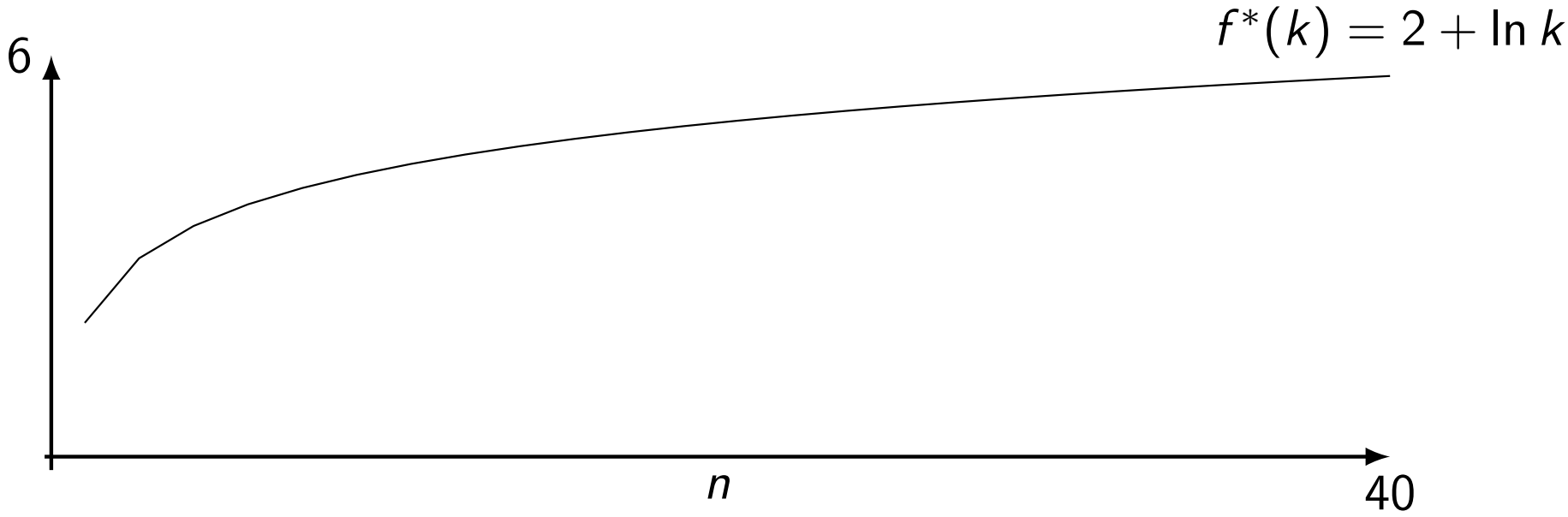


General gap costs

- We also have to extend the cost function to the new kinds of edit operations.
- There are several choices and we start with the general gap cost model.
- This model assigns costs by an arbitrary computable function f^* on the length of the gap.
- That is, for each $w \in \mathcal{A}^+$, define

$$\delta^*(w \to \varepsilon) = \delta^*(\varepsilon \to w) = f^*(|w|)$$

Figure 2: Example of gap cost function $f^*(k) = 2 + \ln k$ and alignment with corresponding gap costs shown above or below the gaps.



$f^*(4) = 3.4$

```
t t g t a t c c a a c g t g t t t t g t t t t t c t t t c a a t a a t - - - - t a g a t c t t a t
t t g t n c - - - - - - - - - - t t t g g - - - - - c t t c n a t a a c c c c a t g g n c c t t n t
```

$f^*(9) = 4.2$ $f^*(5) = 3.6$

General gap costs

- The costs for an alignment A with respect to the general gap cost model is defined as the sum of the costs of the generalized edit operations, the alignment consists of.
- It is denoted by $\delta^*(A)$.

General gap costs

- To handle general gap costs, one extends the edit graph $G(u, v)$ by additional edges, to accommodate all kinds of gaps.
- A gap $u[i'+1 \ldots i] \to \varepsilon$ is represented by an edge from (i', j) to (i, j) for some $0 \leq j \leq n$, $1 \leq i \leq m$ and $0 \leq i' \leq i-1$.
- A gap $\varepsilon \to v[j'+1 \ldots j]$ is represented by an edge from (i, j') to (i, j) for some $0 \leq i \leq m$, $1 \leq j \leq n$ and $0 \leq j' \leq j-1$.
- Thus, for each (i, j), $0 \leq i \leq m$, $0 \leq j \leq n$, the edit graph is extended by additional edges

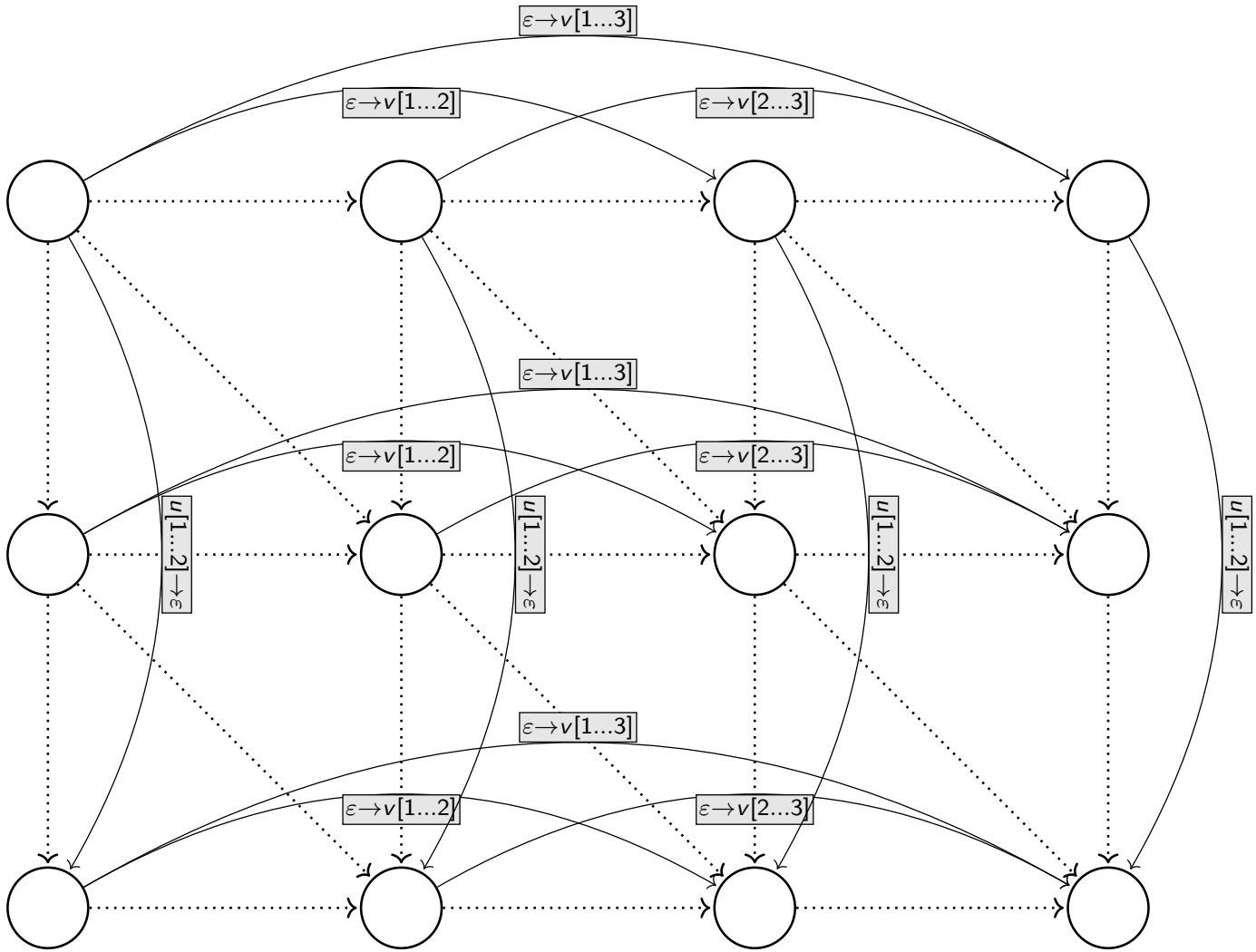
$$(i', j) \xrightarrow{u[i'+1 \ldots i] \to \varepsilon} (i, j)$$

for all i', $0 \leq i' \leq i-1$ and edges

$$(i, j') \xrightarrow{\varepsilon \to v[j'+1 \ldots j]} (i, j)$$

for all j', $0 \leq j' \leq j-1$.

Figure 3: The general-gap edit graph for $u[1 \ldots 2]$ and $v[1 \ldots 3]$. The dotted edges also occur in the standard edit graph. The additional thin edges are marked by deletions and insertions of substrings of length at least 2.



General gap costs

- Thus, each node in the edit graph has on average $(n/2-1)+(m/2-1)$ additional edges, see also Figure 3.
- That is, there are $O(m+n)$ edges into each node and the total number of edges becomes $O(mn(m+n))$.
- To compute the edit distance according to the general gap cost model, we compute an $(m+1)\times(n+1)$-table GGC_{δ,f^*} such that $GGC_{\delta,f^*}(i,j)$ is the generalized edit distance of $u[1\ldots i]$ and $v[1\ldots j]$ with respect to the cost functions δ and f^*.
- Here δ is a cost function assigning costs to replacements only.
- From the considerations above, one derives the following recurrences:

$$GGC_{\delta,f^*}(i,j) = \min \begin{cases} 0 & \text{if } i=j=0 \\ \min\{GGC_{\delta,f^*}(i',j)+f^*(i-i') \mid 0\le i'\le i-1\} & \text{if } i>0 \\ \min\{GGC_{\delta,f^*}(i,j')+f^*(j-j') \mid 0\le j'\le j-1\} & \text{if } j>0 \\ GGC_{\delta,f^*}(i-1,j-1)+\delta(u[i]\to v[j]) & \text{if } i,j>0 \end{cases}$$

General gap costs

- Note the compact description of the four different cases.
- The conditions involving i and j determine whether the value to the left is included in the minimization:

1. if $i > 0$ and $j = 0$, the overall minimum is the second minimum,
2. if $i = 0$ and $j > 0$, the overall minimum is the third minimum
3. if $i > 0$ and $j > 0$, the overall minimum is the minimum of the last three expressions.

- It is obvious that each entry in table GGC_{δ,f^*} is computed in $O(m+n)$ time.
- Thus we obtain an algorithm which runs in $O(mn\,(m+n))$ time.
- The algorithm is the Needleman and Wunsch Algorithm [Needleman and Wunsch, 1970].

Affine gap costs

- A simplified cost model for gaps is the *affine gap model*.
- In this, each gap gets the sum of the costs for each symbol involved in the gap plus a *gap initiation cost* for the introduction of the gap.
- That is, for each $w \in \mathcal{A}^+$, define

$$\delta_{\text{affine}}(w \to \varepsilon) = g + \sum_{i=1}^{|w|} \delta(w[i] \to \varepsilon) \text{ and } \delta_{\text{affine}}(\varepsilon \to w) = g + \sum_{j=1}^{|w|} \delta(\varepsilon \to w[j])$$

where

- the constant g is the gap initiation cost,
- $\delta(w[i] \to \varepsilon)$ is the cost of deleting $w[i]$ and
- $\delta(\varepsilon \to w[j])$ is the cost of inserting $w[j]$.

Affine gap costs

- The costs for an alignment A with respect to the affine gap cost model is defined as the sum of the costs of all edit operations, where gaps have costs according to function δ_{affine}.
- The affine gap cost is denoted by $\delta_{\text{affine}}(A)$.
- The edit distance problem with affine gap costs consists of finding alignments A of u and v such that $\delta_{\text{affine}}(A)$ is minimal.
- Example 1 illustrates the effect of the affine gap cost model.

Example 1

– Consider the following alignment whose cost is determined as follows:

		linear model	affine model	
match	mismatch	indel	gap init	gap ext.
0	1	3	2	1

```
       G A A T T C C G T T A
       |   | |   |   |     |
       G G A T - C - G - - A
linear 0 1 0 0 3 0 3 0 3 3 0 = 13
affine 0 1 0 0 3 0 3 0 3 1 0 = 11

       G A A T T C C G T T A
       |   | |     | |     |
       G G A T - - C G - - A
linear 0 1 0 0 3 3 0 0 3 3 0 = 13
affine 0 1 0 0 3 1 0 0 3 1 0 = 9
```

– make first gap larger, align second C in first sequence with first C in second sequence

$\Rightarrow$ linear cost remains constant, affine cost is reduced by 2

– second alignment with smaller affine gap cost is the preferred alignment

- This example demonstrates the difference between affine gap costs and linear gap costs.
- Affine gap costs provide incentive for the alignment algorithm to keep gaps together where possible rather than scattering many small gaps.
- Generally this is the more desirable behavior, and so the use of affine gap costs makes sense.

Affine gap costs

- Affine gap costs are a special case of the general gap cost model and thus could be handled by the algorithm described in Section 3.
- We will however show that they can be handled more efficiently.
- The idea, first presented in [Gotoh, 1982], is to extend the edit graph to an affine-cost edit graph.
- This allows to recognize which edge begins a gap and which edge extends a gap and in turn assign appropriate costs to the two kinds of edges.
- That is, if an edge starts a gap, then we add the gap start costs g.
- Technically, we split each node (i, j) into three different nodes $(i, j, \mathtt{R})$, $(i, j, \mathtt{D})$, and $(i, j, \mathtt{I})$.
- The type of the node (`R`, `D`, and `I`) specifies that every path in the edit graph ending at that node, corresponds to an alignment that ends with the corresponding (single symbol) edit operation.

Affine gap costs

– A replacement edge $(i-1, j-1) \xrightarrow{u[i] \mapsto v[j]} (i, j)$ leads to three edges

$$(i-1, j-1, x) \xrightarrow{u[i] \mapsto v[j]} (i, j, \mathtt{R})$$

for all $x \in \{\mathtt{R}, \mathtt{D}, \mathtt{I}\}$, in the affine-cost edit graph.

– A deletion edge $(i-1, j) \xrightarrow{u[i] \mapsto \varepsilon} (i, j)$ leads to three edges

$$(i-1, j, x) \xrightarrow{u[i] \mapsto \varepsilon} (i, j, \mathtt{D})$$

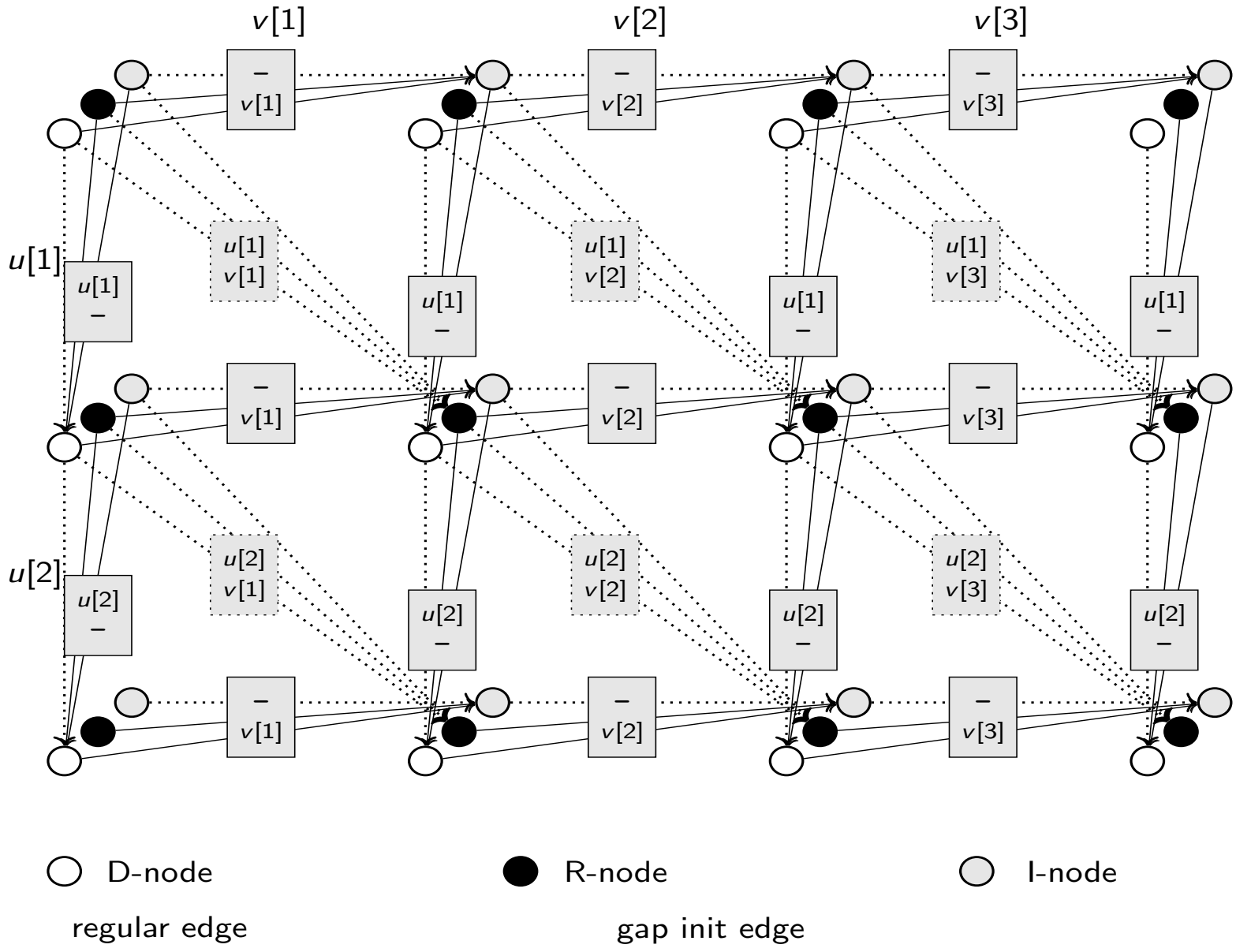
for all $x \in \{\mathtt{R}, \mathtt{D}, \mathtt{I}\}$, in the affine-cost edit graph.

– An insertion edge $(i, j-1) \xrightarrow{\varepsilon \mapsto v[j]} (i, j)$ leads to three edges

$$(i, j-1, x) \xrightarrow{\varepsilon \mapsto v[j]} (i, j, \mathtt{I}),$$

for all $x \in \{\mathtt{R}, \mathtt{D}, \mathtt{I}\}$, in the affine-cost edit graph.

Figure 4: The affine-cost edit graph for $u[1 \ldots 2]$ and $v[1 \ldots 3]$.



Affine gap costs

The following table shows how to assign costs to the different kinds of edges in the affine-cost edit graph.

kind of edge	edit op.	cost assignment
$(i-1, j-1, x) \xrightarrow{u[i] \to v[j]} (i, j, \mathtt{R})$	replacement	$\delta(u[i] \to v[j])$
$(i-1, j, \mathtt{D}) \xrightarrow{u[i] \to \varepsilon} (i, j, \mathtt{D})$	gap extension	$\delta(u[i] \to \varepsilon)$
$(i, j-1, \mathtt{I}) \xrightarrow{\varepsilon \to v[j]} (i, j, \mathtt{I})$	gap extension	$\delta(\varepsilon \to v[j])$
$(i-1, j, \mathtt{R}) \xrightarrow{u[i] \to \varepsilon} (i, j, \mathtt{D})$	gap start	$g + \delta(u[i] \to \varepsilon)$
$(i-1, j, \mathtt{I}) \xrightarrow{u[i] \to \varepsilon} (i, j, \mathtt{D})$	gap start	$g + \delta(u[i] \to \varepsilon)$
$(i, j-1, \mathtt{R}) \xrightarrow{\varepsilon \to v[j]} (i, j, \mathtt{I})$	gap start	$g + \delta(\varepsilon \to v[j])$
$(i, j-1, \mathtt{D}) \xrightarrow{\varepsilon \to v[j]} (i, j, \mathtt{I})$	gap start	$g + \delta(\varepsilon \to v[j])$

Affine gap costs

- Node splitting does not affect the properties of the edit graph, i.e. a path from node (i', j', x) to (i, j, y) in the affine-cost edit graph corresponds to an alignment of $u[i'+1 \ldots i]$ and $v[j'+1 \ldots j]$, that ends with an edit operation according to the value y.
- The only difference is that the gaps in the alignments have costs according to the affine gap cost model.
- To compute optimal global alignments, we use $(0, 0, \mathtt{R})$, $(0, 0, \mathtt{D})$, and $(0, 0, \mathtt{I})$ as start entries and $(m, n, \mathtt{R})$, $(m, n, \mathtt{D})$, and $(m, n, \mathtt{I})$ as final entries.

Affine gap costs

According to the considerations above, the edit graph has three times as many edges and nodes as before. But there are still $O(mn)$ nodes and edges. To compute the optimal alignment, one fills an $(m+1) \times (n+1) \times \{\mathtt{R}, \mathtt{D}, \mathtt{I}\}$-table A_{afn} such that the following holds:

- If $i = j = 0$, then $A_{\text{afn}}(i, j, \mathtt{R}) = 0$ and $A_{\text{afn}}(i, j, y) = g$ for $y \in \{\mathtt{D}, \mathtt{I}\}$. The latter handles the case for an alignment with an initial gap.
- If $i > 0$ or $j > 0$, then

$$A_{\text{afn}}(i, j, y) = \min\ (\{\infty\} \cup \{\delta_{\text{affine}}(A) \mid A \text{ is an alignment of } u[1 \ldots i] \text{ and } v[1 \ldots j] \text{ and } A \text{ ends with an edit operation according to the value of } y\})$$

One can derive the following recurrences for A_{afn}. As the boundary cases are already defined, we can restrict to the case that $i > 0$ or $j > 0$.

Affine gap costs

$$A_{\text{afn}}(i,j,\mathtt{R}) = \begin{cases} \infty & \text{if } i=0 \text{ or } j=0 \\ \min\{A_{\text{afn}}(i-1,j-1,y) + \delta(u[i] \to v[j]) \mid y \in \{\mathtt{R},\mathtt{D},\mathtt{I}\}\} & \text{otherwise} \end{cases}$$

$$A_{\text{afn}}(i,j,\mathtt{D}) = \begin{cases} \infty & \text{if } i=0 \\ \min\ (\{A_{\text{afn}}(i-1,j,\mathtt{D}) + \delta(u[i] \to \varepsilon)\} \cup & \text{otherwise} \\ \qquad \{A_{\text{afn}}(i-1,j,x) + g + \delta(u[i] \to \varepsilon) \mid x \in \{\mathtt{R},\mathtt{I}\}\}) & \end{cases}$$

$$A_{\text{afn}}(i,j,\mathtt{I}) = \begin{cases} \infty & \text{if } j=0 \\ \min\ (\{A_{\text{afn}}(i,j-1,\mathtt{I}) + \delta(\varepsilon \to v[j])\} \cup & \text{otherwise} \\ \qquad \{A_{\text{afn}}(i,j-1,x) + g + \delta(\varepsilon \to v[j]) \mid x \in \{\mathtt{R},\mathtt{D}\}\}) & \end{cases}$$

- The cost of optimal affine gap cost alignment is

$$\min\{A_{\text{afn}}(m,n,x) \mid x \in \{\mathtt{R},\mathtt{D},\mathtt{I}\}\}$$

  Obviously, each entry in table A_{afn} can be evaluated in constant time.
- Thus the algorithm runs in $O(mn)$ time and space.
- As there are approximately three times as many nodes and edges in the affine cost edit graph, compared to the standard edit graph, the practical running time with increase by a factor of about 3.


Gotoh, O. (1982).
An Improved Algorithm for Matching Biological Sequences.
JMB, 162:705–708.


Needleman, S. and Wunsch, C. (1970).
A General Method Applicable to the Search for Similarities in the Amino-Acid Sequence of Two Proteins.
JMB, 48:443–453.