Bioinformatics

The slides are based in material from:

Dan Gusfield, **Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press**

Techniques for analysis and comparison of biological data sequences

- Approximate Pattern Matching
- Multiple Sequence Alignment
- Applications in Molecular Biology Problems

Παρατηρήσεις

- Global Alignment is called Needleman-Wunsch alignment (1970)
- Local alignment is called Smith-Waterman alignment (1981)
- Smith-Waterman can find regions with high similarity by simply performing a trace-back from any cell (i,j) backwards and locate a pair with similarity v(i,j).

Basic definitions (α)

- **Edit Distance:** for 2 strings we define as edit distance the minimum number of operations that are needed in order to transform the first string to the second. The basic operations are **insertion**, **deletion** and **replacement** of symbols.

- Example: S_1: vintner and S_2: writers
- edit-distance(S_1->S_2)=5

Basic definitions (β)

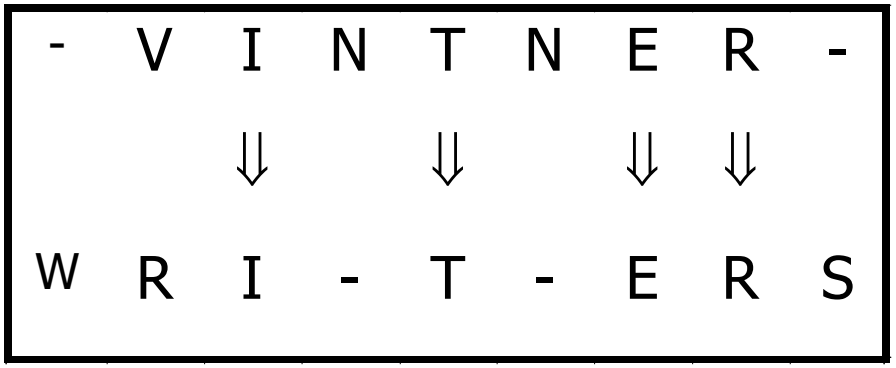
- **Edit Transcript:** for the transformation of a string, we define the sequence of operations that are needed in order to transform the first string to the second. The basic operations are:
  - **Insertion: I**
  - **Deletion: D**
  - **Replacement: R**
  - **Matching: M**
- Example: S_1: vintner and S_2: writers
- edit-distance(S_1->S_2)= RIMDMDMMI

Example of Substitution Matrix

```
BLOSUM Clustered Scoring Matrix in 1/2 Bit Units
#  Cluster Percentage: >= 62
#  Entropy =   0.6979, Expected =  -0.5209
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  4 -1 -2 -2  0 -1 -1  0 -2 -1 -1 -1 -1 -2 -1  1  0 -3 -2  0 -2 -1  0 -4
R -1  5  0 -2 -3  1  0 -2  0 -3 -2  2 -1 -3 -2 -1 -1 -3 -2 -3 -1  0 -1 -4
N -2  0  6  1 -3  0  0  0  1 -3 -3  0 -2 -3 -2  1  0 -4 -2 -3  3  0 -1 -4
D -2 -2  1  6 -3  0  2 -1 -1 -3 -4 -1 -3 -3 -1  0 -1 -4 -3 -3  4  1 -1 -4
C  0 -3 -3 -3  9 -3 -4 -3 -3 -1 -1 -3 -1 -2 -3 -1 -1 -2 -2 -1 -3 -3 -2 -4
Q -1  1  0  0 -3  5  2 -2  0 -3 -2  1  0 -3 -1  0 -1 -2 -1 -2  0  3 -1 -4
E -1  0  0  2 -4  2  5 -2  0 -3 -3  1 -2 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
G  0 -2  0 -1 -3 -2 -2  6 -2 -4 -4 -2 -3 -3 -2  0 -2 -2 -3 -3 -1 -2 -1 -4
H -2  0  1 -1 -3  0  0 -2  8 -3 -3 -1 -2 -1 -2 -1 -2 -2  2 -3  0  0 -1 -4
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4  2 -3  1  0 -3 -2 -1 -3 -1  3 -3 -3 -1 -4
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4 -2  2  0 -3 -2 -1 -2 -1  1 -4 -3 -1 -4
K -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5 -1 -3 -1  0 -1 -3 -2 -2  0  1 -1 -4
M -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5  0 -2 -1 -1 -1 -1  1 -3 -1 -1 -4
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6 -4 -2 -2  1  3 -1 -3 -3 -1 -4
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7 -1 -1 -4 -3 -2 -2 -1 -2 -4
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4  1 -3 -2 -2  0  0  0 -4
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5 -2 -2  0 -1 -1  0 -4
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11  2 -3 -4 -3 -2 -4
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -1 -3 -2 -1 -4
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4 -3 -2 -1 -4
B -2 -1  3  4 -3  0  1 -1  0 -3 -4  0 -3 -3 -2  0 -1 -4 -3 -3  4  1 -1 -4
Z -1  0  0  1 -3  3  4 -2  0 -3 -3  1 -1 -3 -1  0 -1 -3 -2 -2  1  4 -1 -4
X  0 -1 -1 -1 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -2  0  0 -2 -1 -1 -1 -1 -1 -4
* -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4  1
```

Sequence Alignment

- **Sequence Alignment:** we put the one sequence below the other so that common characters are put in the same positions.



Sequence alignment allowing gaps

- **We align two sequence by introducing 7 gaps in 4 positions, that are translated to changing the DNA sequence in the respective positions.**

c t t t a a c - - a - a c
c - - - c a c c c a t - c

Computing the score-function when aligning sequences

1	2	3	4	5	6	7
g	*a*	*g*	-	*t*	*c*	*t*
g	*a*	*c*	*c*	*t*	*c*	-

S	**a**	**c**	**g**	**t**	**_**
a	1	-1	-2	0	-1
c		3	-2	-1	0
g			0	-4	-2
t				3	-1
_					0

- **Score-function=** 0+1-2+0+3+3-1=4

The method of Dynamic Programming

- **Dynamic programming:**

  consider 2 sequences S_1 and S_2, we define as D(i,j) the edit distance between prefixes $S_1[1..i]$ and $S_2[1..j]$, that is the minimum number of operations that are needed in order to transform the i first characters of S_1 to the j first characters of S_2.

- Use 3 basic techniques:
  - recurrence relation,
  - tabular computation,
  - traceback.

Example of Dynamic Programming Table

D(i,j)			***w***	***r***	***i***	***t***	***e***	***r***	***s***
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	0	0	1	2	3	4	5	6	7
v	**1**	1	1	2	3	4	5	6	7
i	**2**	2	2	2	2	3	4	5	6
n	**3**	3	3	3	3	3	4	5	6
t	**4**	4	4	4	4	*			
e	**5**	5							
r	**6**	6							
s	**7**	7							

Recurrence relationship

- **Recurrence relationship:**

$$D(i,j)=min[D(i-1,j)+1,D(i,j-1)+1,D(i-1,j-1)+t(i,j)]$$

  - **D(i,j-1)+1:** we must insert $S_2[j]$
  - **D(i-1,j)+1:** we must delete $S_1[i]$,
  - **D(i-1,j-1)+1:** in order to map $S_1[i]$ to $S_2[j]$ we must replace $S_1[i]$, with $S_2[j]$,
  - **D(i-1,j-1):** matching

Example: recurrence relationship

D(i,j)			*w*	*r*	*i*	*t*	*e*	*r*	*s*
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	0	0	1	2	3	4	5	6	7
v	**1**	1	1	2	3	4	5	6	7
i	**2**	2	2	2	2	3	4	5	6
n	**3**	3	3	3	3	3	4	5	6
t	**4**	4	4	4	4	*			
e	**5**	5							
r	**6**	6							
s	**7**	7							

D(4,4)= D(3,3)=3, αφού $S_1(4)=S_2(4)=t$.

Traceback relationship

- **Traceback:**
  - From (i,j) to (i,j-1) αν $D(i,j) = D(i,j-1)+1$ (character insertion)
  - From (i,j) to (i-1,j) if $D(i,j) = D(i-1,j)+1$ (character deletion)
  - From (i,j) to (i-1,j-1) if $D(i,j) = D(i-1,j-1)+t(i,j)$ (character replacement or matching)

Adding traceback pointers

D(i,j)			w	r	i	t	e	r	s
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	0	0	? 1	? 2	? 3	? 4	? 5	? 6	? 7
v	**1**	?1	? 1	? ? 2	? ? 3	? ? 4	? ? 5	? ? 6	? ? 7
i	**2**	?2	? ?2	? 2	? 2	? 3	? 4	? 5	? 6
n	**3**	?3	? ?3	? ?3	? ?3	? 3	? ?4	? ?5	? ? 6
t	**4**	?4	? ?4	? ?4	? ?4	? 3	? ?4	? ?5	? ? 6
e	**5**	?5	? ?5	? ?5	? ?5	?4	? 4	? ?5	? ? 6
r	**6**	?6	? ?6	? ?6	? ?6	?5	? 4	? ?5	? ? 6
s	**7**	?7	? ?7	? 6	? ? ?7	?6	?5	? 4	? 5

Interpreting traceback pointers

D(i,j)			w	r	i	t	e	r	s
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	0	0	? 1	? 2	? 3	? 4	? 5	? 6	? 7
v	**1**	?1	? 1	? ? 2	? ? 3	? ? 4	? ? 5	? ? 6	? ? 7
i	**2**	?2	? ?2	? 2	? 2	? 3	? 4	? 5	? 6
n	**3**	?3	? ?3	? ?3	? ?3	? 3	? ?4	? ?5	? ? 6
t	**4**	?4	? ?4	? ?4	? ?4	? 3	? ?4	? ?5	? ? 6
e	**5**	?5	? ?5	? ?5	? ?5	?4	? 4	? ?5	? ? 6
r	**6**	?6	? ?6	? ?6	? ?6	?5	? 4	? ?5	? ? 6
s	**7**	?7	? ?7	? 6	? ? ?7	?6	?5	? 4	? 5

V I N T N E R -
W R I T - E R S

Complexity of Dynamic Programming Methods

- Initialization: O(n) + O(m)
- Recurrence relationship: O(n*m)
- Traceback pointers: O(n+m)
- Complexity: $O(n^2)$
- Equivalence with a problem of graph theory where every node has as label (i,j)

Basic definitions (γ)

- **Weighted Edit Distance:** the minimum number of operations that must be done to transform the first string to the second. Every operation has a specific weight-cost. Assume that the basic operations have the following weights:
  - Insertion or deletion: d
  - Replacement : r
  - Matching : m.
- Example : S_1: vintner and S_2: writers
- weighted edit-distance(S_1->S_2)= **r+4d+4m**.

Recurrence relationship with weights

- **Recurrence relationship:**

$$D(i,j)=min[D(i-1,j)+d,D(i,j-1)+d,D(i-1,j-1)+t(i,j)],$$

- where:
  - $t(i,j)=$ e, if $S_1(i)=S_2(j)$,
  - $t(i,j)=r$, if $S_1(i)\neq S_2(j)$ and
  - $D(i,0)=i*d$ and $D(0,j)=j*d$.

Dynamic programming & sequence similarity based on alphabet

- **Recurrence relationship for sequence similarity :**

$$V(i,j)= max[V(i-1,j-1)+s(S_1(i), S_2(j)), V(i-1,j)+ s(S_1(i),\_), V(i,j-1)+ s(\_,S_2(j))],$$

- where:
  - s(x,y): the alignment value of character x and y
  - $V(0,j)=\sum s(\_,S_2(k)),\ 1<k<j$ and $V(i,0)= \sum s(S_1(k),\_),\ 1<k<i.$

EXTENTIONS

- Weighted Edit Distance.
- In Molecular Biology applications the character subsitution weights are stored in substitution matrices - Substitution Matrix: PAM και BLOSUM
- It is better to face it as alignment and embed the score.
- Every match is positive , everything else is 0 or negative

Extentions

- Longest Common Subsequence (match weight 1, otherwise 0)
- End-space free variant that encourages one string to align in the interior of the other, or the suffix of one to align with the prefix of the other (initial conditions 0, everything is countable in the last row and the last column) – shotgun sequence assembly
- Approximate occurrence of P in T (the optimal alignment of P to a substring of T has distance δ from the optimal alignment) (initial conditions 0).

  -- locate a cell (n,j) with value greater than δ.

  -- traverse backpointers from (n.j) to (0,k).

  -- occurrence in T[k,j]

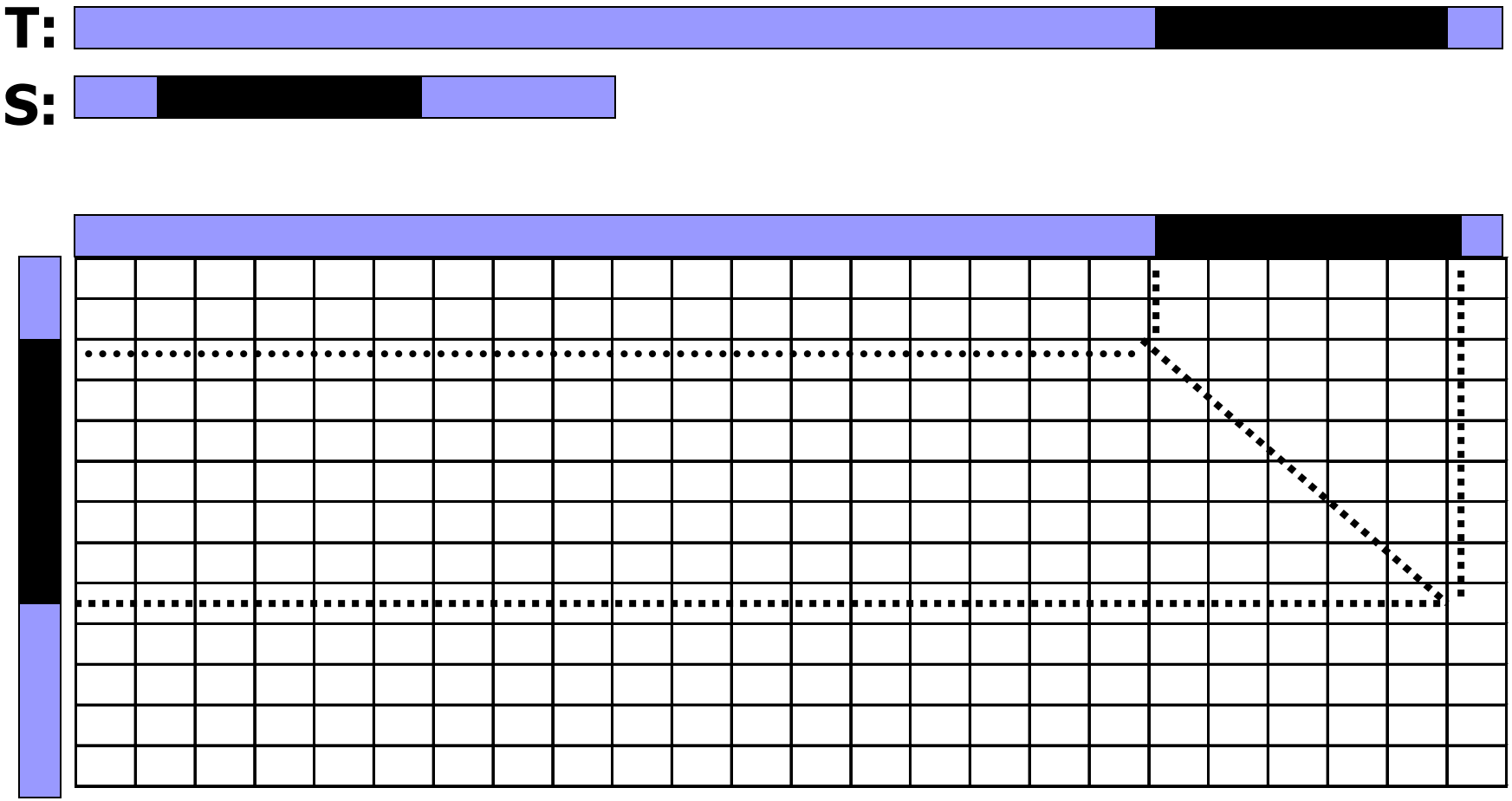
Local Suffix Alignment Problem

- **Local suffix alignment problem:** for two sequence S_1 and S_2 locate a suffix α of $S_1[1..i]$ (with the probability of being empty) and a suffix β of $S_2[1..j]$ (probably empty) so that V(α,β) has the maximum value of all possible pairs of suffices of $S_1[1..i]$ and $S_2[1..j]$. We symbolize as u(i,j) the maxuimum local suffix alignment for the values i and ,j (i<n and j<m).

- Initial conditions v(i,0)=0 and v(0,j)=0 as we can select an empty suffix.

  - $u(i,j)=max[0,u(i-1,j-1)+s(S_1(i), S_2(j)), u(i-1,j)+ s(S_1(i),\_), u(i,j-1)+ s(\_,S_2(j))]$

Locally Similar Strings



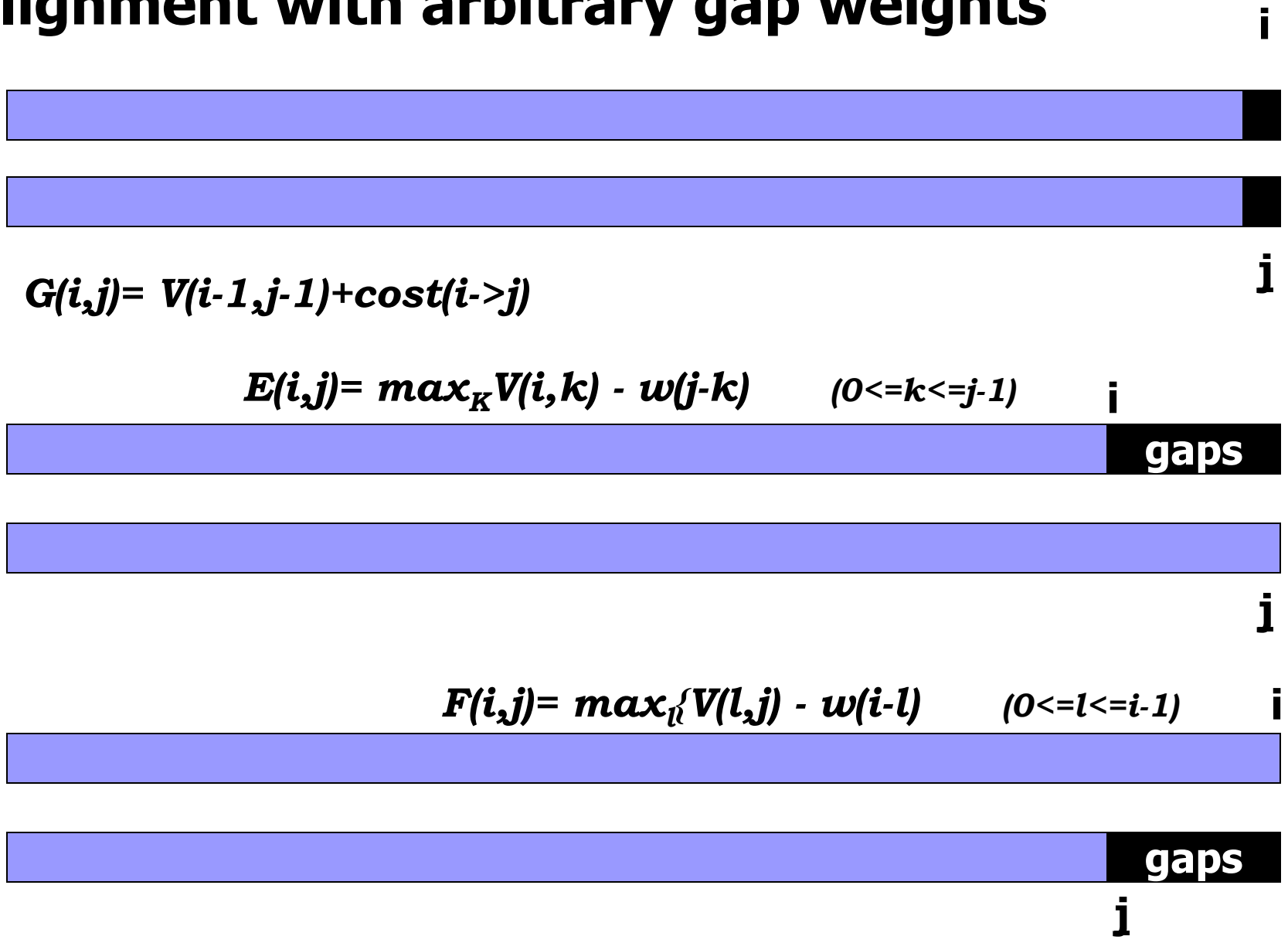
Remarks

- Global Alignment is called Needleman-Wunsch alignment
- Local alignment is called Smith-Waterman alignment
- Smith-Waterman can find regions with high similarity by simply performing a trace-back from any cell (i,j) backwards and locate a pair with similarity v(i,j).

Sequence alignment with gaps

- **Gap:** contiguous spaces, we want to control the distribution of gaps.
- **Introducing gaps:** In order to compute the gap the gap introduces in aligning 2 sequences, we can with a simple approach consider that every gap contribute a constant weight W_g, independent of its length.
- Value of alignment for "k" gaps: $$\sum_{i=1}^{l} s(S_1'(i), S_2'(i)) - kW_g$$
- A better approach is using a function that is a function of the gap length. Then we can fill a matrix: V(i,j)=max[E(i,j), F(i,j), G(i,j)]

Alignment with arbitrary gap weights



- $V(i,j)=\max\{E(i,j), F(i,j), G(i,j)\}$
- $V(i,0)=-w(i)$
- $V(0,j)=-w(j)$
- $E(i,0)=-w(i)$
- $F(0,j)=-w(j)$
- $G(0,0)=0$
- Assuming that $|S_1|=n$ and $|S_2|=m$ the recurrences can be evaluated in $O(nm^2+n^2m)$