Computational Biology

Lecture 5

Saad Mneimneh

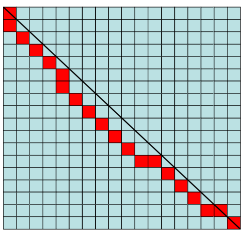
Similar Sequences

- If we believe that the two sequences are similar, then we can align them faster.
- For simplicity assume $m = n$ (since they are similar).
- If x and y align perfectly, this corresponds to a diagonal in the A matrix.
- Therefore, we expect the alignment not to deviate a lot from the diagonal.

Saad Mneimneh

Illustration

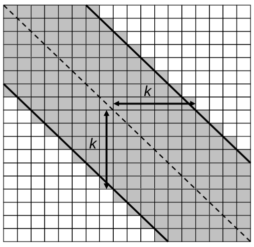
```
-GCGC-ATGGATTGAGCGA
TGCGCCATGGAT-GAGC-A
```



Saad Mneimneh

Bounded Dynamic Programming

- Only update matrix in a band of size k around the diagonal
- If $A(i,j)$ inside band then $|i - j| \le k$



Saad Mneimneh

Bounded Dynamic Programming

- Initialization
  - $A(0,j) = -d.j \quad j \le k$
  - $A(i,0) = -d.i \quad i \le k$

Running time $O(kn)$

- Main iteration
  for i= 1 to n
  for $j = \max(1, i - k)$ to $\min(n, i + k)$

$$A(i,j) = \max \begin{cases} A(i-1, j-1) + s(x_i, y_j) \\ A(i-1, j) - d \text{ if } |i - 1 - j| \le k \\ A(i, j-1) - d \text{ if } |i - j + 1| \le k \end{cases}$$

- Termination
  $A^{OPT} = A(n,n)$

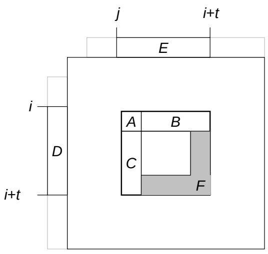
Saad Mneimneh

The 4 Russians Speedup
(for speeding up DP)

- Partition the table into t-blocks (blocks of size txt).
- Compute the values in the table one t-block at a time rather than one cell at a time.
- The goal is to spend only $O(t)$ time per block, rather than $O(t^2)$.
- Achieve a factor of t speedup.

Saad Mneimneh

t-blocks

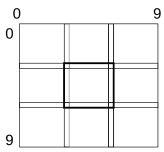


- *t*-block at position (i,j)
- F depends only on (A,B,C,D,E), i.e. Given input A, B, C, D, E, we can compute output F.
- Both input and output have size $O(t)$
- Assume the existence of a **block function** that computes F from A, B, C, D, E in $O(t)$ time.

Saad Mneimneh

DP with *t*-blocks

- Assume m=n=k(t – 1) for simplicity
- Divide table into intersecting t-blocks
- Last row of *t*-block is shared with first row of *t*-block below
- Last column of *t*-block is shared with first column of *t*-block to the right

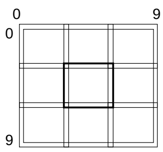


n=9
4-block
k=3

Saad Mneimneh

DP algorithm

- Initialize values in the first row and column of the full table A.
- In a rowwise manner, use the block function to determine values in the last row and last column of each block.
- The output of one block will be input of other.
- Return the value in A(n,n).



Saad Mneimneh

Running time analysis

- There are n^2/t^2 *t*-blocks.
- The output for each *t*-block is obtained in $O(t)$ time using the block function.
- Total time is $O(n^2/t)$
- But how to make the block function $O(t)$?

Saad Mneimneh

Block function

- Enumerate all possible inputs to the block function.
- For each input, pre-compute the resulting output (i.e. a row and a column) in $O(t^2)$
- Store the outputs indexed by the inputs.
- True running time of DP is
  $O(n^2/t)$ + *time of pre-computation*

Saad Mneimneh

Pre-computing the block function

- Number of input choices to the block function is large!
  - A *t*-length row (or column) can have
    around $L^t\alpha^t$ possible values
    where
    L: possible values for a cell, $L >> n$
    α: the size of the alphabet.
- So, we have $L^{2t}\alpha^{2t}$ possible inputs
- Need $L^{2t}\alpha^{2t}t^2$ time!

Saad Mneimneh

Observation

In any row, column, or diagonal of the DP table, two adjacent cells differ by at most 3.

A	*B*
C	*D* (*i*,*j*)

same row
$C - D \leq 2$ because $D \geq C - 2$ (update rule).

replace y_j with a gap in the alignment of $x_1..x_i$ and $y_1..y_j$ with score D, we get an alignment of $x_1...x_i$ and $y_1...y_{j-1}$ with score at least $D - 3$. So $C \geq D - 3 \Rightarrow D - C \leq 3$.

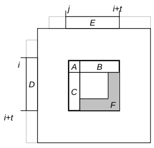
same column
Symmetric argument.

Diagonal
$A - D \leq 1$ because $D \geq A + s$ (update rule) and $s \geq -1$.
we can also show that $D - A \leq 1$.

Saad Mneimneh

Trick: Offset Encoding

- An offset vector is a t-length vector of values in {-3, -2, -1, 0, 1, 2, 3}, where the first entry must be 0.
- A row or a column can be represented as an offset vector, for instance 5, 4, 4, 5 can be represented as 0, -1, 0, +1.
- Given A,
  AB and AC as offset vectors,
  $x_i...x_{i+t-1}$, and $y_j...y_{j+t-1}$,
  we can compute the output row and column as offset vectors.



Saad Mneimneh

Example using Offset Encoding

	T	G	C	
	0 *A*	-2 *A–2*	-2 *A–4*	-2 *A–6*	0
T	-2 *A–2*	*A+1*	*A–1*	*A-3*	3
T	-2 *A–4*	*A–1*	*A*	*A–2*	1
T	-2 *A–6*	*A–3*	*A–2*	*A–1*	1
	0	3	1	1	

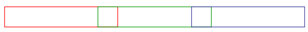
real values

offsets

Saad Mneimneh

Result

- Let v be the concatenation of all offset vectors corresponding to the last row



- $A(n,n) = A(n,0) + \Sigma_i v_i$

Time Analysis

- Number of different inputs to block function is now $7^{2t}.4^{2t} = 28^{2t}$
- Time = $28^{2t}.t^2$
- Let $t \approx 0.5\log_{28}n$ => time = $O(n.\log^2 n)$
- Total = $O(n.\log^2 n + n^2/\log n)$

Saad Mneimneh

The 4 Russians

V. L. Arlazarov, ← Only 1 Russian
E. A. Dinic,
M.A. Kronrod,
I.A. Faradzev

paper in 1970

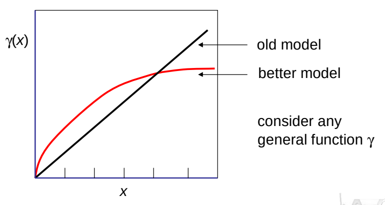
Saad Mneimneh

Gap Penalty

- So far, we assumed linear penalty
  - $\gamma(x) = dx$
- This is not realistic because gaps occur in bunches
  - A gap of length *k* is more likely than *k* gaps of length 1.
- Use a function that does not penalize additional gaps as much as the first gaps

Saad Mneimneh

Penalty function



Saad Mneimneh

New Model => New Problem

Score not additive across a gap

```
xxxxxxxxxxxxxx
yyyyy-----yyyy
```

$$\text{score} \neq \begin{pmatrix}\text{xxxxxxx}\\ \text{yyyyy}\underbrace{\text{--}}_{-\gamma(2)}\end{pmatrix} + \begin{pmatrix}\text{xxxxxxx}\\ \underbrace{\text{---}}_{-\gamma(3)}\text{yyyy}\end{pmatrix}$$

$$\gamma(5) \neq \gamma(2) + \gamma(3)$$

Saad Mneimneh

Revisit the three cases

"Three" Possible cases for aligning $x_1...x_i$ and $y_1...y_j$

1. $x_1..x_{i-1} \mid x_i$ / $y_1..y_{j-1} \mid y_j$ — $A(i,j) = A(i-1, j-1) + \begin{cases} m, \text{ if } x_i = y_j \\ -s, \text{ if not} \end{cases}$

2. $x_1..x_{i-1} \mid x_i$ / $y_1..y_j \mid -$ — $A(i,j) = A(i-1, j) - d$!not additive!

3. $x_1..x_i \mid -$ / $y_1..y_{j-1} \mid y_j$ — $A(i,j) = A(i, j-1) - d$!not additive!

Saad Mneimneh

Note

The *"Two"* unpleasant cases for aligning $x_1...x_i$ and $y_1...y_j$

2. $\boxed{x_1..x_{i-1}} \; x_i$ / $\boxed{y_1..y_j} \; -$

3. $x_1..x_i \; -$ / $y_1..y_{j-1} \; y_j$

Even if $A(i-1, j)$ corresponds to an alignment that aligns x_{i-1} with y_j, which means the score $A(i,j) = A(i-1, j) - \gamma(1)$ is additive, it does not mean that this is the best score for aligning x_i to a gap.

Why?

Because if x_{i-1} is optimally aligned with y_j, it does not mean that the alignment $x_1...x_i$ to $y_1...y_j-$ optimally aligns x_i to a gap. Think about it.

Saad Mneimneh

Fixing the problem

If the optimal alignment ends with a gap, the gap has a certain length. Take the gap length into account for the possibilities

2_1. $x_1..x_{i-1} \mid x_i$ / $y_1..y_j \mid -$

2_2. $x_1..x_{i-2} \mid x_{i-1}\; x_i$ / $y_1..y_j \mid -\; -$

2_3. $x_1..x_{i-3} \mid x_{i-2}\; x_{i-1}\; x_i$ / $y_1..y_j \mid -\; -\; -$

2_i. $\mid x_1 \,..\, x_{i-2}\; x_{i-1}\; x_i$ / $y_1..y_j \mid - \,..\, -\; -\; -$

3_1. $x_1..x_i \mid -$ / $y_1..y_{j-1} \mid y_j$

3_2. $x_1..x_i \mid -\; -$ / $y_1..y_{j-2} \mid y_{j-1}\; y_j$

3_3. $x_1..x_i \mid -\; -\; -$ / $y_1..y_{j-3} \mid y_{j-2}\; y_{j-1}\; y_j$

3_j. $x_1..x_i \mid - \,..\, -\; -\; -$ / $\mid y_1 \,..\, y_{j-2}\; y_{j-1}\; y_j$

Saad Mneimneh

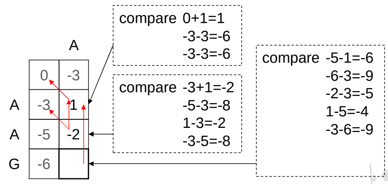
Update Rule
Needleman-Wunsch

$$A(i,j) = \max \begin{cases} A(i-1,j-1) + s \\ A(i-k,j) - \gamma(k) \quad \text{for } k = 1 \ldots i \\ A(i,j-k) - \gamma(k) \quad \text{for } k = 1 \ldots j \end{cases}$$

Initialization should satisfy γ.
Termination as before.

Example

- x = AAG, y = A
- $\gamma(1) = 3$, $\gamma(2) = 5$, $\gamma(3) = 6$



AAG
A - -

Running time

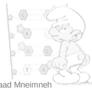
- To compute $A(i,j)$ we need $(1 + i + j)$ time.
- Total = $\Sigma_{i,j}(1 + i + j) \approx mn + mn^2 + nm^2$

There is a catch!

- We did not assume anything about $\gamma(x)$.
- In fact, this new algorithm will not work correctly for any arbitrary $\gamma(x)$.
- It has to satisfy

$$\gamma(x+1) - \gamma(x) \leq \gamma(x) - \gamma(x-1)$$

- The book has a solution that does not require the above condition.



Saad Mneimneh