Central Issues in Biological Sequence Comparison

Definitions: What is one trying to find or optimize?

Algorithms: Can one find the proposed object optimally or in reasonable time optimize?

Statistics: Can one's result be explained by chance?

In general there is a tension between questions. A definition that is too simple may allow efficient algorithms, but may not yield results of biological interest. However, a definition that includes most of the relevant biology may entail intractable algorithms and statistics. The most successful approaches find a balance between these considerations.

The Problem

Given: Two protein or DNA sequences

$$X \equiv x_1 x_2 x_3 \ldots x_m$$
$$Y \equiv y_1 y_2 y_3 \ldots y_n$$

where the x_i and y_i are chosen from a finite alphabet $\boldsymbol{\mathcal{A}}$, e.g. $\{A, C, G, T\}$.

How can one define the *distance* between the sequences X and Y, or alternatively their *similarity*?

We shall adopt the somewhat more flexible formalism of *similarity*, with higher values considered better.

Although there are other possibilities, similarity is generally defined with reference to a *sequence alignment*, in which individual letters from each sequence are placed into correspondence.

Examples of Sequence Alignment

```
groan     colo-r          theatre     theatre
|||:|     |||| |          |||||::     |||||✕
grown     colour          theater     theater
```

```
elephant          vermiform     vermiform-----
|||: |||          ::||:::::          ||||
eleg-ant          formation     -----formation
```

```
disestablishment       disestablishment
|||        |  |||       |||          :|||
dis-------s--ent       dis---------sent
```

Applications

Sequence alignment arises in many fields:

- Molecular biology
- Inexact text matching (e.g. spell checkers; web page search)
- Speech recognition

In general:

The precise definition of what constitutes an alignment may vary by field, and even within a field.

Many different alignments of two sequences are possible, so to select among them one requires an objective (score) function on alignments.

The number of possible alignments of two sequences grows exponentially with the length of the sequences. Good algorithms are required.

NCBI BLAST Protein Database Search

BLAST is a widely-used program for searching DNA and protein sequence databases for sequences similar to a query sequence.

Here is one alignment returned by a BLAST protein database search:

```
>sp|Q99728.2|BARD1_HUMAN
Length=777

 GENE ID: 580 BARD1 | BRCA1 associated RING domain 1 [Homo sapiens]

 Score = 53.1 bits (126),  Expect = 3e-07, Method: Composition-based stats.
 Identities = 32/111 (29%), Positives = 55/111 (50%), Gaps = 15/111 (14%)

Query  24   THVVMKTDAEFVCERTLKYFLGIAGGKWVVSYFWVTQSIKERKMLNEHDFEVRGDVVNGR  83
            THVV+  DA    + TLK  LGI  G W++ + WV   ++ +    E  +E+
Sbjct  605  THVVVPGDA---VQSTLKCMLGILNGCWILKFEWVKACLRRKVCEQEEKYEIP-------  654

Query  84   NHQGPKRARESQDR---KIFRGLEICCYGPFTNMPTDQLEWMVQLCGASVV  131
              +GP+R+R ++++   K+F G     +G F + P D L  +V   G  ++
Sbjct  655  --EGPRRSRLNREQLLPKLFDGCYFYLWGTFKHHPKDNLIKLVTAGGGQIL  703
```

Elements of Global Sequence Alignment

No crossings allowed. For algorithmic reasons, it is fortunate that, although there are natural mechanisms (mutations) that lead to amino acid or nucleotide substitutions, insertions and deletions, there are none that yield transpositions, unlike with keyboard-produced text. In contrast, when analyzing RNA folding, one may choose for algorithmic reasons to exclude "pseudoknots", which do in fact occur naturally.

Gaps. An arbitrary number of *null* characters (represented by dashes) may be placed into either sequence, and aligned with letters in the other sequence. Two nulls may not be aligned. Depending upon one's perspective, the alignment of a letter with a null may be understood as the *insertion* of a letter into one sequence, or the *deletion* of a letter from the other. Therefore, a letter aligned with a null is sometimes called an *indel*.

Alignment scores. The score for an alignment is taken to be the sum of scores for aligned pairs of letters, and scores for letters aligned with nulls. Each such pairing is called an *alignment column*.

Substitution scores. Scores for aligned pairs of letters are called *substitution scores*, whether the letter aligned are identical or not. Most simply, substitution scores may take the form of *match* scores and *mismatch* scores.

Gap scores. The score for a letter aligned with a null is called a *gap score*. Usually gap scores are letter-independent.

Global alignment. All letters and nulls in each sequence must be aligned.

Sequence Similarity

Define the *similarity* of two sequences as the score of their highest-scoring (optimal) alignment.

How do we find the an optimal alignment of two sequence, and its score?

Brute force enumeration is impractical, because the number of possible alignments becomes astronomically large for even fairly short sequences.

Fortunately, the problem is soluble efficiently using a technique called *dynamic programming*.

Dynamic Programming and Global Alignment

Dynamic programming is a method by which a larger problem may be solved by first solving smaller, partial versions of the problem. We demonstrate here how it may be applied to global sequence alignment, where at first we are interested only in the similarity of two sequences, and not the alignment that yields this score.

Definitions:

$s(a, b)$	the substitution score for aligning letters a and b
g	the gap score for aligning any letter to a null
X_i	the partial sequence consisting of the first i letters of $X \equiv x_1 x_2 \ldots x_m$
Y_j	the partial sequence consisting of the first j letters of $Y \equiv y_1 y_2 \ldots y_n$
$SIM(i, j)$	the similarity of X_i and Y_j

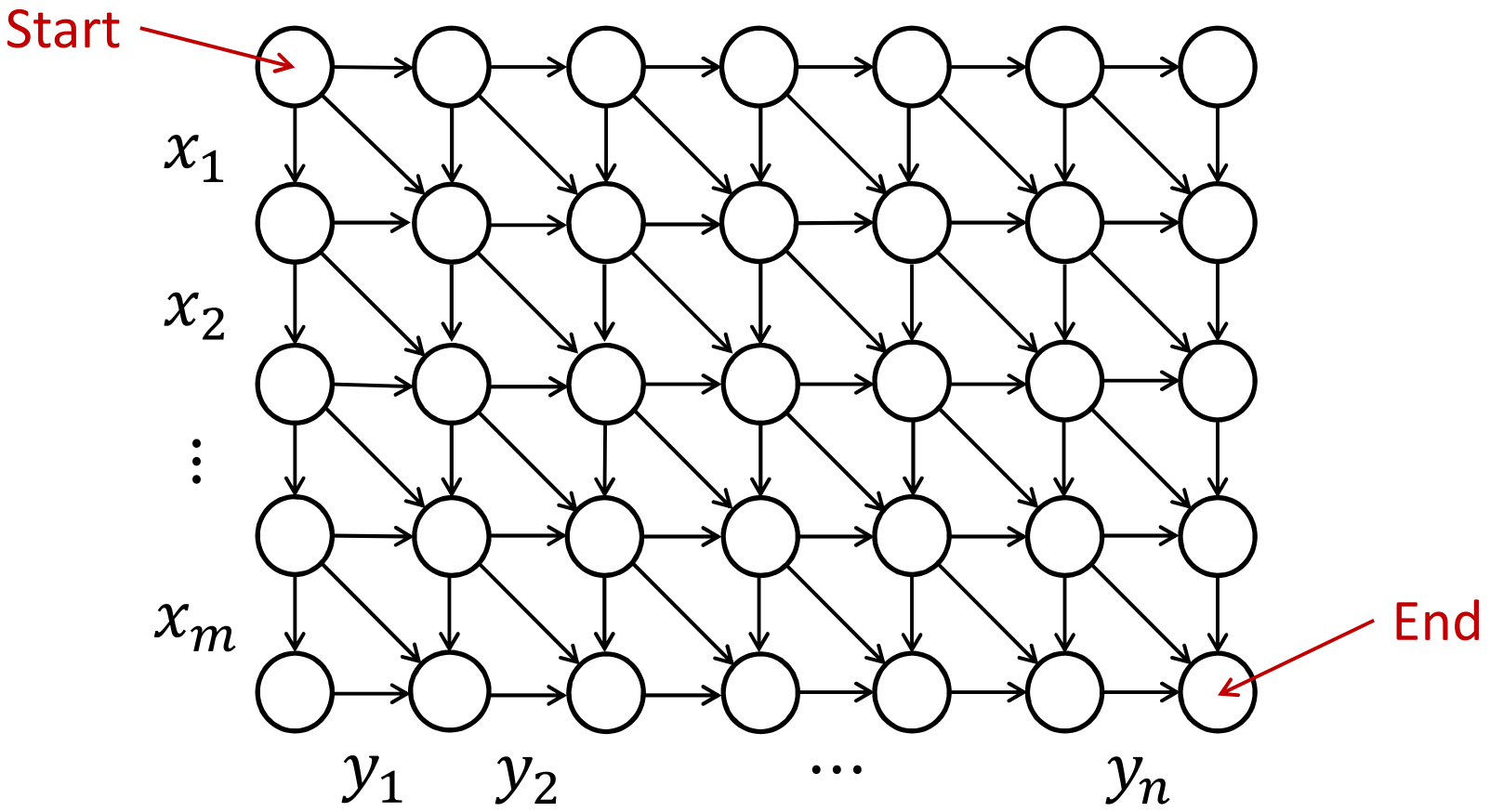
Consider the *last column* of an optimal alignment of X_i and Y_j. This column either aligns x_i to y_j, or x_i to a null, or y_j to a null. Because we do not allow "crossing", there are no other possibilities. This observation yields the following recurrence:

$$\boldsymbol{SIM(i,j) = max}\begin{cases} \boldsymbol{SIM(i-1, j-1) + s(x_i, y_j)} & x_i \text{ and } y_j \text{ aligned} \\ \boldsymbol{SIM(i-1,\ j) + g} & x_i \text{ aligned with a null} \\ \boldsymbol{SIM(i,\ j-1) + g} & y_j \text{ aligned with a null} \end{cases}$$

In brief, we can solve for $SIM(m, n)$ by solving smaller versions of the problem first.

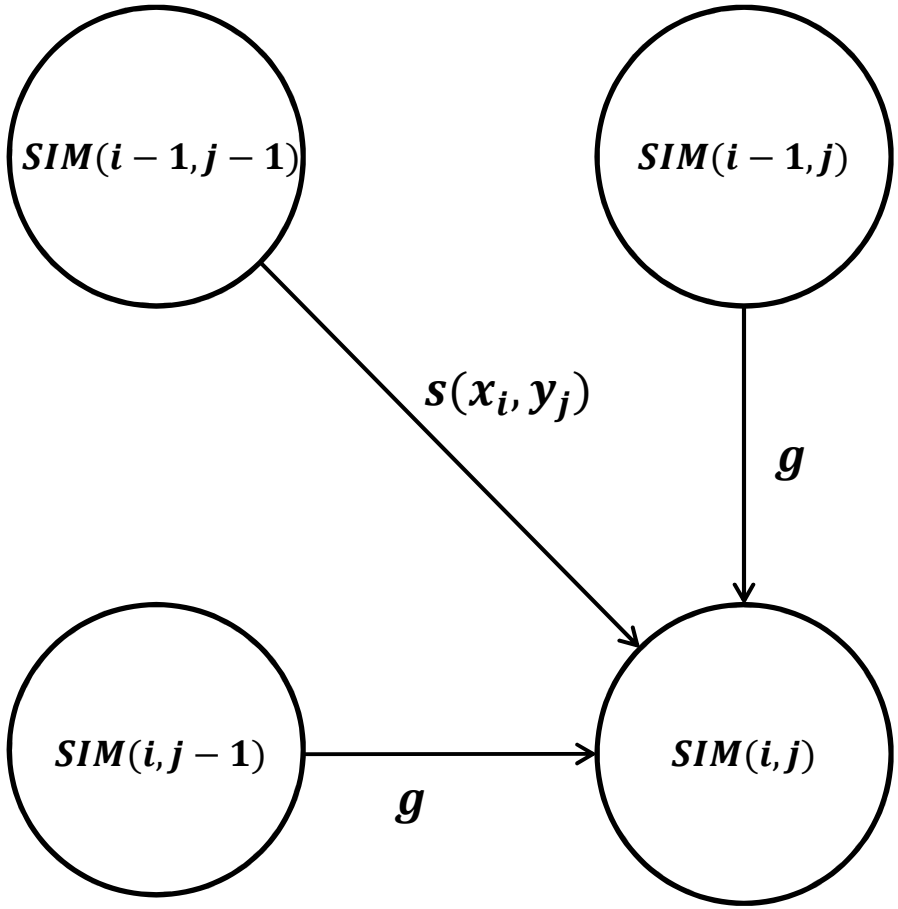
Path graphs

A global alignment may be viewed as a path through a directed *path graph* which begins at the upper left corner and ends at the lower right. Diagonal steps correspond to substitutions, while horizontal or vertical steps correspond to indels. Scores are associated with each edge, and the score of an alignment is the sum of the scores of the edges it traverses. Each alignment corresponds to a unique path, and vice versa.

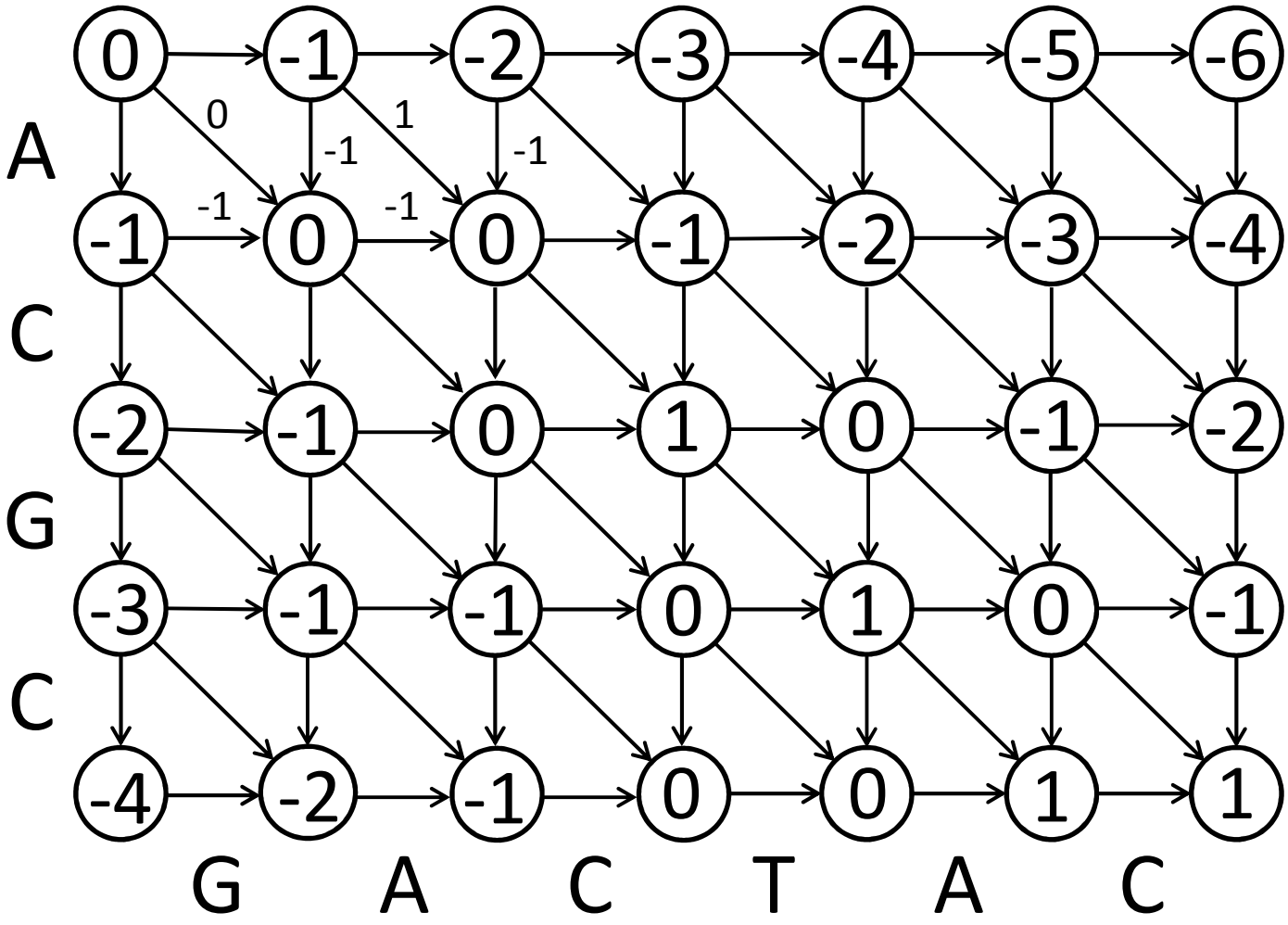


Dynamic programming on path graphs

One may associate a partial similarity with each node of a path graph. If the values of $SIM(i-1, j-1)$, $SIM(i-1, j)$ and $SIM(i, j-1)$ are known, the value of $SIM(i, j)$ may be calculated.

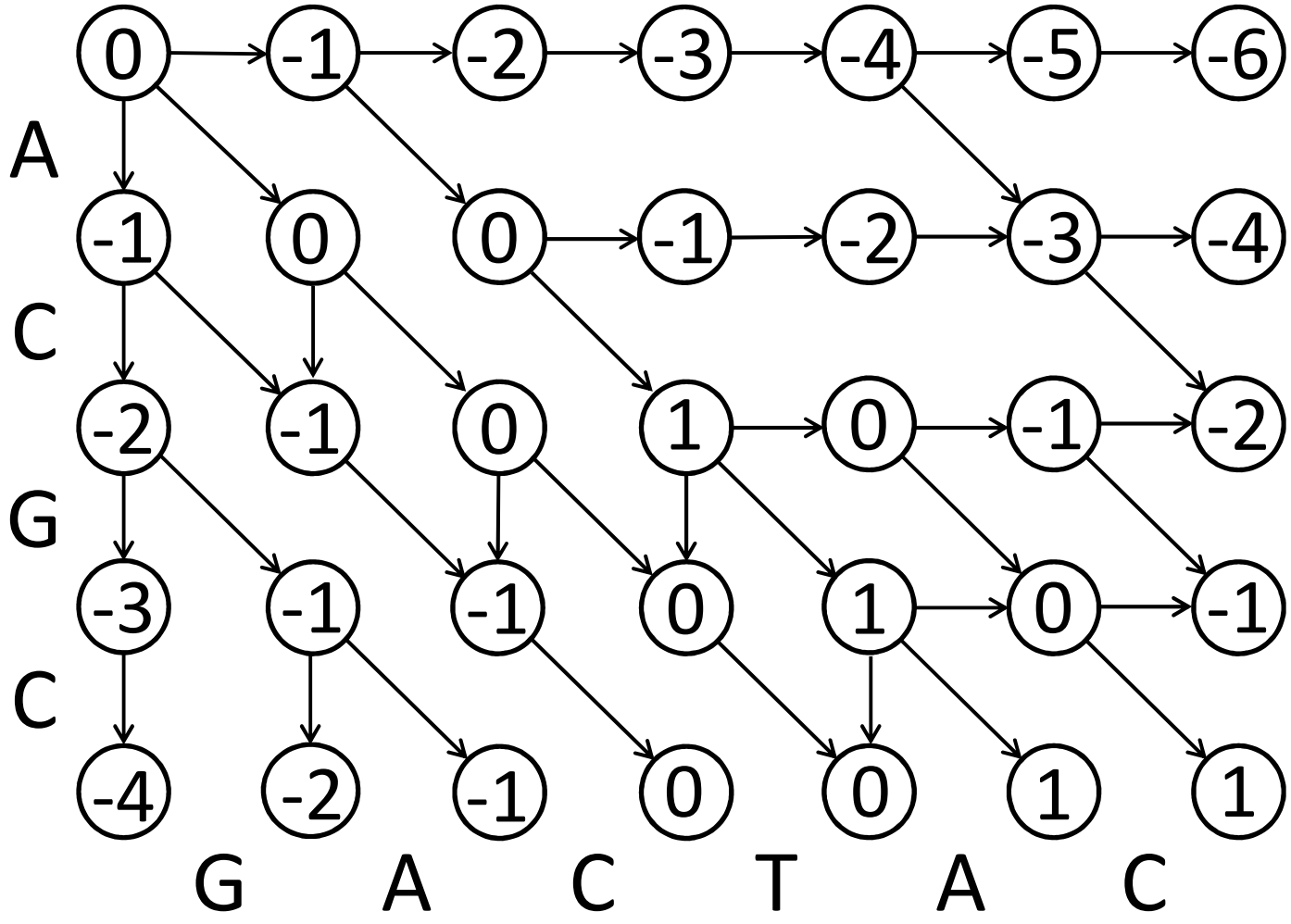


An Example



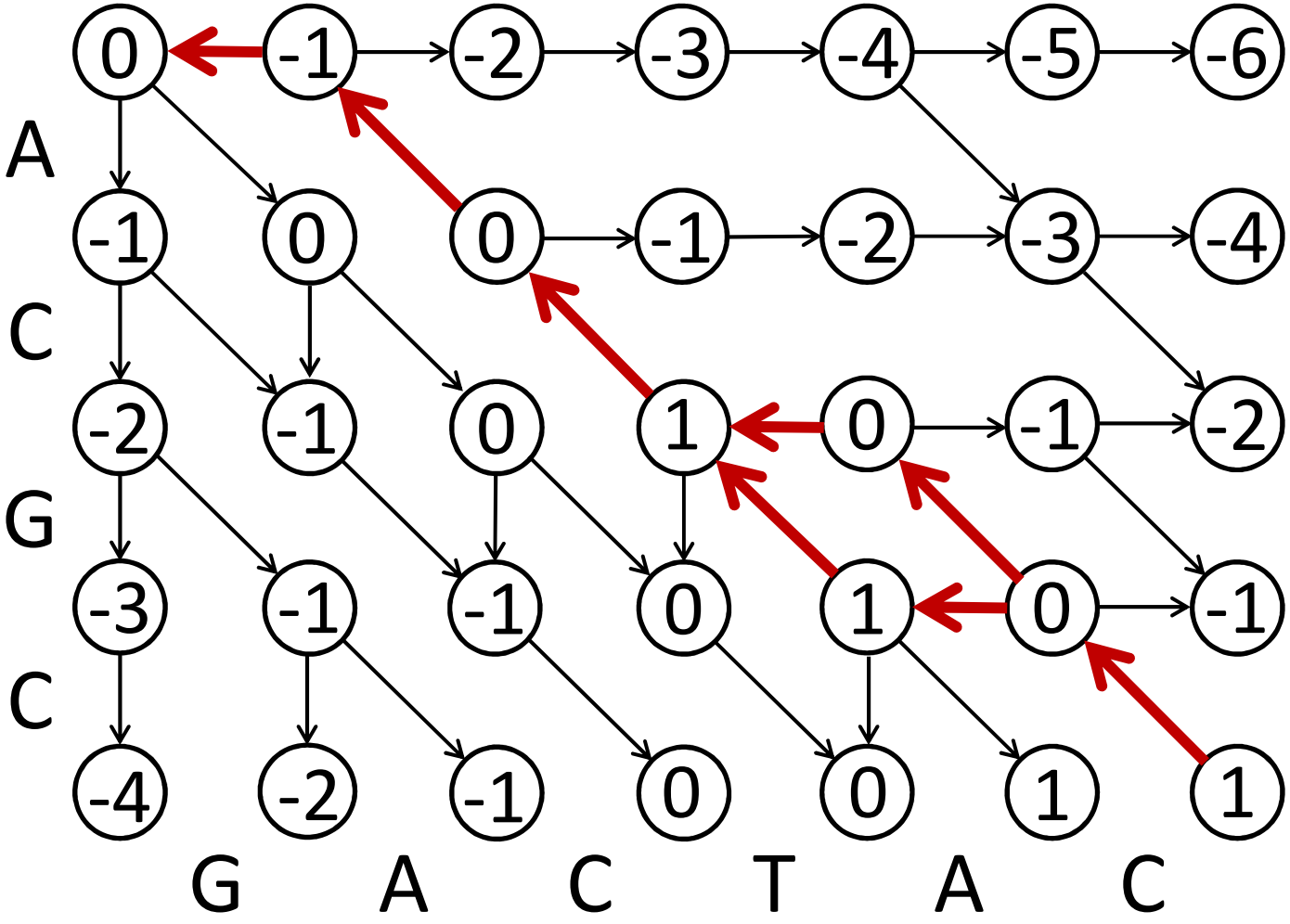
Scores: Match +1 Mismatch 0 Gap -1

What is the optimal alignment?



Record *traceback information*: Which edge or edges led to the optimal score at each node?

Follow the traceback edges from the final node



Optimal alignments:

–ACG–C
GACTAC

and

–AC–GC
GACTAC

Pseudocode for Finding Sequence Similarity

```
Similarity(X,Y):
   For i = 0,...,m:  SIM[i,0] = i*g
   For j = 1,...,n:   SIM[0,j] = j*g
   For i = 1,...,m:
      For j = 1,...,n:
         SIM[i,j] = max(
            SIM[i-1,j-1] + s(X[i],Y[j]),
            SIM[i-1,j]+g,
            SIM[i,j-1]+g
         )
      EndFor
   EndFor
Return SIM[m,n]
```

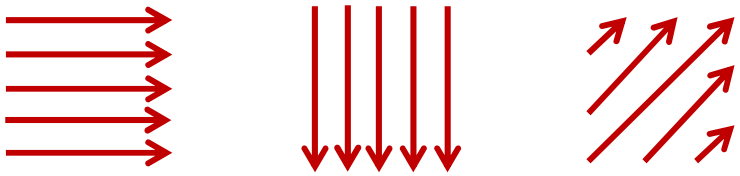
Exercise: Generalize the code to include traceback information, and produce one optimal alignment.

Note: This is generally known as the *Needleman-Wunsch algorithm*, after the first paper in the field of computational molecular biology to apply dynamic programming to the global alignment problem. However, the paper actually describes a somewhat different algorithm which is almost never used.

Needleman, S.B. & Wunsch, C.D. (1970) "A general method applicable to the search for similarities in the amino acid sequences of two proteins." *J. Mol. Biol.* **48**:443-453.

Observations and Generalizations

The nodes can be expanded in a variety of orders, so long as all nodes that "feed into" a given node are expanded before that node. Possible expansion orders are:



The time complexity of the algorithm is $O(mn)$.

If only the similarity is desired, the space complexity is $O[\min(m,n)]$; if an optimal alignment is sought, the space complexity is $O(mn)$, but as we shall see, this too can be reduced to $O[\min(m,n)]$.

It is possible to save time (but in general no more than a constant factor) by not expanding nodes that can not possibly participate in an optimal path.

Fickett, J.W. (1984) *Nucl. Acids Res*. **12**:175-180; Spouge, J.L. (1989) *SIAM J. Appl. Math*. **49**:1552-1566.

Global Alignment Scores

Multiplying all substitution and gap scores by a positive constant does not change the optimal alignment. Why?

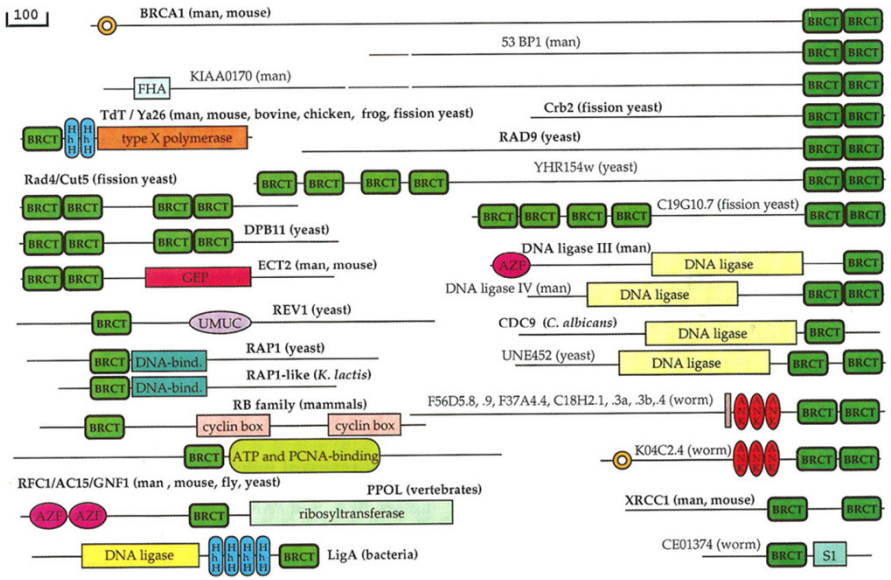
Adding a constant k to all substitution scores, and $k/2$ to all gap scores, does not change the optimal alignment. Why?

A global alignment scoring system with the three nominal parameters of match score a, mismatch score b, and gap score g, in fact has a single free parameter. For example, assuming $a > g$, one can always construct an equivalent scoring system with $a = 1$ and $g = 0$. What is the scoring system of this form equivalent to $(a = 1,\ b = 0,\ g = -1)$?

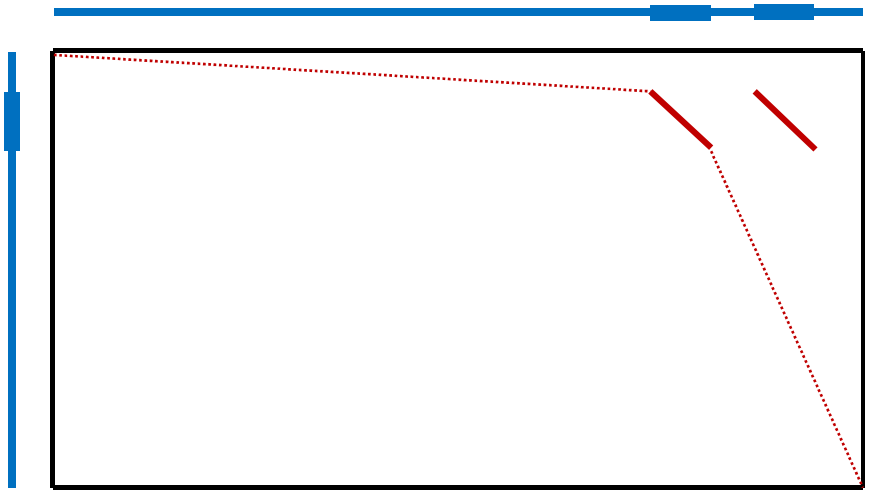
Modifying global alignment scores so that $g = 0$ can speed up the inner loop of the dynamic programming algorithm.

Local Alignment: Motivation

In the early days of protein sequence comparison, most known related proteins, were related over their whole lengths. However, soon proteins that shared only isolated regions of similarity were found. A schematic of a protein *superfamily* is shown here, with related *domains* represented by similar boxes.



The measure of global sequence similarity, and the Needleman-Wunsch alignment algorithm, was not well-adapted to finding such domains. A new definition of local similarity was required, along with a new algorithm for finding locally optimal alignments.



Local Alignment: Definition

During the 1970s and early 1980s, a variety of definitions for local alignment were proposed. The one that eventually gained the greatest popularity, along with an associated algorithm, is due to Smith & Waterman.

Smith & Waterman proposed simply that a local alignment of two sequences allow arbitrary-length segments of each sequence to be aligned, with no penalty for the unaligned portions of the sequences. Otherwise, the score for a local alignment is calculated the same way as that for a global alignment.

It would at first appear that the problem of finding an optimal local alignment should be significantly more complex than the problem of finding an optimal global alignment, because the start and stop positions of the alignment must be located as well. However, only a constant factor more calculation is necessary.

Smith, T.F. & Waterman, M.S. (1981) "Identification of common molecular subsequences." *J. Mol. Biol.* **147**:195-197.

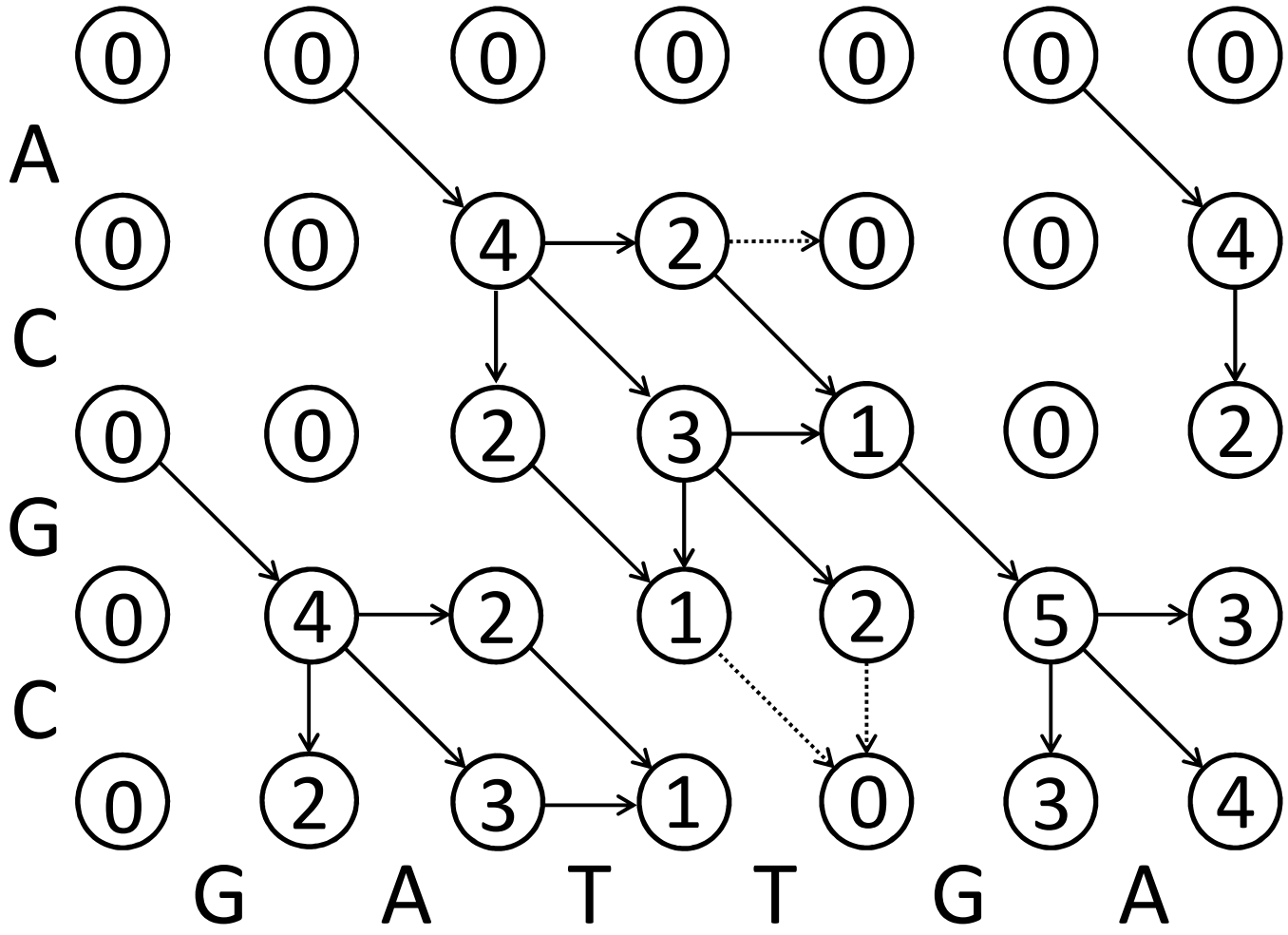
The Smith-Waterman Algorithm

Two modifications to Needleman-Wunsch:

1) Allow a node to start at 0.

2) Record the highest-scoring node, and trace back from there.

Why does this algorithm yield an optimal local alignment?



Scores: Match +4 Mismatch -1 Gap -2

Pseudocode for Finding Local Sequence Similarity

```
Local_Similarity(X,Y):
   S=0
   For i = 0,...,m:  SIM[i,0] = i*g
   For j = 1,...,n:   SIM[0,j] = j*g
   For i = 1,...,m:
      For j = 1,...,n:
         SIM[i,j] = max(
            0,
            SIM[i-1,j-1] + s(X[i],Y[j]),
            SIM[i-1,j]+g,
            SIM[i,j-1]+g
         )
         S=max(S,SIM[i,j])
      EndFor
   EndFor
Return S
```

Exercise: Generalize the code to include traceback information, and produce one optimal local alignment.

Multiplying all substitution and gap scores by a positive constant does not change the optimal alignment. Why?

Adding a constant k to all substitution scores, and $k/2$ to all gap scores, *can* change the optimal alignment. Why?

The Smith-Waterman Algorithm: Traceback

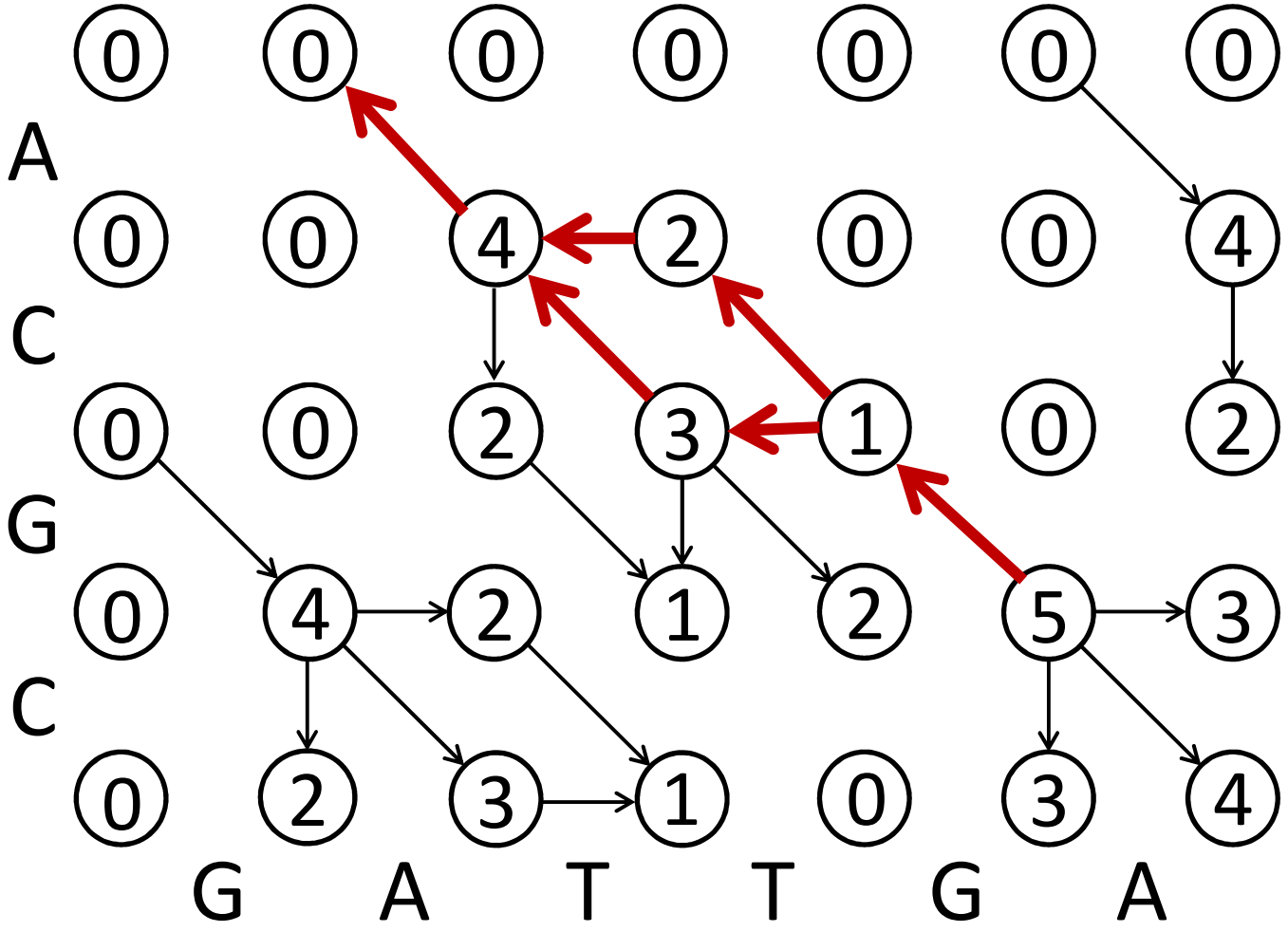
Optimal local alignments, or *subalignments*:

AC–G / ATTG and **A–CG / ATTG**

Questions:

Can one find other *locally optimal* subalignments?

How can they be defined?



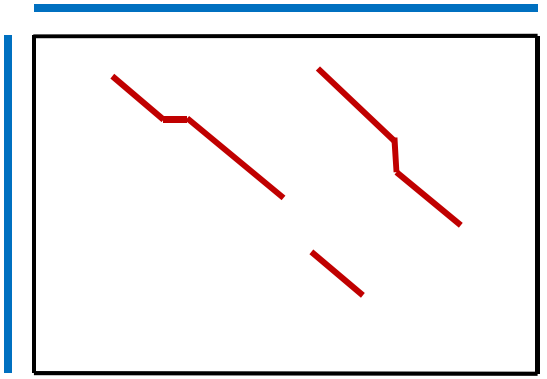
Scores: Match +4 Mismatch -1 Gap -2

Local optimality: Definitions and Algorithms

A definition of local optimality was proposed in 1984, along with an algorithm to find all locally optimal subalignments. [Sellers, P.H. (1984) *Bull. Math. Biol*. **46**:501-514.]

A subalignment is *locally optimal* if its score is greater than or equal to that of any subalignment it "touches".

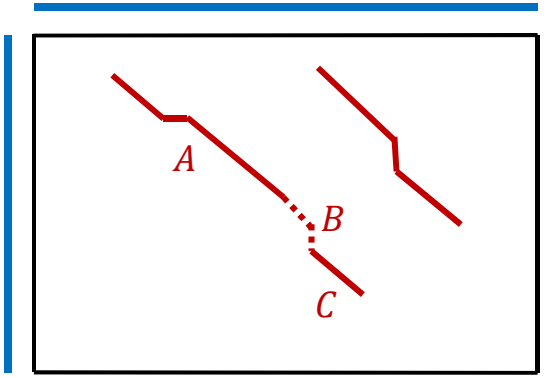
A provably $O(mn)$ algorithm for finding all locally optimal subalignments was subsequently described. [Altschul, S.F. & Erickson, B.W (1986) *Bull. Math. Biol*. **48**:633-660.]



Problem: By Sellers' definition, a strong subalignment can suppress, by means of intermediaries, subalignments it does not actually touch. This can be a particular problem if one is seeking internal approximate repeats.

One may advance an alternative definition to address this problem: A subalignment is *weakly locally optimal* if it touches no weakly locally optimal subalignment that has greater score (Altschul & Erickson, 1986). This definition is not circular, but recursive.

No $O(mn)$ algorithm for finding all weakly locally optimal subalignments of two sequences has been described, although several incorrect ones have been published.



$SIM(A) > SIM(B) > SIM(C)$

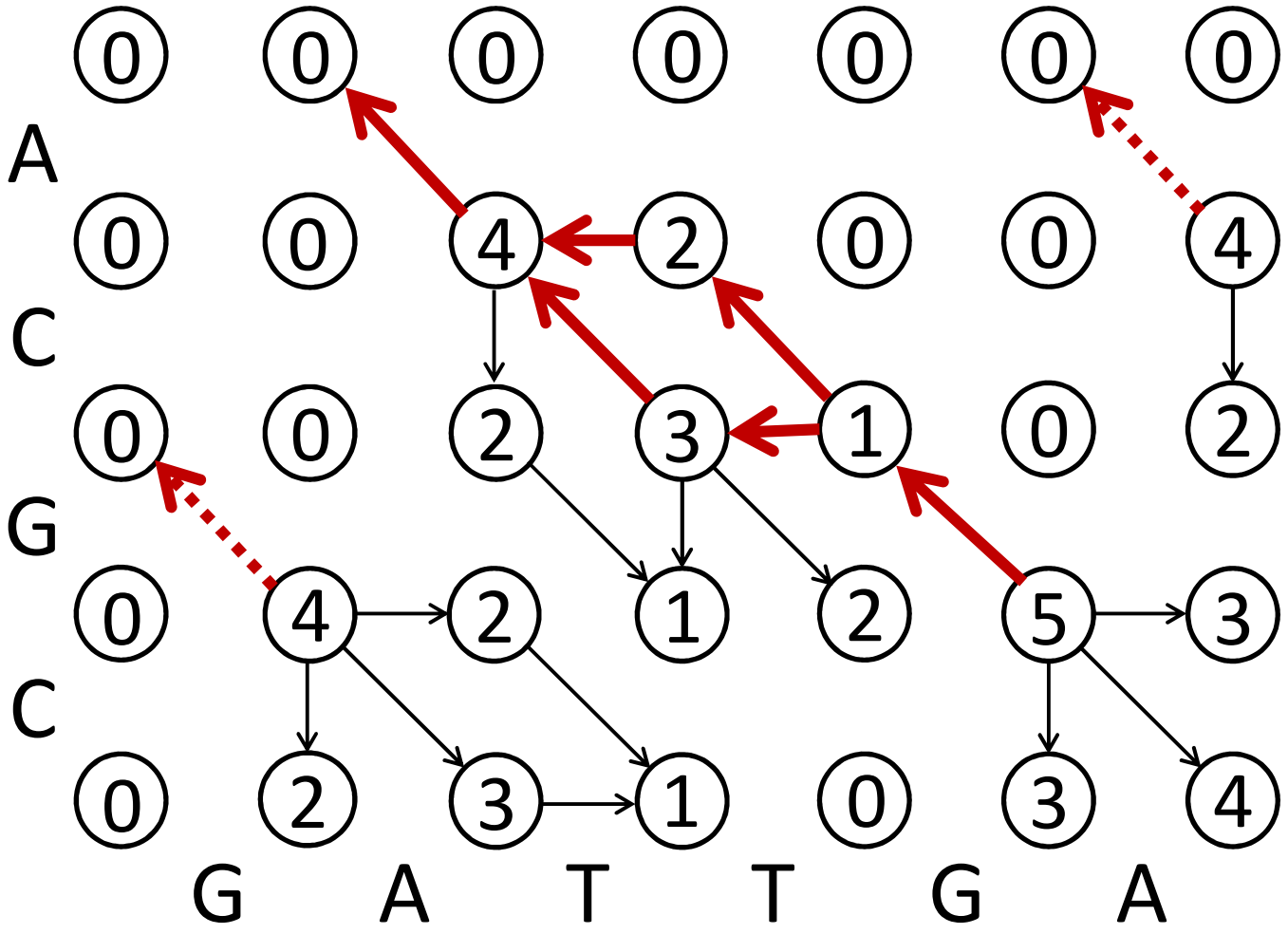
Locally Optimal Subalignments

Optimal subalignments:

```
AC-G     and    A-CG
ATTG            ATTG
```

Additional, locally optimal subalignments:

```
G     and    A
G            A
```

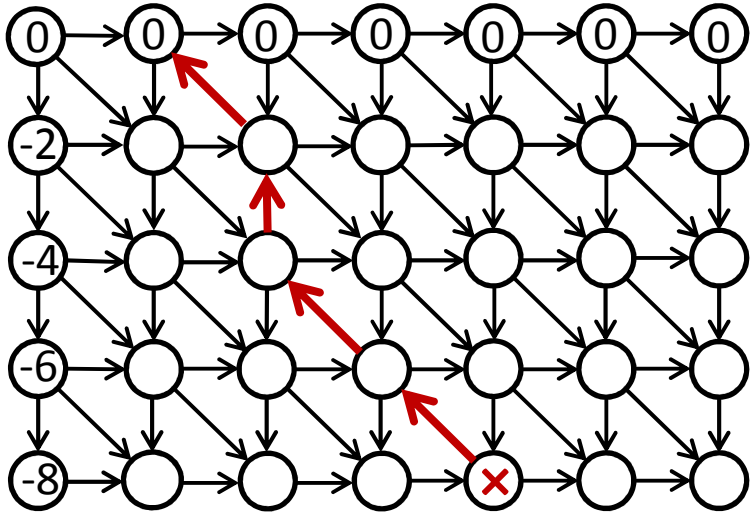


Scores: Match +4 Mismatch -1 Gap -2

Semi-Global Alignment

<u>Biological problem</u>: Find approximate matches to a given pattern within a large sequence. For example, seek promoters within a DNA sequence, or a copies of a domain within a protein sequence.

<u>Solution</u>: *Semi-global alignment*. Needleman-Wunsch algorithm with two modifications: 1) Penalize end gaps in the pattern, but not in the long sequence; 2) Trace back from the highest scoring node along the long edge of the path graph.



Erickson, B.W. & Sellers, P.H. (1983) "Recognition of patterns in genetic sequences." In *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff & J.B. Kruskal (eds.), pp. 55-91, Addison-Wesley, Reading, MA.