Global Alignment and Local Alignment

In *global alignment*, two sequences to be aligned are assumed to be generally similar over their entire length. Alignment is carried out from beginning to end of both sequences to find the best possible alignment across the entire length between the two sequences. This method is more applicable for aligning two closely related sequences of roughly the same length. For divergent sequences and sequences of variable lengths, this method may not be able to generate optimal results because it fails to recognize highly similar local regions between the two sequences. Known also as Needleman-Wunsh algorithm for global alignment.
Local alignment, on the other hand, does not assume that the two sequences in question have similarity over the entire length. It only finds local regions with the highest level of similarity between the two sequences and aligns these regions without regard for the alignment of the rest of the sequence regions. This approach can be used for aligning more divergent sequences with the goal of searching for conserved patterns in DNA or protein sequences. The two sequences to be aligned can be of different lengths. Known also as Smith-Waterman algorithm for local alignment.

Gap Penalties

Performing optimal alignment between sequences often involves applying gaps that represent insertions and deletions. Because in natural evolutionary processes insertion and deletions are relatively rare in comparison to substitutions, introducing gaps should be made more difficult computationally, reflecting the rarity of insertional and deletional events in evolution. If the penalty values are set too low, gaps can become too numerous to allow even nonrelated sequences to be matched up with high similarity scores. If the penalty values are set too high, gaps may become too difficult to appear, and reasonable alignment cannot be achieved, which is also unrealistic. Another factor to consider is the cost difference between opening a gap and extending an existing gap. It is known that it is easier to extend a gap that has already been started. Thus, gap opening should have a much higher penalty than gap extension. This is based on the rationale that if insertions and deletions ever occur, several adjacent residues are likely to have been inserted or deleted together. These differential gap penalties are also referred to as *affine gap penalties*. The normal strategy is to use preset gap penalty values for introducing and extending gaps. For example, one may use a −12/ −1 scheme in which the gap opening penalty is −12 and the gap extension penalty −1.

The total gap penalty (W) is a linear function of gap length, which is calculated using the formula:

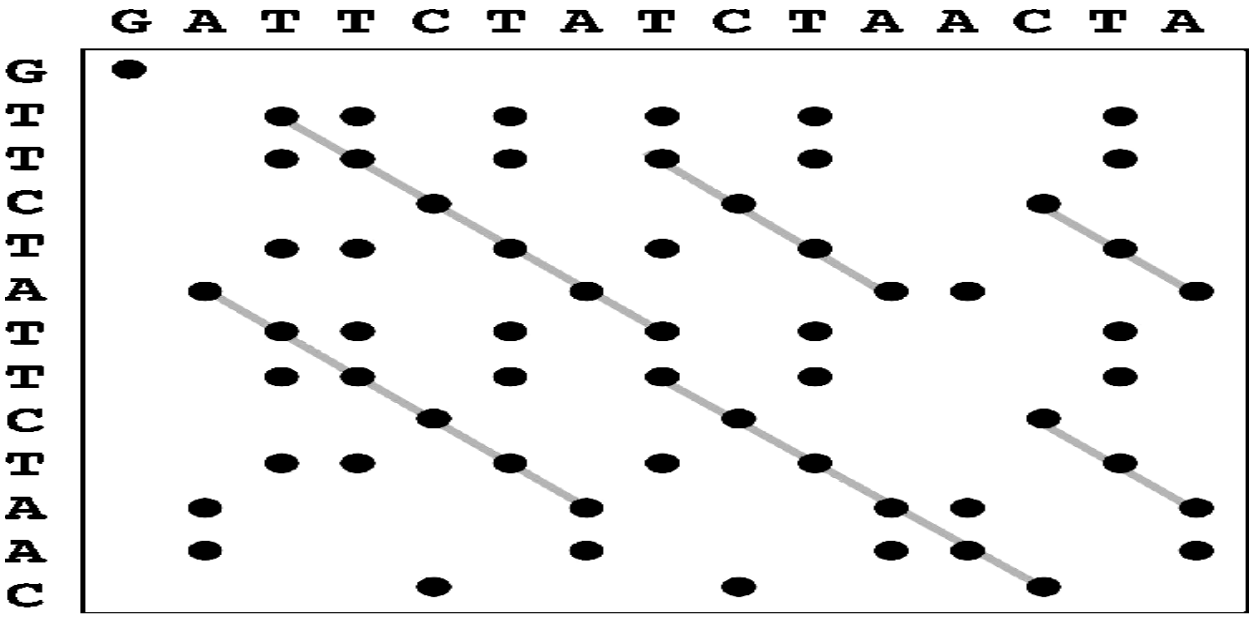
$W = \gamma + \delta \times (k-1)$

Where γ is the gap opening penalty, δ is the gap extension penalty, and k is the length of the gap. Besides the affine gap penalty, a *constant gap penalty* is sometimes also used, which assigns the same score for each gap position regardless whether it is opening or extending. However, this penalty scheme has been found to be less realistic than the affine penalty.

Dot Matrix Method

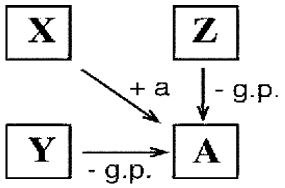
The most basic sequence alignment method is the dot matrix method, also known as the *dot plot method.* It is a graphical way of comparing two sequences in a two dimensional matrix. In a dot matrix, two sequences to be compared are written in the horizontal and vertical axes of the matrix. The comparison is done by scanning each residue of one sequence for similarity with all residues in the other sequence.

If a residue match is found, a dot is placed within the graph. Otherwise, the matrix positions are left blank. When the two sequences have substantial regions of similarity, many dots line up to form contiguous diagonal lines, which reveal the sequence alignment. If there are interruptions in the middle of a diagonal line, they indicate insertions or deletions. Parallel diagonal lines within the matrix represent repetitive regions of the sequences. A problem exists when comparing large sequences using the dot matrix method, namely, the high noise level. In most dot plots, dots are plotted all over the graph, obscuring identification of the true alignment. For DNA sequences, the problem is particularly acute because there are only four possible characters in DNA and each residue therefore has a one-in-four chance of matching a residue in another sequence. To reduce noise, instead of using a single residue to scan for similarity, a filtering technique has to be applied, which uses a “window” of fixed length covering a stretch of residue pairs. When applying filtering, windows slide across the two sequences to compare all possible stretches. Dots are only placed when a stretch of residues equal to the window size from one sequence matches completely with a stretch of another sequence. This method has been shown to be effective in reducing the noise level. The window is also called a *tuple*, the size of which can be manipulated so that a clear pattern of sequence match can be plotted. However, if the selected window size is too long, sensitivity of the alignment is lost.



Dynamic Programming Method

Dynamic programming is a method that determines optimal alignment by matching two sequences for all possible pairs of characters between the two sequences. It is fundamentally similar to the dot matrix method in that it also creates a two dimensional alignment grid. However, it finds alignment in amore quantitative way by converting a dot matrix into a scoring matrix to account for matches and mismatches between sequences. By searching for the set of highest scores in this matrix, the best alignment can be accurately obtained. The best score is put into the bottom right corner of an intermediate matrix. This process is iterated until values for all the cells are filled. Thus, the scores are accumulated along the diagonal going from the upper left corner to the lower right corner. Once the scores have been accumulated in matrix, the next step is to find the path that represents the optimal alignment. This is done by tracing back through the matrix in reverse order from the lower right-hand corner of the matrix toward the origin of the matrix in the upper left-hand corner. The best matching path is the one that has the maximum total score. If two or more paths reach the same highest score, one is chosen arbitrarily to represent the best alignment. The path can also move horizontally or vertically at a certain point, which corresponds to introduction of a gap or an insertion or deletion for one of the two sequences.



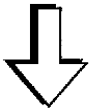
A is the maximum score from one of the three directions plus matching score at the current position

	A	T	T	G	C
A	1	0	0	0	0
G					
G					
C					

	A	T	T	G	C
A	1	0	0	0	0
G	0	1			
G					
C					

	A	T	T	G	C
A	1	0	0	0	0
G	0	1	1		
G					
C					

	A	T	T	G	C
A	1	0	0	0	0
G	0	1	1	2	
G					
C					



	A	T	T	G	C
A	1	0	0	0	0
G	0	1	1	2	2
G	0	1	1	3	3
C	0	1	1	3	4

	A	T	T	G	C
A	1	0	0	0	0
G	0	1	1	2	2
G	0	1	1	3	3
C	0	1	1	3	4

Final Alignment:

```
A T T G C
|     | |
A - G G C
```

Dynamic Programming for Global Alignment

The following is an example of global sequence alignment using Needleman/Wunsch techniques. For this example, the two sequences to be globally aligned are

G A A T T C A G T T A (sequence #1)
G G A T C G A (sequence #2)

So M = 11 and N = 7 (the length of sequence #1 and sequence #2, respectively)

A simple scoring scheme is assumed where

- $S_{i,j}$ = 1 if the residue at position i of sequence #1 is the same as the residue at position j of sequence #2 (match score); otherwise
- $S_{i,j}$ = 0 (mismatch score)
- w = 0 (gap penalty)

Three steps in dynamic programming

1. Initialization
2. Matrix fill (scoring)
3. Traceback (alignment)

Initialization Step

The first step in the global alignment dynamic programming approach is to create a matrix with M + 1 columns and N + 1 rows where M and N correspond to the size of the sequences to be aligned.

Since this example assumes there is no gap opening or gap extension penalty, the first row and first column of the matrix can be initially filled with 0.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Matrix Fill Step

One possible (inefficient) solution of the matrix fill step finds the maximum global alignment score by starting in the upper left hand corner in the matrix and finding the maximal score $M_{i,j}$ for each position in the matrix. In order to find $M_{i,j}$ for any i,j it is minimal to know the score for the matrix positions to the left, above and diagonal to i, j. In terms of matrix positions, it is necessary to know $M_{i-1,j}$, $M_{i,j-1}$ and $M_{i-1,\ j-1}$.

For each position, $M_{i,j}$ is defined to be the maximum score at position i,j; i.e.

$M_{i,j}$ **= MAXIMUM[**
$M_{i-1,\ j-1}$ **+** $S_{i,j}$ (match/mismatch in the diagonal),
$M_{i,j-1}$ **+ w** (gap in sequence #1),
$M_{i-1,j}$ **+ w** (gap in sequence #2)**]**

Note that in the example, $M_{i-1,j-1}$ will be red, $M_{i,j-1}$ will be green and $M_{i-1,j}$ will be blue.

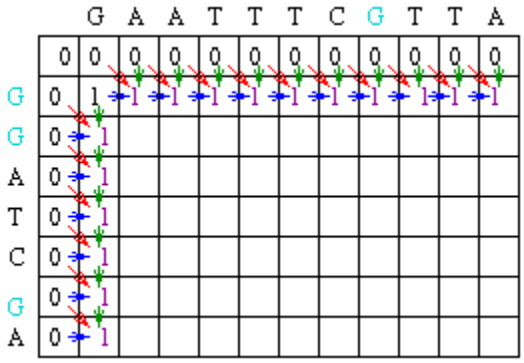
Using this information, the score at position 1,1 in the matrix can be calculated. Since the first residue in both sequences is a G, $S_{1,1} = 1$, and by the assumptions stated at the beginning, w = 0. Thus, $M_{1,1} = MAX[M_{0,0} + 1, M_{1,\ 0} + 0, M_{0,1} + 0] = MAX\ [1, 0, 0] = 1$.

A value of 1 is then placed in position 1,1 of the scoring matrix.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1										
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Since the gap penalty (w) is 0, the rest of row 1 and column 1 can be filled in with the value 1. Take the example of row 1. At column 2, the value is the max of 0 (for a mismatch), 0 (for a vertical gap) or 1 (horizontal gap). The rest of row 1 can be filled out similarly until we get to column 8. At this point, there is a G in both sequences

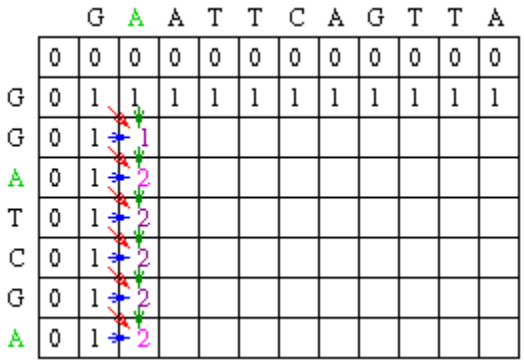
(light blue). Thus, the value for the cell at row 1 column 8 is the maximum of 1 (for a match), 0 (for a vertical gap) or 1 (horizontal gap). The value will again be 1. The rest of row 1 and column 1 can be filled with 1 using the above reasoning.



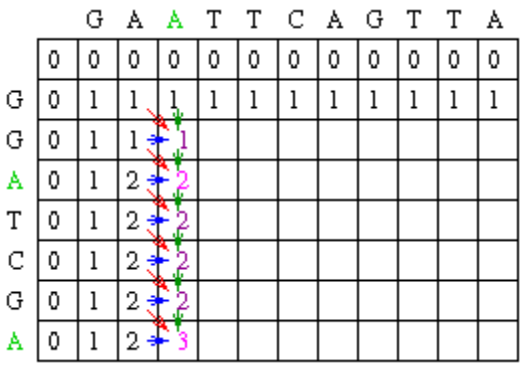
Now let's look at column 2. The location at row 2 will be assigned the value of the maximum of 1(mismatch), 1(horizontal gap) or 1 (vertical gap). So its value is 1.

At the position column 2 row 3, there is an A in both sequences. Thus, its value will be the maximum of 2(match), 1 (horizontal gap), 1 (vertical gap) so its value is 2.

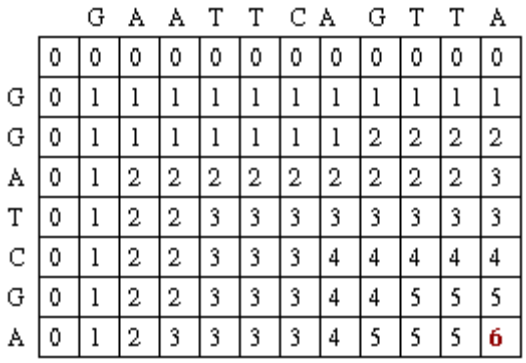
Moving along to position colum 2 row 4, its value will be the maximum of 1 (mismatch), 1 (horizontal gap), 2 (vertical gap) so its value is 2. Note that for all of the remaining positions except the last one in column 2, the choices for the value will be the exact same as in row 4 since there are no matches. The final row will contain the value 2 since it is the maximum of 2 (match), 1 (horizontal gap) and 2(vertical gap).



Using the same techniques as described for column 2, we can fill in column 3.



After filling in all of the values the score matrix is as follows:



Traceback Step

After the matrix fill step, the maximum alignment score for the two test sequences is 6. The traceback step determines the actual alignment(s) that result in the maximum score. Note that with a simple scoring algorithm such as one that is used here, there are likely to be multiple maximal alignments.

The traceback step begins in the M,J position in the matrix, i.e. the position that leads to the maximal score. In this case, there is a 6 in that location.

Traceback takes the current cell and looks to the neighbor cells that could be direct predacessors. This means it looks to the neighbor to the left (gap in sequence #2), the diagonal neighbor (match/mismatch), and the neighbor above it (gap in sequence #1). The algorithm for traceback chooses as the next cell in the sequence one of the possible predacessors. In this case, the neighbors are marked in red. They are all also equal to 5.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1	2	2	2	2
A	0	1	1	2	2	2	2	2	2	2	2	3
T	0	1	2	2	3	3	3	3	3	3	3	3
C	0	1	2	2	3	3	4	4	4	4	4	4
G	0	1	2	2	3	3	4	4	5	5	5	5
A	0	1	2	3	3	3	4	5	5	5	5	6

Since the current cell has a value of 6 and the scores are 1 for a match and 0 for anything else, the only possible predacessor is the diagonal match/mismatch neighbor. If more than one possible predacessor exists, any can be chosen. This gives us a current alignment of

```
(Seq #1)       A
                |
(Seq #2)       A
```

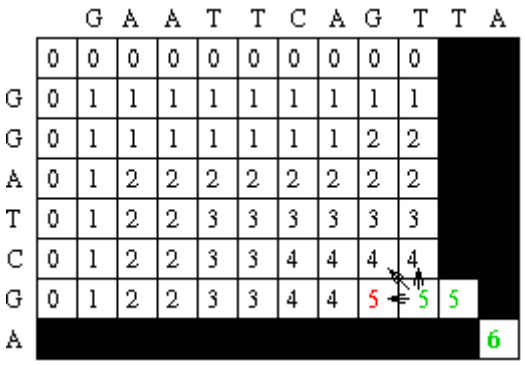
So now we look at the current cell and determine which cell is its direct predacessor. In this case, it is the cell with the red 5.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	
G	0	1	1	1	1	1	1	1	1	1	1	
G	0	1	1	1	1	1	1	1	2	2	2	
A	0	1	2	2	2	2	2	2	2	2	2	
T	0	1	2	2	3	3	3	3	3	3	3	
C	0	1	2	2	3	3	4	4	4	4	4	
G	0	1	2	2	3	3	4	4	5	5	5	
A												6

The alignment as described in the above step adds a gap to sequence #2, so the current alignment is

```
(Seq #1)       T A
                  |
(Seq #2)       _ A
```

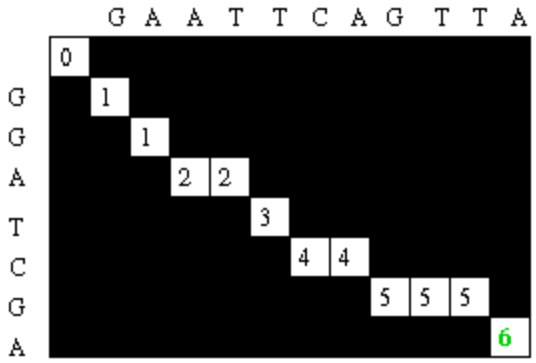
Once again, the direct predacessor produces a gap in sequence #2.



After this step, the current alignment is

```
(Seq #1)      T T A
                  |
              _ _ A
```

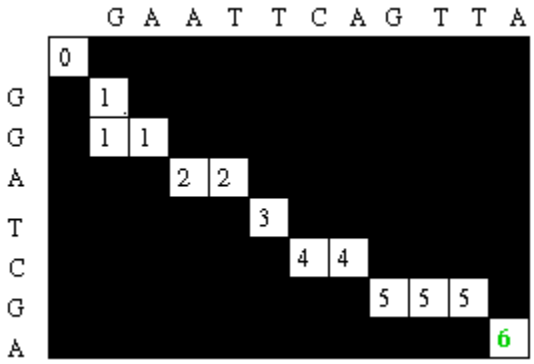
Continuing on with the traceback step, we eventually get to a position in column 0 row 0 which tells us that traceback is completed. One possible maximum alignment is :



Giving an alignment of :

```
G A A T T C A G T T A
|   |   | |   |     |
G G A _ T C _ G _ _ A
```

An alternate solution is:



Giving an alignment of :

```
G _ A A T T C A G T T A
|     |   | |   |     |
G G   A   T C   G     A
```

There are more alternative solutions each resulting in a maximal global alignment score of 6. Since this is an exponential problem, most dynamic programming algorithms will only print out a single solution.

Dynamic Programming for Local Alignment

$$M_{x,y} = \text{MAXIMUM}[$$
$$M_{x-1,\,y-1} + S_{x,y} \text{ (match/mismatch in the diagonal)},$$
$$M_{x,y-1} + w \text{ (gap in sequence \#1)},$$
$$M_{x-1,y} + w \text{ (gap in sequence \#2)},$$
$$0 \quad]$$