CHAPTER III

DIALIGN ALGORITHM

The sequence alignment problem is an optimization problem which goal is to find the best alignment. Thus the very most principal subject in sequence alignment is how to characterize a good alignment. When aligning nucleic acid sequences, biologists are able to assess the quality of alignments from experience and knowledge but computer program need a mathematically defined objective function or scoring scheme by which alignments can be evaluated. Finding a biologically relevant objective function turns out to be another problem of sequence alignment.

DIALIGN was proposed to introduce a new scoring scheme for only locally related sequences. Instead of comparing residue to residue, DIALIGN finds the optimal alignment by comparing pair of segment or diagonal. Here a consistent set of diagonals with maximum score is termed maximum alignment or optimal alignment where the score itself depends on a value which is gained from diagonal weight. The diagonal weight is based on a probabilistic model that represents match probability of diagonals. In this chapter we will discuss the detail of DIALIGN algorithm in producing pairwise sequence alignment starts from how to construct the diagonal and and its consistency issue.

3.1 DIAGONAL CONSTRUCTION

The DIALIGN algorithm is based on the comparison of all segments of the sequences under consideration. It considers all possible gapless aligned segments, called diagonals, calculates a score for each, and a weight to each according to the probability of the number of matches of the corresponding diagonal selected from two sequences. The final alignment is a consistent combination of diagonals which will maximize the total weight that will be discussed later.

Since the algorithm focuses on segment -to-segment of pairwise alignment problem, the first thing to do by the algorithm is constructing diagonals from a pair of sequence. Given two sequences :

$X = X_1 X_2 \ldots X_i \ldots X_{L1}$

$Y = Y_1 Y_2 \ldots Y_j \ldots Y_{L2}$

there are as many $O(L^3)$ diagonals where L is the maximum length of both sequences [Mor 98].

We will find all possible diagonals which is considered by the positions (i,j) where $1 \le i \le L_1$ and $1 \le j \le L_2$. For every pair of positions (i,j) there will be as many as $\min(i,j)$ diagonals. Since we find all diagonals by going backward, we require an index k where $k \ge 0$ and $k \le \min(i\text{-}1, j\text{-}1)$ such that it will form the diagonal $D_k = (X_{i-k}, Y_{j-k}; \ldots; X_i, Y_j)$ which consists of residue at position $(i\text{-}k, j\text{-}k)$ up to residue at position (i,j). In example 3.1 we will illustrate the diagonal construction from a pair of sequence.

Example 3.1

Let,

X = C T G

Y = C G

From sequences above we know that the length of sequence X is three and the length of sequence Y is two, so L_1=3 and L_2=2. We will construct diagonals for all position (i,j) where $1 \le i \le 3$ and $1 \le j \le 2$ such that there will be position (1,1), (1,2), (2,1), (2,2), (3,1), and (3,2). Each position may have more than one diagonal, depends on the number of k. For example, we will construct the diagonals from position (3,2).

As mentioned before that for every pair of positions (i,j) we will construct diagonals D_k with $k \le \min(i-1, j-1)$ and $k \ge 0$. At position (3,2) with i=3 and j=2 there will be min(3,2) = 2 diagonals such that at this position there are two diagonals, D_0 and D_1, where for k=0 forms diagonal $D_0=(X_{3-0}, Y_{2-0})$=(G,G) (see Figure 3.1.a) and for k=1 forms diagonal $D_1=(X_{3-1}, Y_{2-1}; X_3, Y_2)$=(T,C; G,G) (see Figure 3.1.b). Table 3.1 shows all possible diagonals for every pair of positions.

G	T G
G	C G
(a)	(b)

Figure 3.1 Diagonals at position (3,2). (a) D_0 and (b) D_1

Table 3.1 Possible diagonals for every position

i \ j	1	2
1	C C	C G
2	T C	T G
		C T C G
3	G C	G G
		T G C G

A maximum alignment is a consistent set of diagonals with maximum score. To identify which diagonals would be included in maximum alignment, we have to weight those diagonals. A weight function is used to give a positive weight score $w(D)$ to every possible diagonal D. Next section covers further explication about this.

3.2 DIAGONAL WEIGHT

A measure that enables us to assess the significance of a diagonal is to select a suitable set of diagonals. This measurement can be seen from match probability of diagonals that lead to diagonal weight. More precisely, an

alignment produced by DIALIGN is composed of equal length segment pairs (diagonals) exhibiting some statistically significant similarity.

Let *D* be a fixed diagonal, *l* be the length of *D*, and *m* be the number of matches contained in *D*. *P*(*l,m*) denotes the probability of a diagonal with length *l* contains at least *m* matches,

$$P(l,m) = \sum_{i=m}^{l} \left(\frac{l}{i} \right) p^i \ (1-p)^{l-i} \tag{3.1}$$

where *p* is the probability of a pair of residue to be a match. We assume *p*=0.25 for DNA sequences according to the number of DNA nucleotides. There are four nucleotides in nucleic acid sequence, namely adenine (A), cytosine (C), guanine (G), and thymine (T). The probability of a pair of DNA nucleotides to be a match is $^1/_4$.

Defined the weight of a diagonal *D* by

$$w(D) = \begin{cases} E(l,m) & \text{if } E(l,m) > T \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where

$$E(l,m) = -\ln P(l,m) \tag{3.3}$$

Intuitively, this function is useful to give a significant positive weight to a diagonal which contains many matches. And *T* is a user-defined threshold to restore diagonals that are likely to be biologically meaningful. Short DNA sequence does not need a threshold so we set *T*=0 [MDW 96]. In this case, DIALIGN algorithm is independent of user-defined parameters. Example 3.2 will give a way of calculating match probabilities and diagonal weights.

Example 3.2

Using sequences in example 3.1 we will show how to calculate weight of diagonals at position (3,2). As given in Figure 3.1, for position (3,2) we have two diagonals, D_0 = (G,G) and D_1 = (T,C; G,G).

As we see, Diagonal D_0 = (G,G) has length of 1 and 1 match. Substituting l=1 and m=1 to the formula (3.1) we obtain match probability of D_0 as

$$P(1,1) = \binom{1}{1}(0.25)^1(0.75)^0 = 0.25.$$

For the reason that sequences in example 3.1 is very short, we set T=0. Thus, weight of D_0 is $w(D_0) = E(1,1) = -\ln P(1,1) = 1.386$.

Table 3.2 Diagonal weights of each position

i \ j	1		2	
	diagonal	weight	diagonal	weight
1	C C	1.386	C G	0.288
2	T C	0.288	T G	0.288
			C T C G	0.827
3	G C	0.288	G G	1.386
			T G C G	0.827

And diagonal D_1=(T,C; G,G) has length of 2 and 1 match. Substituting l=2 and m=1 to the formula (3.1) we obtain match probability of D_1 as

$$P(2,1) = \binom{2}{1}(0.25)^1(0.75)^1 + \binom{2}{2}(0.25)^2(0.75)^0 = 0.4375.$$

Since $E(l,m)$ obtains positive weight value, using T=0, we get w(D) = $E(l,m)$. Thus, weight of D_1 is w(D_1) = E(2,1) = -ln P(2,1) = 0.827. Complete weight of possible diagonals of the sequences in example 3.1 is provided in Table 3.2.

The overall score of an alignment is then defined as the sum of weights of diagonals. How to compute a maximum alignment is presented in next section.

3.3 ALIGNMENT SCORE

In DIALIGN, alignments are composed from equal length segment pairs (diagonals), and the score of an alignment is defined as the sum of weights of those diagonals. To help compute the alignment score we use a value which is denoted by $\sigma(D)$. The $\sigma(D)$ is gained by adding the score of maximum alignment before diagonal D and the weight of this diagonal D which satisfies the relation

$$\sigma(D) = score(i-k-1, j-k-1) + w(D) \qquad (3.4)$$

Next, at each position (i,j) we compute its score. Score (i,j) means the score of a maximum alignment of the prefixes ($X_1, ..., X_i$) and ($Y_1, ..., Y_j$).

Thus, the score (*i,j*) depends on the score (*i,j*-1) and score (*i*-1,*j*). Recursively, the value of score at position (*i,j*) can be computed as

$$score(i, j) = \max\{score(i, j-1), score(i-1, j), \sigma(D_{i,j})\} \quad (3.5)$$

where $D_{i,j}$ is the longest diagonal ending at the point (*i,j*) that satisfies

$$\sigma(D_{i,j}) = \max\{\sigma(D) \mid \mathrm{D\ ends\ at\ the\ point}\ (i, j)\} \quad (3.6)$$

In other word, $D_{i,j}$ is diagonal which has the largest weight at position (*i,j*). This $D_{i,j}$ might be included in alignment. As an initialization of this algorithm, we define score (0,0) = score (*i*,0) = score(0,*j*) = 0, where $1 \le i \le L_1$ and $1 \le j \le L_2$ for the reason that at position i=0 or j=0 can not be formed a diagonal.

Using sequences in example 3.1 we will demonstrate how to compute score at position (*i,j*) with $1 \le i \le 3$ and $1 \le j \le 2$. Example 3.3 gives an illustration of computing score (1,1) up to score (2,2).

Example 3.3

As explained before, we define score (0,0) = score (1,0) = score(2,0) = score (3,0) = score(0,1) = score(0,2) = 0. In table 3.1 given all possible diagonals at every position. It can be seen that at position (1,1) we have only one diagonal, it is D_0 = (C,C). Also, in table 3.2 given the weight of diagonal D_0 at position (1,1). Having all of these, we now compute score (1,1) which depends on the score (1,0) and score (0,1). Applying equation (3.4), diagonal D_0 = (C,C) has a value of σ(*D*) as

$$\sigma(D) = score(2,1) + w(D) = 1.386 + 1.386 = 2.772$$

and σ($D_{1,1}$) as

$$\sigma(D_{1,1}) = \max\{\sigma(D) \mid D \text{ ends at the point } (1,1)\} = \max\{1.386\} = 1.386 .$$

It means that D_0 is the longest diagonal ending at point (1,1). Next, the score (1,1) is gained by substituting score (1,0), score (0,1), and σ($D_{1,1}$) to the equation (3.5) which obtains

$$score(1,1) = \max\{score(1,0), score(0,1), \sigma(D_{1,1})\} = \max\{0,0,1.386\} = 1.386 .$$

From table 3.1, it can be seen that at position (1,2) we have only one diagonal, it is D_0 = (C,G). Also, from table 3.2 given the weight of diagonal D_0 at position (1,2). Having all of these, we now compute score (1,2) by applying equation (3.4) first. Diagonal D_0 = (C,G) has a value of σ(D) as

$$\sigma(D) = score(0,1) + w(D) = 0 + 0.288 = 0.288$$

and σ($D_{1,2}$) as

$$\sigma(D_{1,2}) = \max\{\sigma(D) \mid D \text{ ends at the point } (1,2)\} = \max\{0.288\} = 0.288 .$$

It means that D_0 = (C,G) is the longest diagonal ending at point (1,2). Next, the score (1,2) is gained by substituting score (1,1), score (0,2), and σ($D_{1,2}$) to the equation (3.5) which obtains

$$\text{score } (1,2) = \max\{\text{score}(1,1), \text{score}(0,2), \sigma(\mathrm{D}_{1,2})\} = \max\{1.386, 0, 0.288\} = 1.386 .$$

At position (2,1), from table 3.1, we have one diagonal only, it is D_0=(T,C). And from table 3.2 given the weight of diagonal D_0 at position (2,1). Having all of these, we can compute score (2,1) by applying equation (3.4) first. Diagonal D_0 = (C,G) has a value of σ(D) as

$$\sigma(D) = score(1,0) + w(D) = 0 + 0.288 = 0.288$$

and σ($D_{2,1}$) as

$$\sigma(D_{2,1}) = \max\{\sigma(D) \mid D \text{ ends at the point } (2,1)\} = \max\{0.288\} = 0.288.$$

It means that D_0 = (T,C) is the longest diagonal ending at point (2,1). Next, the score (2,1) is gained by substituting score (1,1), score (2,0), and σ($D_{2,1}$) to the equation (3.5) which obtains

$$score(2,1) = \max\{score(2,0), score(1,1), \sigma(D_{2,1})\} = \max\{0, 1.386, 0.288\} = 1.386.$$

At position (2,2), from table 3.1, we have two diagonals, D_0=(T,G) and D_1=(C,C; T,G). And from table 3.2 given the weight of diagonal D_0 and D_1 at position (2,2). Having all of these, we can compute score (2,2) by applying equation (3.4) first. Diagonal D_0 = (T,G) has a value of σ(D_0) as

$$\sigma(D) = score(1,1) + w(D) = 1.386 + 0.288 = 1.674$$

and diagonal D_1=(C,C; T,G) has a value of σ(D_1) as

$$\sigma(D) = score(0,0) + w(D) = 0 + 0.827 = 0.827.$$

From σ(D_0) and σ(D_1) we obtain the value of $D_{3,2}$,

$$\sigma(D_{2,2}) = \max\{\sigma(D) \mid D \text{ ends at the point } (2,2)\} = \max\{1.674, 0.827\} = 1.674.$$

It means that D_0 = (T,G) is the diagonal ending at point (2,2) which has the largest weight for the reason that σ($D_{2,2}$) = σ(D_0). Next, the score (2,2) is gained by substituting score (2,1), score (1,2), and σ($D_{2,2}$) to the equation (3.5) which obtains

$$score(2,2) = \max\{score(2,1), score(1,2), \sigma(D_{2,2})\} = \max\{1.386, 1.386, 1.674\} = 1.674.$$

Table 3.3 summarizes all scores for every positions (*i,j*).

Table 3.3 Scores at each position

i \ j	1			2		
	diagonal	σ (D)		diagonal	σ (D)	
1	C C	1.386	$\sigma(D_{1,1})$=1.386 score=1.386	C G	0.288	$\sigma(D_{1,2})$=0.288 score=1.386
2	T C	0.288	$\sigma(D_{2,1})$=0.288 score=1.386	T G	1.674	$\sigma(D_{2,2})$=1.674 score=1.674
				C T C G	0.827	
3	G C	0.288	$\sigma(D_{3,1})$=0.288 score=1.386	G G	2.772	$\sigma(D_{3,2})$=2.772 score=2.772
				T G C G	0.827	

Having the maximum score (3,2) from table 3.3, now the problem is tracing a path of diagonal that has a contribution in producing the maximum score. When sequences are long, there might be many equally optimal alignments so we have to trace all of them. Instead, only one path is followed, leading to the result of only one of the many potential optimal alignments. This process is known as backtracking. Next section will describe a backtracking process to find an alignment path.

3.4 BACKTRACKING PROCESS

As discussed previously in equation (3.5), the formula of score (*i,j*) consists of three components and based on those components we will trace the paths which produce the optimal alignment. Thus, we have three possibilities, score (*i,j*) is equal to score (*i,j*-1), equal to score (*i*-1,*j*) or equal to $\sigma(D_{i,j})$.

We trace back starts from the end point (L_1,L_2). There are three cases:

1. We check whether the score (L_1,L_2) is equal to score (L_1,L_2-1). If they are equal then we trace back from position (L_1,L_2-1) and do the same action as before.
2. We check whether the score (L_1,L_2) is equal to score (L_1-1,L_2). If they are equal then we trace back from position (L_1-1,L_2) and do the same action as before.
3. We check again whether the score (L_1,L_2) is equal to $\sigma(D_{L1,L2})$. If they are equal then the diagonal at position (L_1,L_2) whose $\sigma(D)$ is maximum will be included in alignment. After taking that diagonal, again we starts trace back from position (*i*-*k*-1,*j*-*k*-1).

These cases are represented in equation (3.10).

Mathematically, the process of backtracking in DIALIGN refers to this formula

$$D_1 = prec(L_1, L_2) \tag{3.7}$$

$$D_{q+1} = \pi\left(D_q\right) \text{ as long as } \pi\left(D_q\right) \text{ is defined} \tag{3.8}$$

where *q* is the number of iteration during backtracking and $\pi(D)$ denotes the diagonal preceding *D*, written as

$$\pi(D) = prec(i-k-1, j-k-1) \tag{3.9}$$

while *prec* defined by

$$prec(i,j) = \begin{cases} prec(i,j\text{-}1) & \text{if } score(i,j) = score(i,j\text{-}1) \\ prec(i\text{-}1,j) & \text{if } score(i,j\text{-}1) < score(i,j) = score(i\text{-}1,j) \\ D_{i,j} & \text{if } score(i,j\text{-}1),\ score(i\text{-}1,j) < score(i,j) = \sigma(D_{i,j}) \end{cases} \tag{3.10}$$

In backtracking process, it might appear many different paths which will be resulting different alignments. In example 3.4 we will trace an alignment path using backtrack formula.

Example 3.4

We use information from Table 3.3. We start at point (L_1,L_2) where L_1=3 and L_2=2. Since score (3,2) is not equal to score (3,1) and score (2,2) but equal to $\sigma(D_{3,2})$, according to case (3) we have

$$D_1 = prec\left(L_1, L_2\right) = prec\left(3,2\right) = D_{3,2}.$$

The diagonal ending at point (3,2) which has maximum $\sigma(D)$ is D_0 = (G,G), so we include D_0 in alignment. From here, we obtain diagonal:

G

G

Because diagonal (G,G) is obtained from case (3) then we move to position (*i*-*k*-1,*j*-*k*-1) or (3-0-1,2-0-1)=(2,1). Since score (2,1) is not equal to score (2,0) but equal to score (1,1), according to case (2) we move to position (1,1). We check again, score (1,1) is not equal to score (1,0) and score (0,1) but equal to $\sigma(D_{1,1})$ then we have

$$D_2 = \pi(D_1) = \pi(D_{3,2}) = prec(2,1) = prec(1,1) = D_{1,1}.$$

The diagonal ending at point (1,1) is D_0 = (C,C) only, so we include this diagonal in alignment. We obtain the second diagonal and we put it in front of the first diagonal:

C G

C G

Because $D_3 = \pi(D_2) = \pi(D_{1,1}) = prec(0,0) = undefined$ then backtracking process stops.

In example 3.1 we have sequence *X* is C T G and sequence *Y* is C G. Since there is no diagonal which contains X_2=T in alignment, X_2=T is aligned to a gap. So, the optimal alignment is

C T G

C – G

Table 3.4 provides a complete information about the process of backtracking including the alignment path.

Table 3.4 Alignment path

i \ j	1			2		
	D	σ (D)		D	σ (D)	
1	C C	1.386	$\sigma(D_{1,1})$=1.386 score=1.386	C G	0.288	$\sigma(D_{1,2})$=0.288 score=1.386
2	T C	0.288	$\sigma(D_{2,1})$=0.288 score=1.386	T G	1.674	$\sigma(D_{2,2})$=1.674 score=1.674
				C T C G	0.827	
3	G C	0.288	$\sigma(D_{3,1})$=0.288 score=1.386	G G	2.772	$\sigma(D_{3,2})$=2.772 score=2.772
				T G C G	0.827	

The resulting path that we have just produced composes diagonals to be the best alignment. But does this optimal alignment satisfy the consistency criterion?

As mentioned before that maximum alignment is a consistent set of diagonals with maximum score. In example 3.4 we have diagonals with maximum score, now we will check whether those diagonals are consistent.

From the previous example we obtain the optimal alignment

C T G

C -- G

which is composed of diagonal C and diagonal G

C G

We will show whether diagonal (C,C) is consistent with diagonal (G,G).

Since the index of position of diagonal (C,C) = (X_1, Y_1) precedes the index of position of diagonal (G,G) = (X_3, Y_2) so we believe that diagonal (C,C) is consistent with diagonal (G,G). Thus, those diagonal satisfy the consistency criterion. Hence, it is true that the optimal sequence alignment produced by DIALIGN is consistent collections of diagonals *D*.

Running the algorithm of DIALIGN on long sequences is not easy because it takes much time. That is why in this skripsi we implement the algorithm only for short sequences. Implementation and simulation of the program is given in chapter four.