9

Superposition and Dynamic Programming

Most methods for comparing structures include some use of superposition and dynamic programming.

9.1 Superposition

Superposition can be used to find and score equivalences, by measuring how close the equivalent pairs can come together. One way of thinking of it is to put the structures on top of each other so that the equivalenced elements from the two structures lie as close as possible. If the geometry of the structures is not changed in this process, it is referred to as *rigid-body* superposition. The score can then be a function of the distances between the elements of each equivalent pair in the equivalence. Commonly, the root of the mean of the squares of the distances is used, and is called the *root mean square deviation* (RMSD). Low RMSD values are best, zero indicates exact equality.

Note that superposition can be used to measure (score) equivalences, not necessarily alignments directly. Two different measures are mainly used.

9.1.1 Coordinate RMSD

Superposition can be done by a *transformation* of structure A over B such that the equivalent pairs come as close as possible.

Let $(\alpha_1, \beta_1), \ldots, (\alpha_r, \beta_r)$ be the coordinate sets of the equivalenced elements of the equivalence E (α_i from A and β_i from B, for three dimensions a coordinate set consisting of three values). The problem is then to find a transformation T for A which minimizes the *coordinate* root mean square deviation, that is,

$$\mathrm{RMSD}_{\mathrm{C}}(E) = \min_{T} \sqrt{\frac{1}{\sum_{i=1}^{r} w_i} \sum_{i=1}^{r} w_i (T\alpha_i - \beta_i)^2}, \qquad (9.1)$$

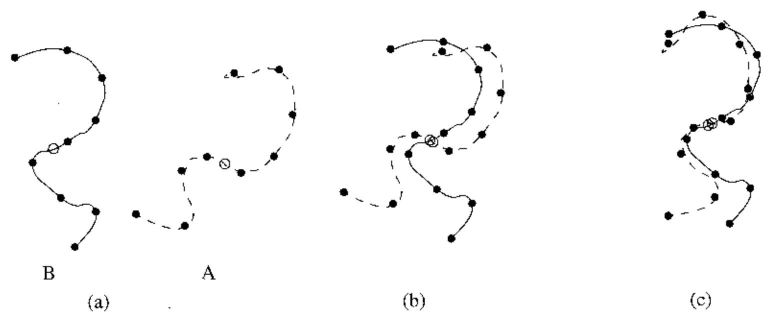


Figure 9.1 Figure illustrating the superposition in two dimensions. (a) The structures; o shows the centroids. (b) A moved so the centroids coincide. (c) A rotated to the minimum RMSD.

where w_i are weights corresponding to each pair (α_i, β_i) (and often set to 1). For residue level structure descriptions, there is usually one coordinate set per residue (e.g. C_α atom).

A transformation can be performed as a *translation* (three distances), and a *rotation* (three angles, around each of the x-, y- and z-axes). (The rotation can also be performed in one operation around a line, the direction of the line has to be calculated for each rotation; cf. Euler's theorem (Gelbert et al. 1977).)

It has been shown that a transformation for the minimum RMSD can be found by first shifting the centroids (geometrical centres) of each structure to the origin of a common coordinate system, and then finding the rotation of A which minimizes the $\mathrm{RMSD_C}$, as shown in Figure 9.1.

A rotation around the origin can be described by an *orthogonal* matrix $R_{3,3}$ (3D space). (There exist equations describing the connections between the angles (3) and the values of the matrix (9) (Gelbert et al. 1977, pp. 530–535).) A matrix is orthogonal if the scalar product of any two different columns is 0, and the result of taking the scalar product of any column with itself is 1. The matrix must be orthogonal to assure that the distances between the points of the same structure are not changed (cf. rigid-body superposition).

The formula can therefore be described by a rotation matrix R and a translation vector t, and we search for a pair (R, t) which minimizes the expression (assuming $w_i = 1$ for all i):

$$\sum_{i=1}^{r}(R\alpha_i + t - \beta_i)^2. \tag{9.2}$$

Example

Let the matrix R be

$$\begin{array}{ccc} C_1 & C_2 & C_3 \\ \left\{\begin{matrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{matrix}\right. & & \left.\right\} \end{array}$$

We can show that R is orthogonal, for example,

$$C_1 \cdot C_2 = \frac{1}{\sqrt{3}}\frac{1}{\sqrt{3}} + \frac{1}{\sqrt{2}}0 + \frac{1}{\sqrt{6}}(-\frac{2}{\sqrt{6}}) = 0$$

and

$$C_1 \cdot C_1 = \frac{1}{\sqrt{3}}\frac{1}{\sqrt{3}} + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\frac{1}{\sqrt{6}} = 1.$$

A point $(1, -1, 1)$ will then be transformed as

$$\left\{\begin{matrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{matrix}\right\} \left\{\begin{matrix} 1 \\ -1 \\ 1 \end{matrix}\right\}$$

to $(\frac{1}{\sqrt{3}}, 0, \frac{2\sqrt{2}}{\sqrt{3}})$. △

The superposition problem can therefore be formulated as finding the orthogonal matrix $R_{3,3}$ which minimizes the function

$$\sum_{i=1}^{r}(R\alpha_i - \beta_i)^2, \tag{9.3}$$

where $(\alpha_1, \beta_1), \ldots, (\alpha_r, \beta_r)$ are now the coordinates after the structures are moved (translated) to a common origin. Algorithms exist for finding such a matrix that either use iterative or direct methods.

In Equation (9.1) a weight is specified. For example, one could let w_k be a measure of how similar the amino acids in the equivalent pair are, and/or how similar the *environments* around the residues of the kth pair are, the environment meaning the spatial position of the neighbouring residues.

9.1.2 Distance RMSD

The distance score method *Distance RMSD* ($\mathrm{RMSD_D}$) alleviates the need for finding a translation and rotation of one of the structures and is given by

$$\mathrm{RMSD_D}(E) = \frac{1}{r}\sqrt{\sum_{i=1}^{r}\sum_{j=1}^{r}(\delta_{ij}^A - \delta_{ij}^B)^2}, \tag{9.4}$$

where δ_{ij}^A is the spatial distance between the elements of A in pairs i and j of the equivalence. Since there is no need to calculate a transformation, it is a faster calculation. However, it has a (sometimes serious) weakness: it is invariant under reflection. This means that if structure B is the mirror image of structure A, then $\mathrm{RMSD_D}(A, B) = 0$ and $\mathrm{RMSD_D}(C, A) = \mathrm{RMSD_D}(C, B)$ for all structures C.

The two measures are experimentally shown to have a close to linear relation (when all weights are equal to 1) as $\text{RMSD}_D = \rho_1 \times \text{RMSD}_C + \rho_2$, where $\rho_1 \approx 0.75$ and ρ_2 is between 0 and 0.2 (Cohen and Sternberg 1980).

9.1.3 Using RMSD as scoring of structure similarities

The problem of pairwise structure comparison is often the problem of finding equivalences with low RMSD value(s). However, several quite different equivalences with similar scores might be found and which of these equivalences represent the 'correct' solution is not an easy task to decide. However, *one always needs to consider how many elements were equivalenced, since for random comparisons the expected RMSD value seems to be proportional to the square root of the number of equivalenced residues*. When taking this into consideration, different measures can be used for evaluating how well two structures can be superposed.

1. Find the equivalence that minimizes the RMSD divided by the square root of the length of the equivalence: $\min_E \text{RMSD}(E(A, B))/\sqrt{n_E}$, where n_E is the number of pairs in the equivalence E.
2. Define a threshold L. Find the maximum number of elements that can be superposed such that RMSD is less than or equal to L.
3. Define a threshold l. Find the maximum number of elements that can be superposed such that the distance between each equivalenced element is less than or equal to l.

The two last methods are mostly used to improve detection of regions of similar topology, excluding structurally unrelated regions.

Most scoring schemes for evaluating equivalences between structure descriptions contain factors related to the RMSD. Many structure comparison programs give as output an equivalence and a resulting RMSD even if they do not use RMSD internally to score equivalences. See Chapter 12 for more discussion of scoring structure comparison.

9.2 Alternating Superposition and Alignment

The methods using alternating superposition and alignment operate on residue level, and the goal is to find a 'best' alignment for the two structures. An initial equivalence (a *seed*) of atoms from each structure is first given, $E_0 = (a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}), \ldots, (a_{i_r}, b_{j_r})$. Note that the A superposition is then performed with respect to the equivalence E_0 using the transformation T_0 which minimizes an RMSD measure. Then the whole structures are superposed using T_0. The distances between *all* pairs of atoms (residues) from the two structures (after superposition), can then be used to define a new scoring matrix R_0, which is used to obtain an alignment A_0 by dynamic programming. Note that usually a scoring matrix gives highest values to 'similar' elements,

hence a score could be the reciprocal of the distance. Note also that R_0 is a *position-dependent scoring matrix*: there is a score for every pair of elements (residues) from A and B.

A new equivalence E_1 of the r equivalences from the alignment with least distances (using R_0) can then be found. From this T_1 is computed, and thereafter R_1, as explained above. This iteration is performed until convergence ($E_{i+1} = E_i$) or some maximum number of cycles is done. Algorithm 9.1 describes the method and Figure 9.7(a) illustrates the approach. Note that in an implementation it is not necessary to use a new variable for each E_p.

Algorithm 9.1. Alternating superposition and dynamic programming.

```
Comparing structures $A$ and $B$ by finding a 'best' equivalence for them.
const
$p_{\max}$         maximum number of cycles
$E_{\text{init}}$         initial equivalence
$r$            the number of pairs in the equivalences
var
$p$            cycle number
$R$            scoring matrix
$E_p$           the equivalence of cycle $p$
$T$            the (minimum) transformation for $E_p$
proc
dist($a_i, b_j$)  the distance between residues
score($d$)     calculate a score from a distance
begin
  $E_0 := E_{\text{init}}$; $p := 0$
  repeat
    $T :=$ the transformation for $\text{RMSD}_C(E_p)$
    $A^* := T(A)$          Superpose $A$ onto $B$ giving $A^*$
                       Calculate the new scoring matrix:
    forall pairs $(i, j)$ do $R_{ij} :=$ score(dist($a_i^*, b_j$)) end
    $(s, P) := \mathcal{A}_R(A, B)$ DP on $(A, B)$ using $R$ (find path $P$ with score $s$)
    $p := p + 1$
                       Pick the $r$ pairs from $P$ with lowest distances in $R$
    $E_p := \{(a_{i_1}, b_{j_1}), \ldots, (a_{i_r}, b_{j_r})\}$
  while $E_p \neq E_{p-1}$ and $p < p_{\max}$
end
```

Example

Let $r = 4$ and the initial equivalence be

$$E_0 = \{(a_1, b_2), (a_2, b_4), (a_5, b_7), (a_9, b_5)\}.$$

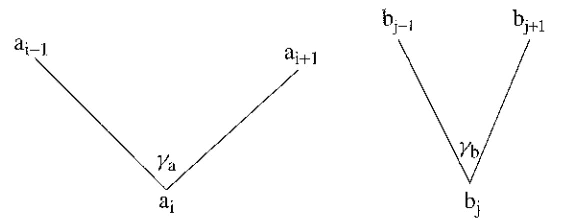


Figure 9.2 Example of a spatial component of a score of (a_i, b_j), the difference of the angles γ_a and γ_b (remember that three points define a plane).

The transformation for best superposition of (a_1, a_2, a_5, a_9) on (b_2, b_4, b_7, b_5) is found, and then used on the whole structure A. The mxn matrix R of *all* pairwise distances (after the superposition) is calculated, and used to define the scoring matrix for a dynamic programming algorithm. Let this result in the best alignment be

$a_1\ a_2\ a_3\ a_4 — — a_5\ a_6\ a_7\ a_8\ a_9$
$— b_1\ b_2\ b_3\ b_4\ b_5\ b_6 — b_7 — b_8$

An equivalence with six pairs is defined by this alignment, and taking the four (r) of them with smallest distances (using R_0) might result in the equivalence

$$E_1 = \{(a_3, b_2), (a_5, b_6), (a_7, b_7), (a_9, b_8)\}.$$

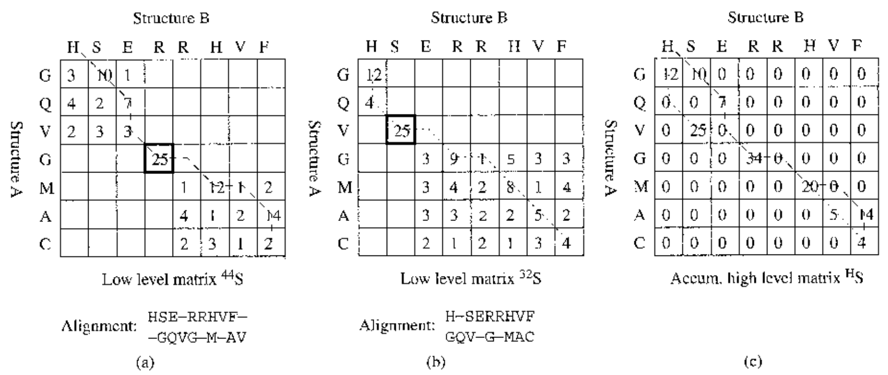
This is the equivalence used in the next cycle. △

The initial seed equivalence is critical to the final result, therefore several initial seeds should be tried. As explained in Section 9.1.3, the final results can be quite different while still giving similar quality measures (RMSD and/or number of aligned residues).

In the presentation we have used r as a constant, instead it could be, for example, the number of pairs $(a^*_{i_p}, b_{j_p})$ on P for which the distances R_{ij} are below a given limit ($a^*_{i_p}$ are the positions of the atoms of A after superposition). In this way only pairs which superpose well are used in the next cycle, hopefully giving faster convergence. However, there is no reason for saying that this is always for the best, so both options should be available.

The scoring of (a_i, b_j) for making the scoring matrix can be changed to include several components, e.g. a sequence component and a local structure component. The sequence component might reflect the similarity between the amino acid types of the two residues (e.g. using a PAM matrix) and the local structure component might reflect, for example, the spatial similarity between (a_{i-1}, a_i, a_{i+1}) and (b_{j-1}, b_j, b_{j+1}). The spatial similarity can, for example, be measured by the difference between two angles, as shown in Figure 9.2.

The method of performing alternating superposition and alignment is also used for refining (postprocessing) the results found by other methods, mostly methods using coarse level description.



Structure B
Structure A
Low level matrix ${}^{44}S$
Alignment: HSE–RRHVF– / –GQVG–M–AV
(a)
Low level matrix ${}^{32}S$
Alignment: H–SERRHVF / GQV–G–MAC
(b)
Accum. high level matrix ${}^{H}S$
(c)

Figure 9.3 Application of the double dynamic programming method. (a), (b) Two of the 56 low-level scoring matrices are shown. The expression used for low-level scores gives the value of 25 to ${}^{ij}S_{ij}$. The best (DP) paths using the scoring matrices and a linear gap penalty with $g = 4$ are shown by broken and dotted lines, respectively. (c) The low-level scores for the two optimal paths are summed up in the high-level scoring matrix.

9.3 Double Dynamic Programming

As explained earlier, traditional dynamic programming cannot alone be used for aligning structures. Since any choice to align two substructures will affect the scoring of the alignment of the complete structures, the independence requirement is violated and DP can no longer guarantee an optimal solution. But if one assumes that the structures are already superposed, reasonable scoring schemes can be devised which allow the structures to be aligned optimally using DP (Section 9.2). However, ideally, one might wish to *simultaneously* align and superpose the structures to optimize a score depending on how well-aligned substructures superpose. Here we describe one method using this approach, the structure sequence alignment program (SSAP) program (developed by Taylor and Orengo (1989)). This is based on a method called double dynamic programming (DDP). The main idea is to use two levels of dynamic programming, constructing a final scoring matrix that can be used on the high level for finding the (best) alignment between the two structures by an ordinary DP algorithm.

Conceptually, the method looks at each residue pair (a_i, b_j), and for each it tries to find how likely it is that this pair is in an optimal alignment. A heuristic for this is to find an optimal alignment under the constraint that (a_i, b_j) is part of the alignment, and to define a (low-level) *position-dependent scoring matrix* under this constraint. In the DDP context this is called a low-level DP, and the scoring matrix uses a *low-level scoring matrix*. For each pair (i, j) there will be defined a separate (low-level) scoring matrix, denoted ${}^{ij}S$. The matrix element ${}^{ij}S_{kl}$ will get a number (score) showing how well the residue a_k fits to b_l under the constraint that a_i is aligned with b_j. Note that these scoring matrices define scores between all pairs of *residues*, not between amino acids.

To force the (low-level) alignment to go through (a_i, b_j), the ordinary DP algorithm can then be used, either by finding an optimal path from (a_i, b_j) to (a_1, b_1) and from (a_i, b_j) to (a_m, b_n), or by giving the score of (a_i, b_j) so high a value that the optimal path is forced to go through it. Since we assume that a_i matches b_j, it is only necessary to calculate the scores of ${}^{ij}S_{kl}$ for $(1 \leqslant k < i,\ 1 \leqslant l < j)$ and $(i < k \leqslant m,\ j < l \leqslant n)$ (that is, the top left and bottom right of ${}^{ij}S_{ij}$).

The results from all low-level computations are 'summed up' in a *high-level scoring matrix* ${}^{H}S$, and final (high-level) dynamic programming is done using ${}^{H}S$. The summing is done by letting the contribution from the low-level matrix ${}^{ij}S$ be all ${}^{ij}S_{pq}$ such that (a_p, b_q) lies on the optimal (low-level) path when ${}^{ij}S$ is used as the scoring matrix. In this way the highest-scoring path from each low-level DP matrix is propagated to the high-level 'summary' scoring matrix. The procedure is illustrated in Figure 9.3, and given in Algorithm 9.2. Note that Figure 9.3 shows scoring matrices, not the 'DP matrices'.

Algorithm 9.2. Double dynamic programming.

```
Aligning the structures $A$ and $B$ using DDP
begin
  ${}^{H}S := \{0\}_{m\times n}$                 Set high-level scoring matrix to zero
  for each pair $(a_i, b_j)$ do
    Create the low-level scoring matrix ${}^{ij}S$
    $(s, P) := \mathbf{DP}^{*}_{{}^{ij}S}(A, B)$       Low-level DP, forced through $(a_i, b_j)$
    forall $(a_p, b_q) \in P$ do        Accumulate from path $P$
      ${}^{H}S_{pq}(P) :={}^{H} S_{pq}(P) +{}^{ij} S_{pq}(P)$
    end
  end
  $(s, P) := \mathbf{DP}_{{}^{H}S}(A, B)$            High-level DP using ${}^{H}S$, best path in $P$
end
```

As ${}^{H}S$ contains a sum of values from low-level matrices, it might contain large values, and before the high-level computation is performed, the values are normalized or their logarithmic values are used.

${}^{H}S$ is constructed as the sum of values from mn low-level matrices, most of them representing pairs which are not in the final alignment. There is a general background or random score associated with the comparison with every dissimilar pair, and to damp this noise it should be removed from the accumulation. One way is to define a cut-off on the score of the low-level alignment and accumulate the scores only from alignments above this cut-off.

Different versions of DDP can be developed, depending, for example, on how the low-level scoring matrices are calculated and which sequence and structure properties are used. Examples are given in the following sections.

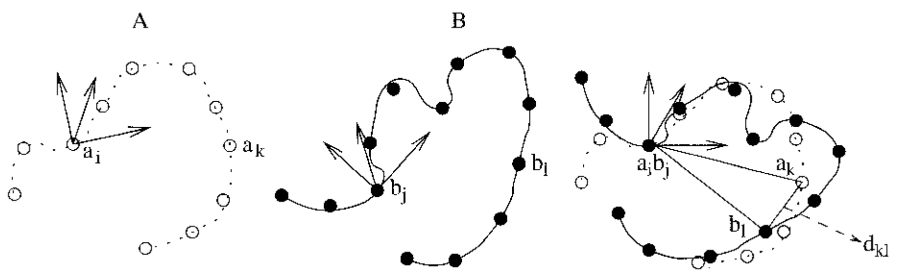


Figure 9.4 Calculation of a low-level scoring matrix ${}^{ij}S$ in double dynamic programming. Two structures A and B are shown schematically, and local coordinate frames are derived from local geometry at a_i and b_j, respectively. The structures are then translated and rotated such that the coordinate frames coincide (right). The scoring ${}^{ij}S_{kl}$ can then be found simply as a function of the distance between a_k and b_l shown as d_{kl} or as a more complex function.

9.3.1 Low-level scoring matrices

There are mn low-level scoring matrices ${}^{ij}S$. All methods for calculating ${}^{ij}S_{kl}$ (showing how good a_k fits b_l when a_i is aligned to b_j) should calculate a superposition of the two structures A and B bond on the choice of i and j. One way of defining the scores is to first define local reference systems at a_i and at b_j, e.g. by using (the C_α of) a_{i-1}, a_i, a_{i+1} and b_{j-1}, b_j, b_{j+1}. With three points one can define a unique reference (coordinate) system (as long as they do not lie on a straight line). The coordinates of the remaining residues are transformed into the respective coordinate systems. The score of aligning a_k, b_l depends on the distance between a_k and b_l in the respective coordinate systems defined at a_i and b_j, as shown in Figure 9.4. One simple scoring scheme is to use a function of the distance between a_k and b_l.

A more comprehensive scoring scheme could also take into account.

- A *direction* component, the difference in the direction of the vectors (a_i, a_k) and (b_j, b_l) (see Figure 9.4).

- An *orientation* component o_{kl}. A local reference frame can also be constructed for a_k and b_l (using the neighbouring residues, as above), after the structures have been transformed to coincident reference frames at a_i and b_j. This component reflects the difference in the reference frames at a_k and b_l, and is measured by the angle between corresponding axes at a_k and b_l.

- A *sequence* distance component g_{kl}, calculated as an increasing function of $|k - i| + |l - j|$. This component should damp the contribution from near neighbours in the sequence (as the matching of local secondary structures). The reason for this component is that the structural similarity at a_k and b_l tends to be higher when they are near a_i and b_j in sequence, and we want all pairs to have 'equal sequential significance' (having a global view).

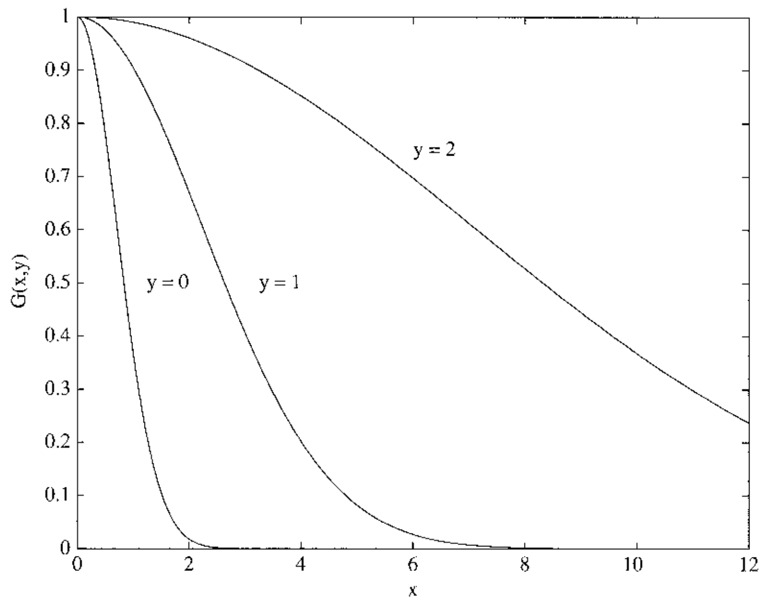


Figure 9.5 The Gaussian function for different values of x and y. We see that the decrease of $\mathcal{G}(x, y)$ is highest for low y.

- A *spatial* distance component h_{kl}, calculated as a decreasing function of $d_{ik} + d_{jl}$, where d_{ik} is the distance between the (the C_α of) a_i and a_k. This will damp the contribution from combined large distances in space.

The two last components together give high weight to residues which are near in space but not near in sequence (i.e. candidates for being in a site around $a_i b_j$).

Combining the components: Gaussian transform function

The low-level scoring matrices are constructed as a combination of several components. Often the score of a component should be the inverse of a measured value, and also some normalization should be done, e.g. in some cases where the measured value is a distance, and we want to use similarity. One such decreasing and normalizing function (all values being in the interval [0,1] for nonnegative x and y) is the Gaussian transform function:

$$\mathcal{G}(x, y) = \exp(-x^2/10^y), \tag{9.5}$$

where x is the measured value which is to be used in a scoring function and y is a (different) constant for each component, defining the slope of $\mathcal{G}$ (see Figure 9.5). The final scoring value ${}^{ij}S_{kl}$ is the product of all the components after transformations.

9.3.2 High-level scoring matrix

The accumulated sums in the high-level matrix could be large, and the logarithms of the sums are used as the scoring values. A linear gap penalty is used. The variation of the values of the scoring matrix is high, with a large difference between ‘good’ and ‘bad’ pairs. A typical alignment will not have gaps in the ‘good’ regions.

9.3.3 Iterated double dynamic programming

As DDP is a heuristic, one might achieve better results by iteration—doing a DDP in each cycle. This, together with an idea of how to limit the work for each DDP cycle, is developed into a program called the structure alignment program (SAP).

The DDP algorithm described above requires a computation time proportional to the fourth power of the sequence length (for two proteins of equal length) as it performs an alignment for all residue pairs. To circumvent this severe requirement, some simple heuristics are devised based on the principle that comparing the environment of all residue pairs is not necessary. Thus some pairs, together referred to as the *seed*, are chosen before the first cycle. Low-level matrices are only constructed for the pairs in the seed, and used for the DDP algorithm in the first cycle. In each cycle the high-level scoring matrix is updated, and the pairs are also selected anew for the next cycle. The overall algorithm is shown in Algorithm 9.3. Two high-level matrices are used: ${}^{H}S$ as before, and a *bias* matrix Q. The reason for this will be explained below. In the algorithm a high-level DP is performed in each cycle; this is for the termination criterion.

Algorithm 9.3. Iterated double dynamic programming.

```
Aligning the structures $A$ and $B$ by iterated DDP
var
$Q$     the high-level bias scoring matrix
$E$     the equivalence used in each cycle
begin
   initialize the bias matrix $Q$
   $E :=$ the seed
   repeat
      ${}^{H}S := \{0\}_{m\times n}$                 Set high-level scoring matrix to zero
                                     Calculate the high-level scoring matrix ${}^{H}S$:
      for each pair $(ij) \in E$ do
         $(s, P) := \mathbf{DP}^{*}_{ij\,S}(A, B)$   Low-level DP forced through $(a_i, b_j)$
         accumulate the low-level result in ${}^{H}S$
      end
      update $Q$ using old $Q$ and ${}^{H}S$
      $(s, P) := \mathbf{DP}_Q(A, B)$         High-level DP
      $E :=$ select new pairs based on $Q$
   until termination criterion is satisfied
end
```

Several of the statements in Algorithm 9.3 have to be described in more detail.

- How is Q initialized and the seed selected?
- How many residue pairs should be selected in each cycle, and how?
- How should Q be updated?
- Should the high-level scoring matrix contribute to the low-level matrices?
- What is the termination criterion?

Initialization

Based on local structure and environment, many residue pairs (indeed most) can be neglected. This selection can be made on secondary structure state (one would not normally want to compare an alpha-helix with a beta-strand and burial (those with a similar degree of burial are most similar) but a component based on the amino acid identity can also be used, giving any sequence similarity a chance to contribute. Contributions from all three components are combined for each pair of positions as a product (to ensure that all three have a reasonable value) giving a matrix Q, which is referred to as the '*bias matrix*' (Q is thus determined by use of the three components). No specific weights are introduced, instead the $\mathcal{G}$ transform is used to give a roughly equal contribution and taking their product makes the size of the component ranges less critical.

The seed can be selected by taking the pairs with highest values in Q, or, for example, using methods for discovering common sites in the structures, e.g. SPratt (see Chapter 13.5).

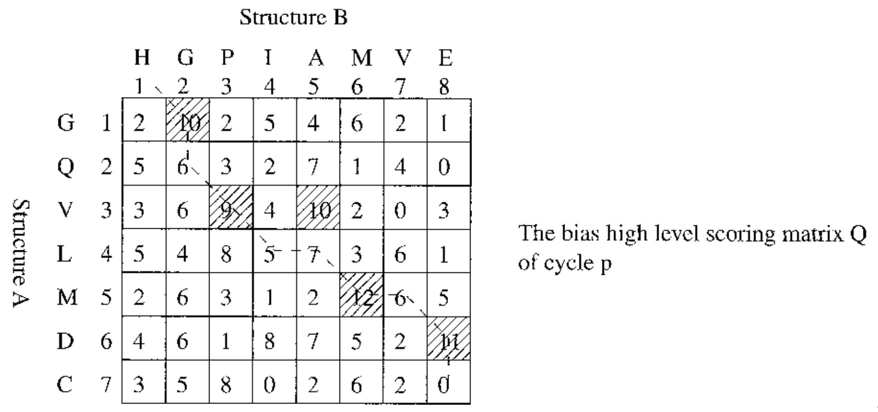
Selecting pairs and updating Q

The highest values in Q for each cycle are used for selecting the pairs to be used in the low-level computation. First a relatively small number of pairs is selected (10–20), but the number is gradually increased with each cycle, as hopefully more 'true pairs' are found. The initial sparse sampling can be unrepresentative of the truly equivalent pairs. To maintain a continuity through the early sparse cycles (hopefully towards the true equivalence) Q is used as a base for incremental revision (the *bias* matrix). If Q^p is the bias matrix on cycle p, then the next revision is calculated as

$$Q^{p+1} = Q^p/2 + \log(1 + {}^H S^{p+1}/20), \tag{9.6}$$

where ${}^H S$ is the matrix as defined for SSAP formed from the summed traces from all the low-level comparisons. The scores in the high-level matrix (${}^H S$) are generally large but the logarithmic damping reduces these into a range with an effective maximum of 1, which is more commensurate with the range of values found in the bias matrix (Q).

To further ensure that the bias matrix does not become dominated by extreme values, its elements are normalized on each cycle.



		H	G	P	I	A	M	V	E
		1	2	3	4	5	6	7	8
G	1	2	▨ 10	2	5	4	6	2	1
Q	2	5	6	3	2	7	1	4	0
V	3	3	6	▨ 9	4	▨ 10	2	0	3
L	4	5	4	8	5	7	3	6	1
M	5	2	6	3	1	2	▨ 12	6	5
D	6	4	6	1	8	7	5	2	▨ 11
C	7	3	5	8	0	2	6	2	0

Figure 9.6 Illustration of the selection procedure and stop criterion for the iterated DDP method. The figure shows the selected pairs in cycle p, when the number of pairs is five (shaded). It also shows the best alignment (by the broken line) for cycle $p + 1$, hence four of the five selected pairs are on this path ($r = 5$, $u = 4$). If all five lay on the alignment the iteration would have converged (and could be terminated).

This form of updating weakens the contribution from the initial Q for each iteration cycle, making the selection of pairs become increasingly determined by the dominant alignment, approaching (or attaining) by the final cycle a self-consistent state in which the alignment has been calculated predominantly (or completely) from pairs of residues that lie on the alignment.

The increasing number of pairs selected can be determined by the following function of the size of the two proteins (m and n) and the cycle number (p):

$$K = 10 + \frac{p+1}{20}\sqrt{mn} \tag{9.7}$$

($p = 0$ for the initial cycle). We see that the number of selected pairs can be larger than the length of the sequences, but this is taken care of in the stop criterion. See Figure 9.6 for an illustration of the selection.

Achieving coherence between the levels

The initial selection of residue pairs might be quite random with respect to the final set of 'true' equivalences and, as a consequence, the comparison of their environments might provide little coherent direction towards the final solution. Although the bias matrix provides a platform from which the selection of pairs can be refined, it has no effect on the scores derived from the low-level matrices. A contribution from the bias matrix as described above can therefore be introduced at this level to provide stability into the early cycles. This is done by adding a contribution from the bias matrix to the low-level matrix that decreases with increasing cycle number. Therefore, instead of using ${}^{ik}S$ as explained in Section 9.3.1, a revised matrix ${}^{ik}S^*$ is used:

$${}^{ik}S^* = {}^{ik}S + Q^p \cdot \mathcal{G}(p, 1), \tag{9.8}$$

were p is the cycle number and $\mathcal{G}$ is the Gaussian transform. On the initial cycle the bias matrix has a full contribution ($\mathcal{G}(0, 1) = 1$), which decreases until after the fifth cycle there is effectively no contribution from the bias matrix ($\mathcal{G}(5, 1) = 0.082$). This provides a smooth transition from the, initially, local information in the bias matrix into the full global view provided by the comparison of the residue environments.

Termination criterion

The iteration should stop when the selected pairs in cycle p coincide with the best (high-level) path in cycle $p + 1$. Let r be the number of selected pairs at cycle p. Then a new Q is calculated, and the best path found. Let u of the selected pairs lay on this path. Then

$$k = \frac{u}{r}$$

can be used as stop criterion, as illustrated in Figure 9.6. If k becomes 1, the iteration stops. However, if the number of selected pairs is more than the length of the true alignment, then $k = 1$ can never be attained so the iteration can also terminate if k stops to increase. In addition, an upper limit for the number of cycles should exist (typically 5–10).

9.4 Similarity of the Methods

SAP and the methods using alternating superposition and alignment can both be looked upon as *two-level* methods. While the latter methods find the DP matrix using an optimal superposition for the current equivalence, SAP does not need to decide (or assume) one exact alignment to calculate the higher-level scoring matrix. Instead the residues pairs that participate in high-scoring low-level alignments receive high values in the high-level scoring matrix and are likely to be included in the final alignment (see Figure 9.7(b)).

9.5 Exercises

1. Regard two 'structures' (A, B) of three atoms in 2D space, defining an equivalence of three pairs:

```
A: (1,4)(4,1)(4,4)
B: (0,0)(2,0)(3,2)
```

   (a) Calculate RMSD$_\text{D}$.

   (b) Calculate the geometrical centres of the 'structures'. Then move the 'structures' so that the geometrical centres are at the origin. Find the new coordinates of the points, and plot them in a diagram.

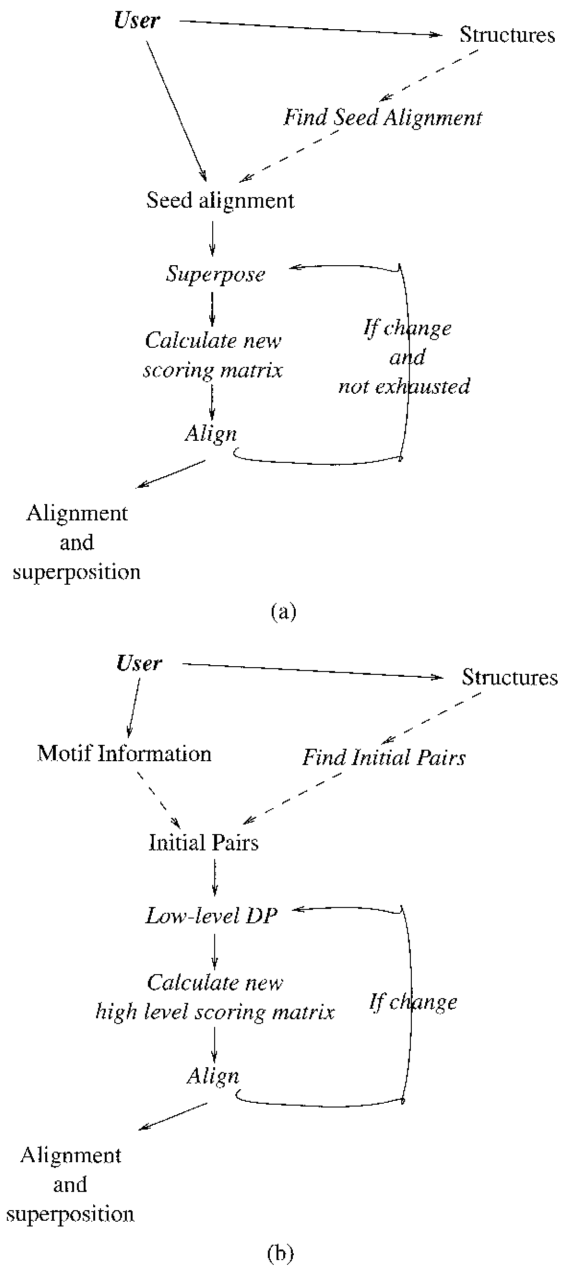


Figure 9.7 (a) Outline of algorithm alternating between alignment and superposition. (b) Outline of the SAP method. Reproduced from Eidhammer et al. (2000) by permission of Mary Ann Liebert.

(c) Calculate RMSD$_C$ (weight 1) for the 'structures' as they are, without rotation.

(d) Look at the 'structures'. Do you think you can find a lower RMSD$_C$ value by rotating one of the 'structures'?

2. This exercise is to illustrate the method of alternating superposition and alignment for finding the best alignment. For simplicity, we consider 'structures' in 2D space. Consider the structures

$$A = (a_1, a_2, a_3, a_4) \quad \text{and} \quad B = (b_1, b_2, b_3, b_4, b_5).$$

Let the position of (atoms representing) the residues be

a_1:	$(2, -0.414)$,	b_1:	$(1.2, -2)$,
a_2:	$(-0.121, 0.293)$,	b_2:	$(1.8, 1.2)$,
a_3:	$(1.293, 3.121)$,	b_3:	$(0.2, -0.4)$,
a_4:	$(4.828, 1)$,	b_4:	$(-1.4, -0.6)$,
		b_5:	$(-1.8, 1.8)$.

(a) Calculate the geometrical centres, and find the coordinates of the atoms when both are moved so that the geometrical centres are at the origin. (Control: the coordinates for a_1 should be $(0, -1.414)$.)

(b) Assume an initial equivalence (subalignment): $(a_2, b_1)(a_3, b_2)(a_4, b_4)$. Assume further that the rotation matrix for minimum RMSD$_C$ for this is

```
-1/1.414    1/1.414
 1/1.414    1/1.414
```

(i) Show that the matrix is orthogonal.

(ii) Rotate all the coordinates of A, and calculate the new coordinates. (Control: the new coordinates of a_1 should be $(-1, -1)$.)

(c) Define a distance matrix D for all pairs (a_i^*, b_j), where a_i^* is the coordinates after rotation.

(d) Define a scoring matrix R by dividing 1 by the distances in D; use one decimal. Use R to find the best alignment for A, B when R is used. Use a linear gap penalty, where the penalty for one blank is the average of the values in R which are less than 1.

(e) Now find the highest-scoring equivalence (subalignment) of length 3.

3. This exercise is to illustrate double dynamic programming. Consider the 'structures' $A = (a_1, a_2, a_3, a_4)$ and $B = (b_1, b_2, b_3, b_4, b_5)$ in 2D space. To define a coordinate system for a_i, choose origin in a_i and the x-axis through a_{i+1}. First choose the pair (a_2, b_3) for low-level dynamic programming. The coordinate in this system are found to be

```
A: (-1,2), (0,0), (2,0), (1,1)
B: (-2,1), (-1,1), (0,0), (1,0), (0,2)
```

(a) Fill in a distance matrix D for all pairs from A, B, and define a low-level scoring matrix ${}^{2,3}S$, by dividing 1 by the distances in D (to one decimal). Let the score for (a_2, b_3) be the double of the highest score. Use a linear gap penalty, and let the gap penalty for one blank be the average of the values in ${}^{2,3}S$, when (a_2, b_3) is not counted. Perform low-level dynamic programming.

(b) Choose (a_3, b_3) for the new low-level matrix. B will have the same coordinates as in (a). To find the new coordinates for (A), you can use the following procedure. Let

- X and Y be the x- and y-axes of the original coordinate system,
- X' and Y' be the x- and y-axes of the coordinate system with origin at a_3 (and x-axis along a_4),
- ϕ be the angle between X and X',
- $g_1 = a_3x$ (the x-coordinate of a_3),
- $g_2 = a_3y$ (the y-coordinate of a_3),
- $g_{11} = \cos(X, X') = \cos(\phi)$,
- $g_{12} = \cos(X, Y') = \cos(90 + \phi)$,
- $g_{21} = \cos(Y, X') = \cos(90 - \phi)$,
- $g_{22} = \cos(Y, Y') = \cos(\phi)$.

If (x, y) are the coordinates of a point in the original coordinate system, then the coordinates (x', y') in the new coordinate system become

- $x' = g_{11}(x - g_1) + g_{21}(y - g_2)$,
- $y' = g_{12}(x - g_1) + g_{22}(y - g_2)$.

You may want to draw a diagram with the coordinate systems for verifying the new coordinates you get. Then define a scoring matrix and perform dynamic programming as explained in (a).

(c) Make the high-level scoring matrix using the two low-level ones, and perform dynamic programming.

(d) Compare the three alignments, and comment the choice of pairs when only two are chosen.

4. When choosing the pairs for the next cycle in double dynamic programming, there is no test for inconsistencies among the pairs (one residue in A could be in two or more pairs). Discuss whether this should have been done.

5. Equation (9.6) shows how the bias matrix Q is updated. Now assume that a cell in the high-level scoring matrix ${}^HS$ in every cycle is calculated to a fixed value K, such that $K = \log(1 + {}^HS^{p+1}/20)$.

(a) Show that when p approaches infinity, the value of the corresponding cell of Q approaches $2K$. Hint: write the equation as $q^{p+1} = q^p/2 + K$ (where q^p is the value of the corresponding cell in cycle p). Then find an expression for q^{p+1} depending on K and q^0.

(b) Let $q^0 = 0.5$. Find the value of q^3 when (i) $K = 0.2$ and (ii) $K = 0.8$.

9.6 Bibliographic notes

Fundamental articles for superposition are McLachlan (1972, 1979), Kabsch (1978) and Cohen and Sternberg (1980).

More about the methods of alternating superposition and alignment can be found in Rao and Rossmann (1973), Rossmann and Argos (1975, 1976), Satow et al. (1986), Cohen (1997), Russel and Barton (1992), Ding et al. (1994), Holm and Sander (1995), Zu-Kang and Sippl (1996), Petitjean (1998) and Gerstein and Levitt (1998),

Double dynamic programming in the program SSAP is described in Taylor and Orengo (1989). An early iterative version is described in Orengo and Taylor (1990), while the algorithm above is described in Taylor (1997a, 1999a).

10

Geometric Techniques

A number of structure comparison methods use geometric techniques to find similar substructures between the structures. Such substructures can subsequently be used in a clustering method and combined into larger equivalences, as described in Chapter 11. In this chapter we focus on the first step, and we consider geometric hashing and distance-based techniques.

10.1 Geometric Hashing

Geometric hashing was originally developed as a computer vision technique for matching geometric features. In order to explain the principles, we first describe the basic ideas using two-dimensional geometric figures, and then show how geometric hashing is used for structure comparison.

10.1.1 Two-dimensional geometric hashing

Assume we have two two-dimensional geometric figures, a *model* A, and a *query* B, described by m and n *points*, respectively. The task is to discover common subfigures, invariant under both rotation and translation (rigid-body transformation, scale could also easily be included, but this will not be dealt with here since all structures are usually given in the same scale). One approach is to try to 'place the query on top of the model', and consider how many points coincide (ignoring the edges). This is illustrated in Figure 10.1, where six points coincide under a given threshold for point coincidence, meaning that the distance between points that coincide is less than the threshold.

Finding this maximal coincidence set is NP-hard. In addition we might not only want to find the maximal coincidence, but *all* coincidences with the number of points over a given threshold. Geometric hashing is a technique used for this problem. The geometric hashing technique defines coordinate systems, called *reference frames*, in both A and B, using, for example, two figure points for each frame (in the 2D case).

Protein bioinformatics: an algorithmic approach to sequence and structure analysis
I. Eidhammer, I. Jonassen and W. R. Taylor © 2004 John Wiley & Sons, Ltd ISBN: 0-470-84839-1