

Figure 14.6: a. A tree with its nodes labeled by a (multi)set of strings. b. A multiple alignment of those strings that is consistent with the tree. The pairwise scoring scheme scores a zero for each match and a one for each mismatch or space opposite a character. The reader can verify that each of the four induced alignments specified by an edge of the tree has a score equal to its respective optimal distance. However, the induced alignment of two strings which do not label adjacent nodes may have a score greater than their optimal pairwise distance.

14.6.2. A bounded-error approximation method for *SP* alignment

Because the exact solution to the *SP* alignment problem is feasible for only a small number of strings, most practical (heuristic) multiple alignment methods do not insist on finding the optimal *SP* alignment. However, little is usually known about how much the alignments produced by those heuristics can deviate from the optimal *SP* alignment. In contrast, in this section we discuss one of the few methods where such error analysis has been possible. We develop here a *bounded-error approximation* method, from [201], for *SP* alignment. That is, the method is provably fast (runs in polynomial worst-case time) and yet produces alignments whose *SP* score is guaranteed to be less than twice the score of the optimal *SP* alignment. This will be the first of several bounded-error approximation methods presented in this book, which will include additional methods for different multiple alignment problems.

An initial key idea: alignments consistent with a tree

The *SP* approximation we present, the improvements of it, and several other methods for other problems all use a key idea that relates multiple alignments to trees.¹⁰ We develop that idea in general before continuing with *SP* alignment in particular. Recall that for two strings, $D(S_i, S_j)$ is the (optimal) weighted edit distance between S_i and S_j .

Definition Let \mathcal{S} be a set of strings, and let T be a tree where each node is labeled with a distinct string from \mathcal{S} . Then, a multiple alignment \mathcal{M} of \mathcal{S} is called *consistent* with T if the induced pairwise alignment of S_i and S_j has score $D(S_i, S_j)$ for each pair of strings (S_i, S_j) that label adjacent nodes in T . For an example, see Figure 14.6.

Theorem 14.6.1. *For any set of strings \mathcal{S} and for any tree T whose nodes are labeled by distinct strings of \mathcal{S} , we can efficiently find a multiple alignment $\mathcal{M}(T)$ of \mathcal{S} that is consistent with T .*

¹⁰ This connection between trees and multiple alignments should not be confused with the connection between *evolutionary* trees and multiple alignments, to be discussed in Sections 14.8 and 14.10.2.

Note that the role of T in this theorem is very special and the induced alignment of two strings S_i and S_j that do not label adjacent nodes will generally have a score greater than $D(S_i, S_j)$.

PROOF OF THEOREM 14.6.1 We will construct the multiple alignment $\mathcal{M}(T)$ one string at a time and show inductively that Theorem 14.6.1 holds after each new string is added to the alignment. To start, choose any two strings S_i and S_j that label adjacent nodes in T and form a two-string alignment of S_i and S_j with distance $D(S_i, S_j)$. The theorem trivially holds at this point. Assume that the theorem holds after some arbitrary number of strings have been added to the multiple alignment. To continue, select any string S' not yet included in the alignment such that S' labels a node adjacent to a node whose label, S_i say, is already in the alignment. In that existing multiple alignment, some spaces may have been inserted into S_i , and we use \bar{S}_i to denote the string S_i with those spaces included. Note that for every string in the existing multiple alignment, each character of that string is in a distinct column with exactly one character of \bar{S}_i .

Next, optimally align string S' with \bar{S}_i using a scoring scheme for two-string alignment, with the added rule that two opposing spaces have zero cost (they are considered a match). It is immediately apparent that the score of that resulting pairwise alignment is exactly $D(S_i, S')$. Now we will add S' to the existing multiple alignment so that the induced alignment of S_i and S' has score $D(S_i, S')$ and so that all the induced scores of the strings already in the alignment remain unchanged. Let \bar{S}' be the string S' with any spaces inserted by the alignment of S' and \bar{S}_i . If the pairwise alignment of S' and \bar{S}_i does not insert any new spaces into \bar{S}_i , then append \bar{S}' to the existing multiple alignment. The result is a multiple alignment with one more string, where the induced score of S_i and S' is $D(S_i, S')$ and where all the induced scores from the previous multiple alignment remain unchanged.

However, if the pairwise alignment does insert a new space in \bar{S}_i between characters l and $l+1$, say, then insert a space between characters l and $l+1$ in every string in the *existing* multiple alignment. This will create new column(s) in the existing multiple alignment in which every character is a space, but otherwise retains the columns of the existing multiple alignment. (For example, suppose that the first four strings from Figure 14.6 have been multiply aligned and that \bar{S}_4 at that point is AY_Z . The alignment of S_5 to \bar{S}_4 inserts an additional space into \bar{S}_4 at the fourth position. The first four rows of Figure 14.6 show the resulting multiple alignment after that space is replicated in strings S_1 , S_2 , and S_3 .) The result of adding a column containing only spaces is a multiple alignment in which the score of all pairwise induced alignments is the same as before any spaces were inserted. Then appending \bar{S}' to that multiple alignment creates a multiple alignment of one more string, where the statement of the theorem holds. The existence of $\mathcal{M}(T)$ then follows by induction.

The time needed to compute $\mathcal{M}(T)$ is dominated by the time to compute $k-1$ pairwise alignments. If each string has length n , then each pairwise alignment takes time $O(n^2)$ and the time to construct $\mathcal{M}(T)$ is $O(kn^2)$. \square

Although Theorem 14.6.1 has become a part of “folklore”, it may have been first explicitly stated in [153]. We can now return to the approximation method for the *SP* alignment problem.

The center star method for *SP* alignment

We will describe the method in terms of an alphabet-weighted scoring scheme for two-string alignment, and let $s(x, y)$ be the score contributed when a character x (possibly a space) is aligned opposite a character y (possibly a space).

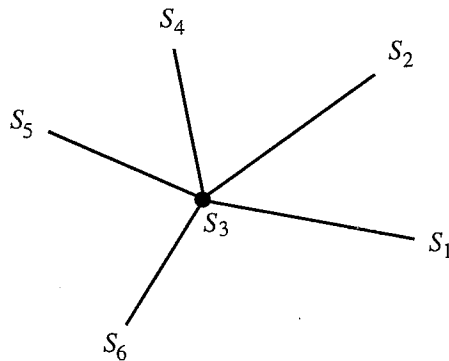


Figure 14.7: A generic center star for six strings, where the center string S_c is S_3 .

Definition A scoring scheme satisfies the *triangle inequality* if for any three characters x , y , and z , $s(x, z) \leq s(x, y) + s(y, z)$.

Triangle inequality makes good intuitive sense in the context of edit distance. It says that the “cost” of transforming x to z directly is no more than the cost of first transforming x to an intermediate character y , and then transforming y to z . It should be noted, however, that not all scoring matrices in use in computational biology satisfy the triangle inequality.

Definition Given a set of k strings \mathcal{S} , define a *center string* $S_c \in \mathcal{S}$ as a string in \mathcal{S} that minimizes $\sum_{S_j \in \mathcal{S}} D(S_c, S_j)$, and let M denote that minimum sum. Define the *center star* to be a star tree of k nodes, with the center node labeled S_c and with each of the $k - 1$ remaining nodes labeled by a distinct string in $\mathcal{S} - S_c$. For an example, see Figure 14.7.

Definition Define the multiple alignment \mathcal{M}_c of the set of strings \mathcal{S} to be the multiple alignment *consistent* with the center star.

Definition Define $d(S_i, S_j)$ as the score of the pairwise alignment of strings S_i and S_j induced by \mathcal{M}_c . Denote the score of an alignment \mathcal{M} as $d(\mathcal{M})$.

Clearly, $d(S_i, S_j) \geq D(S_i, S_j)$, and $d(\mathcal{M}_c) = \sum_{i < j} d(S_i, S_j)$. Also, since \mathcal{M}_c is consistent with the star tree, and string S_c is at the center of the star, $d(S_i, S_c) = D(S_i, S_c)$ for each string S_i . We will show that $d(\mathcal{M}_c)$ is at most twice the score of the optimal *SP* multiple alignment of \mathcal{S} , provided that the scoring scheme used for pairwise alignment satisfies the triangle inequality.

Lemma 14.6.1. *Assume that the two-string scoring scheme satisfies the triangle inequality. Then, for any strings S_i and S_j in \mathcal{S} , $d(S_i, S_j) \leq d(S_i, S_c) + d(S_c, S_j) = D(S_i, S_c) + D(S_c, S_j)$.*

PROOF Consider any single column in the multiple alignment \mathcal{M}_c , and let x , y , and z be the three characters in this column from the three strings S_i , S_c , and S_j , respectively. By the triangle inequality, $s(x, z) \leq s(x, y) + s(y, z)$, and so the claimed inequality follows by the definition of d . The claimed equality follows because the pairwise alignment of S_i and S_c induced by \mathcal{M}_c is an optimal alignment of S_i and S_c , and this is true also for the alignment of S_c and S_j . \square

Definition Let \mathcal{M}^* be the optimal multiple alignment of the k strings of \mathcal{S} . Let $d^*(S_i, S_j)$ be the score of the pairwise alignment of strings S_i and S_j induced by \mathcal{M}^* . Then $d(\mathcal{M}^*) = \sum_{i < j} d^*(S_i, S_j)$.

We can now state and prove the main theorem of this section.

Theorem 14.6.2. $d(\mathcal{M}_c)/d(\mathcal{M}^*) \leq 2(k - 1)/k < 2$.

PROOF First, define $v(\mathcal{M}_c) \equiv \sum_{(i,j)} d(S_i, S_j)$ and $v(\mathcal{M}^*) \equiv \sum_{(i,j)} d^*(S_i, S_j)$, where the pair (i, j) is an *ordered* pair in each case. Clearly, $v(\mathcal{M}_c) = 2d(\mathcal{M}_c)$ and $v(\mathcal{M}^*) = 2d(\mathcal{M}^*)$, and so the ratios $d(\mathcal{M}_c)/d(\mathcal{M}^*)$ and $v(\mathcal{M}_c)/v(\mathcal{M}^*)$ are equal. It is more convenient to work with the second ratio. Recall that the minimum sum of distances, M , is defined as $\sum_j D(S_c, S_j)$. Now $v(\mathcal{M}_c) = \sum_{(i,j)} d(S_i, S_j) \leq \sum_{(i,j)} [D(S_i, S_c) + D(S_c, S_j)]$, by Lemma 14.6.1. For any fixed j , $D(S_c, S_j)$ (which equals $D(S_j, S_c)$) shows up in this expression exactly $2(k-1)$ times. So $v(\mathcal{M}_c) \leq 2(k-1) \times \sum_j D(S_c, S_j) = 2(k-1)M$.

From the other side, $v(\mathcal{M}^*) = \sum_{(i,j)} d^*(S_i, S_j) \geq \sum_{(i,j)} D(S_i, S_j) = \sum_i \sum_j D(S_i, S_j) \geq k \times \sum_j D(S_c, S_j) = kM$ (by the choice of S_c). So $d(\mathcal{M}_c)/d(\mathcal{M}^*) = v(\mathcal{M}_c)/v(\mathcal{M}^*) \leq 2(k-1)M/kM = 2(k-1)/k = 2 - 2/k < 2$. \square

Note that for $k = 3$ the guaranteed upper bound is $4/3$. That is, for three strings, the multiple alignment produced by the center star method will never be more than 34% more than the optimal *SP* score. Translated into lower bounds this says that for $k = 3$, $d(\mathcal{M}^*) \geq .75d(\mathcal{M}_c)$. For $k = 4$ the upper bound is only 1.5, and for $k = 6$ (a problem size considered to be too large for efficient exact solution with strings of length 200) the bound is still only 1.67.

Corollary 14.6.1.

$$kM \leq \sum_{i < j} D(S_i, S_j) \leq d(\mathcal{M}^*) \leq d(\mathcal{M}_c) \leq [2(k-1)/k] \sum_{i < j} D(S_i, S_j).$$

In practice one can better measure the goodness of \mathcal{M}_c by the ratio $d(\mathcal{M}_c)/\sum_{i < j} D(S_i, S_j)$. By Corollary 14.6.1 this ratio is always less than two, but the analysis used there is worst case, so one can expect the ratio to often be considerably less than two. Similarly, one should expect that $d(\mathcal{M}_c)/d(\mathcal{M}^*)$ will often be considerably less than two, since typically $\sum_{(i,j)} D(S_i, S_j)$ will be considerably larger than kM ; that $d(\mathcal{M}^*)$ will not generally be close to $\sum_{i < j} D(S_i, S_j)$ for any strings except those that are very similar; and that $D(S_i, S_j)$ will be less than $D(S_i, S_c) + D(S_c, S_j)$ for most typical strings.

Corollary 14.6.1 is also useful in the MSA speedup discussed in Section 14.6.1 for the exact solution to *SP* alignment, since that method requires knowing an efficiently computed upper bound z on the optimal *SP* alignment score.

14.6.3. Weighted *SP* alignment

A generalization of *SP* alignment was introduced by Altschul and Lipman [20], in which the induced pairwise score of each pair (S_i, S_j) is multiplied by a weight $w(i, j)$. Then the score of a multiple alignment \mathcal{M} is $\sum_{i < j} w(i, j)d(S_i, S_j)$, where $d(S_i, S_j)$ is the score of the pairwise alignment induced by \mathcal{M} . The weights change the importance given to different pairs of strings and are often intended to reflect known evolutionary distance between the organisms from which the strings are obtained. Using weights, one can try to induce the multiple alignment to more accurately reflect known evolutionary history.

The optimal *weighted SP* alignment can again be computed in exponential time by dynamic programming, but little is known about approximating weighted *SP* alignment. However, the weights are primarily desired as a reflection of evolutionary distance, and there is a different multiple alignment objective function that addresses the same goal of using evolutionary history to influence the resulting multiple alignment. That objective function is contained in the *phylogenetic alignment* problem, for which there is an approximation method quite related to the center star method. We will discuss those topics in Section 14.8. But before doing so, we discuss the consensus objective function and an approximation algorithm even more related to the center star method.