Proc. Natl. Acad. Sci. USA
Vol. 82, pp. 3073–3077, May 1985
Biochemistry

Simultaneous comparison of three protein sequences

(sequence alignment/copper proteins)

M. MURATA*, J. S. RICHARDSON†, AND JOEL L. SUSSMAN‡§

Departments of *Inorganic Chemistry and †Pure Mathematics, University of Sydney, New South Wales 2006, Australia; and ‡Laboratory of Molecular Biology, National Institute of Arthritis, Diabetes, and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD 20205

Communicated by Harry B. Gray, December 14, 1984

ABSTRACT **Currently there are several computer algorithms available for aligning two biological sequences. When more than two sequences are to be aligned, however, pairwise comparisons using these methods rarely lead to a consistent alignment of the sequences. One obvious solution to this problem is to compare all the sequences simultaneously. Here we present an algorithm for the simultaneous comparison of three biological sequences. The algorithm is an extension of the method developed by S. B. Needleman and C. D. Wunsch, but it decreases the almost prohibitively long computing time required by a direct naive extension to a practical level: it takes time proportional to the cube of the mean sequence length, in comparison to the fifth power time taken by the direct extension. Simultaneous comparison not only gives a consistent alignment of the three sequences, but it could also reveal homologous residues in the sequences that might be overlooked by the pairwise comparisons. As an example of the application of the algorithm, three copper-containing proteins, plastocyanin, stellacyanin, and cucumber basic blue protein, are compared.**

For finding similarities between the amino acid sequences of two proteins, the method developed by Needleman and Wunsch (1) has been widely used. Their algorithm not only gives a rational way of aligning the two sequences, but it can also be used to evaluate the statistical significance of the similarity between them.

When more than two proteins are compared, however, the result of the sequence alignment by pairwise comparisons may not be consistent for certain residues. Suppose that three sequences A, B, and C are compared. From the first two pairwise comparisons of A with B and A with C, the *i*th residue of A and the *j*th residue of B and the *i*th residue of A and the *k*th residue of C might be aligned. In this case, however, the third pairwise comparison of B with C might not align the *j*th residue of B with the *k*th residue of C. This is because the pairwise comparisons maximize only the alignment of the two sequences concerned.

The above problem can be solved if the three sequences are compared simultaneously. A simultaneous sequence comparison may also facilitate finding segments that are homologous in all the sequences involved.

The algorithm of Needleman and Wunsch can be extended, as these authors indicated in their paper, to be used for simultaneous comparison of more than two amino acid sequences. The usefulness of such an extension has been pointed out by Doolittle (2) and Smith *et al.* (3). However, the calculation requires a larger amount of computer memory and a longer CPU time, as it involves an *n*-dimensional matrix for the comparison of *n* sequences.

A computer program has been written for simultaneous comparison of three proteins. In this communication, an algorithm that decreases the CPU time to a reasonable level is described. The method was applied to three small so-called blue or type 1 copper proteins: plastocyanin (Pc), stellacyanin (Sc), and cucumber basic blue protein (CBP).

The need for a nonsubjective sequence alignment program has become much more important with the advent of the huge growth in the number of DNA and protein sequences available. In addition, a reliable sequence alignment program is a prerequisite for building a model structure based on an amino acid sequence of a protein that is assumed to be similar to another protein whose three-dimensional structure has already been determined. Such model building is at present playing a key role in genetic engineering attempts to design proteins with somewhat altered specificity or functional properties. The algorithm described here provides a step toward a reliable, automated, and nonsubjective alignment program of the kind required.

ALGORITHM

Let the three protein sequences A, B, and C have lengths m, n, and p, and denote by A(i), B(i), and C(i) the ith residues in the respective sequences. To each possible triplet of residues A(i), B(j), C(k), we assign a value $\kappa(i,j,k)$. This is obtained by first comparing the three pairs A(i) and B(j), A(i) and C(k), B(j) and C(k), and assigning weights to the pairs from a suitable matrix such as the genetic code matrix; the matrix developed by McLachlan (4); the mutation data matrix of Dayhoff *et al.* (ref. 5, Fig. 84); or any other matrix providing a measure of the similarities between pairs of amino acids. (If necessary, a suitable constant should be added to make all matrix entries non-negative.) Then $\kappa(i,j,k)$ is the sum, average, product, or other appropriate combination of the weights for the three pairs, depending on the nature of the data on which the matrix is based.

In this study, the three weighting systems mentioned above were used, with the following variation: the values for histidine versus histidine, cysteine versus cysteine, and methionine versus methionine were doubled, because, in light of the evidence from plastocyanin (6), these residues are likely candidates for copper ligands in the type of proteins involved and, therefore, matching them is of particular importance. The sum of the pairwise weights was used to calculate κ:

$$\kappa(i,j,k) = w[\mathrm{A}(i),\mathrm{B}(j)] + w[\mathrm{A}(i),\mathrm{C}(k)] + w[\mathrm{B}(j),\mathrm{C}(k)].$$

(The values of κ need not be stored: they may be computed quite cheaply each time they are required from stored values for the 20×20 array w.)

Now imagine a three-dimensional rectangular prism divided into a lattice of cells (i,j,k) corresponding to the residue triplets A(i), B(j), C(k), for $i = 1, \ldots, m$, $j = 1, \ldots, n$, $k = 1, \ldots, p$. We wish to find a path through the prism from a starting cell on the top ($i = 1$), front ($j = 1$), or left ($k = 1$) face to

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

Abbreviations: CBP, cucumber basic blue protein; Pc, plastocyanin; Sc, stellacyanin.

§Permanent address: Department of Structural Chemistry, Weizmann Institute of Science, 76100 Rehovot, Israel.

a final cell on the bottom ($i = m$), back ($j = n$), or right ($k = p$) face such that the sum of the values κ over the cells in the path is a maximum.

Note that successive cells (i,j,k) and (i',j',k') in a path must have the property that each of $i' - i, j' - j$, and $k' - k$ is ≥ 1 and at least one of them is exactly 1. We can rephrase this more succinctly in terms of "subshells." Define the *subshell* of a cell (i,j,k) to be the set of cells in the three planar regions

$$\{(i,y,z): j \leqq y \leqq n,\ k \leqq z \leqq p\}$$

$$\{(x,j,z): i \leqq x \leqq m,\ k \leqq z \leqq p\}$$

$$\{(x,y,k): i \leqq x \leqq m,\ j \leqq y \leqq n\}.$$

Then the above condition says that (i',j',k') must be in the subshell of $(i + 1, j + 1, k + 1)$.

It may be desirable to lessen the occurrence of gaps in the chosen path—that is, places where $i' - i, j' - j$, and $k' - k$ are not all exactly 1. This can be done by subtracting a fixed non-negative gap penalty γ from the sum of the values κ for each gap in the path.

The method of computation used, following that of Needleman and Wunsch, is to work backwards from the cell (m,n,p), calculating the maximum total value λ for paths from each cell. In more detail, define $\lambda(i,j,k)$ to be the maximum, over all paths from cell (i,j,k) to the bottom, back, or right face, of the sum of the values κ over the cells in the path minus γ times the number of gaps in the path.

It will simplify explanation of various algorithms for calculating λ if we define a further quantity: $\mu(i,j,k)$ will be the maximum value of λ over all the cells in the subshell of (i,j,k). [For full understanding of the methods used to calculate μ described below, one should bear in mind that as a consequence of its definition, $\mu(i,j,k)$ is also the maximum value of $\lambda(x,y,z)$ over all cells (x,y,z) with $x \geqq i$, $y \geqq j$, and $z \geqq k$.]

The following "summation" algorithm may be used to calculate λ and μ:

```
Alg. 1: for i := m downto 1 do
            for j := n downto 1 do
            for k := p downto 1 do
  begin
λ(i,j,k) := κ(i,j,k) +
                max[λ(i+1,j+1,k+1), μ(i+1,j+1,k+1) − γ];
μ(i,j,k) := max[λ(i,j,k), μ(i+1,j,k),
                                μ(i,j+1,k), μ(i,j,k+1)]
  end.
```

Here we assume that initial values of zero have been assigned to $\lambda(i,j,k)$ and $\mu(i,j,k)$ for $i > m, j > n$, or $k > p$.

Once the matrix λ has been calculated, the successive cells in the best path may be printed out as follows:

```
Alg. 2: (r,s,t) := (1,1,1);
        while r ≦ m and s ≦ n and t ≦ p do
  begin
        (x,y,z) := (r,s,t) [best location yet];
        b := λ(r,s,t) [best value so far];
        d := 0 [shortest distance];
        for each (i,j,k) in subshell of (r,s,t) do
            if λ(i,j,k) − γ < b or
              [λ(i,j,k) − γ = b and
            distance from (r,s,t) to (i,j,k) < d]
        then
        begin
            (x,y,z) := (i,j,k);
            b := λ(i,j,k) − γ;
            d := distance from (r,s,t) to (i,j,k)
        end;
            print (x,y,z);
            (r,s,t) := (x+1,y+1,z+1)
  end.
```

Note that among those cells in the subshell of the current cell (r,s,t) with the best possible value of $\lambda - \gamma$ [or just λ for (r,s,t) itself], we choose one as close to (r,s,t) as possible. The usual Euclidean distance (or, for speed, its square) may be used.

In Alg. 2 as shown above, one complication has been neglected. The stated algorithm will subtract the gap penalty γ in the case of a gap occurring at the start of the path, but not for one at the end of the path. We chose not to impose a penalty for a gap at either end, and we modified the start of Alg. 2 slightly to achieve this: terminal residues were permitted to align with nonterminal residues without penalty. Note that imposing a penalty at terminals would presume homologous terminal residues. Therefore, as mentioned by Fitch *et al.* (7), without a prior knowledge that the terminal residues are aligned, external gaps should not be penalized. [If, on the other hand, it were considered desirable to penalize external gaps, another reasonably simple modification to the algorithms could be made: null residues could be added to the sequence terminals, as suggested for two-sequence comparisons by Smith *et al.* (3).]

We now consider some of the details involved in practical implementations of the above algorithms.

First, we examine the time and storage requirements of the summation algorithm Alg. 1. It is possible to save space at the expense of time by not storing the values $\mu(i,j,k)$, but instead calculating them when needed by scanning the subshell of (i,j,k) to find the maximum value of λ. In fact, Needleman and Wunsch may have used the two-dimensional analogue of this approach: at least they make no mention in their paper of having stored what we call μ.

Let l be the geometric mean of m, n, and p. Then the time taken by Alg. 1 is proportional to l^3 if μ is stored, but to l^5 if μ is recalculated. In practice, the time penalty for not storing μ will be much worse on many computers: where "virtual memory" is used—that is, where a relatively small amount of fast memory is supplemented by a much larger quantity of slow backing storage (e.g., on disk)—numerous accesses to values of λ that are not stored contiguously will be very time consuming.

To store λ requires an amount of memory proportional to l^3; if μ is stored as well, twice as much memory will be used. The latter was not convenient in our case. The proteins under consideration have lengths of 96, 99, and 107 residues, so $l^3 = 1{,}016{,}928$. The choice of any of the similarity matrices mentioned above as a basis for κ dictates that 2-byte integers be used for values of λ; thus 2,033,856 bytes are required to store λ. This was well within available capacity, but twice as much memory was not conveniently accessible.

Fortunately, it is not in fact necessary to store the whole of the array μ at once: there are variations of Alg. 1 that take time proportional to l^3 but storage proportional to only

$l^2(l+1)$. In the following algorithm, for example, we use a two-dimensional array $\mu_1(j,k)$, which (roughly speaking) successively stores the values of $\mu(i,j,k)$ for decreasing values of i.

Alg. **1B**: for $i := m$ downto 1 do
for $j := n$ downto 1 do
for $k := p$ downto 1 do
begin
$\lambda(i,j,k) := \kappa(i,j,k) +$
$\max[\lambda(i+1,j+1,k+1), \mu_1(j+1,k+1) - \gamma]$;
if $j < n$ and $k < p$ then
$\mu_1(j+1,k+1) := \max[\lambda(i,j+1,k+1), \mu_1(j+1,k+1),$
$\mu_1(j+2,k+1), \mu_1(j+1,k+2)]$
end.

Again, for ease of exposition we have assumed that initial values of zero have been assigned to $\lambda(i,j,k)$ and $\mu_1(j,k)$ for $i > m, j > n$, or $k > p$. In practice, additional storage would not be used; extra tests on i, j, and k would be added instead. Alg. **1B** is explained in more detail in Fig. 1.

The path-tracing Alg. **2** is more difficult to implement efficiently in practice. Again, a theoretical saving of time at the expense of space is possible: as Needleman and Wunsch suggest (ref. 1, p. 446), one could record during the summation phase the origin of the number added to κ in calculating λ for each cell (i,j,k)—that is, the location of the first cell in the best path leading from (i,j,k). One could then trace the overall best path in time proportional to l. However, the amount of storage required for this approach would be of the order of $l^3 \log_2(l^3)$ bits, which is prohibitively large in practice.

A direct implementation of Alg. **2** as given above requires no significant storage apart from λ, and it takes time proportional to l^3 (provided enough fast memory is available to store the whole of λ: the calculation is much slower if virtual memory is required). This was the method we used.

Note that because of the requirements of significance testing, the bulk of calculations performed will usually not be with the actual proteins, but with random permutations thereof. In this case, the best path itself is of no interest: only the value of that path (i.e., the maximum match) is needed. Thus, no path tracing is required. In fact, a modified form of Alg. **1B** may be constructed (at the expense of a little more storage) in which μ_1 does not "lag behind" λ but keeps up; this can be arranged so that $\mu_1(1,1)$ contains the value of the best path at the end of the computation.

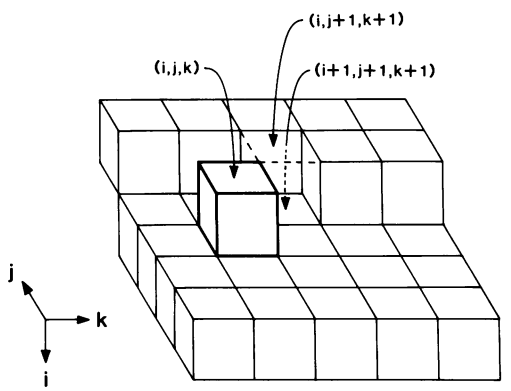

FIG. 1. Illustration of Alg. **1B** at the point where the value of λ for the cell (i,j,k) (shown with heavily outlined edges) is calculated. Cells with lightly outlined edges are those for which the array μ_1 holds the current value of μ just before the assignment to $\mu_1(j+1,k+1)$. After the assignment, $\mu_1(j+1,k+1)$ holds the value of μ for the cell $(i,j+1,k+1)$, indicated by dashed edges. As the figure suggests, one may think of the array μ_1 as holding the values of μ for a layer of cells with a moving ridge or faultline in it: the dashed cell is about to replace the one below it as part of this layer.

RESULTS AND DISCUSSION

The above algorithm was applied to the sequences of three copper proteins, Pc from poplar (quoted in ref. 6), Sc (8), and CBP (9). (Note that residue 37 of the cucumber protein has not been identified, but the presence of serine is suspected.) Fig. 2 shows part of a computer printout of the comparison of the three copper proteins using the weighting system of McLachlan. The CPU time was 81 sec (on a VAX-11/780), which was short enough to run a reasonable number of comparisons of scrambled sequences for statistical analysis. In this work, a sample size of 100 was used to obtain the mean and standard deviation of the scores from the comparisons of scrambled sequences. The standardized score was calculated in the usual way as the number of standard deviations by which the score of the real sequences differs from the mean score for scrambled sequences. The gap penalty used for the comparison shown in Fig. 2 was 12. The choice of this value is discussed below. The arrows in the figure were added later to indicate the locations of gaps or deletions. A complete alignment is shown in Fig. 3.

The alignment of sequences found by the algorithm described above depends on the weights assigned for amino acid pairs and on the gap penalty used. Shown in Table 1 are the results of comparisons obtained using three weighting systems with various gap penalties. The gap penalties listed in the table were selected from the values around the average weights for the identical amino acid pairs, namely 3.45, 9.5, and 16.3 for the genetic code matrix, McLachlan matrix, and mutation data matrix, respectively. In the algorithm used here, no penalty was imposed for a gap at either end of the sequences. Thus, only internal gaps were counted in Table 1. Furthermore, consecutive internal gaps were counted as 1 regardless of their length. This is because in successive summations of the matrix described above, no distinction was made (beyond the presence or absence of a gap) as to the location on the three planes of the maximum value that was used to obtain the final cell values. The table also lists the total gap length—i.e., the sum of the number of sequence elements in the gaps.

Table 1. Alignment scores

	Gap penalty	Standardized score	Matches		Internal gaps	
			Triple	Double	Number	Length
Genetic code matrix	0	12.23	11	43	39	47
	3	11.87	11	41	13	33
	6	11.12	4	51	4	27
	9	10.98	4	51	4	24
	12	11.30	4	50	4	24
Similarity matrix (McLachlan)	4	14.31	15	39	18	44
	8	15.90	16	37	16	41
	12	16.18	14	39	14	37
	16	16.91	9	46	10	35
	20	17.46	7	46	5	35
	24	17.55	7	47	4	33
Mutation data matrix	8	13.72	10	40	14	20
	12	14.30	10	39	14	20
	16	14.96	9	38	9	17
	20	15.01	9	38	8	17
	24	15.62	8	40	8	17
	28	15.16	9	39	8	17

PC		SC		CBP		SCORE
1	I	1	T	1	A	1271
2	D	2	V	2	V	1263
3	V	3	Y	3	Y	1253
4	L	4	T	4	V	1238
5	L	5	V	5	V	1227
6	G	6	G	6	G	1209
7	A	7	D	7	G	1185
8	D	8	S	8	S	1176
9	D	9	A	9	G	1162
10	G	10	G	10	G	1153
11	S	11	W	11	W	1129
13	A	14	P	12	T	1126
14	F	15	F	13	F	1116
15	V	25	W	18	W	1101
16	P	26	A	19	P	1088
17	S	27	S	20	K	1072
18	E	28	N	21	G	1058
19	F	29	K	22	K	1048
20	S	30	T	23	R	1040
21	I	31	F	24	F	1028
22	S	32	H	25	R	1013
23	P	33	I	26	A	1001
24	G	34	G	27	G	994
25	E	35	D	28	D	970
26	K	36	V	29	I	952
27	I	37	L	30	L	944
28	V	38	V	31	L	926
29	F	39	F	32	F	908
30	K	40	K	33	N	881
31	N	41	Y	34	Y	865
32	N	42	D	35	N	852
33	A	43	R	36	P	834
34	G	44	R	37	S	825
35	F	45	F	38	M	815
37	H	46	H	39	H	808
38	N	47	N	40	N	760
39	I	48	V	41	V	736
40	V	50	K	43	V	730
41	F	51	V	44	V	718
42	D	52	T	45	N	704
43	E	53	Q	46	Q	693
44	D	54	K	47	G	675
45	S	55	N	48	G	666
46	I	56	Y	49	F	655
47	P	57	Q	50	S	643
48	S	58	S	51	T	633
49	G	59	C	52	C	615
50	V	60	N	53	N	595

PC		SC		CBP		SCORE
51	D	61	D	54	T	585
52	A	62	T	55	P	571
53	S	63	T	56	A	561
54	K	64	P	57	G	549
55	I	65	I	58	A	540
56	S	66	A	59	K	528
57	M	67	S	60	V	518
58	S	68	Y	61	Y	510
59	E	69	N	62	T	495
60	E	70	T	63	S	484
67	G	71	G	64	G	483
68	E	72	N	65	R	459
69	T	73	N	66	D	449
71	E	74	R	67	G	450
72	V	75	I	68	I	437
73	A	76	N	69	K	419
74	L	77	L	70	L	409
75	S	78	K	71	P	385
77	K	80	V	72	K	388
78	G	81	G	73	G	376
79	E	82	Q	74	Q	352
80	Y	83	K	75	S	334
81	S	84	Y	76	Y	327
82	F	85	Y	77	F	312
83	Y	86	I	78	I	291
84	C	87	C	79	C	277
85	S	88	G	80	N	223
86	P	90	P	82	P	224
87	H	92	H	84	H	212
88	Q	93	C	85	C	164
89	G	94	D	86	G	146
90	A	95	L	87	S	137
91	G	96	G	88	G	129
92	M	97	Q	89	M	105
93	V	98	K	90	K	83
94	G	99	V	91	I	71
95	K	100	H	92	A	63
96	V	101	I	93	V	53
97	T	102	N	94	N	35
98	V	103	V	95	A	21
99	N	104	T	96	L	7

Gap penalty = 12

Maximum match score = 1271

FIG. 2. Sequence alignment of three copper proteins, Pc, Sc, and CBP, by simultaneous comparison. The gap penalty used is 12. Maximum match scores corresponding to the residue triplets are listed in the last column. Arrows were inserted later to indicate locations of unmatched residues (which are not themselves shown). Note that in the final alignment (see Fig. 3), these unmatched residues are explicitly shown, with a corresponding gap in one or both of the other two sequences. Amino acids designated by standard one-letter abbreviations.

In terms of the standardized score, both the mutation data matrix and the matrix of McLachlan are superior to the genetic code matrix. Between the two alternatives, because of the slightly higher standardized score and the considerably greater number of triply matched residues, the weighting system of McLachlan is preferred to the mutation data matrix in the comparison of the three proteins described here. For this reason, the following discussion will be made on the results obtained by using the weighting system of McLachlan. However, it is noteworthy that the alignments produced by the mutation data matrix contain fewer gaps than those produced by the other two matrices.

The probability of obtaining the score with any of the gap penalties listed in the table by chance alone is very low. Consequently, the best alignment was chosen based on the number of residues matched and on the size and pattern of the gaps in the final alignment. As the gap penalty increases beyond 12, the number of three-residue matches drops noticeably, but there is only a small corresponding improvement in the total gap length. Thus, the alignment obtained using the penalty 12 was chosen as the best alignment.

For the sake of comparison, the same sequences were aligned in pairs (Fig. 4) by using the original algorithm of Needleman and Wunsch and the weighting system of McLachlan. Normalized scores were obtained by comparing the score of the real sequences and the mean of the scores from 100 pairs of scrambled sequences for gap penalties 0, 1, 2, ..., 10.

If we wish to construct a three-sequence alignment by superimposing, for example, the alignment of Sc/Pc onto that

FIG. 3. Alignment of the three sequences based on Fig. 2. Identical amino acids (designated by standard one-letter abbreviations) in the homologous positions are enclosed in boxes.

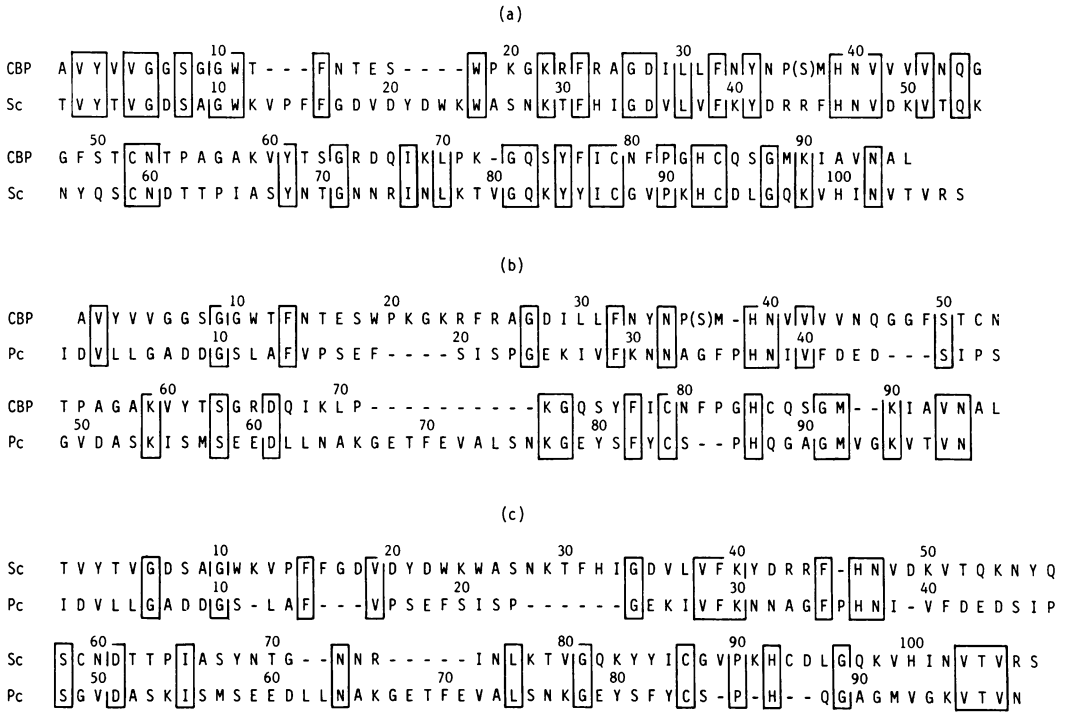

FIG. 4. Sequence alignments of the three copper proteins by pairwise comparisons according to the algorithm of Needleman and Wunsch. For each comparison, scores from 100 pairs of scrambled sequences were used to obtain the standardized score. Gap penalty (g.p.) and standardized score (S) are as follows: (*a*) CBP vs. Sc, g.p. = 6, S = 15.57; (*b*) CBP vs. Pc, g.p. = 7, S = 5.12; (*c*) Sc vs. Pc, g.p. = 4, S = 3.64. Amino acids designated by standard one-letter abbreviations.

of CBP/Sc, only four regions of the alignment can be found that are consistent with the third pairwise alignment of CBP/Pc. They are, by the residue numbers of CBP, 27–38, 39–41, 74–80, and 84; this corresponds to 25% of the CBP sequence. All the other regions require adjustments. Consider, for example, the residues around 10–20 of the three sequences. The pairwise comparisons of CBP/Sc and CBP/Pc align 13–17 of CBP with 16–20 of SC and the same residues of CBP with 14–18 of Pc, respectively. However, the residues 16–20 of Sc are not aligned with 14–18 of Pc in the alignment of Sc/Pc.

In the above example, because of the number of residues involved, it is almost impossible to construct a consistent three-sequence alignment by manipulating the results of pairwise comparisons. Even if this is done, the result will be highly subjective. Furthermore, the local adjustments may change the optimal alignment originally obtained. If that is the case, the entire sequence, including the consistent part, should be adjusted to attain a new optimal alignment. This is the main reason why the method of using pairwise comparisons is not suitable for aligning three sequences.

Among the three proteins, the copper ligands have been identified only in Pc (6); they are histidine-37, cysteine-84, histidine-87, and methionine-92. A sequence comparison between Sc and Pc has already been carried out (10). The alignment of the last section of the two sequences was very similar to the pairwise alignment shown in Fig. 4*c*. Both align histidine-100 of Sc next to methionine-92 of Pc. Because of this closeness of the position of the histidine to the ligand methionine, it was proposed that histidine-100 in Sc serves as one of the ligands. However, the result of the simultaneous three-way comparison aligns this histidine further away from the two methionine residues of Pc and CBP. Thus, although histidine-100 in stellacyanin may indeed be a ligand, the above-mentioned evidence becomes less convincing.

The advantage of simultaneous comparison over the conventional pairwise comparison in aligning three sequences is 2-fold. First, it gives a consistent alignment of three sequences without need of any manual adjustment. Second, it could reveal homologous residues in the sequences that might be overlooked by the pairwise comparisons. It should be noted, however, that the result obtained from simultaneous comparison could not be used to reject the alignment of two sequences obtained by pairwise comparison. The simultaneous comparison of three sequences could provide alternative solutions to the problem of aligning three sequences, which in turn could provide more information on the homology between the sequences under consideration.

M.M. thanks Professor H. C. Freeman for discussions. This work was supported by Grant C80/15377 from the Australian Research Grants Scheme to H. C. Freeman. The participation of J.L.S. was made possible by a travel grant from the Australian Friends of the Weizmann Institute of Science.

1. Needleman, S. B. & Wunsch, C. D. (1970) *J. Mol. Biol.* **48**, 443–453.
2. Doolittle, R. F. (1981) *Science* **214**, 149–159.
3. Smith, T. F., Waterman, M. S. & Fitch, W. M. (1981) *J. Mol. Evol.* **18**, 38–46.
4. McLachlan, A. D. (1971) *J. Mol. Biol.* **61**, 409–424.
5. Dayhoff, M. O., Schwartz, R. M. & Orcutt, B. C. (1978) in *Atlas of Protein Sequence and Structure*, ed. Dayhoff, M. O. (Natl. Biomed. Res. Found., Washington, DC), Vol. 55, pp. 345–352.
6. Colman, P. M., Freeman, H. C., Guss, J. M., Murata, M., Norris, V. A., Ramshaw, J. A. M. & Venkatappa, M. P. (1978) *Nature (London)* **272**, 319–324.
7. Fitch, W. M. & Smith, T. F. (1983) *Proc. Natl. Acad. Sci. USA* **80**, 1382–1386.
8. Bergman, C., Gandvik, E., Nyman, P. O. & Strid, L. (1977) *Biochem. Biophys. Res. Commun.* **77**, 1052–1059.
9. Murata, M., Begg, G. S., Lambrou, F., Leslie, B., Simpson, R. J., Freeman, H. C. & Morgan, F. J. (1982) *Proc. Natl. Acad. Sci. USA* **79**, 6434–6437.
10. Ryden, L. & Lundgren, J.-O. (1979) *Biochimie* **61**, 781–790.