TR/14/84 November 1984

An Application of Dynamic Programming To Pattern Recognition

by

J.P. Warwick and R.I. Phelps

Introduction

Dynamic Programming, [1], is an Operational Research (OR) technique which has found only limited usage in the Industrial and Business Management fields which are home to many other of the OR techniques. Its main application has been in problems of sequential decision making [2] but with the comparatively recent work in Artificial Intelligence, itself prompted by the advent of Fifth Generation Computing, Dynamic Programming has been found a useful tool and has been able to make significant contributions in certain areas.

Within the field of Artificial Intelligence, the main area of application has been in connection with Pattern Recognition, and much work has been conducted in this field already e.g. [3].

This paper illustrates the way that this powerful technique may be used in a Pattern Recognition context by way of a particular well defined problem, and then gives, by way of illustration, a demonstration of the method using a biochemical problem in protein analysis.

The Problem and General Approach

We shall first describe the problem under study and the way in which a solution can be found via Dynamic Programming.

The general problem to be considered is one where we need to predict the sequence in which units of differing sorts will occur, where the only information available is a measure of the probability or likelihood with which each type of unit will occur at each point in the sequence. As an example, let us suppose that we have three different sorts of unit, A, B and C and that we wish to predict a sequence of 10 such units. The probability profiles for each of the units over the sequence are found, or estimated, to be as in Figure 1. It may be the case that these distributions are not exact, either because they were estimated from data or because they could be subjective estimates, but it is assumed that over the whole length of the sequence the probability profile for a particular type of unit will predict the correct number of such units.

So, in the example we seek to find the correct ordering of 3 A type units, 4 B type units and 3 C type units, based on the information supplied by the three profiles.

Several approaches to the problem are. possible, two of which will be mentioned when a demonstration of this method is discussed later. For the present we will confine ourselves to just one way in which Dynamic Programming can be used to attack a problem of this sort.

An inspection of the profiles by eye can give an indication as to where we might expect the units to occur in the sequence. For example, it seems that we might expect type A units at positions 8, 9 and possibly 7, and type B units seem likely around positions 2 to 5. It is not readily apparent where type C units will occur just by looking at its profile since there are no

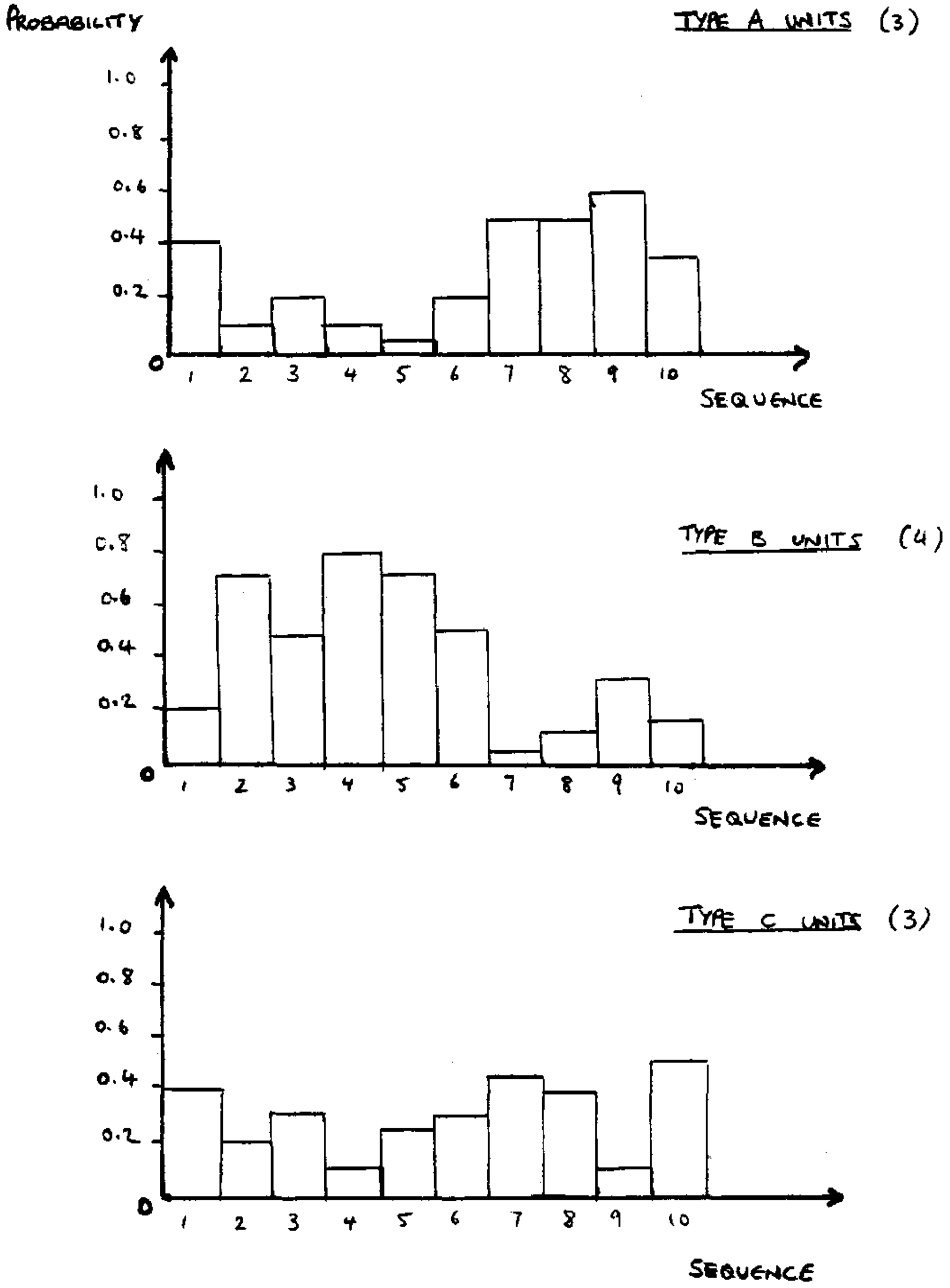
PROBABILITY
TYPE A UNITS (3)
1.0
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8
9
10
SEQUENCE
TYPE B UNITS (4)
1.0
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8
9
10
SEQUENCE
TYPE C UNITS (3)
1.0
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8
9
10
SEQUENCE

Fig 1

outstanding peaks other than at the right hand end, and even these seem to clash with the type A peaks.

In an ideal world, of course, one would like in this situation to have three profiles each with clearly defined peaks and troughs, so that it would be clear at which positions units of each type would fall, or at least were very likely to fall.

In the extreme case each profile would be a step function with value unity where a unit of that type occured and zero where it did not, making the prediction a trivial process. However, due to the inaccuracies in obtaining or estimating the profiles the peaks and troughs are smoothed out somewhat becoming less distinct and making accurate prediction from the profiles very difficult.

The method developed here attempts to convert each of the profiles into a step function by reducing the profile at some points in the sequence and increasing other parts by the same amount, effectively redistributing the area under the profile until it takes the form of a step function with values of zero or unity at each point in the sequence. (At all times the area under the profile is maintained so that the same total number of units are predicted in the sequence).

The problem now becomes one of carrying out this redistribution on each of the profiles simultaneously so that we ensure that only one profile is non zero at each point in the sequence i .e.

only one type of unit is predicted to occur at each point. In addition, it is desirable to maintain the overall shape of the profiles as far as possible, in that it would be nice to carry out the redistribution process disturbing the profiles as little as possible. To do this we consider the cost of moving an area of profile to be the area moved multiplied by the distance it is moved. The total cost of transmuting a profile into a step function is then simply the sum of all these individual costs. It is then possible to carry out the redistribution, producing three step functions, in such a way as to minimise the sum of the costs over all three functions.

Returning to the original example we could adjust the profiles as in Figure 2 to produce three step functions as shown in Figure 3. These would then produce a prediction as to the possible sequence of units. This particular redistribution is in fact optimal in the sense of minimising the cost function described above. The total cost of this redistribution is 7.75.

The problem therefore is one of redistributing the area of the profiles into step functions simultaneously and at minimum cost. This problem is ideally suited to a solution by Dyanmic Programming.

Dynamic Programming Formulation

The problem as described was augmented by including an additional constraint before being formulated as a Dynamic Program. In some cases (for example the application illustrated later) it may be

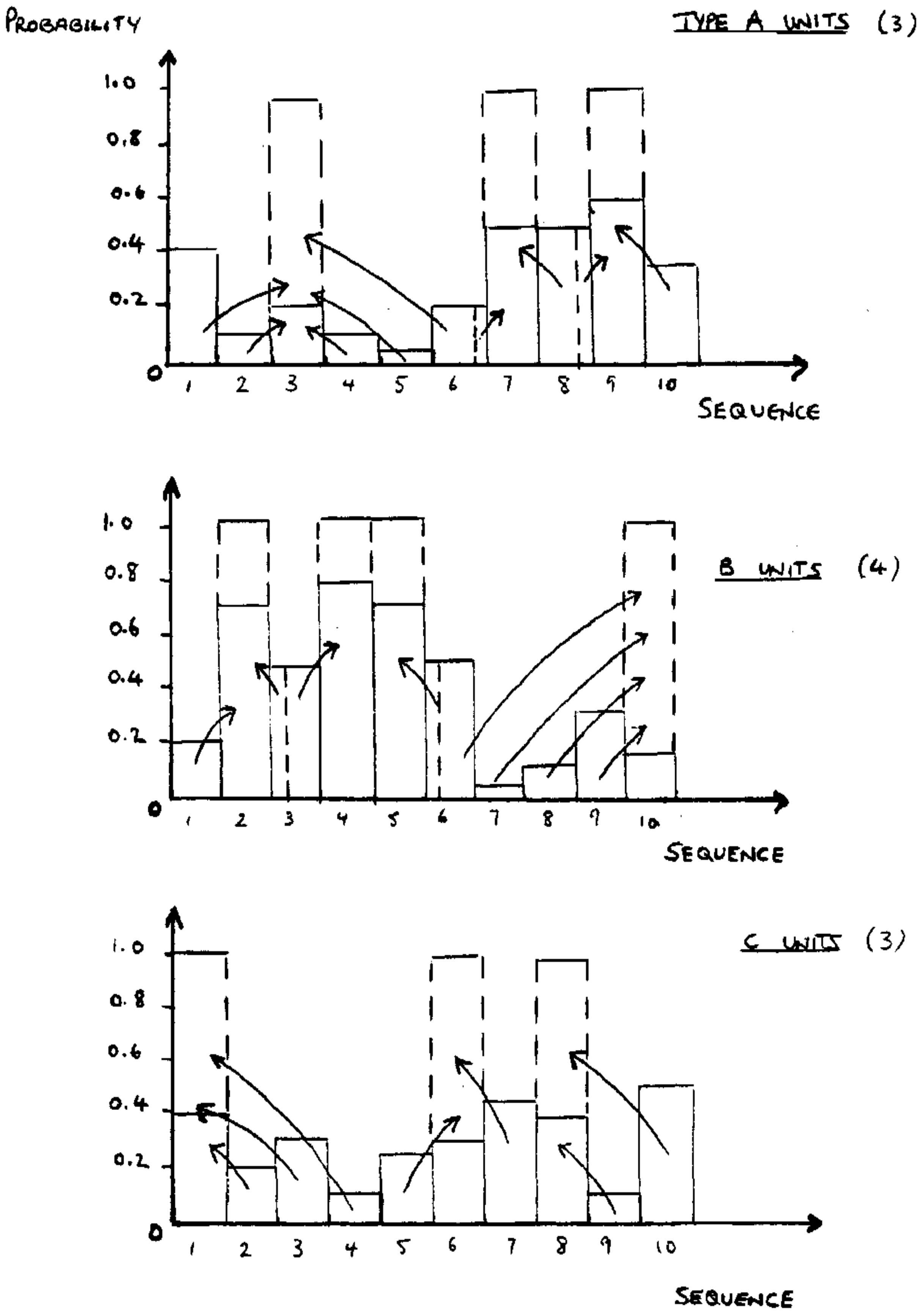
PROBABILITY
TYPE A UNITS (3)
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE
B UNITS (4)
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE
C UNITS (3)
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE

Fig 2

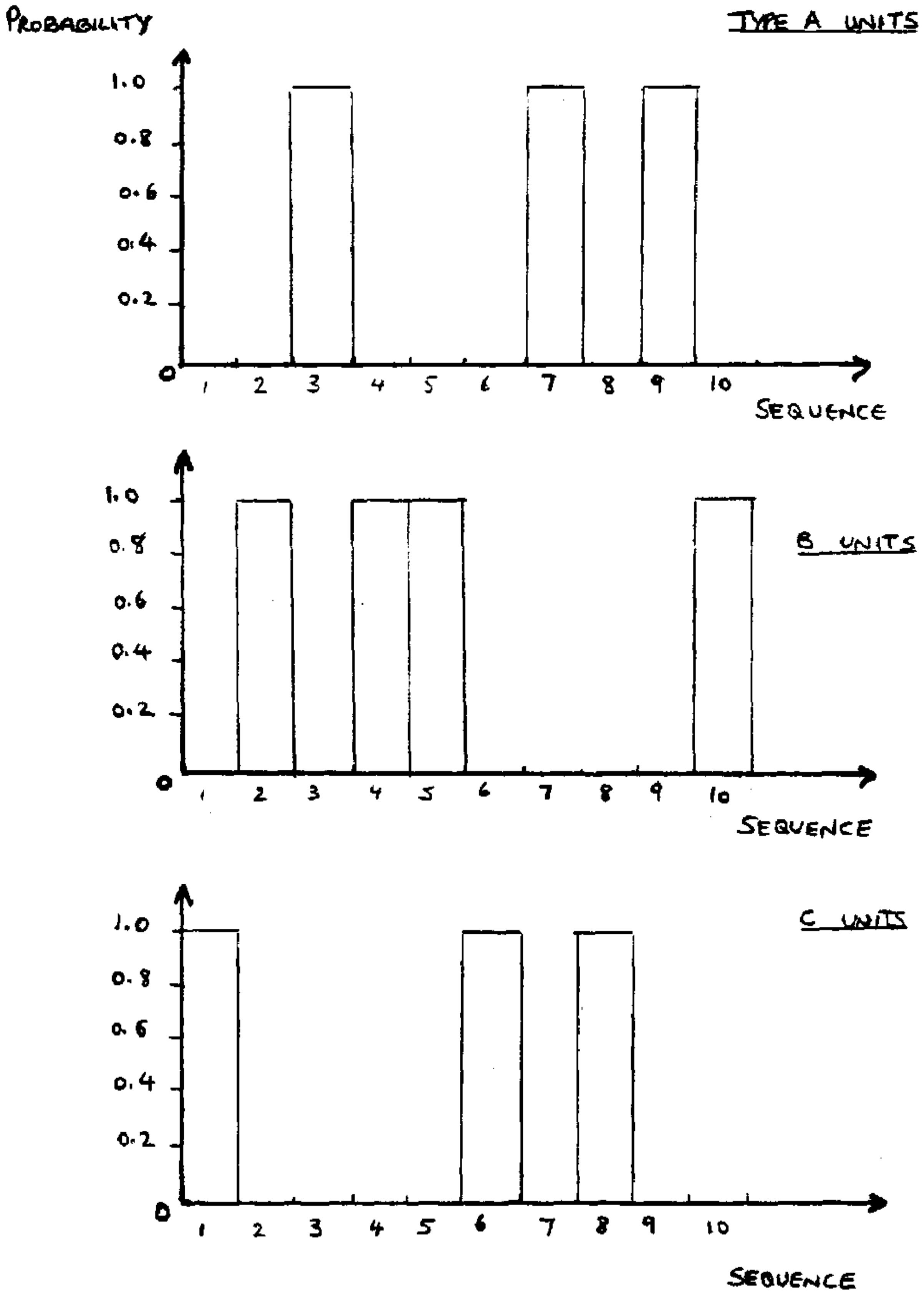
PROBABILITY
TYPE A UNITS
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE
B UNITS
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE
C UNITS
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE

Fig 3

desirable that one or more of the type of units should occur in groups or runs within the sequence, runs whose maximum and minimum lengths can be specified beforehand.

The first stage in the process is to work through each probability profile marking off points in the sequence which bound an area of approximately unity, see Figure 4. This process is necessary in calculating the cost of fitting, say the second type A unit at position 4, having fitted the first A unit beforehand. The cost would be calculated as

$(3 \times 0.5) + (4 \times 0.5) = 3.5$

We shall use the notation C(A, i, N) to be the cost of fitting a unit of type A at position i having already fitted N_A units of A.

So here we have that C(A, 4, 1) = 3.5

Now, in order to calculate the cost of fitting a particular unit in say position i of the sequence we need to know how many units of that type have already been fitted. In addition, the constraint concerning the allowable length of a run of these units means that we must also know the type of unit at position i-1, and how far along a run of units it occured. If, for example the unit at position i-1 was of type A and was the sixth in a run of As, then if the maximum length of an run of As in six we can not fit an A at position i. Similarly, if the previous unit at position i-1 was an A and was the second in a run of As, if the minimum run length for As is four, we must fit another A at position i.

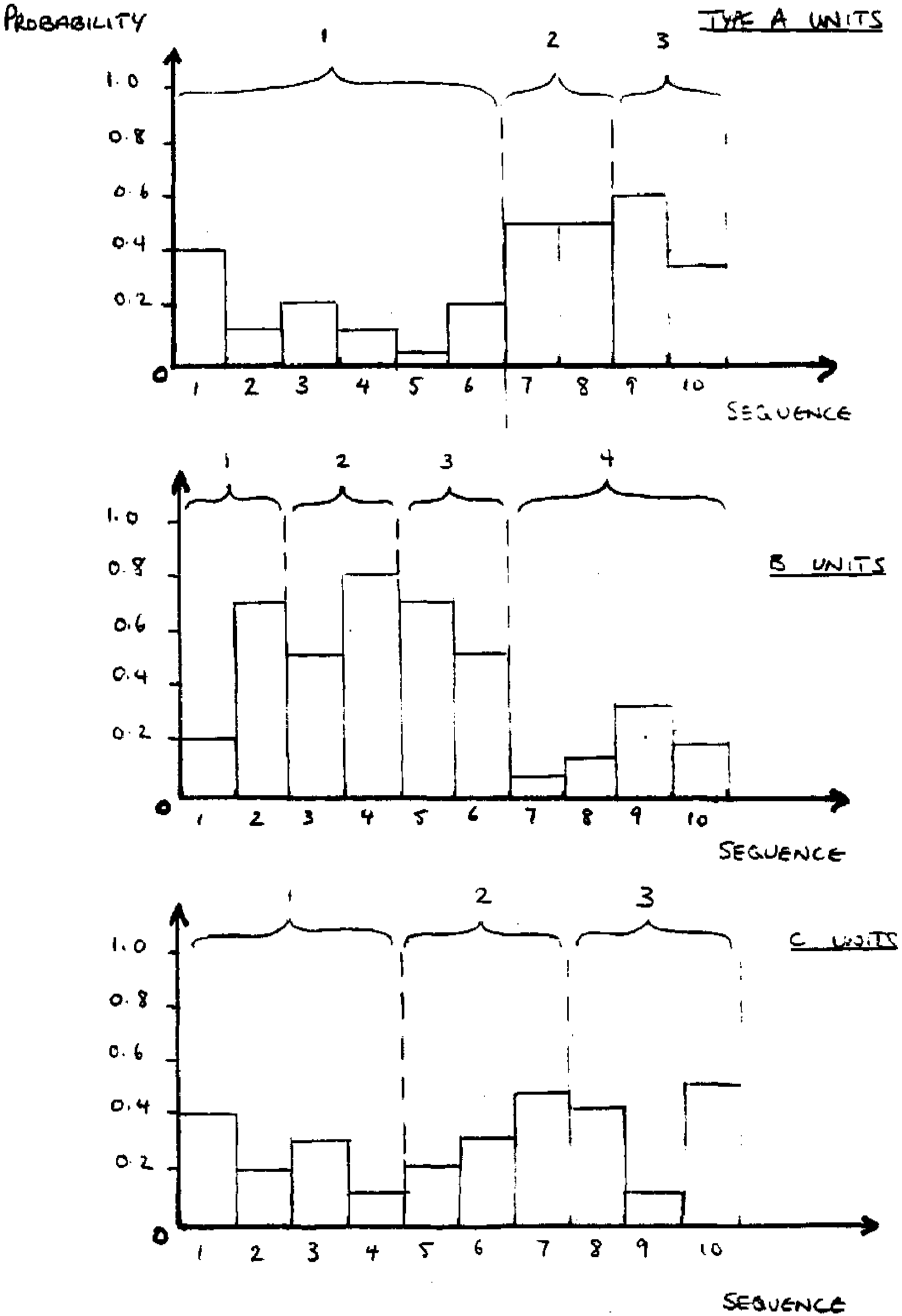
PROBABILITY
TYPE A UNITS
1
2
3
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE
1
2
3
4
B UNITS
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE
1
2
3
C UNITS
1.0
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
SEQUENCE

Fig 4

Now, the Dynamic Program will progress stage by stage fitting a unit at each point in the sequence in turn. In our example, the state description of the Dynamic Program at any stage i contains four elements;

(i) the number of type A units already fitted; denoted as N_A

(ii) the number of type B units already fitted; denoted as N_B

(the number of type C units can be calculated as $N_c=i-N_A-N_B-1$)

(iii) the type of unit at position i-1; denoted as $U = A, B$ or C

(iv) the length of the run of units up to, and including, position i-1; denoted as R.

We denote the optimal (minimum) cost from position i <u>onwards</u> to the end as

$V_i(N_A, N_B, U, R)$

and the optimal sequence of units from positon i onwards as:-

${}^{P}{}_{i}\ (N_A,\ N_B,\ U,\ R)$

Each combination of N_A, N_B, U and R represents one possible state of the sequence upto the point i-1. Clearly, some of these combinations are not ' legal ' in that they can not possibly occur. For example, we must have that

$N_A + N_B < i$

i.e. that we can not have fitted more than i-1 unit so far. (N_A

and N_B are always $\geqslant 0$). Also, as another example, we must have

$N_A \geqslant R$ if $U = A$

$N_B \geqslant R$ if $U = B$ and

$N_C \geqslant R$ if $U = C$

that is we must have previously fitted at least as many units of one type as there are in the latest run.

Each of the constraints are incorporated into the Dynamic Program and if a particular combination of N_A , N_B , U and R is in violation of one of these constraints it can be excluded by setting the cost for that combination infinitely large.

As with many Dynamic Programs the process begins at, the end of the sequence and works backwards, so that the first stage fits a unit at the last position in the sequence and successive stages fit units successively backward along the sequence . Hence we always know the optimal sequence of units from any point i onwards to the end, given any possible sequence from positions 1 to i-1.

Which type of unit is fitted in position i will clearly depend on what is fitted in position 1 to i-1, and so the process for considering position i is as follows; we iterate through all the legal combinations of N_A , N_B , U and R (describing the sequence upto i-1) and for each combination we decide which type of unit to fit at position i so as to minimise cost from i onwards. As an example let us consider the calculation of V_i $(N_A', N_B'\ A, R')$ in which the minimum length of a run of A units is MINA and the maximum length is MAXA.

We would have:

IF $R' < \text{MINA}$ we must fit another A unit so that

$$^{V}i\,(N_A', N_B', A, R') = C(A, i, N_A') + V_{i+1}(N_A'+1, N_A', A, R'+1)$$

IF $R' = \text{MAXA}$ we must not fit another A unit

$$V_i\,(N_A', N_B', A, R') = \min \begin{cases} C(B, i, N_B') + V_{i+1}(N_A', \; N_B'+1, \; B, \; 1) \\ C(C, i, N_C) + V_{i+1}(N_A', \; N_B', \; C, \; 1) \end{cases}$$

If $\text{MINA} < R' < \text{MAXA}$ we could fit either and A, B or C unit

$$V_i\,(N_A' \; N_B' \; A' \; R') = \min \begin{cases} C(A, \; i, \; N_A') + V_{i+1}\,(N_A'+1, \; N_B', \; A, \; R'+1) \\ C(B, \; i, \; N_B') + V_{i+1}\,(N_A', \; N_B'+1, \; B, \; 1) \\ C(C, \; i, \; N_c') + V_{i+1}\,(N_A', \; N_B', \; C, \; 1) \end{cases}$$

Having established which type of unit we should fit at position i, we update the appropriate optimal sequence, $P_{i+1}(.,\; .,\; .,)$ to include the unit fitted at position i to give $P_i\,(N_A', N_B', A, R')$. This process continues until we arrive at the First position in the sequence. Clearly this has nothing before it, so it must be first in .a sequence, so we have :-

$$\text{Total Cost} = \text{Min} \begin{cases} C(A, \; 1, \; 0) + v_2\,(1, \; 0, \; A, \; 1) \\ C(B, \; 1, \; 0) + v_2\,(1, \; 0, \; B, \; 1) \\ C(C, \; 1, \; 0) + v_2\,(1, \; 0, \; C, \; 1) \end{cases}$$

Once we have determined the type of unit fitted in position one we have the optimal sequence of units from positon one onwards and so the process is finished.

Computational Difficulties

The Dynamic Program described above, although correctly formulated in theory could not infact be used to solve any problem with more than a few units in the sequence. The problem lies in the fact that the four dimensional state space has an enormous number of possible combinations (even discounting the illegal ones), too many to hold on the mainframe computer available.

In addition to this, the time taken for a computer to iterate through all these combinations is very long, long enough to make the solution of a sequencing problem of reasonable length infeasible even with modern day computing power.

To circumvent this problem, the following was adopted. By examining the probability profiles for each type of unit it was possible to make a guess at the sequence of units being careful to include the correct number of each type in the sequence. This initial solution was then put into the computer and the Dynamic Program altered in the following way. At each stage the number of A type units and B type units predicted in the initial solution (I_A and I_B) are calculated and the Dynamic Program is only allowed to change them by +/- 5 units at any stage. So we are in effect imposing the additional constraints that at any position in the sequence we must have:-

$I_A - 5 \leqslant N_A \leqslant I_A +5$

$I_B - 5 \leqslant N_B \leqslant I_B + 5.$

This reduces the state space considerably, but means that the program effectively now makes only local adjustments to the inital sequence. However, by running the program again using the latest prediction asthe input sequence the method can slowly transform the initial prediction into the minimum cost solution.

An Application of the Method

To illustrate the use of the method we consider a problem of biochemical nature - that of predicting the secondary structure of a protein.

Given any protein it is reasonably easy to determine its primary structure, that is the sequence of Amino Acids that are present. There are 20 Amino Acids which may occur in proteins and Figure 5 shows a section of one such protein with its sequence of Amino Acids.

What is of more importance is to be able to use this basic information of the primary structure to predict the higher level structures formed by the protein which are difficult and time consuming to determine in practice (usually by x-ray crystallography).

As a first step the method proposed in this paper can be used to predict the occurance of three secondary structures - the α-helix, the β sheet and the coil (anything not α-helix or β sheet). We are interested in predicting the way the protein adopts these structures as we move along its sequence of Amino

Acids. In othe words we need to predict whether an α, β, or c unit occurs at each position in the sequence. The actual structure formed by the section of protein shown in Figure 4 is given in Figure 5.

In order to do this we need to generate probability profiles for each of the α, β, and c structures for the length of the protein and then apply the Dynamic Programming method.

A number of proteins have been analysed, and their secondary structure is now known. Statistical analysis of these proteins have been reported by Garnier et al [4] and Robson [5] the result of which was the publication of tables showing the likelihood of a particular Amino Acid forming an α, β or c structure at its position in the sequence, and also the effect it will have in promoting α, β, and c structures in the positions eight on either side of it.

Using these tables, and the Amino Acid sequence for a protein, it is possible to construct probability profiles for each of the α, β and c structures, demonstrated in Figure 6.

Two methods of secondary structure prediction already exist using the profiles, those of Robson [5] and Taylor [6],

The method of Robson involves predicting at each point in the sequence the most likely type of unit judged from the profiles. To increase the accuracy of prediction he used the method on the proteins of know secondary structure and then adjusted the

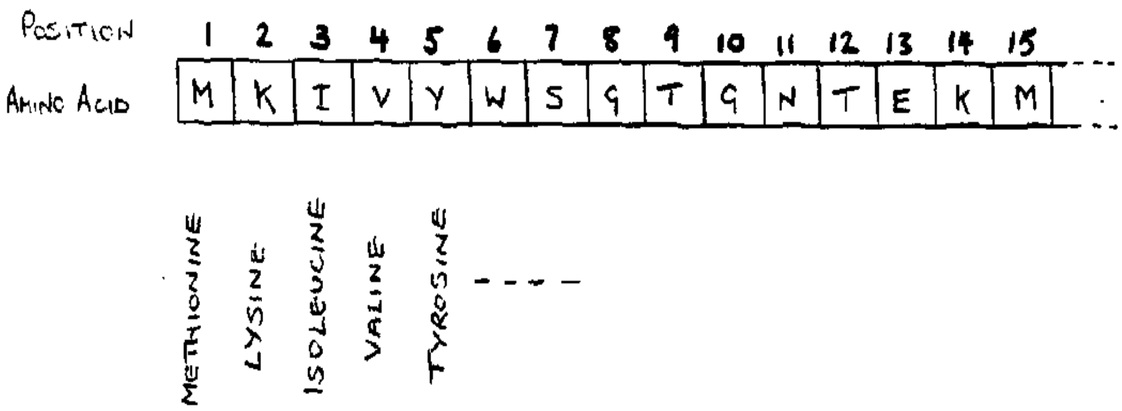
POSITION
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
AMINO ACID
M K I V Y W S G T G N T E K M
METHIONINE
LYSINE
ISOLEUCINE
VALINE
TYROSINE

Fig 5

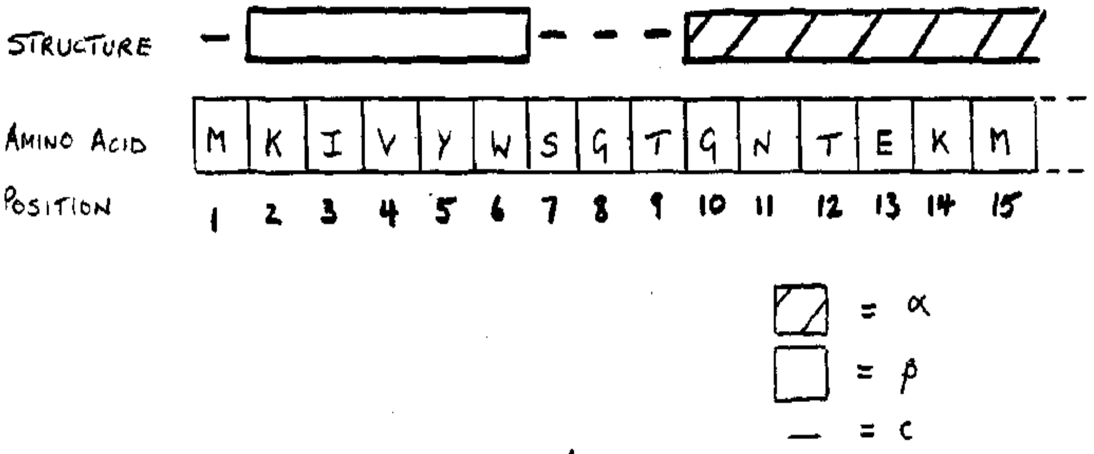
STRUCTURE
AMINO ACID
M K I V Y W S G T G N T E K M
POSITION
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
= α
= β
= c

Fig 6

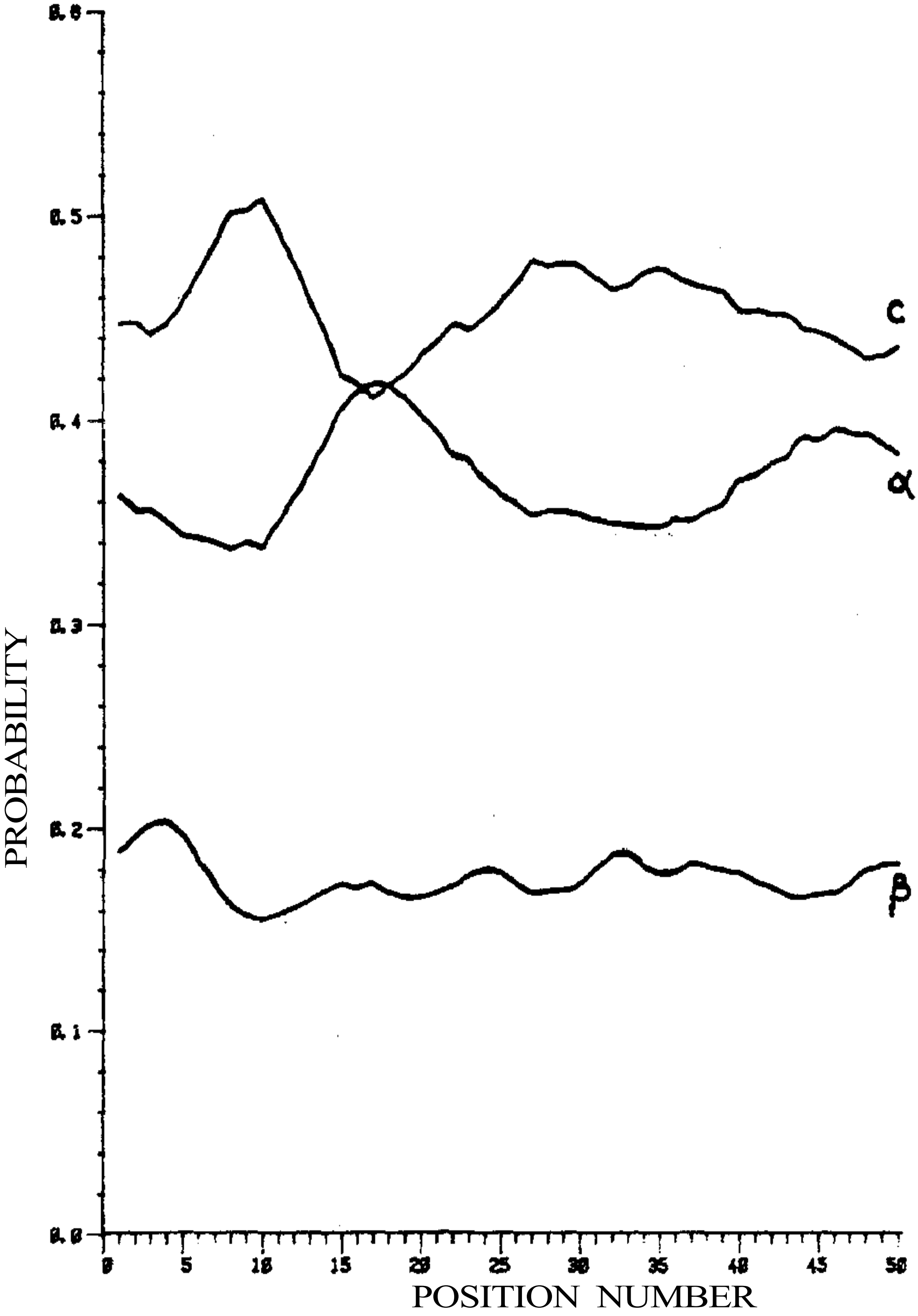
0.6
0.5
0.4
0.3
0.2
0.1
0.0
PROBABILITY
0
5
10
15
20
25
30
35
40
45
50
POSITION NUMBER
c
α
β

Fig 7.

proteins of known secondary structure and then adjusted the profiles by adding a constant to each probability within a profile, so adjusting its height relative to the others. By finding a suitable constant for each profile he obtained the best fit he could against the proteins of known secondary structure.

The method of Taylor stems from his observation that the three types of secondary structure often occur in a form which he termed the β α β structure. This structure took the form of a number of β units followed by c units followed by α units, more c units and finally β units. He analysed proteins to find the average length of runs within the structure so producing a template which he could fit to the profiles. He then fitted this template at points along the sequence, scaling it up or down where appropriate. Each template was fitted so as to maximise the goodness-of-fit (F). F was in fact the areas under each profile in the range of their corresponding template sections. He then raised the profile levels in the region of their corresponding template sections by an amount proportional to F. A secondary structure prediction was then obtained by applying the method of Robson on the new profiles. Thus, Taylors method aims to improve the fit given by Robson. In fact the fitting and scaling of the template is another problem which could well be attacked by Dynamic Programming and is shortly to be investigated.

To illustrate the use of the method described in this paper we consider one protein, part of the probability profiles for which are shown in Figure 7. Analysis of the secondary structure of a number of proteins indicated the maximum and minimum lengths of α, β, and c structures and these together with a guessed sequence of structures were input to the Dynamic Program. The input sequence and successive solutions provided by the method are shown in Figure 8. The final solution and the actual sequence of structures are shown in Figure 9. The method in this case correcly predicted the structure at 72.4% of the positions in the protein sequence .

The percentages correcly predicted by the methods of Robson and Taylor were 60.18 and 81.28 respectively. Using the method on a small number of proteins it was found to be in general as good as the best predictions of Robson but not as good as those of Taylor, whose method is effectively a two stage process. This is an encouraging result since the improvement in Robsons prediction by including Taylors template stage could equally well apply to the method described here (although this has still to be investigated), and the method may be considered better than that of Robson on methodological grounds since it does not require the rather arbitrary use of constants to improve the fit to a satisfactory level .

Although the results generated by the Dynamic Programmming method described here at present show no consistent improvement over other methods, it is a measure of the power of Dynamic Programming that other researchers ideas such as the notion of a template proposed by Taylor could be incorporated into the

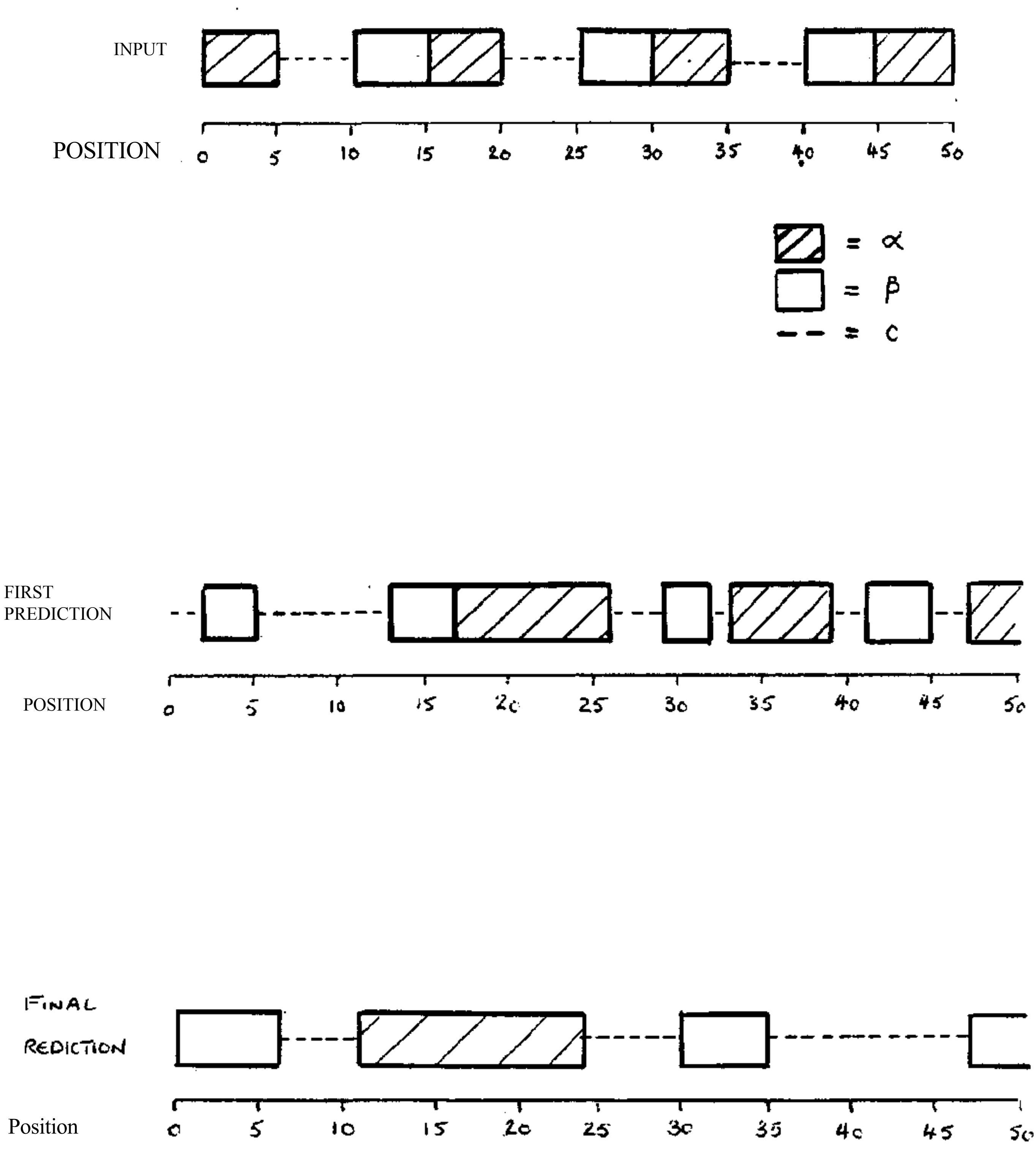
INPUT
POSITION
0 5 10 15 20 25 30 35 40 45 50
= α
= β
= c
FIRST PREDICTION
POSITION
0 5 10 15 20 25 30 35 40 45 50
FINAL
REDICTION
Position
0 5 10 15 20 25 30 35 40 45 50

Fig 8

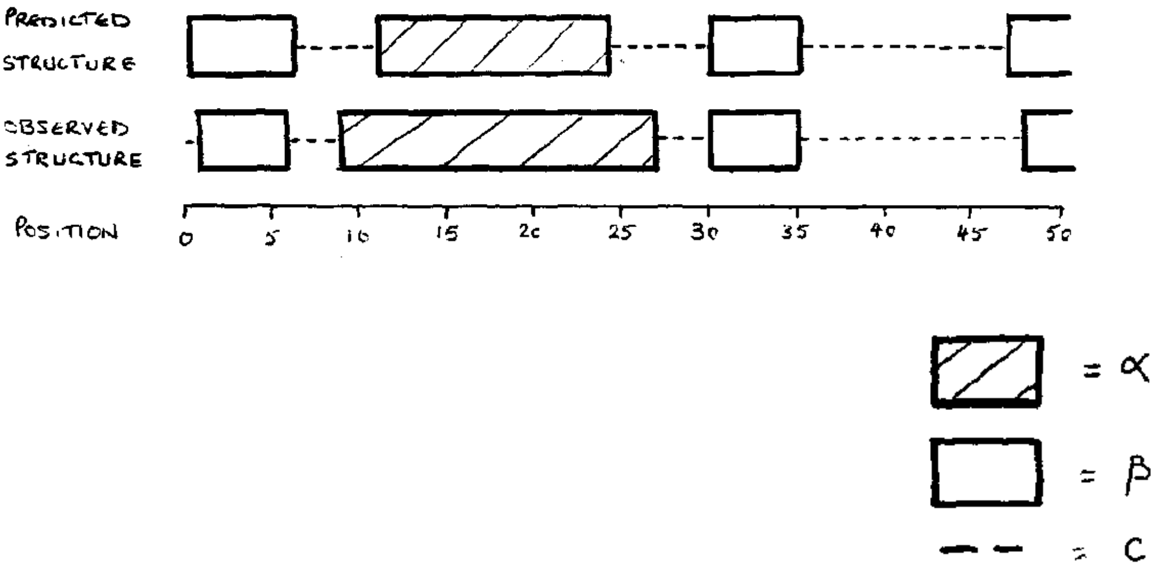
PREDICTED STRUCTURE
OBSERVED STRUCTURE
POSITION
0
5
10
15
20
25
30
35
40
45
50
= α
= β
= C

Fig 9

formulation easily, thereby almost certainly improving the performance of the model.

The importance of this paper lies not so much in the particular application and results presented, but more in demonstrating the ease with which a technique little used in Industrial and Business settings becomes a powerful tool when applied to problems of comparing and matching often found in the field of pattern recognition and A.I.

References

[1] White, D. J., Dynamic Programming, Oliver and Boyd, Edinburgh.

[2] Brennan, J. T., 'Alternative Solution Methods for the Inventory Replenishment Problem under Increasing Demand', J. Opl. Res. Soc., Vol 31, pp 615 - 620.

[3] Russell, M. J. et al., 'Automatic Speech Recognition using Local Timeseale Variability Information', Proc. Inst. Acoustics, November 1982.

[4] Garnier, J. et al., 'Analysis of the Accuracy and Implications of Simple Methods for Predicting the Secondary Structure of Globular Proteins', J. Mol. Biol. (1978), 120, 97-120.

[5] Robson, B. , 'Analysis of the Code Relating Sequence to Conformation in Globular Proteins', Biochem. J. (1974), 141, 853-867.

[6] Taylor, W. R. and Thornton, J. M., 'Prediction of Super Secondary Structure in Proteins, Nature, Vol 310, pp 540-542.