Chapter 4

Sequential Assignment

4.1 Implementation and Practical Considerations

4.1.1 Introduction

With TOCSY or COSY spectra, it is possible to find patterns of peaks corresponding to *types* of amino acid. However, such spectra do not provide information on *where* each of these amino acids fits into the sequence for the protein under study. For this, one needs so-called "backbone" spectra, which contain inter-residue peaks. Assigning chemical shifts to specific spins within specific residues in the protein is known as *sequential assignment*.

A program, called *chain*, has been developed to perform sequential assignment automatically. As input, it requires a set of backbone results files, a pattern file, the sequence of the protein, and a parameter file. Program operation can be broken into two steps:

- **Chaining.** Take the presented backbone results, and try to fit them together in chains following the sequence. In general, many such chains (sequence fragments or "assignments") will be produced, as a result of natural breakpoints in the sequence (eg. at prolines), and also due to ambivalent input data. A set of complete or partial putative sequential assignments for the protein is generated, ordered according to a "goodness" factor.

- **Assignment list processing.** After generation of putative assignments, the sequence fragments are subjected to further processing to reduce the amount of data presented to the user.

A flow diagram for the *chain* program is shown in Figure 4.1.

In the following sections, the requirements for the input data and the design of the two program stages are described in more detail.

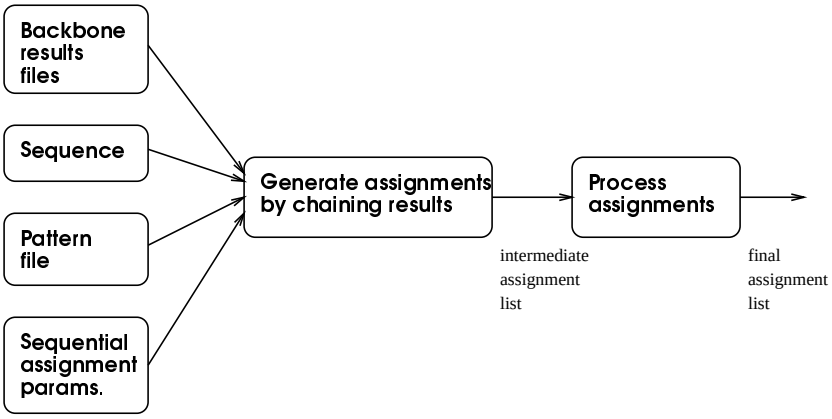


Figure 4.1: **Flow diagram for the sequential assignment program.**

4.1.2 Input Data

Before running *chain*, results files for amino acid pairs, triples or higher tuples, must first be generated by *patt_recog* from a set of backbone spectra (eg. HNCA/HN(CO)CA or NOESY); *chain* cannot do a sequential assignment directly from a spectrum. The purpose of doing this is to find the patterns corresponding to inter-residual connectivities in the spectra.

In addition, *chain* also needs to have the pattern file that was used during the *patt_recog* run which produced the results files. This tells the program the conditions under which the *patt_recog* run was done. Furthermore, the program needs a file specifying the parameters that control the way in which the sequential assignment is performed.

Optionally, one may supply spin system assignment results, e.g. those derived from HCCH-COSY and -TOCSY data. These can be used to impose additional constraints on the chaining process.

4.1.3 Chaining

The central data structure of the *chain* program is the sequence of the protein being assigned. The program steps along the sequence one residue at a time, and attempts to use each position as a starting point for "growing" sequence fragments; these represent putative *assignments*. This process is outlined in Figure 4.2.

The fragment grows from the N-terminus end towards the C-terminus end of the protein. Growth is initiated by looking into the supplied results files, to see if a result can be found to start off the fragment. It must fit into the current position in the sequence to qualify for this. Then a second result is sought, which, to qualify, must fit in the sequence after the first result, *and* have enough chemical shift values in common with the first result to be a reasonable sequential match. This process continues until the fit criterion cannot be satisfied by any of the remaining results.

The reason why many fragments can be grown from the same starting point in the sequence is that many results may fit to this position in the sequence. For each of these results, the program attempts to grow a new sequence fragment. The fragments generated may be of varying lengths. These sequence fragments will correspond to various portions of the entire sequence; breaks may occur as a result of prolines, weak or confusing signals or insufficient input data.

Sequence fragments are stored in a list, ranked according to a score which measures their "goodness" relative to various criteria. When a new fragment has been found by the program, it is fed into the list of assignments, and "bubbled" up until it finds an assignment with a higher score. This means that the assignment list is always ordered according to score.

Sequence fragments are represented by the program internally as a data structure with the following components:

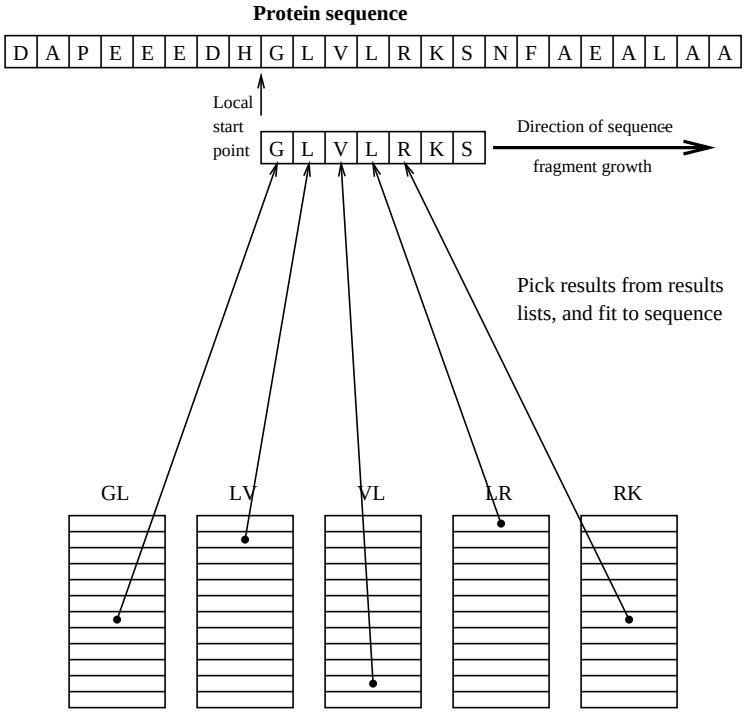


Figure 4.2: **Chaining results together to produce assignments.**
The sequential assignment algorithm steps the "local starting point" along the protein sequence one residue at a time. For each starting point position, a set of sequence fragments are generated, by selecting results from the results lists and fitting them to the sequence and to the chemical shifts of the spins shared with the predecessor result. Each sequence fragment thus generated represents a possible assignment, and is stored in an *assignment list*. The results lists are obtained from searches in backbone spectra for the peaks corresponding to amino acid pairs.

- A copy of the portion of the main protein sequence spanned by the fragment;
- A list of the backbone results which match the residues in the sequence fragment;
- A score.

For a given fragment, this score value is computed as a sum of local scores, one for each *pair* of results in the fragment. These local scores in turn are the products of two components:

1. The score on the *second* of the two results in the pair, normalised to a value between 0 and 1. This is the score given to the result by *patt_recog* during the backbone pattern search.
2. A chemical shift matching factor. This is a value between 0 and 1. The more nearly identical the chemical shifts for the spins shared by the two results, the larger this value will be.

Figure 4.3 shows how the score is calculated for a sequence fragment.

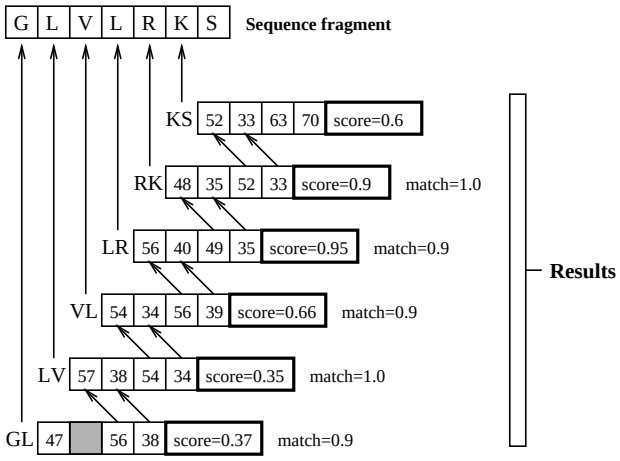
It should be noted with regard to chemical shift matching that it makes no sense to match two chemical shifts which are too far apart. For instance, one would never match a Cα=54 ppm in one result with a Cα=48 ppm in another result. Hence, a cutoff or threshold chemical shift difference will normally be specified to prevent such implausible matches. This means that before a new result is added to a growing fragment, it must first satisfy the condition that the chemical shift differences between the spins that it has in common with the last result in the fragment are less than the threshold values. This has the beneficial sideffect of speeding up the program, since it reduces the size of the search tree.

Results originating in crowded parts of the spectrum may be imperfect, and have some wrong chemical shifts. Therefore, an option exists for making the matching condition less stringent by specifying a maximum mismatch count. This allows some spins to break the matching criterion, yet still be accepted, giving some limited compensation for errors in the original results list. Figure 4.4 shows these ideas graphically.

In order to reduce the memory consumed by the program, the size of the assignment list is limited, so that only the highest scoring assignments are retained.

4.1.4 Assignment List Processing

The sequential assignment process can generate many thousands of potential assignments. To make this list more amenable to manual assessment, two sets of processing algorithm have been implemented:



$$\text{Sequence fragment score} = \frac{(0.6+0.9+0.95+0.66+0.35)}{5} * \frac{(1.0+0.9+0.9+1.0+0.9)}{5}$$

Figure 4.3: **Calculating the score of a sequence fragment.**
In the sequence fragment shown, there are 5 pairs of amino acids, with a corresponding set of 5 results. The *score* values shown for each result are normalised versions of the values produced by the *patt_recog* program. The *match* values shown are measures of chemical shift fit between pairs of results. The chemical shifts shown are notionally for Cα and Cβ spins. The arrows show the fits between pairs of results. The overall score for the fragment is the product of the mean result score and the mean chemical shift match.

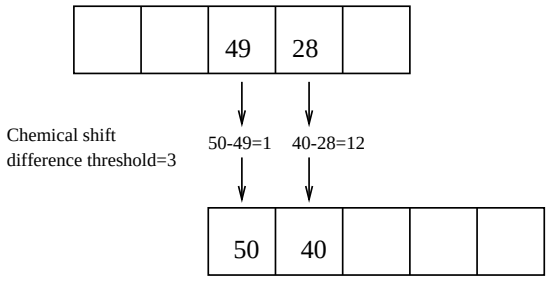


Figure 4.4: **The result matching conditions.**
In this example, two results are shown, where one pair of chemical shifts falls within the allowed chemical shift difference but the other does not. The second result will therefore only be added to the growing fragment if the maximum mismatch count is 1 or greater.

- **Penalisation.** These reorder the list, to emphasise results which are "good" according to various heuristic criteria. This is done by multiplying the score of an assignment by a fractional value between 0 and 1; the nearer this value is to 1, the better the assignment is, according to the current criterion.

- **Deletion.** These algorithms look at an assignment, and decide if it should remain in the list or not - if not, they delete it. They often work by comparing pairs of assignments and deleting the one with the lowest score, so the choice of penalisation algorithms can also affect the operation of the deletion algorithms.

The following special symbols are employed in the definition of the sequence fragment penalisation and deletion algorithms:

Symbol	Meaning
e, f	Indeces to results in a sequence fragment.
E_{frag}	The total number of results in a sequence fragment.
Γ	The set of results assigned to the current sequence fragment.
$\delta_{com(n)}$	The nth spin that a result has in common with other results.
$\underline{n_{com}}$	A list of the spin numbers in a result which are common with other results.
N_{com}	The number of spins that a result has in common with other results.
S	The length of the entire protein sequence.
S_{frag}	The length of a sequence fragment.
s_{start}, s_{end}	Start and end points on a sequence fragment.
Σ	The set of all sequence fragments.

The following penalisation algorithms are available (see also Figure 4.5):

- **Similar residue penalisation.** If two amino acid residues in different parts of the sequence fragment for a given assignment share similar or identical chemical shifts, then it is likely that the program has used near repeats, which are unlikely to arise in real spectra. The more similar residue pairs there are in the spectrum, the smaller the penalisation factor. Symbolically,

$$F = \prod_{e=1}^{E_{frag}} \prod_{f=1}^{E_{frag}} sp(e, f)$$

  where E_{frag} is the number of results in the current sequence fragment and $sp(e, f)$ is a function that derives a penalisation factor from similar results:

$$sp(e, f) = \begin{cases} 1 & \text{if } T(e, f, N_z, \Delta_z) \\ F_s & \text{otherwise} \end{cases}$$

  where F_s is a constant penalisation factor, which will be applied if the penalisation criterion holds, and $T(e, f, N_z, \Delta_z)$ is a logical function which determines if N_z or more spins in result f are within the chemical shift difference Δ_z of the corresponding spins in result e - this function has already been defined in Section 3.1.4 (page 27).

- **Low count penalisation.** This penalises assignments with short sequence fragments - the shorter the number of residues in a sequence fragment, the smaller the penalisation factor. Symbolically,

$$F = \begin{cases} 1 & \text{if } E_{frag} \geq E_{frag(min)} \\ F_s & \text{otherwise} \end{cases}$$

  where F_s is a constant penalisation factor, which will be applied if the penalisation criterion holds, and $E_{frag(min)}$ is the minimum allowable number of results in a sequence fragment.

- **Imperfect connectivity penalisation.** "Perfect" connectivity between two results occurs when the chemical shifts for all shared spins are *identical.* The fewer residues in an assignment for which this is true, the smaller the penalisation factor. Symbolically,

$$F = \prod_{e=1}^{E_{frag}} ipc(e)$$

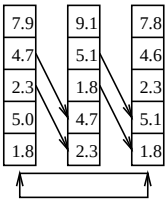
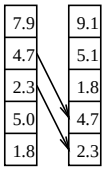
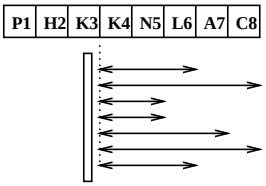
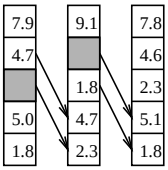
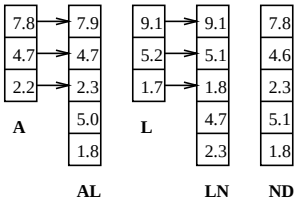
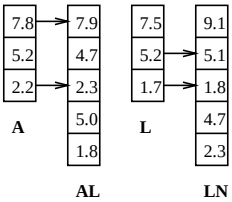
Name	Situation (graphical representation)	Situation (in words)	Action
Similar residue		The results assigned to two different positions in the sequence fragment are very similar.	Penalise according to the count of similar residues.
Low count.		The current sequence fragment contains rather few results.	Apply a penalisation proportional to the number of results in the fragment.
Excessive start point		Too many sequence fragments starting at the same position in the main sequence.	Penalise by a factor inversely proportional to the number of sequence fragments with the same starting pooint.
Uninstantiated spins		Some chemical shift values are missing (uninstantiated), allowing a match to any chemical shift in a subsequent result.	Penalise according to a factor inversely proportional to the number of uninstantiated chemical shifts.
Spin system overlap		Not all of the backbone results associated with the current sequence fragment have matching spin system results from sidechain assignment runs.	Penalise by a factor inveresely proportional to the number of results without matching sidechain results.
Incomplete spin system match		Not all chemical shifts match between backbone results and spin system results.	Penalise by a factor inveresely proportional to the number of non-matching chemical shifts.

Figure 4.5: **Penalisation algorithms for sequential assignment.**

where $ipc(e)$ returns a penalisation factor dependant on whether result e and result $e+1$ have all matching common spins or not:

$$ipc(e) = \begin{cases} 1 & \text{if } \sum_{n=1}^{N} t_{com}(e,n) = N_{com} \\ F_s & \text{otherwise} \end{cases}$$

where

$$t_{com}(e,n) = \begin{cases} 1 & \text{if } n \in \underline{n_{com}} \wedge (\delta_{e(n)} = \delta_{e+1(n)}) \\ 0 & \text{otherwise} \end{cases}$$

- **Excessive start point penalisation.** Penalises assignments if many assignments have a common starting point in the sequence. Symbolically,

$$F_s = S/start(\Sigma_s)$$

where *start* is a function that counts the number of sequence fragments which have the same start point as Σ_s:

$$start(\Sigma_s) = \sum_{t=1}^{S} st(\Sigma_t, \Sigma_s)$$

$st(\Sigma_t, \Sigma_s)$ is a logical function, which determines if Σ_t has the same start point as Σ_s, returning 1 if yes, 0 otherwise.

- **Uninstantiated spins penalisation.** During peak pattern searching with the *patt_recog* program, it is possible to allow the search to ignore some missing peaks, with the result that not all spin chemical shifts will be assigned. If the results making up a sequential assignment contain many such "uninstantiated" spins, this leads to rather doubtful connectivities, since uninstantiated spins can match any other spin. The penalisation factor produced by this algorithm is smaller when the number of uninstantiated spins in the current assignment is large.

  This can be expressed as follows: if F_e is the penalisation for a single result due to uninstantiated spins (as defined for **Uninst-chm-shft** on page 45), then the penalisation for a sequence fragment is:

$$F = \sum_{e=1}^{E_{frag}} Fe/E_{frag}$$

- **Mismatch count penalisation.** Sequence fragments which contain non-matching chemical shifts between results are permissable, if the maximum mismatch count is greater than 1. However, fragments containing many such mismatches are unlikely to be correct. They can be penalised according to the number of mismatches thus:

$$F = B_{mismatch}/B$$

where B is the total number of spins that *should* match between pairs of results, and $B_{mismatch}$ is the count of spins which fail to satisfy the matching condition:

$$B = \sum_{e=1}^{E_{frag}} \sum_{n=1}^{N} M(n)$$

$$M(n) = \begin{cases} 1 & \text{if } n \in \underline{n_{com}} \\ 0 & \text{otherwise} \end{cases}$$

and

$$B_{mismatch} = \sum_{e=1}^{E_{frag}} \sum_{n=1}^{N} M_{mismatch}(e,n)$$

$$M_{mismatch}(e,n) = \begin{cases} 1 & \text{if } n \in \underline{n_{com}} \wedge t(e,e+1,n,\Delta) \\ 0 & \text{otherwise} \end{cases}$$

The arithmetic function $t(e,f,n,\Delta)$ determines whether the chemical shifts for spin n in results e and f are less than chemical shift difference Δ apart, as defined in Section 3.1.4 (page 37).

- **Spin system overlap.** This type of penalisation algorithm uses the degree of overlap between the backbone results in the sequence fragments in the assignment list, and the results produced by *spin system* pattern searching, to re-order assignments in an assignment list.

  The spin system assignments are supplied to the program as a set of results lists. They will in general be the outcome of a pattern search over spectra such as COSY or TOCSY. It is to be expected that these lists will contain many correctly assigned spin systems, but there will also be partially correct spin systems and totally incorrect spin systems. Some spin systems will be missing altogether.

If such a set of results lists is available, the *chain* program will automatically check each *backbone* result in each sequence fragment against the appropriate spin system results lists. The best matching spin system results will be noted, assuming that any fit.

A match factor will be calculated for each sequence fragment, which embodies the degree of chemical shift overlap between the spin system assignments and the corresponding backbone assignments. The poorer the degree of match, the smaller this factor.

The net outcome is that assignments with many overlapping spin system results will be strongly promoted, and will survive the various clustering algorithms. Although this cannot guarantee that these assignments are genuine sequential fragments, it significantly increases the probability that this will be the case, if the sidechain results are good.

The prerequisite for this type of penalisation is that a set of spin system results has already been generated. The more spin system types, the better. Eg. if the spin system results lists for Gly, Ala, Amx, Ser, Glx, Thr, Val, Ile, Pro and Leu are available, then a useful outcome can be expected. If only results for Gly and Glx are available, this kind of penalisation will probably not help very much. Symbolically,

$$F = 1 - E_{smatch}/E_{frag}$$

where E_{smatch} is the number of results in the sequence fragment that also have matching spin system results.

- **Incomplete spin system match.** This penalisation algorithm also requires results lists from spin system searches. It looks at each of the spin system results associated with the backbone results. For each one, it compares the *actual* count of overlapping (or matching) chemical shifts between spin system and backbone with the *expected* count of overlapping chemical shifts. The bigger the difference, the harsher the applied penalisation. This will tend to hit instances where, for example, a spin system result only matches a backbone result for the Cα but not for the Cβ chemical shift values.

The following are all deletion algorithms (see also Figure 4.6):

- **Subsequence.** If one assignment is a simple subsequence of another assignment, delete it.

$$k_s = subs(s, t)$$

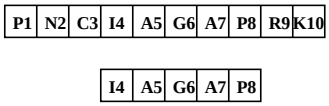
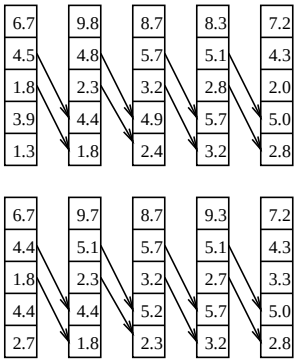
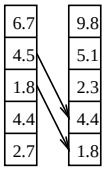
Name	Situation (graphical representation)	Situation (in words)	Action
Subsequence		One sequence fragment is actually a subsequence of a longer fragment.	Delete the shorter of the two fragments.
Secondary cluster		Two sequence fragments contain results with similar chemical shifts.	Delete the lowest scoring fragment.
Too short		There are so few results in the sequence fragment that it is not worth condsidering.	Delete it.

Figure 4.6: **Deletion algorithms for sequential assignment.**

where $subs(s,t)$ determines if sequence fragment s is a subsequence of sequence fragment t:

$$subs(s,t) = ids(s,t,e+1,2) \wedge (1 \leq e \leq E_{frag(s)}) \wedge (\Gamma_{se} = \Gamma_{t1})$$

and $ids(s,t,e,f)$ is a recursive function which determines if the subsequence of s starting at result number e is identical to the subsequence of t starting at result number f:

$$ids(s,t,e,f) = \begin{cases} 1 & \text{if } e == E_{frag(s)} \\ ids(s,t,e+1,f+1) & \text{if } \Gamma_{se} == \Gamma_{tf} \\ 0 & \text{otherwise} \end{cases}$$

- **Secondary cluster.** This kind of clustering compares sequence fragments pairwise, to determine the degree of chemical shift overlap. If there is more than a given percentage of similar sequential results common to both sequence fragments, then the lowest scoring sequence fragment is deleted.

$$k_s = secl(s,t)$$

where $secl(s,t)$ determines if sequence fragment s can be clustered with sequence fragment t:

$$\begin{aligned} secl(s,t) \quad = \quad & T_{secl}(s,t,e,1,N_l,\Delta_l) \wedge T_{secl}(s,t,e,1,N_u,\Delta_u) \wedge \wedge \\ & cls(s,t,e+1,2) \wedge (1 \leq e \leq E_{frag(s)}) \end{aligned}$$

where $T_{secl}(s,t,e,f,N_l,\Delta_l)$ and $T_{secl}(s,t,e,f,N_u,\Delta_u)$ are defined in a similar way to the functions $T(e,f,N_l,\Delta_l)$ and $T(e,f,N_u,\Delta_u)$ given in Section 3.1.4 on page 37, except in this case, we are making comparisons between two different results sets, those for sequence fragments s and t.

$cls(s,t,e,f)$ is a recursive function which determines if the subsequence of s starting at result number e is identical to the subsequence of t starting at result number f:

$$cls(s,t,e,f) = \begin{cases} 1 & \text{if } e == E_{frag(s)} \\ cls(s,t,e+1,f+1) & \text{if } T_{secl}(s,t,e,f,N_l,\Delta_l) \wedge T_{secl}(s,t,e,f,N_u,\Delta_u) \\ 0 & \text{otherwise} \end{cases}$$

- **Too short.** All assignments containing fewer than or equal numbers of results to the specified factor will be deleted. Symbolically:

$$k = E_{frag} < E_{min}$$

Sequence fragments will accumulate in the assignments list as the program proceeds along the sequence. By default, processing will be performed only once, after all residues in the sequence have been visited. However, if the sequence is lengthy, then large numbers of assignments will be generated, and the program will soon hit the maximum assignments list limit. As a consequence, potentially good assignments may be lost, since there will not be room in the list for them. A feature has therefore been included to allow the user to tell the program to do assignment list processing at *every* point along the sequence, if required. In this case, the deletion algorithms will ensure that the intermediate assignment list remains small, and the danger of list overflow will thereby be significantly reduced.

After processing, the final list of assignments is deposited in an ASCII file, where it can be browsed by the user.

4.2 Experimental

The *chain* program is able to take the results obtained from pattern searches in backbone spectra, and to string them together according to chemical shift values, to derive a sequential assignment.

The first attempt to do this was made using the data presented in Section 3.5.2 (page 87). The outcome was not very satisfactory; the connectivities found by the program showed many errors. Often, several results would be correctly chained together, but assigned to the wrong positions in the sequence, due to the other results included into the fragment. Investigation showed that this was caused by the pattern used to generate these results, which was tailored to producing results for human evaluation. The number of results produced by backbone pattern searches is usually large - generally several thousand. One of the principal concerns when presenting data to a user is to reduce this number to a quantity which could reasonably be dealt with - no more than 1.5 times the sequence length. Hence, in the above mentioned test, secondary clustering was aggressively applied, with the consequence that at least 20% of the results contained either i) a wrong chemical shift or ii) uninstantiated chemical shifts.

In case i), this can lead the sequential assignment algorithm to make incorrect choices about the successor to the current residue during the chaining process; in the case of ii) the choice of possible successors to the current residue may be very large, since an uninstantiated chemical shift can be matched with anything.

In order to overcome these problems, it was decided to experiment with using patterns containing only very limited secondary clustering. The hypothesis was the following: if the correct results are in the results list produced by the pattern search, even if this is a very large results list, then the *chain* program should find correct sequential assignment fragments, because the correct fragments should be the longest and have the best matching chemical shifts, and hence score very highly.

Only the XX pattern was used in this test, to simplify things. This means, of course, that all possible connections through glycines will be missed. The pattern is shown in Figure 4.7. A *patt_recog* run generated 4714 results using this pattern. Rather than reproducing the whole results set here, a selected subset will be presented, shown in Figures 4.8 and 4.9. The results in this list have been identified as being 100% correct by comparing with the list of manually found assignments. They have been ordered according to position in the sequence rather than score, and it can easily be seen that two reasonably long subsequences could, in principle, be found by *chain*: i) from R13 to A22 and ii) from A67 to Y77.

The results from this run were fed into *chain*, along with the chaining parameter set shown in Figure 4.10. This parameter set was the outcome of a number of previous experiments, whose main purpose was to determine the usefulness of the various penalisation and deletion algorithms. The following assumptions are thus built into the parameter set:

- Rewarding sequence fragments starting with a proline did little to improve the assignments, probably because of the number of breaks in the sequence data extracted from the XX run results list;
- Penalising sequence fragments with few chemical shifts in common with results obtained from the spin system searches reported in Section 3.5.1 (page 87). also produced no significant improvement in performance. This came as something of a surprise, because it was assumed that this data would increase the probability of finding correct sequence fragments. However, since, on average, about 50% of the sidechain results are incorrectly assigned, these results are as likely to cause a correct fragment to be penalised as an incorrect fragment. This is something which could be improved upon in future, eg. by using the scores on the sidechain results to decide how significant they are;
- Penalisation of imperfect connectivity was a powerful tool for selecting correct sequence fragments.

The program was run on a Silicon Graphics MIPS R8000 processor machine; execution time was about 40 minutes. The full set of sequence fragments assigned by the program is presented in Appendix D.

A total of 20 sequence fragments were found, numbered 0 to 19. Two of these were substantially correct (these are shown in Figure 4.11); furthermore,

Spectrum size parameters:

	HNCO			CBCANH		
	w1	w2	w3	w1	w2	w3
Size (data pts)	128	128	512	128	128	512
Sub-block (data pts)	16	4	16	4	4	16
Sweep width (ppm)	39.460	19.878	6.847	39.456	80.813	6.847
Offset (ppm)	98.870	167.561	4.810	98.131	5.765	4.810

Results list processing parameters:

Sec-clus: N_l=6 Δ_l=0.00 Δ_u=1.00
Fract-score: ϵ_{low}=0.030
Exs-penal: M_x=3
Excess-peaks: $P_{expected}$=4 r_l=13000
Uninst-chm-shft: Active.

Chemical shift ranges:

Resid.	N	H	CO	Cα	Cβ
X	106.5→133.5	6.00→10.55	167.6→187.4	45.0→66.6	13.0→72.3

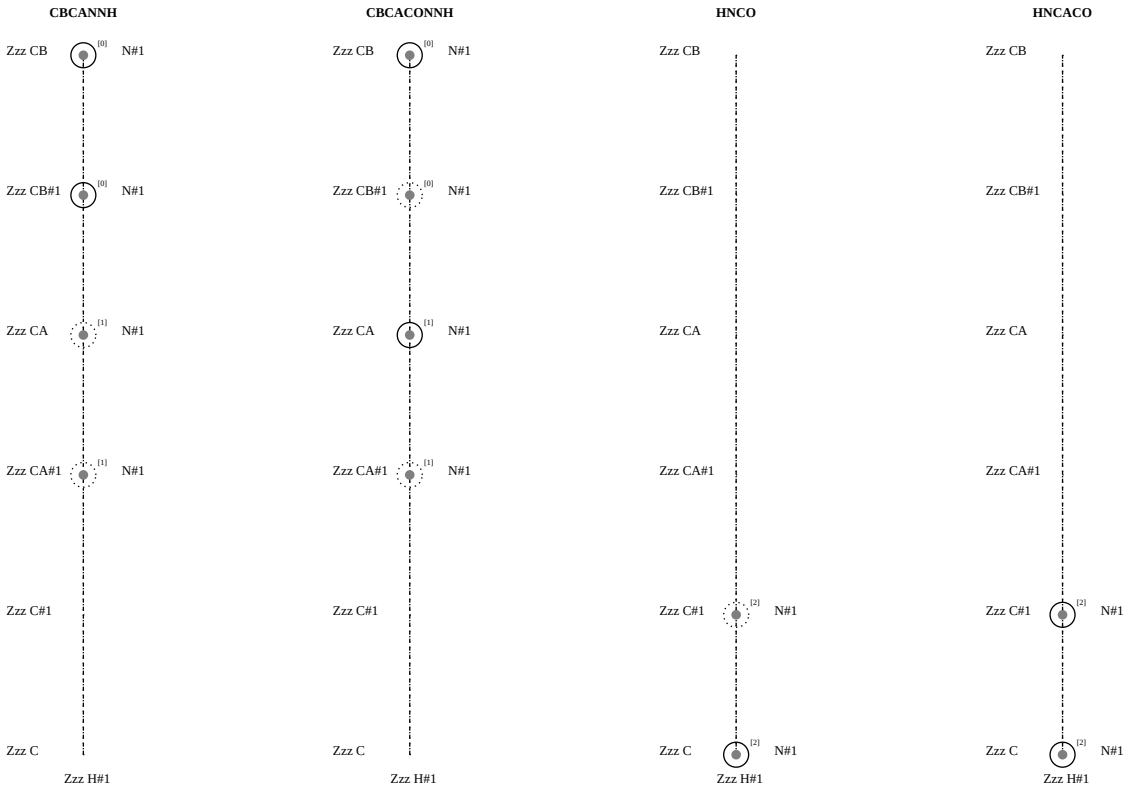


Figure 4.7: **XX pattern used in the sequential assignment test.**

H#1	N#1	CB#1	CB	CA#1	CA	CO	CO#1	Resp	Orig	Assignment
8.46	**127.4**	**17.8**	**41.8**	**51.2**	**55.0**	**175.2**	**175.3**	843355	8433555	A2
8.71	**124.0**	**30.4**	**32.3**	**57.5**	**63.2**	**176.9**	**175.5**	375364	3753645	E4
8.37	**121.6**	**33.5**	**31.0**	**55.6**	**57.5**	**175.5**	**175.8**	161320	1613205	E5
8.43	**125.9**	**31.7**	**43.6**	**63.2**	**56.3**	**175.6**	**175.2**	71139	811865	V11
9.00	**119.1**	**34.8**	**44.3**	**54.4**	**53.7**	**174.4**	**176.3**	156492	2079645	R13
9.09	**124.3**	**31.7**	**34.8**	**61.3**	**54.4**	**176.3**	**178.4**	191890	1918905	K14
8.18	**110.2**	**63.8**	**31.7**	**60.7**	**61.3**	**178.4**	**175.3**	158969	2483905	S15
7.63	**118.8**	**39.9**	**63.8**	**52.5**	**60.7**	**175.3**	**177.7**	124466	1944795	N16
8.01	**124.6**	**40.5**	**39.9**	**63.8**	**53.1**	**176.9**	**176.3**	143649	1436495	F17
8.86	**120.3**	**17.8**	**40.5**	**55.6**	**63.8**	**176.3**	**181.4**	216194	2161945	A18
8.23	**120.3**	**29.1**	**17.8**	**60.1**	**55.6**	**181.2**	**178.7**	289592	2895925	E19
7.83	**124.6**	**19.0**	**29.1**	**55.6**	**60.1**	**178.7**	**179.7**	196252	1962525	A20
7.63	**119.1**	**42.4**	**19.0**	**57.5**	**55.6**	**179.5**	**177.7**	181606	1816065	L21
7.18	**117.6**	**17.8**	**42.4**	**53.7**	**57.5**	**177.7**	**179.1**	95421	1517645	A22
8.33	**124.6**	**32.9**	**30.4**	**60.1**	**55.0**	**172.2**	**175.0**	163900	1639005	K25
8.25	**116.6**	**42.4**	**32.9**	**55.6**	**59.4**	**175.0**	**173.5**	159290	1592905	Y26
9.80	**128.0**	**34.8**	**44.9**	**62.6**	**54.4**	**174.7**	**173.2**	113094	1130945	V29
9.04	**122.2**	**42.4**	**40.5**	**55.6**	**57.5**	**175.6**	**169.9**	124119	1416485	Y32
8.76	**117.9**	**34.2**	**30.4**	**64.5**	**59.4**	**177.7**	**176.3**	67878	902045	C39
7.54	**120.9**	**18.4**	**32.3**	**55.0**	**60.1**	**178.7**	**179.2**	182323	2080725	A41
7.40	**122.2**	**15.2**	**43.0**	**57.5**	**56.9**	**177.7**	**176.7**	133685	1525655	A43
7.07	**119.7**	**30.4**	**31.0**	**58.8**	**66.4**	**180.6**	**177.7**	149943	1499435	E45
8.76	**125.6**	**39.2**	**30.4**	**63.2**	**58.8**	**177.7**	**177.3**	113727	1137275	Y46
7.89	**119.1**	**18.4**	**39.2**	**55.6**	**63.2**	**177.3**	**180.9**	180967	2065255	A47
7.99	**122.2**	**32.9**	**18.4**	**60.7**	**55.0**	**180.9**	**180.0**	208475	2084755	K48
8.30	**124.6**	**17.8**	**32.9**	**55.6**	**60.1**	**180.0**	**178.3**	204645	2046455	A49
7.94	**120.6**	**41.8**	**32.3**	**58.2**	**60.1**	**178.3**	**178.3**	119637	1365335	L53
7.77	**120.9**	**32.3**	**41.8**	**59.4**	**58.8**	**178.9**	**180.5**	182825	1828255	K54
8.03	**124.0**	**17.8**	**32.3**	**55.6**	**59.4**	**180.5**	**179.4**	208714	2087145	A55
7.51	**115.4**	**30.4**	**17.8**	**56.3**	**55.0**	**179.4**	**177.0**	208750	2087505	E56
9.25	**125.9**	**43.6**	**32.3**	**55.6**	**52.5**	**173.2**	**175.8**	118784	1187845	L62
9.50	**122.2**	**26.0**	**43.6**	**50.0**	**55.6**	**175.8**	**176.4**	124486	1420675	A63
9.57	**122.5**	**36.1**	**26.0**	**55.0**	**50.0**	**176.4**	**172.4**	146882	1468825	K64
8.78	**131.1**	**20.9**	**38.6**	**53.7**	**52.5**	**175.8**	**177.0**	127510	1275105	A67
8.65	**110.5**	**68.9**	**20.9**	**64.5**	**53.7**	**176.9**	**175.8**	98940	1545955	T68
6.92	**123.4**	**31.0**	**68.9**	**57.5**	**63.8**	**175.8**	**177.2**	88981	1390335	E69
8.33	**121.2**	**27.2**	**31.7**	**53.1**	**57.5**	**177.3**	**176.9**	151543	1729455	E70
7.54	**116.9**	**63.2**	**27.2**	**62.0**	**53.1**	**176.9**	**177.5**	110418	1725295	S71
8.51	**122.8**	**39.9**	**63.2**	**57.5**	**62.0**	**177.5**	**179.1**	112983	1765365	D72
7.85	**122.2**	**43.6**	**40.5**	**58.2**	**58.2**	**178.9**	**178.0**	337209	3372095	L73
7.97	**120.3**	**18.4**	**43.6**	**55.6**	**58.2**	**179.2**	**179.4**	180102	1801025	A74
8.22	**117.9**	**28.5**	**18.4**	**59.4**	**55.6**	**179.4**	**180.0**	277499	2774995	Q75
8.13	**122.8**	**27.9**	**27.9**	**59.4**	**59.4**	**180.0**	**177.3**	341676	3416765	Q76
7.12	**114.2**	**38.0**	**27.9**	**59.4**	**59.4**	**177.3**	**175.0**	302068	3020685	Y77
9.82	**132.0**	**39.9**	**72.7**	**60.7**	**64.5**	**172.7**	**174.7**	48261	754085	I85
9.41	**130.5**	**41.8**	**42.4**	**57.5**	**55.6**	**173.5**	**173.9**	117768	1177685	F88
8.97	**123.4**	**33.5**	**41.8**	**53.7**	**57.5**	**173.9**	**176.6**	168285	1682855	R89
9.95	**124.3**	**38.0**	**33.5**	**55.0**	**53.7**	**176.6**	**175.6**	111818	1118185	N90
8.37	**117.9**	**70.2**	**41.1**	**63.8**	**54.4**	**177.2**	**175.2**	137714	2151795	T93
8.37	**125.3**	**20.3**	**70.2**	**52.5**	**63.8**	**175.3**	**178.4**	122503	2184445	A94
7.91	**113.5**	**63.8**	**19.7**	**56.3**	**53.7**	**178.6**	**172.2**	127631	2275895	S95
9.06	**126.2**	**31.0**	**34.8**	**56.9**	**55.0**	**176.4**	**175.6**	147943	1479435	E98
8.49	**128.0**	**39.2**	**31.0**	**58.8**	**56.9**	**175.5**	**176.1**	92613	1056935	Y99
7.97	**117.9**	**31.0**	**29.8**	**55.6**	**56.3**	**175.0**	**177.8**	184820	1848205	E104
9.84	**126.5**	**19.7**	**31.7**	**56.9**	**55.6**	**178.0**	**179.2**	111324	1113245	A105
8.92	**115.1**	**40.5**	**19.7**	**58.2**	**57.5**	**179.2**	**178.3**	265949	2659495	D106
7.51	**122.5**	**39.2**	**41.1**	**65.7**	**58.2**	**178.6**	**178.1**	97885	1117095	I108

Figure 4.8: **The correctly assigned results from the XX pattern used in the automated assignment test, part 1. Continued in Figure 4.9.**

H#1	N#1	CB#1	CB	CA#1	CA	CO	CO#1	Resp	Orig	Assignment
8.10	**118.5**	**32.3**	**39.2**	**68.3**	**65.7**	**178.1**	**177.8**	127193	1271935	V109
8.27	**123.1**	**30.4**	**38.0**	**62.6**	**56.9**	**177.8**	**179.4**	130105	1301055	W111
8.39	**119.7**	**43.0**	**30.4**	**58.2**	**62.6**	**179.2**	**180.0**	114261	1303985	L112
8.47	**120.3**	**32.3**	**43.0**	**60.1**	**58.2**	**180.0**	**179.2**	114026	1140265	K113
7.43	**118.8**	**32.3**	**32.3**	**58.2**	**60.7**	**179.1**	**178.1**	212047	2120475	K114
7.42	**117.6**	**31.0**	**32.3**	**56.3**	**58.2**	**178.3**	**176.6**	177332	1773325	R115
7.22	**108.6**	**70.8**	**31.7**	**62.0**	**55.6**	**176.4**	**174.4**	132235	2745765	T116
8.39	**125.6**	**19.0**	**32.3**	**52.5**	**63.8**	**176.9**	**176.4**	969135	9691355	A119
7.75	**129.9**	**20.3**	**19.7**	**54.4**	**52.5**	**176.4**	**182.6**	1266664	12666645	A120

Figure 4.9: **The correctly assigned results from the XX pattern used in the automated assignment test, part 2. Continued from Figure 4.8.**

Two chemical shift values were considered "close" if they were less than or equal to 1.4 ppm apart. It was assumed that every point along the sequence, except for residues preceding a proline, was a valid starting point for chain growth. Chain growth was automatically halted if a proline was encountered. The following tables of penalisation and deletion parameters utilise the notation introduced in Chapter 3.

Penalisation algorithms:

Similar residue	F_s=0.5
Imperfect connectivity	true
Uninstantiated spins	true
Low count	F_s=0.7
Excessive start point	true

Deletion algorithms:

Too short	E_{min}=3
Subsequence	true
Secondary cluster	$E_{frag(s)}$=1.0, N_u=0.8*E_u, Δ_l=0.5, Δ_u=70

Figure 4.10: **Chaining parameter set.**

Putative sequential assignment 14: score=2689324, orig_score=19340730, 9 results
Sequence fragment: D7 H8 V9 L10 V11 L12 R13 K14 S15 N16
Penalisations: low_count=0.70 imperfect_connectivity=0.21 excessive_start_pt=0.97

H#1	N#1	CB#1	CB	CA#1	CA	CO	CO#1	Assignment
7.93	120.6	32.3	32.3	64.5	60.7	175.0	176.3	53
8.39	125.6	32.3	32.3	52.5	63.8	176.9	176.4	119
9.95	124.3	38.0	33.5	55.0	53.7	176.6	175.6	90
7.98	124.9	32.9	39.9	53.1	53.1	175.0	173.2	17
9.25	125.9	43.6	32.3	55.6	52.5	173.2	175.8	62
8.98	119.4	34.8	44.3	54.4	53.7	176.3	176.3	13
9.09	124.3	31.7	34.8	61.3	54.4	176.3	178.4	14
8.18	110.2	63.8	31.7	60.7	61.3	178.4	175.3	15
7.63	118.8	39.9	63.8	52.5	60.7	175.3	177.7	16

Putative sequential assignment 15: score=2914289, orig_score=23394866, 11 results
Sequence fragment: D66 A67 T68 E69 E70 S71 D72 L73 A74 Q75 Q76 Y77
Penalisations: low_count=0.89 imperfect_connectivity=0.14 excessive_start_pt=0.99

H#1	N#1	CB#1	CB	CA#1	CA	CO	CO#1	Assignment
8.38	125.3	19.0	32.3	52.5	63.8	171.0	174.2	119
8.66	110.5	68.9	20.9	64.5	53.7	174.7	175.9	68
6.92	123.4	31.0	68.9	57.5	63.8	175.8	177.2	69
8.33	121.2	27.2	31.7	53.1	57.5	177.3	176.9	70
7.54	116.9	63.2	27.2	62.0	53.1	176.9	177.5	71
8.51	122.5	39.9	63.2	58.8	62.0	177.5	179.1	72
7.85	122.2	43.6	40.5	58.2	58.2	178.9	178.0	73
7.95	120.0	18.4	43.6	55.6	58.2	176.7	181.5	74
8.22	117.9	28.5	18.4	59.4	55.6	181.1	179.2	75
8.13	122.8	27.9	27.9	59.4	59.4	180.0	177.3	76
7.12	114.2	38.0	27.9	59.4	59.4	177.3	175.0	77

Figure 4.11: **Two sequential assignment fragments.**
Each fragment starts with three summary lines: i) The first line gives the position of the fragment in the assignment list, the score and the "original" score before any penalisations were applied; ii) The second line shows the residues putatively found, in single letter code notation; iii) The third line shows the penalisation algorithms which were activated by the fragment and the severity of the penalisation applied in each case. After the summary lines, the results that were chained together to make the fragment are shown, in the order they appear in the fragment. The chemical shift connectivities are always fairly clear; take CB, for example: for a given result, the CB entry in column 4 of the chemical shift table should match fairly closely to the CB in column 3 of the result above.

they score highly, indicating that the penalisation and deletion algorithms have been effective in promoting good assignments and deleting poor assignments. These two fragments correspond substantially with the two long subsequences expected (ie. R13 to A22 and A67 to Y77). It is concluded that it was justifiable to use input data (from *patt_recog*) to which secondary clustering has *not* been applied. With the addition of further spectra containing other nuclei, such as Hα or Hβ, additional constraints would be obtained, which should produce a significant improvement in the quality of the assignments. Using spectra which have been obtained under consistent conditions would also help to improve the outcome of sequential assignment.