Journal of Biomolecular NMR, 4 (1994) 385–410
ESCOM

J-Bio NMR 184

Evaluation of an algorithm for the automated sequential assignment of protein backbone resonances: A demonstration of the connectivity tracing assignment tools (CONTRAST) software package

John B. Olson Jr. and John L. Markley*

Department of Biochemistry, College of Agriculture and Life Sciences, University of Wisconsin-Madison, 420 Henry Mall, Madison, WI 53706, U.S.A.

Received 25 May 1993
Accepted 22 November 1993

Keywords: Automated assignment; Proteins; Software

SUMMARY

The peptide sequential assignment algorithm presented here was implemented as a macro within the CONnectivity TRacing ASsignment Tools (CONTRAST) computer software package. The algorithm provides a semi- or fully automated global means of sequentially assigning the NMR backbone resonances of proteins. The program's performance is demonstrated here by its analysis of realistic computer-generated data for III^{Glc}, a 168-residue signal-transducing protein of *Escherichia coli* [Pelton et al. (1991) *Biochemistry*, **30**, 10043–10057]. Missing experimental data (19 resonances) were generated so that a complete assignment set could be tested. The algorithm produces sequential assignments from appropriate peak lists of nD NMR data. It quantifies the ambiguity of each assignment and provides ranked alternatives. A 'best first' approach, in which high-scoring local assignments are made before and in preference to lower scoring assignments, is shown to be superior (in terms of the current set of CONTRAST scoring routines) to approaches such as simulated annealing that seek to maximize the combined scores of the individual assignments. The robustness of the algorithm was tested by evaluating the effects of imposed frequency imprecision (scatter), added false signals (noise), missing peaks (incomplete data), and variation in user-defined tolerances on the performance of the algorithm.

INTRODUCTION

The development of homonuclear ^1H 2D NMR has allowed the proton NMR resonances of many small proteins (M_r < 10 kDa) to be unambiguously assigned through use of intraresidue through-bond J-coupling connectivities and interresidue through-space NOE connectivities (Wüthrich, 1986). The process of assigning the resonances of a protein is one of the most time-

*To whom correspondence should be addressed.

consuming and tedious steps in the analysis of proteins by NMR, but it is a necessary prerequisite to further NMR analysis. The assignment process often requires simultaneous consideration of multiple pathways and careful bookkeeping to distinguish among diverging connectivity paths. It is not surprising that there has been great interest in developing methods to automate the procedure.

Semi-automated approaches for obtaining assignments from homonuclear ^{1}H 2D NMR data for small proteins (M_r < 10 kDa) have required what can only be assumed to be clean, precise data sets and have met with modest success (Billeter et al., 1988; Cieslar et al., 1988; Eads and Kuntz, 1989; Kraulis, 1989; Weber et al., 1989; Catasti et al., 1990; Pfändler and Bodenhausen, 1990; Van de Ven, 1990; Eccles et al., 1991; Kleywegt et al., 1991). These methods are less efficient for larger proteins (M_r > 10 kDa), owing to their increased spectral overlap. The recent introduction of multinuclear 2D (Markley, 1989), 3D and 4D NMR techniques (Clore and Gronenborn, 1991; Bax and Grzesiek, 1993) has extended the size of proteins that can be analyzed by NMR. These methods reduce the spectral overlap and increase the number of correlations that can be made among peaks from different spectra, thus reducing the amount of ambiguity in the assignment process.

Several automated assignment procedures have been developed for use with specific sets of 3D or 4D data (Cieslar et al., 1990; Ikura et al., 1990; Vuister et al., 1990; Oschkinat et al., 1991; Bernstein et al., 1993). These procedures are tied to particular kinds of experimental data, and all require the sequence of the protein as input. For example, the procedures of Cieslar et al. (1990) and Oschkinat et al. (1991) were developed for use with 3D homonuclear data. The ALFA program developed by Bernstein and co-workers (1993), which uses 3D TOCSY-HMQC and 3D NOESY-HMQC (Fesik and Zuiderweg, 1988; Marion et al., 1989) data, employs a method of combinatorial minimization to order spin systems according to the primary sequence of the protein, a list of spin systems classified by residue type, a list of potential sequential neighbors, and optional structural information. The procedure of Ikura et al., which uses data from the 3D TOCSY-HMQC experiment (Fesik and Zuiderweg, 1988; Marion et al., 1989) and several triple-resonance experiments, i.e., HNCA, HNCO, HCACO, HCA(CO)N (Kay et al., 1990), and HN(CO)CA (Bax and Ikura, 1991), traces unambiguous pathways along the backbone. It stops at positions with multiple possibilities for tracing a connectivity and relies on the user's ability to identify spin systems in the assigned fragments and to match them with the known sequence of the protein (Ikura et al., 1990). Recently the trend has been to use more powerful 3D and 4D experiments in order to simplify the assignment problem, and several simple automated or automatable procedures have been suggested (Ikura et al., 1990; Campbell-Burk et al., 1992; Logan et al., 1993; Seip et al., 1993). As these experimental techniques allow larger and larger proteins to be analyzed by NMR, more ambiguity will be introduced into the assignment problem by the greater spectral overlap and greater linewidths. More sophisticated assignment algorithms will be needed to cope with these difficulties.

Ideally, assignment algorithms should be insensitive to the presence of false peaks (noise), missing peaks (gaps), imprecision in resonance frequencies (scatter), and the degree of overlap in the data. The CONTRAST (CONnectivity TRacing ASsignment Tools) package has been designed with these problems in mind. The program makes it possible to test the dependence of assignment algorithms on these factors. Perhaps the most important features in the design of the CONTRAST package described here are that: (i) it can be adapted to make use of any set of multidimensional NMR experiments (2D, 3D, 4D,...); (ii) its results can be used to check the

primary sequence of the protein, since the program does not require this information as input; (iii) it can accommodate different filters and scoring routines; and (iv) it evaluates the ambiguity of each resonance that it assigns and each pairing that it makes and provides ranked alternatives that can be considered manually in cases of high ambiguity.

Not all combinations of experiments can be expected to yield complete assignments. No matter how good an algorithm is, it cannot create an assignment if the necessary information is lacking. The generality of the CONTRAST routines and the flexibility of the macro approach allow the program to be used even when the quantity or quality of input data provides only partial assignments. The program also allows the use of redundant data to increase the certainty of the assignments. Since the output of the CONTRAST package gives an indication of the amount of ambiguity in each assignment, this alerts the user to situations where the input data contain insufficient information for unambiguous assignment.

The CONTRAST sequential backbone assignment (SBA) algorithm presented here consists of two stages: (i) In the first stage, overlapping fragments of connectivities are built up by starting from a designated *source spectrum* and by adding connectivities to it from additional data sets until all of the available data sets have made their contributions. Connectivities are added in the order of increasing ambiguity, in order to minimize the number and length of false pathways that get explored. (ii) In the second stage, these fragments are ordered by maximizing the degree of overlap between adjacent fragments: a process which corresponds to sequentially ordering the residues of the protein. Since the functions used to accomplish these two objectives are all adaptable, and since the use of the macro language allows different functions to be inserted or omitted, the algorithm can be easily altered to work with other data sets or assignment strategies. The Methods section describes the CONTRAST program and some of the functions used to implement the CONTRAST SBA algorithm. An analysis of the effectiveness of the algorithm as implemented here is included in the Results and Discussion section.

METHODS

The CONTRAST program is written in standard ANSI 'C', and it currently runs under UNIX on Silicon Graphics Iris workstations. The program (Olson and Markley, 1993) is compiled from over 20 000 lines of code, and it accesses a 50-page on-line manual. Input for the program is in the form of ASCII lists of peak frequencies and intensities and ASCII macro files, which are tailored to the set of data to be analyzed.

The published assignment table for the 168-residue protein III^{Glc} (Pelton et al., 1991) was used to generate 'data sets' to simulate peak-picked files from HNCO, HNCA, HCACO, HCA(CO)N (Kay et al., 1990), HN(CO)CA (Bax and Ikura, 1991), and TOCSY-HMQC (Fesik and Zuiderweg, 1988; Marion et al., 1989) experiments. This group of experiments is the same as that used by Ikura et al. (1990) for their semi-automated assignment procedure. The published data for III^{Glc} did not provide assignments for some of the ^{1}H, ^{13}C and ^{15}N atoms in the molecule. In order to complete the set of test data, the missing chemical shifts were generated artificially by adding 0.1 ppm (in the case of ^{1}H atoms) or 1.0 ppm (in the case of ^{13}C or ^{15}N atoms) to the chemical shifts of corresponding atoms in nearest-neighbor residues of the same residue type.

The use of a full data set permitted more general testing of the effects of scatter, noise and missing peaks in the data sets. Scatter in the data was simulated by adding to or subtracting from

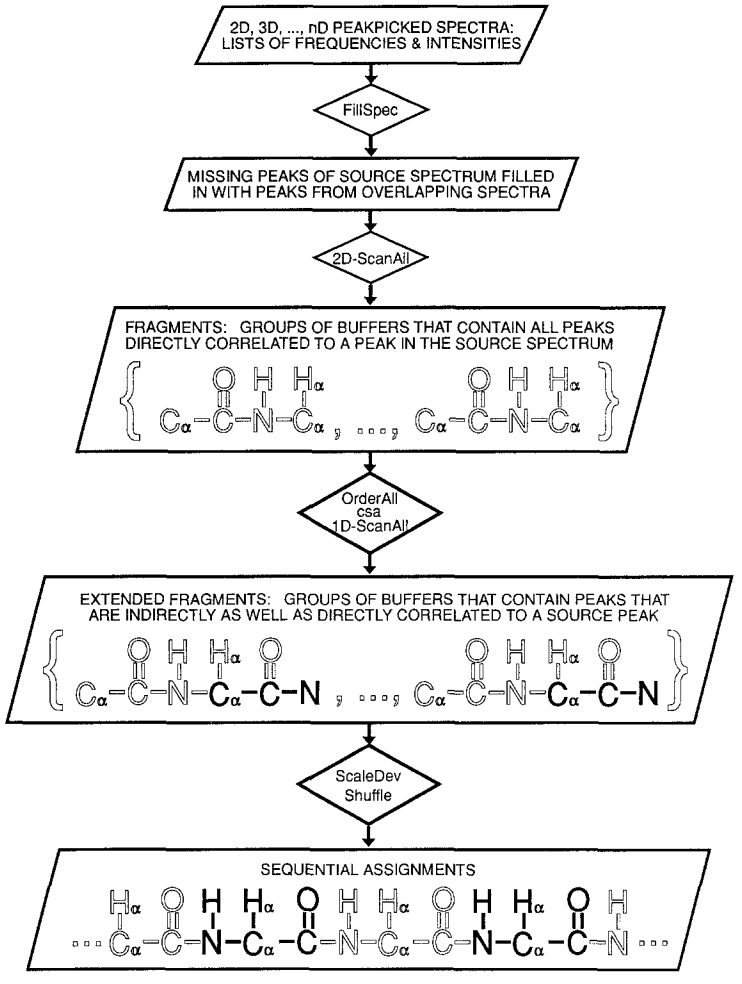


Fig. 1. Flowchart of the SBA (sequential backbone assignment) algorithm as implemented with the CONTRAST software package. The algorithm uses HNCO as the source spectrum and HNCA, HN(CO)CA, HCACO, HCA(CO)N and TOCSY-HMQC spectral data sets to generate and order fragments.

each chemical shift a value, randomly generated with a Gaussian central tendency within a designated range. Noise was added to a spectrum in the form of spurious peaks, generated such that each new noise peak contained one frequency coordinate that was randomly generated within the range of chemical shift values appropriate to that nucleus; the other frequency coordinates were set equal to chemical shift values from real peaks in the spectrum, before scatter had been added to them. The intensity values for these 'noise' peaks were all taken to be equal to the intensities of the related 'real' peaks. Missing peaks were simulated by deleting real (not 'noise') peaks at random from each generated 'spectrum' list, such that the number of deleted peaks was the same for each spectrum. The total time required for the program to determine sequential assignments for the backbone resonances of III^{Glc} from one of the above data sets on a Silicon Graphics Iris Indigo 4000 was generally less than 30 s.

***CONTRAST SBA* ASCII Macro File**

```
lf hnco.con                                          **load spectrum file hnco.con
lf hnca.con                                          **load spectrum file hnca.con
lf hncoca.con                                        **load spectrum file hncoca.con
lf hcaco.con                                         **load spectrum file hcaco.con
lf hcan.con                                          **load spectrum file hcan.con
lf tocsy.con                                         **load spectrum file tocsy.con
timer                                                **starts timer
fillspec 1 2 !(d1 <.05> %1 && d2 <.4> %2) d3=0.0     **fills in missing hnco peaks
scanall 1, 1 2 3 6 (d1 <.02> %1 && d2 <.1> %2)       **scans hnco, hnca, hncoca & tocsy
**prunebuffer 1 |hnco -c 95                          **deletes peaks from buffers
**prunebuffer 2 |hnca -c 95                          **deletes peaks from buffers
**prunebuffer 3 |hncoca -c 95                        **deletes peaks from buffers
**prunebuffer 6 |tocsy -c 95                         **deletes peaks from buffers
orderall d                                           **orders peaks in the buffers
csa 1, 4 (d1 <.02> d3|tocsy,f3 & d2 <.1> d3|hnca,f3) **combinatorial search of hcaco
csa 1, 5 (d1 <.02> d3|tocsy,f3 & d3 <.1> d3|hnca,f3) **combinatorial search of hcan
**prunebuffer 4 |hcaco -c 95                         **deletes peaks from buffers
**prunebuffer 5 |hcan -c 95                          **deletes peaks from buffers
scanall 1, 4 (#|hcaco < 1 && d1 <.02> d3|tocsy,f3) |SHhcaco **1D conditional scan
scanall 1, 4 (#|hcaco < 1 && d2 <.1> d3|hnca,f3) |SChcaco   **1D conditional scan
scanall 1, 5 (#|hcan < 1 && d1 <.02> d3|tocsy,f3) |SHhcan   **1D conditional scan
scanall 1, 5 (#|hcan < 1 && d3 <.1> d3|hnca,f3) |SChcan     **1D conditional scan
**prunebuffer 4 |SHhcaco -c 95                       **deletes peaks from buffers
**prunebuffer 4 |SChcaco -c 95                       **deletes peaks from buffers
**prunebuffer 5 |SHhcan -c 95                        **deletes peaks from buffers
**prunebuffer 5 |SChcan -c 95                        **deletes peaks from buffers
orderall d                                           **orders peaks in new buffers
scaledev                                             **scales peaks in buffers
set sn >scores.sn                                    **takes scoring routines from file
shuffle 1                                            **shuffles fragments using hnco
timer                                                **prints time to screen
shuffletofile 1 >.shuf                               **writes CONTRAST SBA output
shuffletospectrum 1, hnco#2                          **creates an ordered hnco spectrum
writespectrum hnco#2 >shuf.con                       **writes ordered spectrum to file
ordercounter >shuf.con                               **calculates algorithm performance
quit                                                 **exits CONTRAST
```

Fig. 2. The ASCII macro used to implement the SBA algorithm with the CONTRAST software package. The CONTRAST program reads macro files one line at a time, executing each function before continuing to the next line. Comments are marked with double asterisks. PruneBuffer commands are shown, but were disabled by addition of double asterisks before the function call.

Data structures and nomenclature

A schematic of the CONTRAST SBA algorithm as applied to the above sample data set is shown in Fig. 1. The ASCII macro used to implement the CONTRAST SBA algorithm is illustrated in Fig. 2. Input data for the CONTRAST SBA strategy is in the form of files from peak-picked N-dimensional spectra, each with N frequencies and one intensity per peak. Peak-picked files are read into memory into structures called *spectra* S_i, where i = [1,2,...,I]. Each of the I spectra contains an identifying number i associated with that spectrum, the spectrum's name, a set of default values to be used by the program in the absence of input values, comments, and a list of J peaks. A given spectrum is referred to by its name or number i, preceded by a '$' symbol. Each peak $S_i \cdot p_j$, where j = [1,2,...,J], of spectrum S_i has associated with it: N frequencies $S_i \cdot p_j \cdot d_n$ for each of the N dimensions, where n = [1,2,...,N]; a peak intensity value $S_i \cdot p_j \cdot d_0$; a comment string; and a total of M *buffers.* Buffers are general structures that contain a set of peaks from one or more spectra. The main function of the buffers is to serve as a work area for holding groups of peaks, formed from searches of other spectra or other buffers. Any number of buffers F can be formed during a session. A given buffer is referred to by its name or number f, where f = [1,2,...,F], preceded by a '|' symbol. Associated with each peak in a buffer is a set of

statistical and bookkeeping values, some of which are explained in the Neighborhood Scoring section of the Appendix.

Searches are performed for chemical shifts that fall within select ranges of given dimensions n = [1,2,...,N] of a spectrum or buffer. The dimensions are specified by d1,d2,...,dN. The target value of a search is defined to be the center of the specified range, and the tolerance of that search is defined to be one-half the width of that range along the dimension of the search. Tolerances are specified by using the notation < tolerance value >. Target values are written either explicitly (e.g., as 4.02 ppm) or denoted by '%n', 'dn', 'dn|buffer', or 'dn$spectrum' to specify that a separate search is to be performed for target values taken from dimension n of each peak in a buffer or spectrum. In searches and conditional statements, the intensity of a peak is treated as an additional coordinate, and the searches in the intensity dimension are specified by 'd0' or simply 'i'. Doubled conjunctions, '&&' (synchronous 'and') and '||' (synchronous 'or'), are used to combine two or more search expressions, so that each expression refers to the same peak or comparable peaks in the specified buffers or spectra. Single conjunctions, '&' (combinatorial 'and') and '|' (combinatorial 'or'), combine two or more expressions, so that all combinations of peaks from the different expressions are compared. Similarly, the combinatorial operators '>' (greater than), '=' (equals), '< >' (within tolerance), '><' (outside tolerance), '!=' (not equal), etc., can be converted to synchronous operators by doubling their symbols (e.g. '==', '<<>>', '!!='). These synchronous and combinatorial operators and conjunctions can be combined with parentheses and '!' ('not' modifiers) to build up very selective searches.

In the CONTRAST SBA strategy, one selected spectrum (the *source spectrum*) serves to direct searches and to initiate the arrangement of buffers into *fragments*. Fragments are groups of buffers, each associated with one of the R peaks in the source spectrum. Each peak in the source spectrum is used as the starting point for a set of searches, and each of these searches generates the buffers in its associated fragment. Ideally, each fragment will contain enough information to unambiguously assign a portion of the spin system. These usually correspond to an amino acid residue with links to its neighbors. Separate routines are then used to order the fragments.

Fragment assembly

The peaks in the source spectrum are used as starting points for building fragments. Since each peak in the source spectrum generates a fragment, each noise peak picked along with the real peaks in that spectrum gives rise to a bogus fragment, which must be discriminated against in later steps. An analysis of the dependence of the strategy on the presence of noise peaks is included in the Results and Discussion section.

Peaks that are missing from the source spectrum (either because they overlap with another peak or because they were not detected) produce breaks in the assignments, unless they are filled in with information from other spectra. The routine FillSpec is used to fill gaps in the source spectrum with information from other spectra that share common chemical shift dimensions (Fig. 1). For example, in the following call to the FillSpec routine, the HN projection of the HNCA spectrum is searched for peaks that do not correspond to peaks in the HN projection of the source HNCO spectrum; these peaks are then added to the HNCO spectrum with the frequency of the C^{α} dimension set to the arbitrary value of 0.0 ppm, which is sufficiently far outside the range of carbonyl resonances that it will not be mistaken for a true carbonyl resonance:

fillspec hnco hnca !(d1 <.05> %1 && d2 <.4> %2) d3 = 0.0

The expression in parentheses gives the criteria for a search for peaks in the HNCA spectrum that match peaks in the HNCO spectrum, and the '!' (not) symbol before the parentheses specifies that the function should transfer the complement of this set to the HNCO spectrum. In general, the notation 'dn', where n = [1,2,...,N], refers to the dimension of the spectrum which is to be searched (the spectrum whose name or number is listed last), and '%' refers to the dimension of the spectrum from which the target values are to be taken (the spectrum listed first). In this case, the HNCA spectrum will be searched for peaks whose first dimension (d1) chemical shift values are within a 0.05 ppm tolerance of the target value %1 (taken from the first dimension of each peak in the HNCO spectrum) and whose second dimension (d2) chemical shift values are within a 0.4 ppm tolerance of the target value %2 (taken from the second dimension of each peak in the HNCO spectrum). Peak intensity levels, or ranges for values in the third dimension, could also have been specified. The peaks in any spectrum can be edited manually in order to increase the performance of the algorithm.

The first step in assembling fragments (Fig. 1) is to use the ScanAll function to scan spectra that overlap the source spectrum in one or more dimensions (see Searches in the Appendix):

scanall hnco, hnco hnca hncoca tocsy (d1 <.02> %1 && d2 <.4> %2)

In this example, the HNCO spectrum is listed first to indicate that it is to be the source spectrum from which the target values are to be taken. The four remaining spectra are examined in turn by using the search string (d1 <.02> %1 && d2 <.4> %2), where d1 and d2 indicate the dimensions to be searched in each of the four spectra, 0.02 and 0.4 indicate the tolerances to be used, and %1 and %2 are target values taken from the first and second dimensions of each peak in the source spectrum. The source spectrum, HNCO, could have been searched with much lower tolerances and in all three dimensions in a separate call to scanall, to insure that only one peak would be selected, but this would have obfuscated any ^{1}H-^{15}N overlap problems. This routine creates a total of 4R buffers: one for each of the four spectra HNCO, HNCA, HN(CO)CA and TOCSY-HMQC for each of the R peaks in the HNCO spectrum. The end result is a series of R fragments as shown in Fig. 3.

The number of peaks that a given buffer contains can be used as an indicator of the degree of confidence in an assignment. In this example (Fig. 3), each buffer contains peaks that intersect with their source peaks in two dimensions. In general, the more dimensions that intersect, the more confidence one will have in peaks found by the search. Another indicator for the degree of confidence is the deviation value for a peak in a buffer; such values are calculated by summing the deviations of the individual dimensions involved in the search. In the present example, these range linearly from 0.2 (for frequencies that deviate from the target value by the entire tolerance value) to 1.2 (for frequencies that are identical to the target value). The ScaleDev routine, used later in the macro, scales the deviation values of the peaks by their proximity to other peaks in the buffer; after scaling, the deviation values become more uniform indicators of the probability that a given peak in a fragment contains the correct assignment.

The PruneBuffer function filters the contents of each buffer in a specified set of buffers by removing from that buffer each peak found to have a significantly higher deviation value in another buffer in the set. In the example

prunebuffer 2 lhnca −c 95

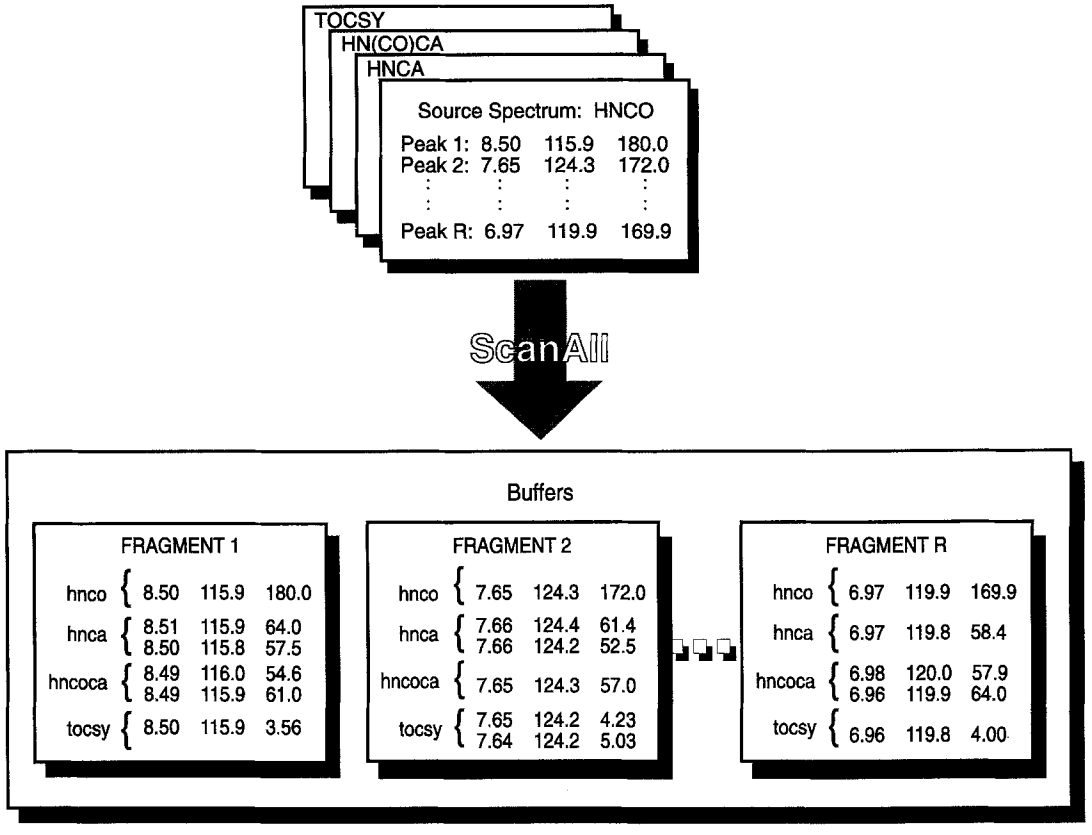


Fig. 3. Schematic of the use of the ScanAll function to create fragments with HNCO as the source spectrum.

the function searches each HNCA buffer to find peaks from the HNCA spectrum (spectrum 2) that are also found in other HNCA buffers. If the deviation value of a peak in one buffer is lower than 95% of the deviation value of that same peak in another buffer, then the peak with the lower deviation value is deleted from its buffer. A '−s' flag (scale) instead of a '−c' flag (cut) causes the function to decrease the deviation values of the peaks rather than to delete them. Although the PruneBuffer function increases the performance of the algorithm for relatively clean data sets, its use makes the algorithm more sensitive to the choice of tolerances set for the program. If the PruneBuffer function is set to delete peaks from the buffers, then the absence of those peaks could lead to a false sense of confidence in the remaining peaks (see the Results and Discussion section). Calls to the PruneBuffer function are shown to be disabled by 'commenting out' in the sample macro (Fig. 2).

The OrderBuffers function below reorders the peaks in each buffer according to the indicated field. In the example

orderbuffers d −d

the 'd' indicates that the peaks in the buffers are to be ordered by deviation values associated with each peak, and the '−d' switch indicates that the peaks are to be put in decreasing order. The

OrderBuffers function can be used to order buffers on the basis of 13 different fields, including peak intensities, coordinate values, composite scores, and the number of times a resonance value is repeated in a buffer. By positioning the 'best' peaks at the top of the buffers, not only is an ordered list of assignments provided for the output of the program, but each function that follows is able to make preferential use of the best information in the buffers.

Data from spectra are abstracted into one or more buffers that collectively represent fragments of the protein backbone. Figure 1 shows a schematic of the fragments assembled thus far from the sample data sets. Confidence in the chemical shifts in a fragment can be increased by the addition of buffers that themselves provide independent information about the chemical shifts of atoms already found in the existing fragment, and fragments can be extended by the addition of buffers that contain chemical shifts of one or more atoms that are not found in the existing fragment. The multiple-dimensional implementation of the ScanAll function provides the best way of extending fragments, since searches in two or more dimensions of a spectrum are much more selective than searches in a single dimension. The selectivity of the CombinedScanAll (csa) function (Fig. 1) lies somewhere between the multiple-dimensional and single-dimensional implementations of ScanAll.

The csa function makes it possible to use frequencies from two different buffers in a fragment, to screen a new spectrum for possible fragment extensions. By contrast, the ScanAll function uses frequencies from either a single spectrum or a single buffer in a fragment to search a new spectrum and to create a new buffer for the searched spectrum in that fragment. The csa routine takes the best p peaks of one buffer in the fragment and combines them with the best q peaks in a second buffer in the fragment, to search two dimensions of a spectrum that does not share more than one dimension with any one of the original buffers (see Searches in the Appendix). The following two csa functions use the first three peaks of the TOCSY-HMQC buffers and of the HNCA buffers to search for matching peaks in the HCACO and HCA(CO)N spectra, respectively:

csa 1, hcaco (d1 <.04> d3|tocsy,f3 & d2 <.3> d3|hnca,f3)

csa 1, hcan (d1 <.04> d3|tocsy,f3 & d3 <.3> d3|hnca,f3)

In the examples above, the ',f3' after the buffer names indicates how many peaks from each buffer will be used in the search. Thus, the first three peaks will each be taken from the TOCSY-HMQC and HNCA buffers. Since the peaks in these buffers have been ordered by their deviations, the three best peaks are chosen. In the first example, dimension one of the HCACO experiment is searched for resonances within 0.04 ppm of a target taken from the third dimension of the first, second or third peak in the TOCSY-HMQC buffer, and the second dimension of the HCACO experiment is searched for a resonance within 0.3 ppm of a target taken from the third dimension of the first, second or third peak in the HNCA buffer. A different search is performed for each of the nine combinations (a single '&' symbol indicates a combinatorial 'and') of the first three peaks of the TOCSY-HMQC buffer and the first three peaks of the HNCA buffer. Figure 1 contains a schematic of the extended fragments created by the ScanAll and CombinedSearchAll routines.

The HNCA and TOCSY-HMQC buffers of a particular fragment must both contain 'correct' peaks in order for the csa function to find 'correct' HCACO or HCAN peaks. If peaks are missing from either the HNCA spectrum or the TOCSY-HMQC spectrum, then the fragments that would normally have contained those peaks will be incomplete, and the csa function will not be able to make use of HCACO or HCAN information. This situation is partially remedied by using two

different single-dimension ScanAll function calls whenever the HCACO or HCAN buffers are empty:

```
scanall hnco, hcaco (#|hcaco !=0 && d1 <.02> d3|tocsy,f3) |SHhcaco
scanall hnco, hcaco (#|hcaco !=0 && d2 <.1> d3|hnca,f3) |SHhcaco
```

In the above example, '#|hcaco' represents the number of peaks in the HCACO buffer in the current fragment. If this number is not zero in the first ScanAll call above, then the first dimension of every peak in the HCACO spectrum will be searched for values that are within 0.02 ppm of the third dimension of the first three peaks in the TOCSY-HMQC buffer. All matching peaks are put in a buffer that is given the name 'SHhcaco'. Such conditional use of the single-dimension ScanAll function prevents the greater numbers of peaks found by the less selective search from swamping out the more reliable data generated by the csa function.

Since the SBA algorithm is implemented as a simple macro in the CONTRAST package, variations on the method are easily implementable by tailoring the macro to the experimental data at hand. A shell call to the command line gives the user the ability to edit the contents of buffers or spectra or to use other interactive functions at any point in the algorithm. The scores (confidence levels) of the peaks in the buffers can be re-evaluated by taking into consideration factors such as peak intensity or the quality of the spectra from which the peaks were derived. The ScanAll and csa functions can be used in conjunction with experiments such as 4D HCCH-TOCSY (Olejniczak et al., 1992), 3D H(CCO)NH or 3D C(CO)NH (Grzesiek et al., 1993) to extend the fragments to include side chains, and these side-chain buffers can then be filtered by using the contents of a TOCSY-HMQC buffer and a CONTRAST Intersection function. The options are nearly limitless.

Sequencing routines

At this point in the algorithm, the extended fragments produced by the ScanAll and csa functions overlap with adjacent extended fragments at the three resonances illustrated in bold letters in Fig. 1. One approach to continuing would be to extend the fragments further by using csa and ScanAll searches, but as more buffers are added to each fragment, the potential for errors increases. For this reason we have chosen a more global approach, in which the fragments are shuffled in order to maximize the amount of overlap between adjacent fragments. Several different approaches to fragment shuffling have been implemented in CONTRAST; they differ only in how they attain the final arrangement of fragments. All approaches contain the same scoring functions, which are implemented in the user-defined macro by the 'set' function command

```
set sn <scores.sn
```

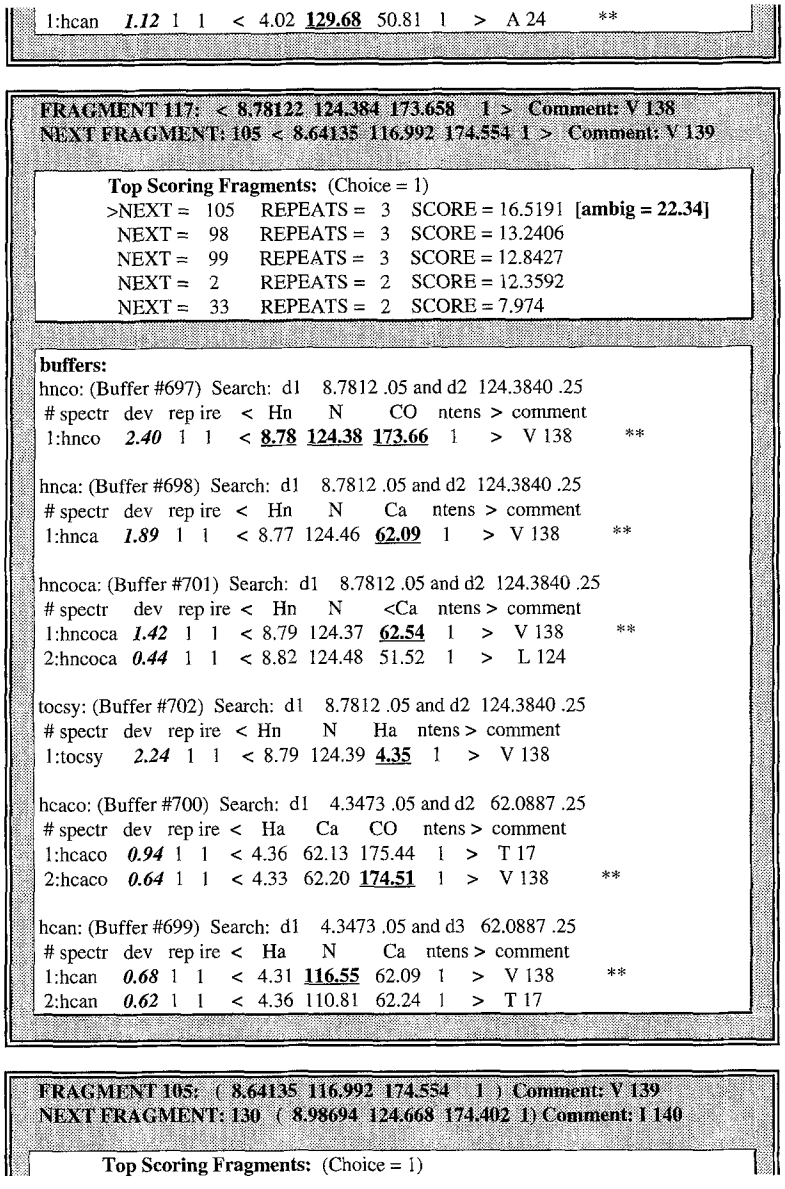
In this example the 'sn' (ScoreNeighbors) variables are set by reading input from the file scores.sn, which contains seven function calls that score the overlap between fragments at the C^{α}, C' and N resonances:

```
bob (d3|hnca <.25> d3|hncoca) -s -b 100%
bob (d3|hcaco <.25> d3|hnco) -s -b 100%
```

```
bob (d2|hcan <.50> d2|hnco) -s -b 100%
bob (d3|SHhcaco <.25> d3|hnco) -s -b 25%
bob (d3|SChcaco <.25> d3|hnco) -s -b 25%
bob (d3|SHhcan <.50> d2|hnco) -s -b 25%
bob (d3|SHhcan <.50> d2|hnco) -s -b 25%
```

The bob (BufferOverlapBuffer) function scores the intersection of a set of specified buffers. (See the section on Neighborhood Scoring in the Appendix.) The bob function can employ conjunctions, so that pairwise or even more complicated comparisons can be made between two or more buffers. For example, bob can be used to score the two-dimensional overlap between CBCA(CO)NH (Grzesiek and Bax, 1992) and HNCACB (Wittekind and Mueller, 1993) or other multiply overlapping buffers, so that the magnitude of the score reflects the higher degree of overlap. The first call to bob in the example above scores the overlap between the third dimension of the HNCA buffer and the third dimension of the HN(CO)CA buffer. The function does this by finding the set of intersecting resonances between the d3 dimension of the HNCA buffer and the d3 dimension of the HN(CO)CA buffer in each fragment, by using 0.3 as the tolerance of the search. The '−b' flag instructs the function to count only the highest scoring pair of resonances from the intersection set, since only one resonance is expected to be common to the two buffers. A '−n' flag (indicating that all of the pairs of resonances in the intersecting set should be scored) would be used to score the intersection of two NOESY-HMQC (Fesik and Zuiderweg, 1988) buffers, since more than one resonance could be expected to be shared between the two buffers from adjacent fragments. A '−s' flag tells the function to scale the score on the basis of a user-defined combination of factors (see Neighborhood Scoring in the Appendix); the default simply returns the number of resonances in the intersection set. The percentages following each function call specify the weights to be used for scaling the score returned by each function; these allow one to weight more important tests more highly than less important ones. In the above example, the scores of the first three bob functions are given equal weight, but the scores of the remaining four bob function calls are given less weight, since they rely on buffers created by the less selective 1D searches. A total score Γ_{jk} between two fragments F_j and F_k is obtained by taking the weighted sum of all individual bob scores. The magnitude of this score is assumed to be proportional to the probability that the two fragments are neighbors in the protein sequence.

Three different approaches employing the scoring routines listed above were tested for their ability to order fragments: Shuffle (a 'best first' approach), Anneal (an approach that uses simulated annealing (Metropolis et al., 1953; Kirkpatrick et al., 1983) to find the global maximum score), and an unnamed combination of the two approaches which was tried in two variations. Shuffle, which proved to be the most succesful algorithm (see the Results and Discussion section), uses a 'best first' approach to finding the correct arrangement of fragments that is similar to the 2D NOESY-based approach used by Kleywegt et al. (1991). In the call to Shuffle (Shuffle hnco L), the name of the source spectrum (HNCO) and the number of highest scoring neighbors that are to be remembered (L), are the only input parameters. The Shuffle algorithm requires J_s^2 comparisons to shuffle the fragments into an order that places the fragments with the highest scores, Γ_{jk}, together in the sequence. The algorithm goes through each fragment F_j in the starting sequence, and by using the scoring functions in memory to calculate Γ_{jk} for every possible



1:hcan *1.12* 1 1 < 4.02 **129.68** 50.81 1 > A 24 **

FRAGMENT 117: < 8.78122 124.384 173.658 1 > Comment: V 138
NEXT FRAGMENT: 105 < 8.64135 116.992 174.554 1 > Comment: V 139

Top Scoring Fragments: (Choice = 1)
>NEXT = 105 REPEATS = 3 SCORE = 16.5191 **[ambig = 22.34]**
NEXT = 98 REPEATS = 3 SCORE = 13.2406
NEXT = 99 REPEATS = 3 SCORE = 12.8427
NEXT = 2 REPEATS = 2 SCORE = 12.3592
NEXT = 33 REPEATS = 2 SCORE = 7.974

buffers:

hnco: (Buffer #697) Search: d1 8.7812 .05 and d2 124.3840 .25
spectr dev rep ire < Hn N CO ntens > comment
1:hnco *2.40* 1 1 < **8.78** **124.38** **173.66** 1 > V 138 **

hnca: (Buffer #698) Search: d1 8.7812 .05 and d2 124.3840 .25
spectr dev rep ire < Hn N Ca ntens > comment
1:hnca *1.89* 1 1 < 8.77 124.46 **62.09** 1 > V 138 **

hncoca: (Buffer #701) Search: d1 8.7812 .05 and d2 124.3840 .25
spectr dev rep ire < Hn N <Ca ntens > comment
1:hncoca *1.42* 1 1 < 8.79 124.37 **62.54** 1 > V 138 **
2:hncoca *0.44* 1 1 < 8.82 124.48 51.52 1 > L 124

tocsy: (Buffer #702) Search: d1 8.7812 .05 and d2 124.3840 .25
spectr dev rep ire < Hn N Ha ntens > comment
1:tocsy *2.24* 1 1 < 8.79 124.39 **4.35** 1 > V 138

hcaco: (Buffer #700) Search: d1 4.3473 .05 and d2 62.0887 .25
spectr dev rep ire < Ha Ca CO ntens > comment
1:hcaco *0.94* 1 1 < 4.36 62.13 175.44 1 > T 17
2:hcaco *0.64* 1 1 < 4.33 62.20 **174.51** 1 > V 138 **

hcan: (Buffer #699) Search: d1 4.3473 .05 and d3 62.0887 .25
spectr dev rep ire < Ha N Ca ntens > comment
1:hcan *0.68* 1 1 < 4.31 **116.55** 62.09 1 > V 138 **
2:hcan *0.62* 1 1 < 4.36 110.81 62.24 1 > T 17

FRAGMENT 105: (8.64135 116.992 174.554 1) Comment: V 139
NEXT FRAGMENT: 130 (8.98694 124.668 174.402 1) Comment: I 140

Top Scoring Fragments: (Choice = 1)

Fig. 4. Excerpt from the output file, generated by the CONTRAST SBA algorithm. Each fragment is associated with a peak in the HNCO source spectrum and uses the position of that peak in the source spectrum as a label. For reference, the peaks in each data set used to generate this output are labeled with the names of the residues that gave rise to them. These comments are ignored by the assignment algorithm and simply aid in the evaluation of its performance. The first white box in each fragment contains the numbers and scores of the five fragments with the highest potential for being next in the sequence. In this example fragment 105 was determined to be next in the sequence, with an ambiguity factor of 22.34. The second white box contains the buffers that comprise the fragment and the peaks that are present in those buffers. Unique shift values used to assign the fragment are underlined. The deviation values (shown in bold italics) of each peak serve as indicators of confidence in the assignment. Deviation values range between 0.0 (no confidence) and 2.4 (high confidence). The presence of double asterisks after a peak indicates that the peak was used by the ordering routine to pair the fragments; this gives the peak assignment additional credibility.

fragment pair, it records the L highest scoring neighbors F_k along with their scores. (The user-defined integer L was set to 10 for the work described here.) F_j is then inserted before its highest scoring neighbor F_k, if F_k does not already have a fragment inserted before it. If F_k already has a fragment F_l inserted before it, then the two scores Γ_{jk} and Γ_{lk} are compared, and the fragment that has the higher score with F_k is inserted before F_k. The fragment with the lower of the two scores will then get inserted before its next-highest scoring neighbor, and the process continues until all of the displaced fragments are either inserted or reach the end of their L highest scores, at which point they are left unconnected to the emerging sequence.

The Anneal algorithm uses simulated annealing to search out the fragment arrangement that gives the highest possible global score, which is given by the sum of the individual Γ_{jk} scores between neighbors. This differs from the Shuffle algorithm, in that high-scoring fragment pairs can become separated if an alternative arrangement gives a higher global score.

The combined approach uses either simulated annealing or steepest descent to refine the Shuffle output. The use of the steepest descent algorithm with the Shuffle algorithm takes the 'best first' arrangement of the Shuffle output sequence and finds the local score maximum nearest that arrangement. The use of simulated annealing at low temperatures yields almost the same result, but allows the algorithm to traverse lower scoring arrangements to get to local maxima near the 'best first' starting point. These 'refinement' routines also can be performed iteratively, by using a 'lock' function to fix the changes made by the 'refinement' routines, so that these changes are locked in place on the next iteration of Shuffle and force the routine to search out alternative arrangements.

RESULTS AND DISCUSSION

The standard output of the CONTRAST SBA algorithm is an assignment file which contains all of the information necessary for making the sequential assignments (Fig. 4). In contrast to data that have been distilled into an assignment table, the assignment file contains the information needed to judge the level of ambiguity in each assignment. In order to interpret the assignment file, one constructs the assignments for each fragment from the first peak in each buffer. The fragments are placed in the sequential order determined by the shuffling routine. The use of a second or lower ranked peak from one of the buffers of a fragment by the shuffling routine in making sequential associations is an indication that the higher ranked peaks do not represent valid candidates for the assignment. A high ambiguity level for a fragment's sequential ordering should serve as a warning that the assignment might be wrong.

The results of the CONTRAST SBA algorithm are dependent on the number, quality and type of experiments in the data set and on the amount of user interaction. The addition of a 4D HCACON (Kay et al., 1991) or a NOESY-HMQC (Fesik and Zuiderweg, 1988) data set, for instance, increases the performance of the algorithm, whereas the exclusion of HCACO or HCA(CO)N data sets could decrease its performance. Since the CONTRAST SBA routines are designed to be as versatile and interactive as possible at all stages of the program, the editing of buffers or spectra can also affect the performance of the algorithm. The results presented here were obtained from realistic, computer-generated HNCO, HNCA, HCACO, HCAN and TOCSY-HMQC data sets, by fully automated, noninteractive use of the standard macro. Use of computer-generated spectra made it possible to vary the number of missing peaks, the amount of

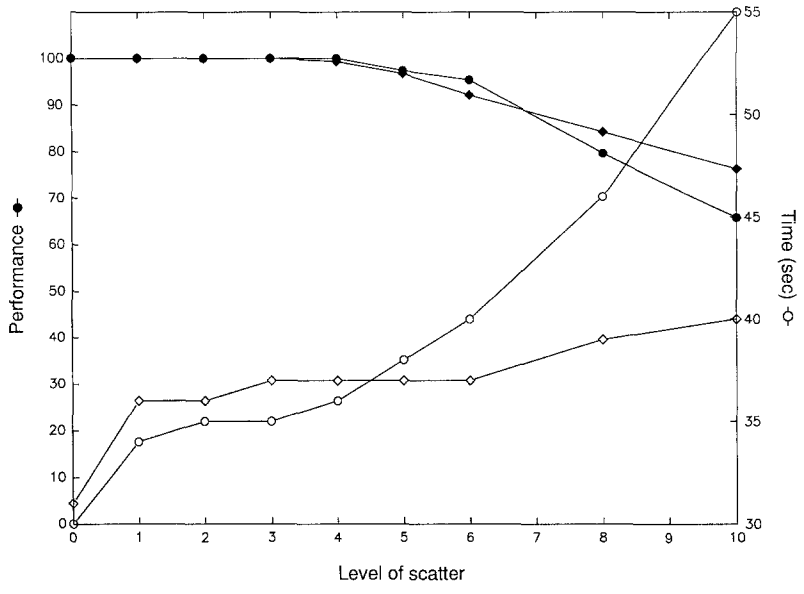


Fig. 5. Dependence of the performance (percentage of possible correct assignments found, filled circles) and execution time (open circles) of the CONTRAST SBA algorithm on the level of uncertainty of resonance frequencies (scatter). Diamonds show the same dependences when PruneBuffer filters are inserted into the basic algorithm; filled diamonds show the performance and open diamonds show the execution time of the modified algorithm. Data points are connected with line segments for clarity. The level of scatter (see text) is the multiple of an initial level of chemical shift uncertainty (scatter = 1) that was added to the 'perfect' data randomly as a Gaussian distribution within assumed ranges: ± 0.005 ppm for protons, ± 0.125 ppm for HCA(CO)N nitrogens, and ± 0.025 ppm for carbons and other nitrogens. The level of scatter that one would expect to find in a protein the size of III^{Glc} (M_r = 18.1 kDa), using conventional Fourier transform processing strategies enhanced by linear prediction, is scatter = 2 (Ikura et al., 1990). Noise was added to each spectrum to a final level of 10%.

scatter and the noise level included in the data sets. This allowed us to evaluate the dependence of the performance of the algorithm on each of these spectral features. Use of actual spectra would give information on whether the experiments chosen were adequate to completely assign a particular protein by this technique, but would give little information on the robustness of the algorithm.

When presented with perfect data sets (with no noise, scatter or missing peaks), the program produced an assignment file which contained the correct assignments for the backbone resonances of all 161 assignable (non-proline) residues, listed in sequential order within the eight segments formed by breaking the sequence at the seven prolines. The number of correct orderings of the residues of the protein were taken as a simple indicator of the performance of the algorithm; thus, the 'perfect' III^{Glc} data yields a score of 152 correct orderings (defined as 100% performance). This performance was independent of the tolerances used by the program for searches and scoring routines, but the execution time of the program increased with increasing tolerance (results not shown).

Inaccuracies in the detected positions of peaks and differences in sample conditions cause small deviations (scatter) in the chemical shifts of resonances in real NMR data. The magnitude of these deviations is dependent on the resolution of the experiment, the type of nucleus, differences in

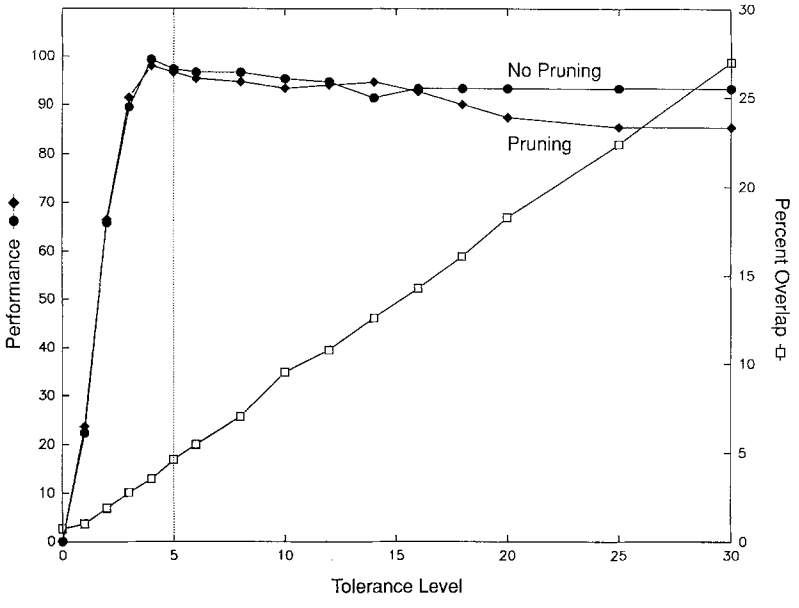


Fig. 6. Dependence of the performance (percentage of possible correct assignments found) of the basic SBA algorithm on the tolerance levels used by the CONTRAST program. Results are shown without (filled circles) and with (filled diamonds) insertion of PruneBuffer filters into the basic algorithm. The apparent percent overlap (open squares) is the percentage of interresonance chemical shift differences that are smaller than the tolerances set in the program. Tolerance levels are multiples of the initial tolerances of ±0.01 ppm for comparison of protons, ±0.05 ppm for comparison of carbons and non-HCA(CO)N nitrogens, and ±0.15 ppm for comparison of HCA(CO)N nitrogens with nitrogens from other experiments. Each data set was generated with scatter = 5 (2.5 times the level of scatter expected in a protein the size of III^{Gle}) and it had noise peaks added to a final level of 10% noise. Data points are connected with line segments for clarity. The vertical dotted line indicates a tolerance level comparable to the level of scatter in the data.

sample conditions between experiments and the methods used to convert time-domain data (free induction decays) to frequencies. Scatter was added to the chemical shift values in the data sets to give realistic experimental reproducibility limits (Ikura et al., 1990) of ± 0.02 ppm for ^{1}H resonances from the same nucleus in different data sets, ± 0.1 ppm for ^{15}N resonances from two H_2O data sets, ± 0.3 ppm for ^{15}N resonances from H_2O and D_2O data sets (to account for isotope effects on chemical shifts) and ± 0.1 ppm for ^{13}C resonances. With this level of scatter and with a 10% noise level in all of the data sets, the performance of the algorithm was again a perfect 152 correct orderings (100% performance). Each noise peak added to a spectrum had the same intensity as the other peaks and was correlated to at least two real peaks by shared chemical shift values. All spectra used in the tests of algorithm performance reported here had noise peaks added to them to a final level of 10%, except for the spectra used to test the dependence of the algorithm on noise levels and those used to generate the profile of ambiguity levels.

Figure 5 shows the performance and execution time of the algorithm, as the scatter in the data sets (and therefore the tolerances used by the program) was increased beyond that typical of experimental data. Performances are shown for data processed both with and without PruneBuffer filters. The scatter was increased in multiples of an initial level of scatter: scatter = 1 represents the level of scatter described above. It can be seen that the performance of the algo-

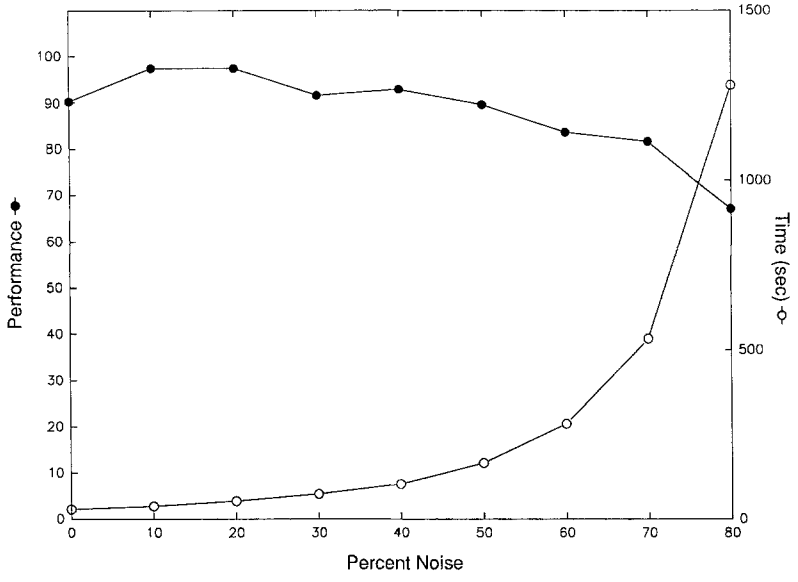


Fig. 7. Dependence of the performance (percentage of possible correct assignments found, filled circles) and execution time (open circles) of the basic CONTRAST SBA algorithm on the percent noise (percent noise = (number of false peaks/(number of false peaks + number of real peaks) × 100) in the spectra. One chemical shift dimension from each noise peak was randomly generated within the chemical shift range characteristic of that dimension, and the other chemical shift dimensions were set equal to chemical shifts from the corresponding dimensions of other peaks in the spectrum. The false peaks thus formed were given the same intensity values as those given to the real peaks. The indicated final percentage of noise was added to each spectrum in the set used for assignments (including the source spectrum). Each data set was generated with scatter = 5 (2.5 times the level of scatter expected in a protein the size of III^{Glc}). Data points are connected with line segments for clarity.

rithm did not decrease until the amount of scatter in the data reached a level of 2.5 times the level of scatter in a typical experimental data set (scatter = 5). As the level of scatter increased, the apparent overlap of the spectra increased. The errors in the output, obtained from the data set which contained five times the initial level of scatter (2.5 times the expected experimental scatter), were examined individually and were found to arise from a common mechanism. In each case, two or more pairs of chemical shift assignments in two or more fragments were found to have traded places. In each case, the assignment made by the algorithm appeared to be the best assignment, given the data. But since these resonance switches always occur within the range of tolerances for the appropriate nuclei, the ambiguity levels generated by the program in such cases are always high. When such problems are recognized, they can be overcome experimentally by decreasing the level of ambiguity in the fragments: either by increasing the digital resolution of one or more of the input data sets or by adding new data from an additional experiment.

When working with real spectra, one cannot be certain of the amount of scatter in the data, and thus the question of what tolerances to use for the program becomes important. For many automated assignment algorithms this choice is critical (Eads and Kuntz, 1989; Ikura et al., 1990; Kleywegt et al., 1991). However, the performance of the basic CONTRAST SBA algorithm is relatively insensitive to the tolerance chosen, as long as it is larger than the sum of the scatter levels and the alignment offset between the spectra being compared. The main penalty paid for

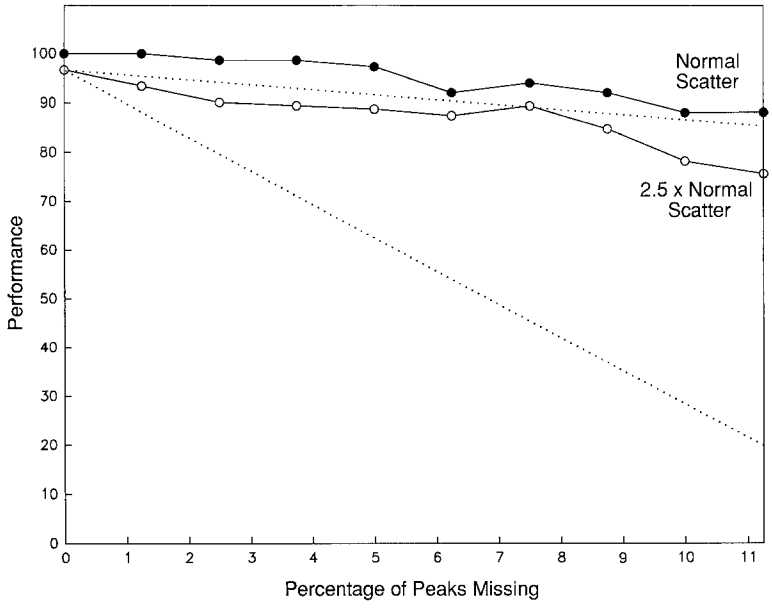


Fig. 8. The effect of missing peaks on the performance (percentage of possible correct assignments found) of the CONTRAST SBA algorithm. Peaks were deleted at random from each spectrum, including the source spectrum, up to the percentage levels indicated on the horizontal axis. The level of noise of each spectrum was maintained at 10%. The filled circles (connected by line segments for clarity) trace the performance of the CONTRAST SBA algorithm, operating on data sets with a normal level of scatter for a protein the size of III^{Glc} (scatter = 2), while the open circles trace the performance of the algorithm on data sets with 2.5 times the normal level of scatter (scatter = 5). The lower dotted line represents the theoretical worst-case performance in which each deletion causes an error, and the upper dotted line represents the performance when only the deletion of HNCA peaks results in mistakes.

using very large tolerances is that the computation time increases. Figure 6 shows the dependence of the algorithm performance on the tolerances used by the algorithm for a data set scatter = 5 and 10% noise peaks. The performance of the algorithm reaches a maximum at tolerances smaller than the amount of scatter added to the data (marked with a vertical dotted line); this occurred because the low tolerance levels eliminated from consideration some of the worst data. Inclusion of PruneBuffer filters in the algorithm is shown to decrease the stability of the algorithm. These filters also reduce the completeness of the output, so that sources of ambiguity may no longer be seen in the output file. The effective overlap of resonances in the spectra is shown to increase throughout the entire range shown in the figure as the tolerances used by the program increase. The effective percent overlap is defined here as the percentage of interchemical shift differences between related nuclei that are smaller than the program's user-set tolerances for searches in the dimensions containing those nuclei.

The performance of automated assignment strategies is dependent on the quality of the spectra and on the discrimination and accuracy of the peak-picking routine used. One is often forced to choose between picking extra noise peaks or missing real data. Thus, it is important to note that the performance of the algorithm did not decrease when levels of noise up to 90% (number of additional or false peaks equal to nine times the number of true peaks) were added to data sets created with scatter at realistic experimental levels (results not shown). Figure 7 shows the

dependence of the performance of the algorithm on the level of noise added to data sets, created with 2.5 times the normal level of scatter (scatter = 5). The initial rise in performance with the addition of noise up to the level of 20% is thought to be due to the increased number of degrees of freedom available to the shuffle algorithm.

Figure 8 shows the dependence of the ability of the algorithm to assemble and correctly order the generated fragments on the number of peaks deleted from the spectra. Peaks were deleted at random from each spectrum, including the source spectrum, up to the percentage levels indicated on the horizontal axis. Since the performance is the percentage of the perfect score for the number of fragments placed together correctly in the output sequence, it does not reflect mistakes in the assignment of individual resonances, which occur when the peaks that contain their chemical shifts are missing. The performance is shown both for data to which a normal level of scatter (scatter = 2) had been added and for data to which 2.5 times the normal level had been added (scatter = 5). Noise peaks were not permitted to be deleted from the data sets. The performance of the algorithm on data sets to which a normal level of scatter had been added is seen to be very good, up to levels where 5% of the peaks had been deleted. The overall performance of the algorithm on the 'normal' data sets drops at a rate of about one percent per percentage of data deleted. The performance of the algorithm for data to which 2.5 times the normal level of scatter had been added drops at a rate of 1.5% per percentage of data deleted. The lower dotted line represents the theoretical performance of the algorithm (on data sets with scatter = 5) if every peak deletion resulted in a mistake by the program, and the upper dotted line represents the theoretical performance if only the deletion of HNCA peaks resulted in mistakes.

The Shuffle routine uses a 'best first' approach to making assignments. Assignments are effectively made for the best data first, so that the highest scoring assignments are given more weight than lower scoring assignments. An alternative approach, which uses simulated annealing (Metropolis et al., 1953; Kirkpatrick et al., 1983) to find the arrangement of fragments that give the highest combined score for the entire sequence, was also explored. The simulated annealing algorithm Anneal usually provided results that were identical to those of the Shuffle algorithm for the assignment of clean, precise data sets; however, the performance of the Anneal algorithm fell behind that of Shuffle as the quality of the data decreased. Figure 9 shows a comparison of the performance of the two algorithms on the same data sets as a function of the level of scatter. When the performance of the Shuffle algorithm was better than that of the Anneal algorithm, the global score of the sequences arranged by Anneal was always higher (better) than that of Shuffle. We believe that the difference in performance is due to a fundamental difference between the two approaches, rather than to an artifact of the scoring routines. Simulated annealing is a global approach, but the information for making assignments is inherently local (unless local secondary structure and medium-range NOE information are used in the process). Figure 10 illustrates how a global optimization routine, such as simulated annealing, can yield a sequential assignment that does not make use of the highest scoring fragment pair (the assignment with the most evidence). The likelihood that simulated annealing will miss high-scoring pairings increases with the ambiguity of the data, since second- and third-place scores for alternative fragment pairings approach the scores for correct pairings with increasing ambiguity. By contrast, the 'best first' approach gives the highest priority to sequential pairings that are least ambiguous, much in the same way that one makes assignments manually.

The ability of any algorithm to assemble and order fragments is dependent on the quality of the

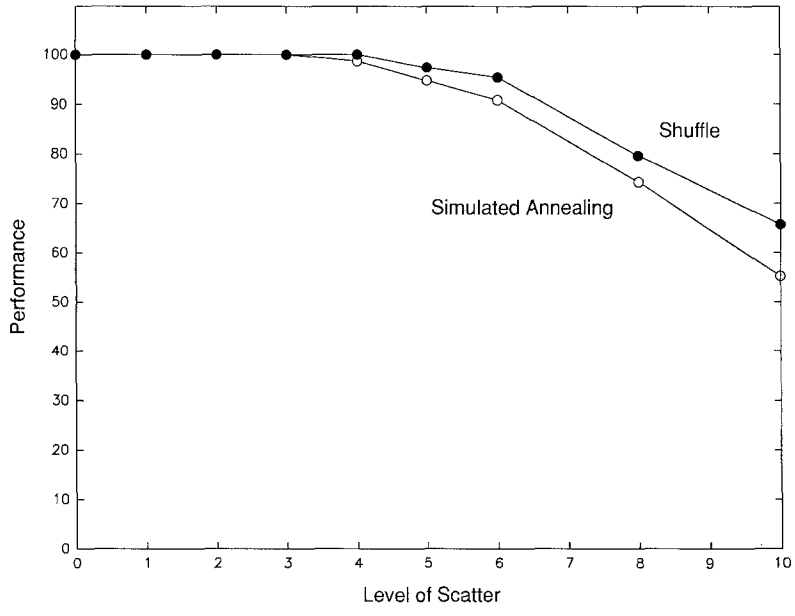


Fig. 9. Comparison of the performance (percentage of possible correct assignments found) of the Shuffle (filled circles) and Anneal (open circles) implementations of the CONTRAST SBA algorithm, as the level of uncertainty of resonance frequencies (scatter) is increased in multiples of an initial level of frequency uncertainty, added to 'perfect' data in a Gaussian distribution within assumed ranges: ± 0.005 ppm for protons, ± 0.125 ppm for HCA(CO)N nitrogens, and ± 0.025 ppm for carbons and other nitrogens. The level of scatter that one would expect to find in a protein the size of III^{Glc} (M_r = 18.1 kDa) is represented by 2.0 on this scale. A noise level of 10% was added to each spectrum in the data sets. Data points are connected with line segments for clarity.

input data. The goal of an automated assignment program should not be to produce an unambiguous result if the data themselves are ambiguous. Rather, the goals should be to quantify the degree of ambiguity of each assignment that the algorithm makes and to indicate the reasons for the ambiguity. This was a prime consideration in the design of the CONTRAST SBA routines. The assembly of fragments by means of the ScanAll, csa and OrderAll routines produces a set of buffers which each contain a list of peaks in order of importance, from the most probable assignment at the top of the list to the least probable at the end of the list (see Fig. 4). The number of peaks in each buffer and the difference between the deviation values of each peak in a buffer is a good measure of the confidence one should have in that assignment. The presence of only a single peak or a large difference in the deviation values between the first peak and subsequent peaks both indicate that the assignment is relatively sound, given the soundness of the peaks that gave rise to the buffer. On the other hand, the presence of several peaks in the buffer, each with about the same deviation values, should be a warning that the first peak may not contain the correct assignment. The consequences of using alternative 'second place' assignments suggested by the program can be explored by using one of several interactive functions, included in the CONTRAST program.

The presence of second or higher place peaks in the buffers making up a fragment allows the shuffling routines to consider all of the relevant data in ordering the fragments. Peaks containing the correct assignments for their particular resonances may be in second or higher places in their

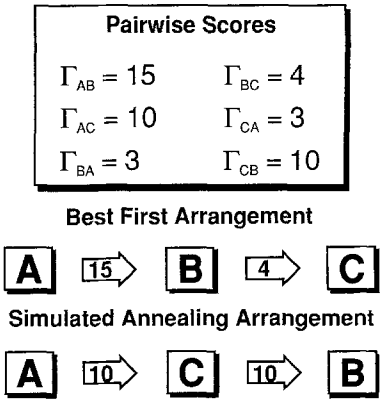


Fig. 10. Illustration of how three fragments, A, B and C, can be ordered differently by the 'best first' (Shuffle) and simulated annealing (Anneal) algorithms, given the listed scores Γ_{jk} between individual fragment pairs. The simulated annealing algorithm finds the arrangement that yields the highest possible global score (20), but in doing so, it fails to pair the fragments (A and B) for which there is the most evidence or highest pairwise score.

buffers. The use of these peaks by the sequential ordering algorithm is an indication that such peaks should be given more credence than their positions or scores would otherwise warrant. These peaks are marked with double asterisks in the CONTRAST output file (Fig. 4). Thus, the process of sequentially ordering the fragments contributes information that can be used in making the final assignment of the resonances in the fragments.

The algorithm orders the assembled fragments into the 'best order' according to the data. If a combination of resonance overlap and unfortuitous scatter causes enough different resonances to effectively 'trade places', then the best ordering of the fragments containing those peaks could be wrong, and the position of the correct fragment which gets displaced by the incorrect one would also be wrong. Although such situations are unavoidable (since the algorithm is making the best assignment, given the data) the ambiguity of each sequential assignment can be quantified in the output, so that the user is alerted to the possibility of errors in the sequential assignment. Figure 11 shows the correlation of the ambiguity factor produced by the SeqAmbig function, with mistakes and natural breaks in the sequential assignments generated by the algorithm. In our experience, an ambiguity factor of 50 or greater indicates the presence of considerable uncertainty in the assignment. All incidences of misassignment in the example shown had factors above this level, and of the 15 correctly assigned fragments with ambiguity factors higher than 50, all showed considerable overlap (10 of them were overlapped to the point that second or third scores were chosen rather than higher scores).

Perhaps the most important feature of the CONTRAST program itself is its adaptability. Several assignment strategies, showcasing different new 3D or 4D experiments, have recently been suggested in the literature (Boucher et al., 1992; Campbell-Burk et al., 1992; Logan et al., 1993; Seip et al., 1993). The CONTRAST SBA algorithm can be modified easily to implement any of these techniques (or even combinations of these techniques). This can be done by adding a few lines to or by deleting a few lines from the macro shown in Fig. 2 or by writing a new macro. Not only can the CONTRAST SBA algorithm be adapted to almost any type of multidimensional NMR data, new algorithms can be constructed from the existing functions to attain different

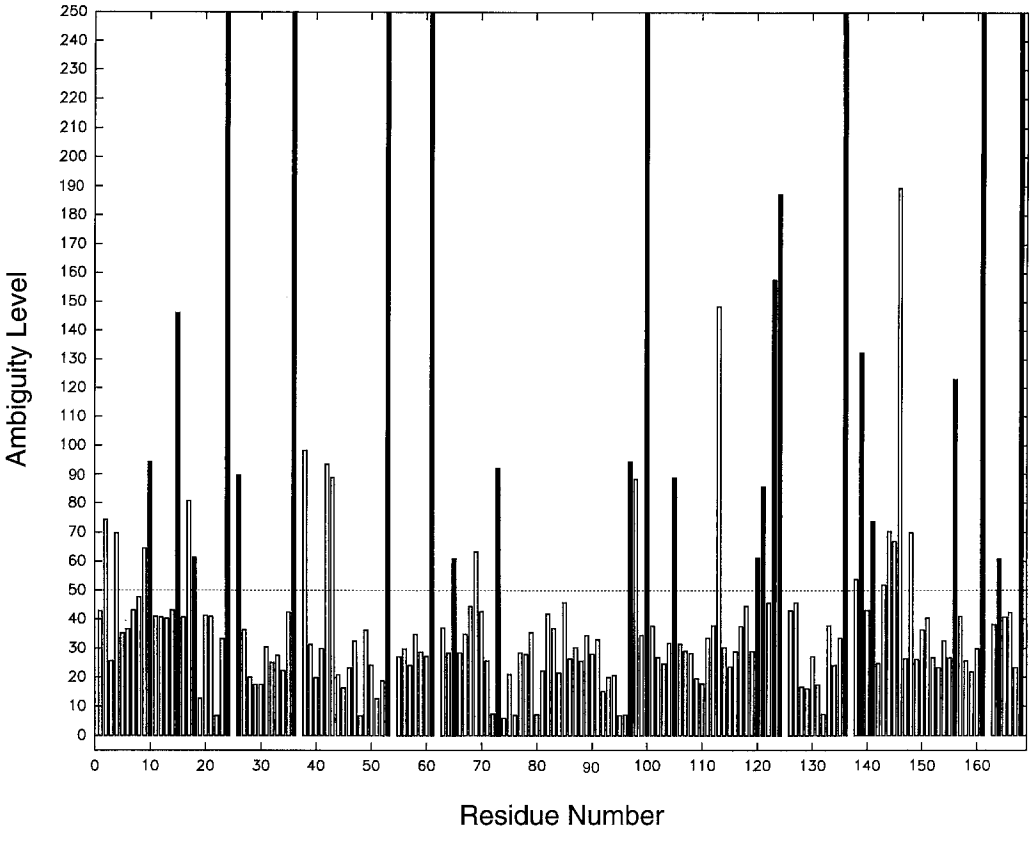


Fig. 11. Ambiguity levels for each residue assignment, plotted as a function of residue number (i). The vertical axis represents the level of ambiguity in the data used by the algorithm to assign the sequential neighbor (i + 1) of residue (i). No ambiguity levels were calculated for prolines, and thus blanks are left at their positions in the sequence. This output was generated from noise-free data sets, to which 2.5 times the level of scatter expected in an 18-kDa protein (scatter = 5) had been added. Empirically, it is found that ambiguity levels at or above 50 indicate possible assignment errors, whereas ambiguity levels at or above 250 indicate a certain break or error. As expected, high ambiguity levels are found at residues preceding proline breaks. Erroneous pairings and breaks are indicated by solid bars.

objectives or to attain the same objective in a different way. This approach should prove useful in evaluating and in determining which data sets are needed to provide an acceptable level of confidence in the assignments.

CONCLUSIONS

The macro-based implementation of algorithms with the CONTRAST software package allows them to be adapted to use almost any NMR data set and allows other variations on the strategies to be tested rigorously. The basic implementation of the SBA algorithm is insensitive to the tolerances used by the program in making assignments. The SBA algorithm is relatively insensitive to noise, and it is able to work around gaps in a spectrum, depending on the amount of redundant information in other spectra.

The CONTRAST package quantifies the ambiguity of each resonance assignment it makes,

and it includes alternative assignments at both the atom and residue (sequential assignment) levels. Alternative assignments at either level can be evaluated by using a set of 'lock' functions, provided in the CONTRAST package. Finally, our experience in using simulated annealing to arrange fragments (by using only local connectivity information with the set of scoring routines supplied with CONTRAST) has suggested that combinatorial optimization approaches, such as simulated annealing, are not as effective as the 'best first' approach which establishes the most secure connectivities first. The CONTRAST package will be made available upon request.

ACKNOWLEDGEMENTS

This work was supported by NIH grants LM 04958 and RR 02301. J.B.O. had additional support from a Steenbock Predoctoral Fellowship. The authors thank Dr. Mark E. Anderson and Dr. Roger A. Chylla for helpful discussions and Amy W. Olson for her aid in preparing the manuscript.

REFERENCES

Bax, A. and Ikura, M. (1991) *J. Biomol. NMR*, **1**, 99–104.
Bax, A. and Grzesiek, S. (1993) *Acc. Chem. Res.*, **26**, 131–138.
Bernstein, R., Cieslar, C., Ross, A., Oschkinat, H., Freund, J. and Holak, T.A. (1993) *J. Biomol. NMR*, **3**, 245–251.
Billeter, M., Basus, V.J. and Kuntz, I.D. (1988) *J. Magn. Reson.*, **76**, 400–415.
Boucher, W., Laue, E.D., Campbell-Burk, S.L. and Domaille, P.J. (1992) *J. Biomol. NMR*, **2**, 631–637.
Campbell-Burk, S.L., Domaille, P.J., Starovasnik, M.A., Boucher, W. and Laue, E.D. (1992) *J. Biomol. NMR*, **2**, 639–646.
Catasti, P., Carrara, E. and Nicolini, C. (1990) *J. Comput. Chem.*, **11**, 805–818.
Cieslar, C., Clore, G.M. and Gronenborn, A.M. (1988) *J. Magn. Reson.*, **80**, 119–127.
Cieslar, C., Holak, T.A. and Oschkinat, H. (1990) *J. Magn. Reson.*, **87**, 400–407.
Clore, G.M. and Gronenborn, A.M. (1991) *Annu. Rev. Biophys. Biophys. Chem.*, **20**, 29–63.
Eads, C.D. and Kuntz, I.D. (1989) *J. Magn. Reson.*, **82**, 467–482.
Eccles, C., Guntert, P., Billeter, M. and Wüthrich, K. (1991) *J. Biomol. NMR*, **1**, 111–130.
Fesik, S.W. and Zuiderweg, E.R.P. (1988) *J. Magn. Reson.*, **78**, 588–593.
Grzesiek, S. and Bax, A. (1992) *J. Am. Chem. Soc.*, **114**, 6291–6293.
Grzesiek, S., Anglister, J. and Bax, A. (1993) *J. Magn. Reson.*, **101**, 114–119.
Ikura, M., Kay, L.E. and Bax, A. (1990) *Biochemistry*, **29**, 4659–4667.
Kay, L.E., Ikura, M., Tschudin, R. and Bax, A. (1990) *J. Magn. Reson.*, **89**, 496–514.
Kay, L.E., Ikura, M., Zhu, G. and Bax, A. (1991) *J. Magn. Reson.*, **91**, 422–428.
Kirkpatrick, S., Gelatt Jr., C.D. and Vecchi, M.P. (1983) *Science*, **220**, 671–680.
Kleywegt, G.J., Boelens, R., Cox, M., Llinas, M. and Kaptein, R. (1991) *J. Biomol. NMR*, **1**, 23–47.
Kraulis, P.J. (1989) *J. Magn. Reson.*, **84**, 627–633.
Logan, T.M., Olejniczak, E.T., Xu, R.X. and Fesik, S.W. (1993) *J. Biomol. NMR*, **3**, 225–231.
Marion, D., Driscoll, P.C., Kay, L.E., Wingfield, P.T., Bax, A., Gronenborn, A.M. and Clore, G.M. (1989) *Biochemistry*, **28**, 6150–6156.
Markley, J.L. (1989) *Methods Enzymol.*, **176**, 12–64.
Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953) *J. Chem. Phys.*, **21**, 1087–1092.
Olejniczak, E.T., Xu, R.X. and Fesik, S.W. (1992) *J. Biomol. NMR*, **2**, 655–659.
Olson Jr., J.B. and Markley, J.L. (1993) *CONTRAST Manual*, available upon request.
Oschkinat, H., Holak, T.A. and Cieslar, C. (1991) *Biopolymers*, **31**, 699–712.
Pelton, J.G., Torchia, D.A., Meadow, N.D., Wong, C.Y. and Roseman, S. (1991) *Biochemistry*, **30**, 10043–10057.
Pfändler, P. and Bodenhausen, G. (1990) *J. Magn. Reson.*, **87**, 26–45.

Seip, S., Balbach, J. and Kessler, H. (1993) *J. Biomol. NMR*, **3**, 233–237.
Van de Ven, F.J.M. (1990) *J. Magn. Reson.*, **86**, 633–644.
Vuister, G.W., Boelens, R., Padilla, A., Kleywegt, G.J. and Kaptein, R. (1990) *Biochemistry*, **29**, 1829–1839.
Weber, P.L., Malikayil, J.A. and Mueller, L. (1989) *J. Magn. Reson.*, **82**, 419–426
Wittekind, M. and Mueller, L. (1993) *J. Magn. Reson., Ser. B*, **101**, 201–205.
Wüthrich, K. (1986) *NMR of Proteins and Nucleic Acids*, Wiley, New York, NY.

APPENDIX

Definition of variables

S	The set of all spectra read into memory by the program. S_i = the ith spectrum.
$S_i.p$	The set of all peaks in spectrum S_i. $S_i.p_j$ = the jth peak in S_i.
$S_i.p_j.d$	The set of all coordinates in $S_i.p_j$. $S_i.p_j.d_n$ = the nth peak in $S_i.p_j$.
$S_i.p.d_n$	The set of all nth coordinates in all peaks in spectrum S_i.
$S_i.p_j.d_0$	The intensity of peak $S_i.p_j$. Also written $S_i.p_j.i$.
B	The set of all buffers (containing peaks) created by the program. B_f denotes the fth buffer.
$B_f.p_k$	The kth peak in B_f.
$B_f.p_k.\delta$	The deviation of the target dimension(s) of $B_f.p_k$ from the target value(s).
$B_f.p_k.i$	The intensity of $B_f.p_k$.
$B_f.p_k.\gamma$	The number of peaks in B_f that are due to the correlation that the experiment is meant to provide. $B_f.p_k.\gamma_{max}$ = the maximum number of correlations possible.
$B_f.p_k.\rho$	The number of times coordinates from $B_f.p_k$ are repeated in B_f.
S_s	The source spectrum from which fragments will be built.
F	The set of all fragments created from S_s. F_j denotes the fragment corresponding to $S_s.p_j$.

Searches

Many of the functions in the backbone-ordering algorithms involve searching specific dimensions of the peaks in a buffer or spectrum for coordinates or intensities within a given tolerance of a target value. When the target values are taken from one set and the search is applied to another set, this is similar to performing a traditional intersection of the two sets, except that the criterion for including an element in the intersection set has been made less stringent by requiring only specified dimensions of the elements of the sets to match within a given tolerance, as opposed to requiring all dimensions of the elements in the sets to be identical. We use the symbol $\cap_{d_x(\tau)d_y}$ to refer to such a pseudointersection with a tolerance τ between specified dimensions d_x and d_y. The single-dimension pseudointersection of sets of peaks X.p and Y.p is defined as

$$X.p \underset{d_x(\tau)d_y}{\cap} Y.p \equiv \{X.p \times Y.p \mid X.p_j.d_x - Y.p_k.d_y \leq \tau, \quad 1 \leq j \leq J, \quad 1 \leq k \leq K\}$$

where $X.p \times Y.p$ is the set of all combinations of ordered pairs of peaks, $\{(X.p, Y.p)\}$, such that

$$\{(X.p, Y.p)\} \equiv \begin{Bmatrix} (X.p_1, Y.p_1) & (X.p_1, Y.p_2) & \cdots & (X.p_1, Y.p_K) \\ (X.p_2, Y.p_1) & (X.p_2, Y.p_2) & \cdots & (X.p_2, Y.p_K) \\ \cdots & \cdots & \cdots & \cdots \\ (X.p_J, Y.p_1) & (X.p_J, Y.p_2) & \cdots & (X.p_J, Y.p_K) \end{Bmatrix}$$

The projection of a set of ordered pairs $\{(X.p, Y.p)\}$ onto its second component $\pi_2(\{(X.p, Y.p)\})$ is given by

$$\pi_2(\{(X.p, Y.p)\}) = \begin{Bmatrix} Y.p_1 & Y.p_2 & \cdots & Y.p_K \\ Y.p_1 & Y.p_2 & \cdots & Y.p_K \\ \cdots & \cdots & \cdots & \cdots \\ Y.p_1 & Y.p_2 & \cdots & Y.p_K \end{Bmatrix}$$

For each spectrum or buffer searched, the ScanAll function creates a buffer B_f, where $1 \leq f \leq J_s$ and J_s is the number of peaks in the source spectrum, for each peak $S_s.p_j$ in the source spectrum S_s. The name of each buffer is determined by the name of the spectrum or buffer being searched. The single-dimension ScanAll search of set $S_i.p$ with $S_s.p$ as the source spectrum, gives the following results

$$\begin{Bmatrix} S_s.p \\ S_s.p_1 \\ S_s.p_2 \\ \vdots \\ S_s.p_J \end{Bmatrix} \xrightarrow{\text{ScanAll}} \begin{Bmatrix} S_i.p \\ S_i.p_1 \\ S_i.p_2 \\ \vdots \\ S_i.p_K \end{Bmatrix} = B_f$$

$$= \left\{ \pi_2\left(\{S_s.p_1\} \underset{d_x(\tau)\, d_y}{\cap} S_i.p \right), \pi_2\left(\{S_s.p_2\} \underset{d_x(\tau)\, d_y}{\cap} S_i.p \right), \ldots, \pi_2\left(\{S_s.p_J\} \underset{d_x(\tau)\, d_y}{\cap} S_i.p \right) \right\}$$

If we define a multidimensional pseudointersection with tolerances τ_t for $t = [1, 2, \ldots, T]$, for specified dimensions d_{xt} of peak set X.p and d_{yt} of peak set Y.p as

$$X.p \underset{\substack{d_{x_1}(\tau_1)\, d_{y_1} \\ \vdots \\ d_{x_T}(\tau_T)\, d_{y_T}}}{\cap} Y.p \equiv X.p \underset{d_{x_1}(\tau_1)\, d_{y_1}}{\cap} Y.p \cap \cdots \cap X.p \underset{d_{x_T}(\tau_T)\, d_{y_T}}{\cap} Y.p$$

then the multidimensional case of the ScanAll function would be given by

$$\begin{Bmatrix} S_s.p \\ S_s.p_1 \\ S_s.p_2 \\ \vdots \\ S_s.p_J \end{Bmatrix} \xrightarrow[\substack{d_{x1}(\tau_1)\, d_{y1} \\ \vdots \\ d_{xT}(\tau_T)\, d_{yT}}]{\text{ScanAll}} \begin{Bmatrix} S_i.p \\ S_i.p_1 \\ S_i.p_2 \\ \vdots \\ S_i.p_K \end{Bmatrix} = B_f$$

$$= \left\{ \pi_2 \left(\{S_s.p_1\} \underset{\substack{d_{x1}(\tau_1)\, d_{y1} \\ \vdots \\ d_{xT}(\tau_T)\, d_{yT}}}{\cap} S_i.p \right), \pi_2 \left(\{S_s.p_2\} \underset{\substack{d_{x1}(\tau_1)\, d_{y1} \\ \vdots \\ d_{xT}(\tau_T)\, d_{yT}}}{\cap} S_i.p \right), \ldots, \pi_2 \left(\{S_s.p_J\} \underset{\substack{d_{x1}(\tau_1)\, d_{y1} \\ \vdots \\ d_{xT}(\tau_T)\, d_{yT}}}{\cap} S_i.p \right) \right\}$$

The csa (CombinedSearchAll) function uses peaks from two buffers, B_x and B_y, in each fragment F_j

$$B_x.p = \left\{ \begin{array}{c} B_x.p_1 \\ \vdots \\ B_x.p_u \\ \vdots \\ B_x.p_J \end{array} \right\}, \qquad B_y.p = \left\{ \begin{array}{c} B_y.p_1 \\ \vdots \\ B_y.p_v \\ \vdots \\ B_y.p_{K1} \end{array} \right\}$$

to create a set of search strings, using all combinations of the first u peaks in $B_x.p$ with the first v peaks in $B_y.p$ and using a total of T coordinates $(d_{x1}, d_{x2}, \ldots, d_{xT})$ and T tolerances $(\tau_1, \tau_2, \ldots, \tau_T)$, to search a spectrum S_i to form a buffer B_z such that

$$B_z.p = \left\{ \begin{array}{l} \pi_2 \left(B_x.p \times B_y.p \underset{\substack{B_x.p_1.d_{x1}(\tau_1)\, B_y.p_1.d_{y1} \\ \vdots \\ B_x.p_1.d_{xT_1}(\tau_T)\, B_y.p_1.d_{yT} \\ \vdots}}{\cap} S_i \right) \cup \cdots \pi_2 \left(B_x.p \times B_y.p \underset{\substack{B_x.p_1.d_{x1}(\tau_1)\, B_y.p_1.d_{y1} \\ \vdots \\ B_x.p.d_{xT_1}(\tau_T)\, B_y.p_v.d_{yT} \\ \vdots}}{\cap} S_i \right) \cup \\ \pi_2 \left(B_x.p \times B_y.p \underset{\substack{B_x.p_1.d_{x1}(\tau_1)\, B_y.p_1.d_{y1} \\ \vdots \\ B_x.p_u.d_{xT_1}(\tau_T)\, B_y.p_1.d_{yT}}}{\cap} S_i \right) \cup \cdots \pi_2 \left(B_x.p \times B_y.p \underset{\substack{B_x.p_1.d_{x1}(\tau_1)\, B_y.p_1.d_{y1} \\ \vdots \\ B_x.p_u.d_{xT_i}(\tau_T)\, B_y.p_v.d_{yT}}}{\cap} S_i \right) \end{array} \right\}$$

Neighborhood scoring

Let the set of ordered pairs of peaks C, formed from the pseudointersection of buffers $B_1.p$ and $B_2.p$ with $B_1.p$ dimension d_x within a tolerance τ of $B_2.p$ dimension d_y, be defined by:

$$C \equiv B_1.p \underset{d_x(\tau)\, d_y}{\cap} B_2.p = \{(B_1.p_j, B_2.p_k)_1, (B_1.p_j, B_2.p_k)_2, \ldots, (B_1.p_j, B_2.p_k)_A\}$$

where A is the number of ordered pairs in C and C_a represents the ath pair in C. The scoring function $g(C_a)$ calculates a score for the two peaks of the ordered pair C_a, by taking a weighted sum of the attributes of each peak and the deviation δ_{12} between the peaks

$$g(C_a) = w_{\delta_i}\, \delta_1 + w_{\delta_2}\, \delta_2 + w_{t_1}\, t_1 + w_{t_2}\, t_2 + w_{\gamma_1} \frac{\gamma_1}{\gamma_{1\,\max}} + w_{\gamma_2} \frac{\gamma_2}{\gamma_{2\,\max}} + w_{\rho_1}\, \rho_1 + w_{\rho_2}\, \rho_2 + w_{\delta_{12}}\, \delta_{12}$$

The weights in the equation (w_{δ_1}, w_{δ_2}, w_{t_1}, w_{t_2}, w_{γ_1}, w_{γ_2}, w_{ρ_1}, w_{ρ_2} and $w_{\delta_{12}}$) are all user-set parameters which, for the example in this text, were all set to 0.0, except for w_{δ_1} and w_{δ_2}, which were set to 1.0, and for w_{12}, which was set to the dimensionality of the searches used to generate the peaks in B_1 and B_2. The deviations δ_1 and δ_2 of the first and second peaks, respectively, of the ordered pair were calculated from the target values V_{1t} and V_{2t}, and tolerances τ_{1t} and τ_{2t} of the searches originally used to place the peaks in the buffers

$$\delta_f = \sum_{t=1}^{T} 1.2 - \frac{|B_f.p_j.d_{xt} - V_{ft}|}{\tau_{ft}}$$

The deviation between the peaks, δ_{12}, was similarly calculated from the equation

$$\delta_{12} = \frac{|B_1.p_j.d_x - B_2.p_k.d_y|}{\tau_c}$$

where τ_c is the tolerance used by the scoring function $g(C_a)$ and d_x and d_y are the dimensions being compared in scoring B_1 and B_2, respectively. The other parameters were not used in this analysis, but are parameters indicating the intensities of the peaks (t_1 and t_2), the ratio of the number of peaks observed in a spectrum for a single correlation (γ_1 and γ_2) to the total possible number of peaks per correlation (γ_{1max} and γ_{2max}), and the number of times the coordinates d_x or d_y are repeated in other spectra in B_1 or B_2 (ρ_1 and ρ_2).

The bob (BufferOverlapBuffer) scoring function call takes the general form

bob (Boolean) [−s] [−b] P%

where P is the weight that is applied to the calculated result, [−s] is an optional 'scaling' flag that signals the function to calculate a score that is scaled by the parameters of the peaks involved in the scoring, and [−b] is an optional flag that sets the function to score only the best matching peaks in the two buffers, as opposed to totaling the scores of all matching peaks. If s = 1 is taken to indicate the presence of the scaling flag and s = 0 its absence, and if b = 1 indicates the presence of the best match flag and b = 0 indicates its absence, then the score G_i, calculated by the *i*th bob function, shows the following dependence

$$G_i = \begin{cases} \max\limits_{a=1}^{A} g(C_a); & s = 1, b = 1 \\ \sum\limits_{a=1}^{A} g(C_a); & s = 1, b = 0 \\ A; & s = 0, b = 0 \\ 1; & s = 0, b = 1, A > 0 \\ 0; & s = 0, b = 1, A = 0 \end{cases}$$

A total score Γ_{jk} between two fragments F_j and F_k is obtained by taking the weighted sum of all I bob scores, obtained by the user-defined weights P_i

$$\Gamma_{jk} = \sum_{i=1}^{I} P_i G_i$$