Sequential Assignment of 2D-NMR Spectra of Proteins Using Genetic Algorithms

R. Wehrens,* C. Lucasius, L. Buydens, and G. Kateman

Laboratory for Analytical Chemistry, Catholic University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Received November 23, 1992

The application of genetic algorithms to the problem of the sequential assignment of two-dimensional protein NMR spectra is discussed. The problem is heavily underconstrained since in most cases more patterns are available than amino acid positions, and uncertainties may exist in the preliminary assignments. The results indicate that relatively large amounts of errors may be present in the input data for the genetic algorithm while useful results may still be obtained.

INTRODUCTION

The bottleneck in the calculation of the three-dimensional structure of proteins from NMR data lies in the spectrum interpretation. Several months of an expert's time may be required to obtain a preliminary interpretation that has to be refined further. Therefore, attempts have been made to automate the process, at least to the extent that the expert is aided in the interpretation process.[1-6] However, no systems are known that are able to obtain a complete interpretation for proteins of reasonable size without human assistance. This is largely due to the anomalities that occur in every spectrum, such as missing peaks because of noise or insufficient resolution, and atypical patterns. The flexibility and robustness that is necessary to solve a problem of this complexity can only be found in an iterative process, where an expert is able to switch back and forth to pursue his ideas freely. Such an iterative process is difficult to implement, however, and therefore most systems aim at providing a best first guess. These results may then be validated manually and included in the next run of the system.

Most systems apply the so-called sequential assignment strategy,[7] in which the interpretation consists of three stages. In the first stage the spin systems of the separate amino acids (hereafter denoted as patterns) are identified in the COSY spectra. Uncertainties can arise whether a peak does or does not belong to a specific pattern. Some programs solve this uncertainty by retaining two versions of the pattern: one with and one without the doubtful peak. Another reason for the number of patterns to be larger than the number of amino acids is the presence of *glycine* amino acids in the sequence. Because of the two C_α protons in a glycine, two patterns instead of one pattern may be identified.

In the second stage, these patterns are assigned to amino acid types. This is done on the basis of characteristic features of the patterns. However, in most cases it will not be totally clear to what kind of amino acid a pattern should be assigned, and therefore some errors are inevitable in this stage, too.

Finally, the patterns are assigned to amino acids at specific positions in the sequence. This is done by identifying a list of possible pattern pairs for each pair of vicinal amino acids in the chain. For a combination of patterns to be valid for an amino acid pair, the types of the patterns must match the types of amino acids, and a sequential connectivity in the NOESY spectra should be found. The pairs of patterns found in this way are then combined to map to the complete amino acid sequence.

The last part is especially difficult, since an exhaustive search will often lead to combinatorial explosion. In this paper, we propose a new approach to the last step that we will call, for convenience, the sequential assignment step, using genetic algorithms.[8] Genetic algorithms (GA's) are problem solvers that are very powerful in searching large solution spaces, mimicking the principle of "the survival of the fittest". Candidate solutions are ranked according to some evaluation function that is written for the specific problem, and successful solutions are allowed to reproduce with other successful solutions. This way, after several generations, the population consists of solutions that have inherited the strong parts of their ancestors and that hopefully form a better solution to the problem. In the past, genetic algorithms have been applied to a wide variety of problems,[9-11] especially to those for which an analytical solution was not possible. The genetic algorithm that is applied here is part of a hybrid expert system, HIPS.[12] It must be stressed, however, that any other system may be used to provide the necessary input files. Because of the nature of the input needed for the genetic algorithm, it is to be expected that most other interpretation programs will be able to produce the necessary information. The GA will be discussed thoroughly in the following section.

The data sets that have been used to test the performance of the genetic algorithm have been obtained by the HIPS program, using COSY, NOESY, DQSY, and RCT spectra. They contain the following information:

(a) The sequence of amino acids.

(b) A list of combinations of vicinal amino acids from the sequence.

(c) For each combination of amino acids, a list of combinations of spin patterns that may match the amino acids.

A sample input file is shown partially in Figure 1.

Ideally, for each combination of amino acids, the number of possible pattern combinations should match the number of occurrences of the amino acid combination in the sequence. In such a case, sequential assignment would be trivial and could easily be performed manually. In practice, however, the problems posed are much larger. In expert problem solving, it is often necessary to reject one's own hypotheses and start anew. This is due to the inherent uncertainty in composing the patterns (does this peak belong to the pattern or not?) as well as in assigning the patterns to amino acids (can this pattern be caused by a valine?). As already stated, most spectrum

```
SEQUENCE INFO >
  size :  58
  aa's : R P D F C L E P P Y T G P C K A R I I R Y F Y N A K A G L
         C Q T F V Y G G C R A K R N N F K S A E D C M R T C G G A

PATTERN INFO >
  nr of patterns :  64
  patterns:
0   CYS-51
1   PATTERN-3
2   GLY-56
  .

  .
63  PROLINE-4

COMBINATION INFO >
  nr of aa combinations: 54

  G  A :
  nr of pattern combinations: 25
27  3
56  3
  .

  .
55  28

  C  G :
  nr of pattern combinations: 16
32  1
35  1
  .

  .
41  59

  .

  .
```

Figure 1. Sample input file (partially shown) of the BPTI data set, obtained with global settings (see text). Pattern identifiers like CYS-51 are optional and are used only for validation purposes here.

interpretation programs handle this by allowing for multiple assignments, but this will increase the number of possibilities in the sequential-assignment step significantly. Moreover, the number of potential sequential connectivities in the NOESY spectra may be quite large, and overlapping patterns may pose difficult problems, too. Thus, the number of pattern combinations may be expected to exceed the necessary number by a large amount. These types of errors may be called *false positive* (FP) errors. Furthermore, some pattern combinations that actually are present in the "correct" sequence may be missing: *false negative* (FN) errors. A sequential connectivity may fall below the noise level, or a pattern may be assigned to the wrong type of amino acid. In such a case, the "correct" pattern combination will not be present in the input file for the sequential assignment module.

Systems that perform an exhaustive search for the best sequence of patterns are not very robust against both types of errors. A large number of FP errors will cause the search to diverge to such an extent that it is completely intractable. FN errors may cause the search to pursue false directions in an early stage, yielding spurious results. The genetic algorithm may be expected to be more robust against both types of error. Superfluous information in the form of FP errors will cause a lot of suboptima to be present in the solution space, but the GA has a reputation of being capable of finding global optima, even in very difficult search spaces.[8] Missing pattern combinations (FN errors) will of course prevent the system from finding a complete solution, but since a complete candidate solution is evaluated at once, nothing prevents the GA from placing the correct patterns at the following positions.

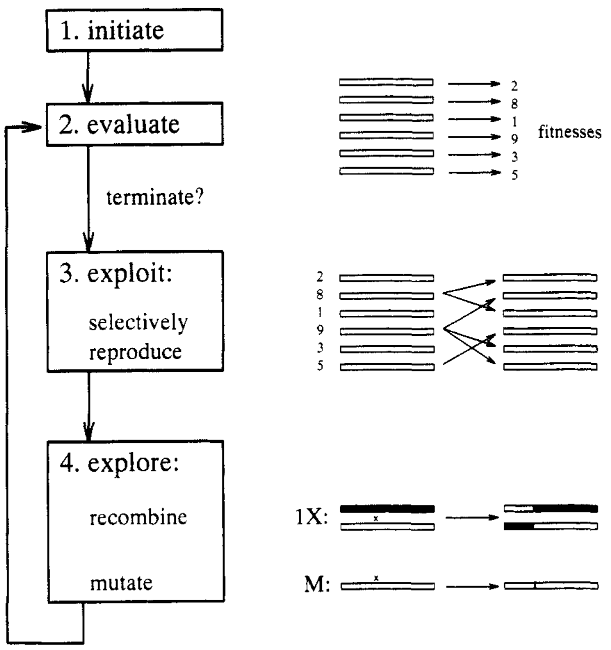

Figure 2. Flow chart of a genetic algorithm.

GENETIC ALGORITHMS

In this section, the fundamentals of genetic algorithms are reviewed briefly and the most important ideas and concepts are discussed. The principal idea is that of the survival of the fittest; a population of candidate solutions is evaluated, and only the best are allowed to reproduce. Each solution is represented as a bit string; each string evaluation yields a fitness. Solutions are ranked to their fitnesses and allowed to reproduce themselves with a probability proportional to their fitness. Reproduction takes place by means of the so-called crossing-over operator. In a crossover, random parts of the parent strings are combined to form a new child string. In this way, large parts of successful solution strings are combined to form new and hopefully even better solution strings. After reproduction, the parent strings are deleted to keep the number of strings constant. At the reproduction stage, some random mutations are introduced to keep the population from premature convergence. These concepts are illustrated in Figure 2. As the number of generations increases, and bad solutions are deleted, the fitness of the best member of the population may be taken as an indication of the success of the search. In Figure 3 a typical GA run is depicted. The evaluation criterion yields discrete fitness values that rapidly increase in the beginning of the search. After a while, an optimum is reached and the search can be terminated. The termination criterion is usually a maximum number of generations or a maximum fitness value.

Whereas the concepts are simple, a lot of small variations and extensions are possible. For instance, so-called two-points crossover operators may be defined that exchange parts of bit strings that are not terminal, and penalty functions may be included in the string evaluation in the case that the solutions become too much alike. Furthermore, crossover and mutation rates may be varied, along with population sizes, and scaling functions may be applied to the fitness. This makes a genetic algorithm a very flexible tool.

However, the most important issue in applying genetic algorithms is the representation of the problem. A suitable way must be found in which solutions can be represented as bit strings and in which they can be manipulated easily.

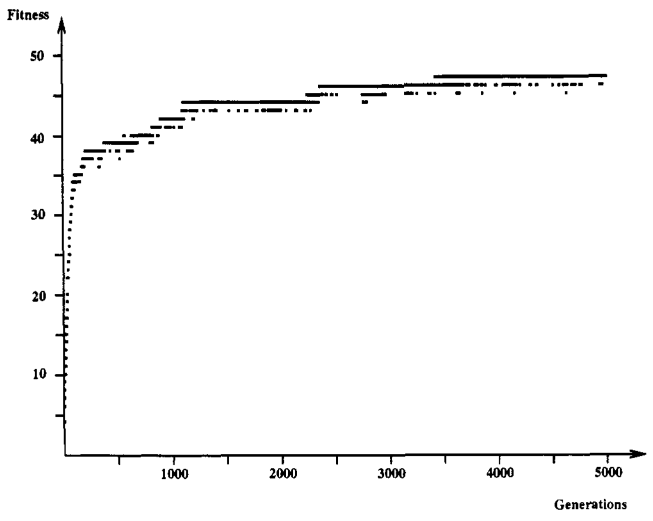

Figure 3. Performance plot of a genetic algorithm.

Furthermore, an appropriate evaluation criterion must be defined. If any of these elements is not optimal, the genetic algorithm will perform poorly. For numerical optimization problems, several established methods have been set up, and application of genetic algorithms in such cases is relatively straightforward.[9] However, subset selection and sequencing problems often require another set of primitives.[13]

Application to the Sequential Assignment. The sequential assignment problem is essentially a subset selection problem where also the sequence of the solution is of importance. In principle, N patterns should be mapped to M positions in the sequence (where $N > M$). The size of the search space is $N!/(N-M)!$. This number can be very large; as an example, one of our data sets consisted of a protein of 74 amino acids and 97 possible patterns, yielding a search space of approximately 10^{130} solutions! The genetic algorithm that is applied in this cases uses a specially developed subset encoding along with special crossover and mutation operators.[13] In short, each solution is represented as a permutation of the N possible elements. Only the first M elements are evaluated in the fitness function so that the subset selection takes place automatically. The crossover operator preserves position as much as possible: an element on a certain position in the parent string will be copied to the same position in the child string, whenever possible.[13] Mutation consists of random swapping of two elements on the bit string and is divided into a *reorder mutation* and a *trade mutation*. The former swaps two elements in the first M elements of the bit string; the latter swaps an element from the first M elements with an element of the last $N - M$ elements of the bit string.

The fitness criterion basically counts the number of pattern combinations in a candidate solution that are present in the list of possible pattern combinations (for each combination of vicinal amino acids). As an example, consider the amino acid combination GA (*glycine alanine*) at the end of the sequence in Figure 1. Twenty-five combinations of patterns are possible here, according to the input; whenever in a candidate solution one of these pattern combinations is at the last positions in the sequence, the candidate solution receives 1 point. In this way, a maximum fitness of $M - 1$ is possible. In that case, all pattern combinations match the combinations of amino acids to which they are mapped.

Throughout all experiments, the crossover rate was kept at a value of 0.8, the reorder mutation rate at a value of 0.02, and the trade mutation rate at a value of 0.01. The mutation rates are kept low to prevent the search from becoming random.

Table I. Characteristics of the Original Data Sets[a]

protein	settings	no. of patterns	no. of pattern pairs	no. of FN
E-L30	global	61	922	29
E-L30	optimal	61	1032	31
BPTI	global	64	1567	21
BPTI	optimal	63	1032	16
Tendami	global	97	4042	19
Tendami	optimal	90	2244	20

[a] E-L30 and BPTI both consist of 58 amino acids; Tendami consists of 74 amino acids. In the E-L30 data set, more than half of the "correct" pattern combinations are not present in the input file. The Tendami sets are difficult because of the large number of pattern combinations that are possible and the large number of superfluous patterns that have been found.

The values used for these parameters are in agreement with the optimal values found using an experimental design. The population size was usually set to 500, and a maximum number of generations of 3000 was used as a stop criterion. Fitnesses were scaled using a sigmoid function, to enable the genetic algorithm to make progress even in flat solution spaces.[13]

The programs were written using the GATES software[14] and were run on a SUN SPARC1 workstation. One run of 3000 generations usually took 4 h of real time.

RESULTS AND DISCUSSION

In this section, the results of a number of test cases with real and simulated data sets will be discussed. With these sets, we can determine the amount of incorrect superfluous, or missing data that the GA can handle in reaching a sensible solution. To be clear, it is not our claim that the GA will find the complete and correct sequence of patterns but rather a large part of the solution that then can be used to obtain the missing parts, either by hand or by a next iteration with the help of an expert system. The test cases serve as a means of illustrating this.

Data Sets. The data sets are derived from three proteins: E-L30, BPTI (bovine pancreatic trypsine inhibitor), and Tendamistat. Fabricated spectra of these proteins[5] were used in the assignment using the expert system HIPS.[12] In HIPS, the conclusions of the expert system can be evaluated using a database of solved cases, and suggestions can be done by the system to improve its performance. This way, several parameters in the expert system can be tuned to obtain optimal results. For each protein, two data sets were produced, one set in which the performance of HIPS was optimized for that particular protein, and one set in which global settings were used that were optimal for the total of the three proteins. These original sets are denoted "optimal" and "global", respectively. Thus, we obtained six realistic data sets (two for each protein), in which different amounts of FP and FN errors are present. These six data sets are described further in Table I. It should be noted that these sets have been obtained using the fully automatic mode of HIPS, and that the only manual intervention has been the training of HIPS to the three data sets to obtain a set of optimal settings for each protein and a set of global settings which yielded an overall maximal performance.

From these data sets, several others were constructed manually to test the abilities of the GA. First of all, the optimal data sets from Table I were pruned manually to reduce the number of possible pattern pairs. This is not an unrealistic situation since in real life the spectrum interpretation program will most probably be used by an expert who is capable of limiting the number of possibilities in an early stage. In this

Table II. Characteristics of the Constructed Data Sets[a]

protein	characteristics	no. of patterns	no. of pattern pairs	no. of FN
E-L30	optimal settings, pruned	61	218	30
	optimal settings, full	61	1063	0
	optimal settings, pruned, full	61	249	0
	global settings, full	61	929	0
BPTI	optimal settings, pruned	63	352	16
	optimal settings, full	63	1050	0
	optimal settings, pruned, full	63	370	0
	global settings, full	64	1590	0
Tendami	optimal settings, pruned	90	520	21
	optimal settings, full	90	2244	0
	optimal settings, pruned, full	90	533	0
	global settings, full	97	4042	0

[a] In case the number of FN errors increases by pruning, a pattern that has been inserted for a missing assignment in literature has been pruned away.

case, the pruning was continued for each combination of amino acids until less than ten possible pattern combinations were left. As the "correct" solutions of patterns of all data sets used here are known, this knowledge was used as the pruning criterion; if a pattern in a combination did not match the amino acid type in that combination, the pattern combination was removed. This pruning procedure ensures that the FP errors continue to be distributed evenly over all combinations of amino acids. In case an amino acid has not been assigned in the literature, a pattern that is not part of the sequence will be inserted at random. For the present purposes, they can be treated as if they were correct. Furthermore, all original data sets, as well as the pruned data sets, were provided with the correct pattern pairs that were missing (column no. of FN in Table I), again using the knowledge of the correct pattern sequence. This yielded nine more data sets in which the fitness of the true solution is the highest fitness possible. The data sets constructed in the above ways have been gathered in Table II.

EXPERIMENTAL SECTION

In each experiment, five runs with the GA were done. From each run, the solution with the highest fitness was selected. If two or more solutions had the same fitness, the last one generated was selected. An assignment of a pattern to a position is considered definite if in at least three of the five selected solutions the pattern is placed at that position. An assignment is considered possible if this is the case with two of the five runs. No attempt is made to check whether a pattern has been assigned to two places at once.

Furthermore, each data set was used in four experiments that used a slightly different fitness evaluation function. In the normal case, as has already been described, the number of correct pattern pairs present in the trial solution is counted. In the three other evaluation functions, extra points are given if two correct pattern pairs follow each other. That is, if the middle pattern of the three is possible in both pattern pairs. The amount that is added in such a case is varied: in this way we can see the influence of the evaluation on the eventual performance of the GA. In the first experiment, 0.5 is added for each pair of correct pattern pairs; in the second 1, and in the third 2.

The sets where the real solution has the highest fitness can be used to assess the sensitivity of the genetic algorithm to superfluous and misleading information, by validating its ability to reach a global optimum in a complex solution space.

Table III. Mean Fitnesses and Standard Deviations of Fitnesses in Five Experiments for the Data Sets[a]

protein	characteristics	"correct" F	mean F	s_F
E-L30	global settings	27	34.2	2.4
	optimal settings	29	33.8	2.8
	pruned	28	27.8	1.9
	global settings, full	57	41.0	5.6
	optimal settings, full	57	39.8	4.5
	pruned, full	57	54.5	2.5
BPTI	global settings	37	42.8	1.6
	optimal settings	42	40.4	2.4
	pruned	42	42.8	2.3
	global settings, full	57	45.8	2.7
	optimal settings, full	57	42.0	2.5
	pruned, full	57	49.2	7.9
Tendami	global settings	55	62.4	2.3
	optimal settings	54	61.0	2.9
	pruned	53	47.8	4.5
	global settings, full	73	66.2	3.0
	optimal settings, full	73	61.6	1.8
	pruned, full	73	54.6	5.7

[a] Fitness is defined here as the number of pattern pairs in the best candidate solution that are present in the pattern pair table in the data set. The "correct" fitness is the fitness of the "correct" solution.

In the original data sets of Table I, we can observe the effects of missing and incorrect information.

RESULTS

In Table III, the results of a series of five runs for the data sets are gathered. As can be seen, in the original data sets, in almost all cases fitnesses are found that are higher than the fitness of the correct solution. This indicates that a lot of local optima exist and that the optimum we are looking for will not even be the highest one. In the completed data sets, of course, the correct solution has the highest possible fitness, and the previous behavior is not observed. The column containing the standard deviations is also very interesting. In three or four cases, markedly high standard deviations are found when the five GA runs are compared: E-L30 with global and completed settings, BPTI with pruned and completed settings, and both incomplete and completed pruned settings of Tendamistat. The BPTI case is the result of one run where the GA got stuck in a local optimum. The standard deviation of the other four runs is much smaller. In the other cases, rather different fitnesses are found, and it appears that the GA has difficulties in finding the correct path to the solution. In the E-L30 case this may be caused by the large number of FN errors and, in the Tendamistat cases, by the large number of superfluous patterns that have to be mapped to the sequence of 74 amino acids. Using the Tendamistat data sets that have not been pruned, high fitnesses can be obtained because of the enormous amount of possible pattern pairs in the input sets. Thus, the number of local optima that are almost as high as the global optimum is very large. We will see shortly that this hypothesis is confirmed by the low number of actually assigned patterns in Tendamistat.

Results of the Original Data Sets. The original data sets in Table I present the most difficult cases since they contain the highest number of missing pattern pairs along with a large amount of superfluous pattern pairs. In Tables IVand V the definite and possible assignments, respectively, for these data sets are tabulated for the different evaluation functions. In all cases, the percentage of possible assignments that is correct is significantly smaller than the percentage of definite assignments. This is as expected, but in most cases the additional number of correct assignments is so small that there

Table IV. Definite Assignments for the Data Sets from Table I with Different Evaluation Functions[a]

data set	bonus = 0	bonus = 0.5	bonus = 1	bonus = 2
E-L30, optimal	9/16	10/16	4/11	10/18
E-L30, global	7/15	5/15	6/15	6/20
BPTI, optimal	18/20	23/28	18/23	18/22
BPTI, global	12/17	17/23	14/16	13/22
Tendami, optimal	6/19	5/18	4/10	3/8
Tendami, global	3/10	3/9	4/10	4/10
total	55/97	63/109	50/85	54/100

[a] The first number of each entry is the number of correct assignments; the second number, the total number of assignments. Bonus is the number that is added to the fitness of a solution in the case of two successive correct pattern pairs. The case where bonus = 1 yields the highest percentage of correct assignments (59%) as well as the smallest number of incorrect assignments.

Table V. Possible Assignments for the Data Sets from Table I with Different Evaluation Functions[a]

data set	bonus = 0	bonus = 0.5	bonus = 1	bonus = 2
F-L30, optimal	13/40	12/47	8/42	14/35
E-L30, global	13/38	9/39	7/31	8/43
BPTI, optimal	26/47	24/44	29/44	30/50
BPTI, global	14/40	28/50	18/39	15/41
Tendami, optimal	16/58	15/57	12/48	10/50
Tendami, global	9/47	9/44	9/53	8/37
total	91/270	97/281	83/257	85/246

[a] In this case, the highest performance is obtained where bonus = 2.

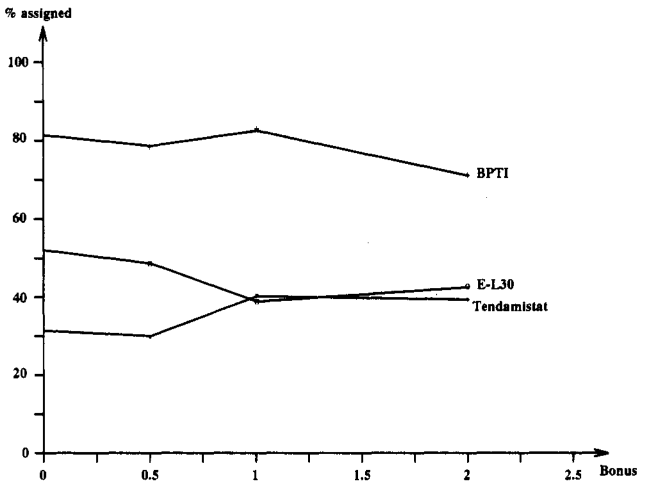

Figure 4. Percentage of correct definite assignments for the three proteins, using different values for the bonus parameter. These results are the mean over the optimal and global data sets (see text).

appears to be no benefit at all in bookkeeping the possible assignments. This will be consistently the case in all subsequent results, so for the sake of brevity we will not give the results of possible assignments further.

In Figure 4 the percentages of correct assignments are gathered for each protein, where the results using both the optimal and global data sets have been combined. It is clear that the GA performs best in the case of BPTI. In this data set, relatively few FN errors are present, and the load of some extra FP errors, compared with E-L30, does not seem to pose a problem. Here, roughly 80% of all assignments are correct. In E-L30, more patterns are incorrectly assigned because of errors in the input set. Tendami represents a different case because of the large amount of superfluous patterns and pattern pairs and the greater length of the amino acid chain. However, the difference between the global and optimal settings is small, especially in the possible assignments. It may be concluded

Table VI. Definite Assignments for the Pruned Data Sets from Table II with Different Evaluation Functions

data set	bonus = 0	bonus = 0.5	bonus = 1	bonus = 2
E-L30	6/26	18/29	18/28	16/23
BPTI	37/44	35/42	37/39	38/44
Tendami	29/35	26/34	22/32	14/23
total	72/105	79/105	77/99	68/90

Table VII. Definite Assignments for the Completed Data Sets from Table II with Different Evaluation Functions

data set	settings	bonus = 0	bonus = 0.5	bonus = 1	bonus = 2
E-L30	optimal	11/19	14/16	17/23	17/19
E-L30	global	25/32	27/29	24/31	19/20
E-L30	pruned	55/56	55/58	57/57	55/58
BPTI	optimal	18/24	36/39	23/27	33/38
BPTI	global	21/28	20/22	14/21	16/20
BPTI	pruned	49/52	51/51	51/53	52/54
Tendami	optimal	6/17	5/9	5/12	4/18
Tendami	global	5/18	9/19	6/11	8/16
Tendami	pruned	35/43	35/39	38/41	37/43

that both data sets are actually too indetermined to let the GA find a reasonable solution. The small numbers of definite assignments reflects the observed behavior that there are many ways in which a trial solution with a high fitness can be formed. Therefore, patterns are rarely assigned to the same place in two or three runs.

Results of the Constructed Data Sets. The constructed data sets from Table II present different situations to the genetic algorithm. In the manually pruned sets, much less superfluous pattern combinations are present, and in the full sets, all pattern combinations of the correct solution are present in the data set. Thus, with these sets, the sensitivity of the genetic algorithm with respect to false negative and false positive errors in the input can be investigated.

Comparison of the results of the pruned sets with the results of the optimal sets in Table IV gives an indication of the influence of FP errors on the eventual outcome. The results of the pruned sets are given in Table VI. Pruning clearly yields a significant performance improvement, as expected. Not only can the correct path to the global optimum be found easier, but also several other local optima are removed. The data sets where the missing pattern pairs are completed can be used to further assess the ability of the GA to find the global optimum, which in these cases comprises the correct solution. These results are gathered in Table VII.

DISCUSSION OF RESULTS

Original Data Sets. First of all, the results of the original data sets can be compared with each other (see Table IV). It is clear that the BPTI data sets give the best results. Despite significant numbers of FN, as well as FP errors in the input files, a reasonable assignment rate is achieved. In the case of E-L30, the number of correct assignments is much smaller, presumably because of the large number of FN errors; more than 50% of the correct pattern combinations is absent from the input files. The Tendamistat data sets present another difficulty: the number of possible pattern combinations is so large that a large number of sequences can be constructed that have an equal or even higher fitness than the correct sequence. In this case, many local optima of a very different nature can be found that have high fitnesses; and most of them have higher fitnesses than the correct solution (see Table III). This situation is adequately reflected by the very small number of assignments, of which, in many cases, only the *proline* assignments are correct. These are treated as dummies in the input data because HIPS does not assign proline patterns.[12]

Pruned Data Sets. Second, the results of the pruned data sets may be used to assess the performance of the GA in the case of only a limited number of FP errors (compare the optimal data sets in Table IV with those in Table VI). In almost all cases, the number of assignments as well as the percentage of correct assignments increases significantly. The difference is most markedly present in the case of Tendamistat, where the original data sets posed for the GA too big a problem; with the pruned Tendamistat data set approximately one-third of all positions is correctly assigned. Also the results on the E-L30 and BPTI data sets show a significant improvement. The interesting thing to note is that the largest pruned data set, Tendami, contains more than half the number of pattern combinations than the smallest original data set, and besides that a larger surplus of patterns for the available positions. The difference in difficulties presented by these data sets thus appears to be rather small. This notwithstanding, results of the pruned Tendamistat data set are significantly better than the results of the original E-L30 set. Almost 40% of the positions have been assigned correctly in the pruned Tendamistat case, whereas in the original data set of E-L30, the best performance consists of a correct assignment of 17% of the positions. However, the percentages of assignments that are correct differ only a little in these cases; it appears that a large number of FP errors predominantly prevents patterns from being assigned to a specific position. From this, a tentative conclusion may be drawn that there is some cutoff value for the number of FP errors above which the GA will not be able to yield useful results. On the basis of these data, one could estimate that the cutoff value would lie somewhere around 10–20 times the number of positions in the sequence. If more FP errors are present, no reliable assignments are to be expected.

Completed Data Sets. Third, the completed data sets may be compared with the incomplete sets to see how much the performance of the GA is hampered by FN errors. Of course, a completely correct interpretation is almost impossible in the presence of such errors, but, as already said, it is already a significant aid in the spectrum interpretation if large parts of the sequence have been assigned. As can be seen in Table VII, the effect of removing the FN errors is largest in the pruned data sets, where the FP errors do not play a predominant role and the number of assignments already is significant. In the pruned E-L30 and BPTI cases, 85–100% of the positions is assigned the correct pattern. In the Tendamistat data set, this figure lies around 50%, indicating that the number of approximately 450 FP errors is still too big to achieve a complete assignment. In the original data sets, not so much of an improvement in the number of assignments has been reached, but more of an improvement in the rate of correct assignments. However, the Tendamistat data sets still are too difficult for the GA to find the global optimum, that now constitutes the real solution.

Evaluation Function. Finally, some comments on the evaluation function may be made. The representation of a candidate solution and its evaluation criterion should not only represent the merits of that solution but should also enable the crossover operator to combine useful parts of different solutions so that an even better one is obtained. This implies that the so-called "fitness landscape", that is effectively sampled by the GA, should have a more or less smooth surface. If it is completely flat with one sharp spike, containing the solution, it is extremely unlikely that search methods will find the optimum, given solution spaces of the current magnitude. On the other extreme, fitness landscapes with a very ragged surface without any apparent coherence will also give bad and unreliable results. In such a case, the GA will not perform much better than a random search. In general, it is best to leave the evaluation function as simple as possible, since complicated evaluation functions often have a tendency to roughen the fitness surface. For instance, the first evaluation function that was tried consisted of a part in which probabilities of patterns belonging to certain types of amino acids were combined with the presence or absence of sequential crosspeaks. Although the fitness gave a very good picture of what constituted a good solution, the results with the GA were very poor.

The current evaluation function, however, is much simpler. The "correct" solution has the highest fitness, but there are a lot of partially correct solutions that approach this fitness. This is essential for a successful operation of the GA. However, it was thought worthwhile to try the enhancement of the evaluation function by rewarding multiple vicinal pattern combinations that were present in the input set. It can be seen, however, that the addition of a bonus in such a case does not seem to have much influence. Although in some cases better results may be obtained, especially when averaging over the three proteins, the general trend is not very convincing, and further performance improvements are more likely to appear if the quality of the input sets is improved. A further improvement could be the inclusion of more GA runs to determine whether or not a pattern is definitely assigned to a specific position.

CONCLUSION

In this paper, the possibilities of using genetic algorithms in the sequential assignment of NMR protein spectra have been assessed. It can be concluded that, provided the data sets are of sufficient quality, good results can be obtained. The amount of errors that is permitted in the data sets has been investigated. Useful results (one-third of the amino acid positions assigned correctly) may still be obtained if the number of pattern combinations exceeds the necessary number of amino acid combinations with an order of magnitude. These FP errors mainly affect the number of assignments that is made by the GA, and not the rate of correct assignments. Missing information (FN errors) mainly affects the quality of the assignments, as may be expected. If a pattern combination is absent from the input data set, the GA will try to fit in a false combination that will be rewarded in the evaluation phase. Data sets with up to 50% missing pattern combinations have been evaluated, with the result of 10–30% of the positions correctly assigned, depending on the number of FP errors. If the number of FP errors is small enough, and the number of FN errors is zero, performances of nearly 100% may be obtained.

The above results indicate that the GA is a promising technique to be used in the automation of the spectrum interpretation process. Especially in cases where an amount of errors is present that would prohibit other techniques from producing any results at all, the GA may still be able to assign a part of the sequence correctly. The quality of the solutions presented by the GA may be assessed by investigating the fitnesses of the solutions as well as the number of assignments that can be made from the runs. Because of the large robustness against both FP and FN errors, it is expected that this technique can be coupled effectively to expert systems or other programs, where uncertainties may lead to a large number of possibilities and incorrect conclusions. However, it must be borne in mind that the spectra from which the data sets were derived have been simulated, albeit as realistically as possible. Results with real spectra must be obtained to adequately assess the power of the method and to be able to compare it with other methods.

REFERENCES AND NOTES

(1) Catasti, P.; Carrara, E.; Nicolini, C. Pepto: an expert system for automatic peak assignment of two-dimensional nuclear magnetic resonance spectra of proteins. *J. Comput. Chem.* **1990**, *11*, 805–818.

(2) Cieslar, C.; Clore, G. M.; Gronenborn, A. M. Computer-aided sequential assignment of protein ^{1}H NMR spectra. *J. Magn. Reson.* **1988**, *80*, 119–127.

(3) Eads, C. D.; Kunz, I. D. Programs for computer-assisted sequential assignment of proteins. *J. Magn. Reson.* **1989**, *82*, 467–482.

(4) Kleywegt, G. J.; Lamerichs, R. M. J. N.; Boelens, R.; Kaptein, R. Toward automatic assignment of protein ^{1}H NMR spectra. *J. Magn. Reson.* **1989**, *85*, 186–197.

(5) van de Ven, F. J. M. PROSPECT, a program for automated interpretation of 2D NMR spectra. *J. Magn. Reson.* **1990**, *86*, 633–644.

(6) Yu, C.; Hwang, J.-F.; Chen, T.-B.; Soo, V.-W. RUBIDIUM, a program for computer-aided assignment of two-dimensional NMR spectra of polypeptides. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 183–187.

(7) Wüthrich, K. *NMR of proteins and nucleic acids*; Wiley: New York, 1986.

(8) Goldberg, D. E. *Genetic algorithms in search, optimization and machine learning*; Addison-Wesley: Reading, MA, 1989.

(9) Davis, L., Ed. *Handbook of Genetic Algorithms*; Van Nostrand Reinhold: New York, 1991.

(10) Lucasius, C. B.; Kateman, G. Understanding and using genetic algorithms. Part 1: Concepts, properties and context. *Chemometrics and Intelligent Laboratory Systems*; 1993, in press.

(11) Lucasius, C. B.; Kateman, G. Understanding and using genetic algorithms. Part 2: Representation, configuration and hybridization. *Chemometric and Intelligent Laboratory Systems*; 1993, submitted for publication.

(12) Wehrens, R.; Lucasius, C.; Buydens, L.; Kateman, G. HIPS, a hybrid self-adapting expert system for NMR spectrum interpretation using genetic algorithms. *Anal. Chim. Acta*, in press.

(13) Lucasius, C. B.; Kateman, G. Towards solving subset selection problems with the aid of the genetic algorithm. In *Proceedings of the 2nd Workshop on Parallel Problem Solving from Nature*, Brussels; Männer, R., Manderick, B., Ed.; Elsevier: Amsterdam, 1992; p 239.

(14) Lucasius, C. B.; Kateman, G. GATES: Genetic Algorithm Toolbox for Evolutionary Search, Software Library in ANSI C, Laboratory for Analytical Chemistry, Catholic University, Nijmegen, 1991.