Performance Enhancement of Vector-Based Search Systems: Application to Carbon-13 Nuclear Magnetic Resonance Chemical Shift Prediction

Robert C. Schweitzer and Gary W. Small*

Center for Intelligent Chemical Instrumentation, Department of Chemistry, Ohio University, Athens, Ohio 45701-2979

Received June 27, 1995[⊗]

A database partitioning algorithm is described for use in increasing the speed of vector-based search systems. When implemented with a library of *n*-dimensional vectors to be searched, the algorithm subdivides the *n*-dimensional data space into multiple levels of *n*-dimensional boxes. The vectors lying within each box are catalogued, and the search for the nearest matches to a target vector is reduced to a search of only the box in the data space containing the target vector and the immediate neighboring boxes. The use of multiple layers of boxes serves to reduce the computer memory requirements for implementing the partitioning scheme while still allowing the number of vector comparisons in the search to be minimized. In the work presented, this algorithm is applied to the prediction of carbon-13 nuclear magnetic resonance chemical shifts. A database retrieval system for chemical shifts is implemented based on encoding the chemical environments of carbon atoms into a seven-dimensional vector representation. The chemical shifts of target carbon atoms are estimated by performing a search of the database for carbon environments that match the targets. The performance enhancement afforded by the partitioning algorithm as well as the search accuracy is found to depend on three design variables. Employing a library of 133 533 environment vectors, these variables are optimized for the chemical shift prediction application by performing searches on sets of 39 074 and 3900 test carbons. Based on the relative number of vector comparisons required to implement the searches with and without partitioning, speed increases by factors of 34.1 and 19.9 are realized for searches in which the 5 and 200 nearest matches are found, respectively.

INTRODUCTION

The growth in size and availability of computer-readable spectral databases has led to a variety of techniques for automated spectral interpretation.[1] Among the most widely used of these methods are vector-based search systems in which entries in the spectral database are coded as multidimensional vectors, and database retrievals are performed through the use of vector-based comparison metrics such as the Euclidean distance. This basic database retrieval approach has been used for a variety of applications such as spectral similarity searches and spectral prediction.[1]

One major application area for automated spectral interpretation algorithms is carbon-13 nuclear magnetic resonance spectroscopy (^{13}C NMR). This technique is widely used in the solution of organic structure elucidation problems due to its great sensitivity to changes in the carbon skeleton of a molecule. This sensitivity to structural changes makes the interpretation of ^{13}C NMR spectra challenging and has led to interest in the development of computational aids to spectral interpretation. One of these approaches, computer-based spectrum simulation, encompasses a set of tools that enables the spectra of candidate structures for an unknown to be simulated based on the encoding of relationships between chemical structural characteristics and experimentally observed chemical shifts. Candidate structures can subsequently be confirmed or rejected through comparisons between the simulated spectra and the spectrum of the unknown.

The two most common approaches to implementing spectrum simulation are direct database retrieval methods[2−7] and empirical modeling techniques.[8−12] The direct retrieval approach employs a database of structures and their experimentally observed and assigned spectra. Each carbon atom in this database is encoded and stored in a manner that correlates the chemical environment of the carbon to its associated chemical shift. To predict each chemical shift in a simulated spectrum, the database is searched for the carbon atom which best matches the chemical environment of the carbon whose predicted chemical shift is sought. The predicted chemical shift is taken as the experimentally observed chemical shift of that closest match. This method can generally predict chemical shifts of carbon atoms in a wide variety of chemical environments, but the accuracy of the predicted chemical shift is often limited.

In the empirical modeling method, mathematical models are computed that relate chemical structural features to observed chemical shifts. In most cases, the models used are of the form

$$s_a = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n \quad (1)$$

where s_a is the chemical shift of atom a, the x_i terms are calculated numerical structural descriptors that encode some aspect of the chemical environment of atom a, and the b_i are weighting coefficients. In the generation of the model, a set of atoms with known chemical shifts is used, and the b_i terms can be calculated by use of regression analysis techniques. Once computed, the models can be used to predict unknown chemical shifts. The empirical modeling method can often predict chemical shifts to an accuracy of

* Author to whom correspondence should be sent. Phone: (614) 593-1748. small@helios.phy.ohiou.edu.
⊗ Abstract published in *Advance ACS Abstracts,* November 1, 1995.

better than 1 ppm, but a given model is only applicable to a narrow range of chemical environments. Pretsch and coworkers have attempted to address this problem through the development of models which sacrifice some accuracy to gain a wider range of applicability.[13−14]

Research in our laboratory is focusing on the combination of the two approaches to gain the relative generality of the database retrieval method and the accuracy of the empirical modeling technique. As in the database retrieval method, the initial step of this combined approach requires the creation of an atom-based database in which each of the carbon atoms is encoded according to its chemical environment and stored along with its experimentally observed chemical shift. For each carbon atom in a given input structure, the direct database retrieval approach is used to select a subset of carbon atoms in the database with chemical environments similar to that of the target carbon atom. This subset of carbons is used to build a model of the type described above, with which the chemical shift of the target carbon atom can be calculated.

One key to the practicality of this approach is the speed with which these steps can be performed. Since the database retrieval step must be performed for each carbon atom in the input structure and the closest matches are drawn from the entire atom-based database, the retrieval algorithm used must be as efficient as possible. In our work, this database searching procedure is based on encoding carbon atom environments in a multidimensional vector format and then employing a conventional vector-based search system to retrieve the most closely matching environments to a desired target carbon.[15] A limitation of this method is the significant computational overhead associated with the requirement to perform potentially hundreds of thousands of Euclidean distance calculations to implement the vector comparisons. There are several strategies which could be used to speed up this vector-based search. One such approach is a hierarchical method which was introduced by Zupan and applied to the retrieval of infrared spectra.[16,17] This method uses a binary tree structure to order the vectors in the database and to allow a rapid retrieval of the closest match for an input vector. The applications of this method in the literature have used databases on the order of only 1000 vectors, however, and it appears that the method would not lend itself well to databases containing hundreds of thousands of vectors. To address this problem, the research described in this paper implements a database partitioning strategy that can easily accommodate large numbers of vectors and represents a general solution to enhancing the speed of vector-based search systems of the type used in our database retrieval approach.

EXPERIMENTAL SECTION

The Sadtler ^{13}C NMR database consisting of 29 966 structures and their accompanying chemical shifts was used for this research (Bio-Rad Laboratories, Inc., Sadtler Division, Philadelphia, PA). From this set of structures, a subset of 21 199 structures was chosen for the current work based on the presence of atoms supported by the environmental encoding algorithm used in this work (C, H, O, N, P, S, F, Cl, Br, I). The data set included organic compounds varying in size from 1−40 carbons and included a large number of different functional groups. The total number of carbons in the 21 199 structures was 241 275. This data set was divided randomly into two sets of structures. The larger set contained 16 959 structures and was used to define a “library” of carbon atom environments for the retrieval experiments. The smaller set of 4240 structures was used as a test set.

The computations described in this paper were implemented on a Silicon Graphics 4D/460 computer system running under Irix (version 5.2, Silicon Graphics, Mountain View, CA) and operating in the Center for Intelligent Chemical Instrumentation at Ohio University. The software used in this work was developed by use of a combination of FORTRAN 77 and C. Details regarding the software organization are provided in the supporting information accompanying this paper.

RESULTS AND DISCUSSION

Environmental Encoding Algorithm. Retrieval of the most structurally similar carbons from a database requires an encoding scheme for the chemical environment of each carbon. The purpose of this scheme is to represent the chemical environment of carbon atoms in a manner that relates to the chemical shift and in a way that facilitates quantitative comparisons between environments. The algorithm used in this work was developed by Small and Jurs[18] and has been used previously to implement chemical shift retrievals.[15] In this scheme, the environment of carbon atom a is represented as an $n+1$ dimensional vector of the form

$$\mathbf{e_a} = (e_0, e_1, ..., e_n) \tag{2}$$

where the first dimension, e_0, characterizes atom a, the second dimension, e_1, describes the influence of atoms one bond removed from atom a on its chemical shift, and each successive dimension describes the collective influence of the atoms one bond further from atom a on the chemical shift of that atom. Experimental chemical shifts of small molecules were used to define a set of parameters for use in computing the elements of $\mathbf{e_a}$.[18] Given these general parameters, the environment vector for a carbon atom can be derived directly from the chemical structure. No experimental chemical shift information for the carbon atom is required. In the majority of the work reported here, n in eq 2 was set to 6, defining a seven-dimensional vector.

Assembly of Atom Database. Environmental vectors were calculated for each carbon atom in the two sets of structures described previously. All duplicate carbons (carbons with the same environmental vectors and chemical shifts) were removed from the environment library. This ensured that no single environmental vector was weighted more heavily in the database than any other vector. Duplicate carbons were also removed from the test set but only within individual structures. This ensured that a complete simulated spectrum could be obtained for each structure. After removing duplicates, the number of library carbons was 133 533, and the number of carbons in the test set was 39 074. Each of the vectors in both the library and test sets has an experimental chemical shift associated with it. However, the purpose of the test set is to simulate the case in which a user inputs a structure in which the chemical shifts are unknown and are to be predicted. Thus, although each of the test vectors has an associated experimental chemical shift, the testing procedure used here assumes that no such

Table 1. Partitioning of Seven-Dimensional Data

dimension	min.	max.	range	bins[a]
1	−9.28	127.25	136.53	18
2	−22.22	265.11	287.33	36
3	−1.04	29.77	30.82	4
4	−0.27	6.70	6.97	1
5	−0.08	2.14	2.22	1
6	−0.02	0.84	0.86	1
7	−0.01	0.40	0.41	1

[a] Partitioning based on a bin size of 8.0.

shift exists in order to test the ability of the retrieval algorithm to predict chemical shifts accurately.

Overview of Data Space Partitioning. The database retrieval is implemented by performing Euclidean distance comparisons between the environment vectors in the test and library sets and finding the nearest matches (i.e., smallest distances) in the library for each test vector. The chemical shifts of these library vectors are then assigned to the test vectors as the predicted chemical shifts for the corresponding carbon atoms. With this procedure, each test vector must be compared to each of the library vectors.

The goal of the work described here was to increase the speed of the library retrieval by reducing the number of required vector comparisons. The method used is a data space partitioning technique which allows the correct nearest matches to be found while requiring that the test vector be compared only to a judiciously chosen subset of the library vectors. The first step in this scheme is the creation of a data structure to which each library vector is assigned in a way that corresponds to its actual position in the *n*-dimensional data space. This step will be discussed in detail in this section. After this data structure has been built and every library vector has been assigned to it, the nearest matches for a given test vector can be found. The data structure creation is based on partitioning each dimension in the data space formed by the library vectors into bins of equal size, where the size is specified in the units of the e_i values in eq 2. In the present application, the data space is thus divided into a number of equally sized seven-dimensional boxes, where seven is the dimensionality of the environment vectors. The boxes are constructed of equal size so that any vector in the data space can be mapped easily into the box to which it belongs for comparisons to the vectors in that box.

The number of boxes generated by the use of a bin size of 8.0 is described in Table 1. The number of bins in each dimension is dictated by the bin size and the maximum and minimum values found in that dimension across the library of environment vectors. The total number of seven-dimensional boxes in the data space is simply the product of the number of bins in each dimension ($18 \times 36 \times 4 \times 1 \times 1 \times 1 \times 1 = 2592$).

After the number of bins in each dimension of the data space has been computed, every vector in the library set is assigned to its corresponding box in the data space. For a given vector, this assignment is based on the calculation of the bin number for each component of the vector. The bin number for component i of vector a, $b_{a,\mathrm{i}}$, is defined as

$$b_{\mathrm{a,i}} = (e_{a,\mathrm{i}} - \mathrm{min_i})/s \tag{3}$$

where $e_{a,\mathrm{i}}$ is the value of component i of vector a, $\mathrm{min_i}$ is the minimum value in dimension i computed across the library, and s is the predetermined bin size, which is equal in every dimension. The computed value of $b_{a,\mathrm{i}}$ is truncated to an integer which indicates the bin number. The first bin in each dimension is numbered zero. This mapping operation is repeated for each vector in the library and results in a list of library vectors for each box in the data space. For example, the first library vector is [101.00, 142.88, 8.91, 1.69, 0.77, 0.0, 0.0]. The minimum values of each dimension of the data space are found in column 2 of Table 1. For a bin size of 8.0, eq 3 can be used to calculate the first bin as $(101.00 - (-9.28))/8.0 = 13$. By repeating this operation for each dimension, the vector is assigned to the box whose coordinates are [13, 20, 1, 0, 0, 0, 0]. The order in which library vectors are assigned to the data structure is arbitrary since the *n*-dimensional box to which a given vector is assigned is not dependent upon the order in which the library vectors are assigned.

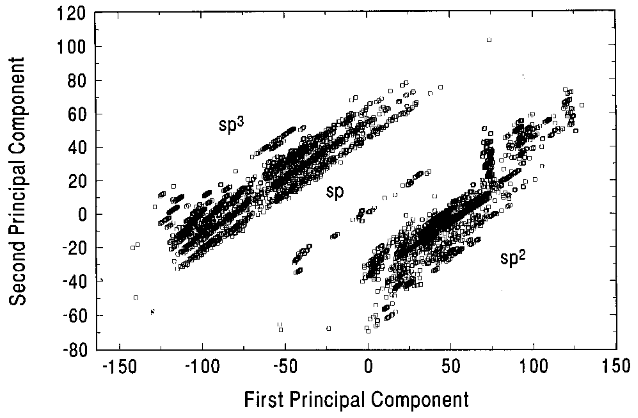

Figure 1. Principal components score plot generated from a subset of 59 501 of the 133 533 seven-dimensional library vectors. As labeled in the figure, the vectors cluster into three broad groups based on the hybridization of the carbon whose environment is described.

For the data space partitioning with a bin size of 8.0, only 521 of the 2592 boxes (20%) have library vectors assigned to them. For a bin size of 2.0, there are 1.27×10^6 total boxes, and of these only 6302 (0.5%) are occupied. The large variance in vector density reflected by the number of unoccupied boxes is related to the fact that carbon atoms are restricted to a limited number of chemical environments. The distribution of vectors in the seven-dimensional space is illustrated in Figure 1 through the use of principal components analysis (PCA). PCA is a data reduction technique which allows the generation of accurate low-dimensional representations of multidimensional data. In the figure, the scores along the first two principal components computed from a random subset of 59 501 of the library vectors are plotted. The first two principal components account for 99.7% of the data variance, confirming that the figure is an accurate portrayal of the relative vector positions in the seven-dimensional space. The regions of the space corresponding to sp^3, sp^2, and sp carbons are labeled in the figure. The variation in vector density in the data space is clearly observed.

After the assignment of the library vectors to the correct boxes, the retrieval of closest matches for carbon atoms in the test set can be performed. To find the nearest matches for a given test vector, the box in the data space to which that test vector maps is found, and the test vector is compared to each of the library vectors in that box and its immediate neighboring boxes. The database search is thus reduced to a comparison of the library vectors which are close to the test vector rather than a comparison to all vectors in the data space.

Three issues are paramount in judging the success of this approach. First, the correct matches must be found through the partitioning scheme. Second, a significant increase in the overall speed of the library retrieval must be realized in order to justify the additional computational overhead and software development time associated with implementing the partitioning algorithm. Finally, the partitioning algorithm must be acceptable in terms of the computer memory required to build the data structure which implements the partitioning of the data space. There is a tradeoff between the amount of memory used and the size of the partitions in the data space. The smaller the bin size (s in eq 3), the smaller the sizes of the boxes in the data space, the smaller the number of library vectors assigned to a given box, and thus the smaller the number of vector comparisons required to find the closest matches for a given test vector. As the bin size is decreased, however, more memory is required since each box in the data space requires one word of memory to serve as a status indicator and pointer to the list of vectors contained in that box. For the data space used in this work, bin sizes of 1.6, 2.0, 4.0, and 8.0 correspond to a requirement of 12.4, 5.08, 0.16, and 0.013 Mb of memory, respectively. From these data, it can be seen that the amount of memory required increases geometrically with decreasing partition size. However, the number of vector comparisons required to find the top p matches is small at a small bin size and increases with increasing bin size. Thus, the tradeoffs among these criteria must be explored if an optimal implementation of the partitioning concept is to be devised.

Multilevel Partitioning of Data Space. The solution to the memory problem described above is to partition the data space into greater than one level of boxes. Under this scheme, the top level of boxes would have a relatively large bin size so that as many boxes as possible at the top level would have library vectors in them. Since there are areas of the data space which have a large density of vectors, some of the boxes will have a large number of vectors. Each of these dense boxes can be partitioned into smaller boxes and thus treated as a new data space. This partitioning process can be repeated recursively for some finite number of levels, thus allowing very dense areas of the data space to be divided into very small boxes. Other areas of the data space that have few vectors would not need to be partitioned further. By having multiple levels of boxes, the number of vector comparisons required is kept low since very dense areas of the data space are divided into very small boxes. At the same time, the amount of memory required is minimized since only those relatively few areas of the data space which are dense have a large number of boxes. Under this scheme, three parameters require optimization: (1) the number of levels of partitioning to be used, (2) the number of library vectors required to define a given box as dense and thus subject to further partitioning, and (3) the number of bins to use in the lower levels for subdividing each dimension of the data space.

The number of bins into which to divide the new data spaces is dictated by memory considerations and indirectly by the number of dimensions in the vector. The advantage

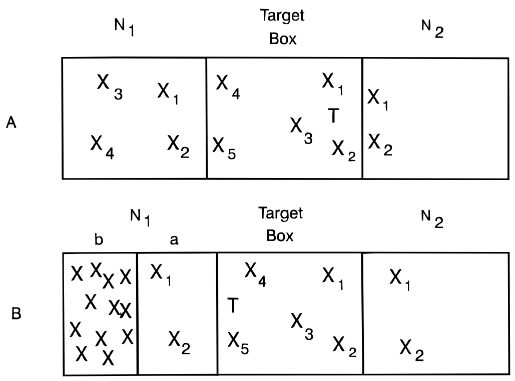

Figure 2. A. Illustration of the necessity for searching neighboring boxes to the target box. The two vectors in box N_2 are closer to the target vector, T, than are two of the vectors in the target box. B. Illustration of the requirement that if a neighboring box at a deeper level than the target box is searched (e.g., box N_{1a}), a minimum number of vectors must lie in that neighboring box in order to help guarantee that some nearest matching library vectors will not be overlooked. In this example, the small number of vectors in box N_{1a} dictates that some nearest matches will be located in box N_{1b}.

of dividing each dimension into a large number of bins is that fewer levels of boxes are needed. For the case in which each dimension is divided into four bins, the bin sizes for successive levels below the top level could be assigned as 8.0, 2.0, 0.5, 0.125. Seven levels would be needed to reach the same smallest bin size using a division into two bins rather than four for each dimension (8.0, 4.0, 2.0, 1.0, 0.5, 0.25, 0.125). The relative practicality of these two schemes is determined by the vector dimensionality used. For example, if a relatively high-dimensional vector (e.g., 13) is used to increase the selectivity of the environmental encoding, the effect on memory utilization is dramatic. The amount of memory required for a division into four bins is prohibitive for a 13-dimensional vector (4^{13} * 4 bytes/box = 268 Mb). However, a division into two bins (2^{13} * 4 bytes/box = 32.7 Kb) is manageable on most modern workstations in which 32 Mb or more of memory is commonly available. If a target memory utilization of 15 Mb is specified for the data structure, approximately 450 such boxes can be partitioned in this way. Experimental evidence suggests that a limit of approximately 450 dense boxes is realistic for the data spaces used in this work. For this reason, the work described here will be based on a subdivision of dense boxes into two bins in each dimension.

Algorithm for Search of Neighboring Boxes. Given an optimal design for the partitioning of the data space, the principal remaining design decision focuses on the algorithm used to determine which boxes will be searched. Based on the partitioning design used, a given test vector will map into a box termed the target box. This target box will reside at some level of the partitioned data structure based on the density of vectors in the corresponding region of the data space. Euclidean distance comparisons of the test vector and library vectors thus begins with the library vectors residing in the target box.

As Figure 2A illustrates, however, some of the closest matches can be missed if only the target box is searched. In this example, a two-dimensional data space contains library vectors which are labeled X and a test vector labeled T. The figure depicts the target box and two of the neighboring boxes, labeled N_1 and N_2. The data space is to be searched for the five library vectors which are closest to T. There are five library vectors in the target box, but it can be seen clearly that the two library vectors in box N_2 are closer to the test vector than vectors X_4 and X_5 in the target box. The closer vectors would be missed if only the target box were searched.

The general search strategy employed here is to search all of the vectors in the target box and in each neighboring box. This strategy rests upon several conditions which must be met. First, the target box must contain the same or greater number of vectors as the number of closest matches desired. Under this scheme, if a test vector maps to a box in a deeper level that has insufficient vectors, the larger parent box at the next higher level is used for the search. A complicating case occurs if the test vector maps into a box at the top level which contains insufficient vectors. The boxes at the top level are large enough, however, such that any vectors not within the target box or a neighboring box are most likely associated with carbons so dissimilar from the target carbon as to be of little use. Thus, if the top level has insufficient vectors, the top level target box is searched along with its neighbors, and the user must be satisfied with a potentially incomplete list of the closest matches.

A second condition is that each of the neighboring boxes must be searched and must be of the same size (i.e., be at the same level) as the target box. As illustrated in Figure 2B, if a smaller neighboring box is searched, some of the closest matches could be missed. This example is similar to the case depicted in Figure 2A, with the exception that box N_1 is dense in Figure 2B and is therefore partitioned into two smaller boxes of equal size, labeled N_{1a} and N_{1b}. The distribution of vectors in N_1 is such that the majority of vectors lie in N_{1b}. In this case, a search of only the smaller box (i.e., N_{1a}) that is adjacent to the target box will result in missing some of the closest matches.

It was determined through experimentation, however, that the number of vector comparisons can be greatly reduced if smaller neighboring boxes can be searched. As a first approximation, the second condition described above can be violated provided that if a neighboring box is searched which is at a deeper level than the target box, that neighboring box must contain some minimum number of vectors. Otherwise, the larger neighboring box located at the same level as the target box is searched. In practice, this minimum value must be greater than or equal to the number of closest matches sought in order to ensure that vectors located in the nonadjacent smaller box (e.g., N_{1b} in Figure 2B) are not significant in the search.

A complete treatment of the neighboring box problem requires consideration of the issue of data space dimensionality. In one dimension, a target box has two neighbors. In n dimensions, a target box has $3^n - 1$ neighbors. For example, in 13 dimensions, a target box has 1.6×10^6 neighboring boxes, whereas in seven dimensions, a target box has 2186 neighboring boxes. Although the vast majority of the neighboring boxes have no vectors, the search algorithm must include a check of each box to see if it contains any vectors. When the dimensionality is 13, this traversal of each neighboring box accounts for a significant amount of the total time spent in the search. In seven dimensions, however, the majority of the total computational time is spent performing vector comparisons. The neighboring box traversal problem in cases of high dimensionality can be alleviated in the initial creation of the library by building a list of every occupied neighboring box for every occupied box in the data space. Then, when a test vector is mapped to a particular box, a list is present of all the neighboring boxes which need to be searched. Instead of checking a huge number of neighboring boxes, only those boxes which contain vectors are searched.

The importance of this step can be illustrated by considering an example implementation. When these lists of neighboring boxes were built for the 13-dimensional case with eight levels of boxes, a density threshold of 500 vectors required for partitioning into a deeper level, and a bin size of 16.0 at the top level, the maximum number of occupied neighboring boxes for any occupied box in the data space was 199. The amount of computational time required to build the data structure and collect the top five matches for each of the 39 074 vectors in the test set was approximately 3.5 h and required 2.13×10^8 vector comparisons. The actual search time was approximately 3 h. The same search conducted without the benefit of the neighbor list took approximately 8 h.

There is one complication which can occur when the searching algorithm uses a list of occupied neighboring boxes for any given target box. If a test vector maps into a box at the top level in which there are no library vectors, there will be no neighbor list for that box since neighbor lists are only built for occupied boxes. In this case, no matches will be found for the test vector. This problem could be solved for these cases by not using the neighbor list approach for target boxes at the top level with no library vectors, by searching the entire data space, or by increasing the size of the boxes in the top level. If there are no library vectors in the target box, however, it is most likely that none of the closest matches will be similar enough to be useful for a chemical shift prediction.

Evaluation of Partitioning Algorithm. The partitioning algorithm and search procedure described above were implemented in computer software and tested to evaluate the increase in search speed obtained through the use of the procedure and to explore the optimal variable settings for use in the partition design. The three variables studied were the number of levels of boxes used, the number of library vectors required to define a box as dense and thus subject to further partitioning (termed the density threshold), and the minimum number of vectors required for the search of a neighboring box that is smaller than the target box. In all experiments, a seven-dimensional data space was used, and the bin size for the top level of boxes was 8.0. In all except one experiment, the search was implemented to find the top 200 matching library vectors for each vector in the test set. In one experiment, only the top five matches were found. Two criteria were used for judging the success of the partitioning algorithm: (1) the increase in speed obtained relative to the search of the full database and (2) the number of discrepancies encountered between the search results obtained with and without partitioning. The speed increase was measured in terms of the total number of vector comparisons required to find the p nearest matching library vectors for the set of test vectors used.

In the first set of experiments, the number of levels of boxes in the seven-dimensional data space was varied from 1−7. Figure 3 shows the results for the case in which the top five matches were found for the full set of 39 074 test

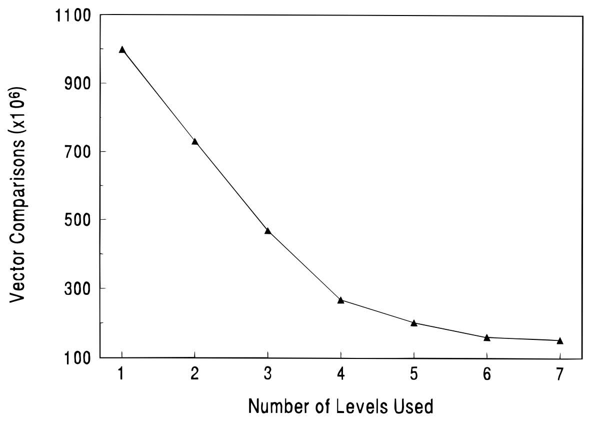

Figure 3. Plot of number of vector comparisons required to find the five nearest matches for the 39 074 test vectors vs the number of levels used in the partitioned data space. Without partitioning, 5.218×10^9 vector comparisons were required.

vectors. A density threshold of 500 was used, and the searching of neighboring boxes of smaller size than the target box was allowed for all such boxes with a minimum of 25 vectors. With one level of boxes, 9.99×10^8 vector comparisons were required. Without the use of partitioning, the total number of comparisons required was 5.218×10^9. This represents a speed increase of a factor of 5.2 relative to the case in which no partitioning was used. Four of the test vectors produced discrepancies in the top five library matches between the results obtained with and without partitioning. In each case, these test vectors mapped to target boxes at the top level with less than five library vectors. As discussed previously, this case defines a situation in which there is no guarantee that the required number of matches will be found among the target box and its immediate neighboring boxes. The number of mismatches remained constant for each data point in Figure 3.

The speed increase achieved by the multilevel partitioning approached an optimal value at four levels, with only a small reduction in the number of vector comparisons being observed for levels 5−7. Past the seventh level, no further boxes met the density threshold of 500 vectors. Based on the number of vector comparisons, the speed increase obtained for partitioning at seven levels was a factor of 34.1 relative to the case in which no partitioning was performed. For this case, the actual computational time required was 1.3 h. Approximately 40 h were required when no partitioning was used. The decrease in elapsed time by a factor of 30.8 afforded by the partitioning is slightly less than the speed increase measured in terms of vector comparisons. This difference is due to the fact that the elapsed time includes the overhead of the bookkeeping steps associated with the partitioning itself.

Figure 4 presents results from an analogous study of the number of vector comparisons required for a partition structure of 1−7 levels for the case in which the top 200 matches were sought. For this experiment, the density threshold was again 500, while 200 was set as the minimum population size to allow the search of smaller neighboring boxes. As noted previously, this is the minimum allowable value for this setting when 200 matches are sought. A randomly chosen subset of 3900 of the 39 074 test vectors was employed for these searches. Without the use of partitioning, 5.208×10^8 vector comparisons were required for this set of test vectors. With one level of boxes, the

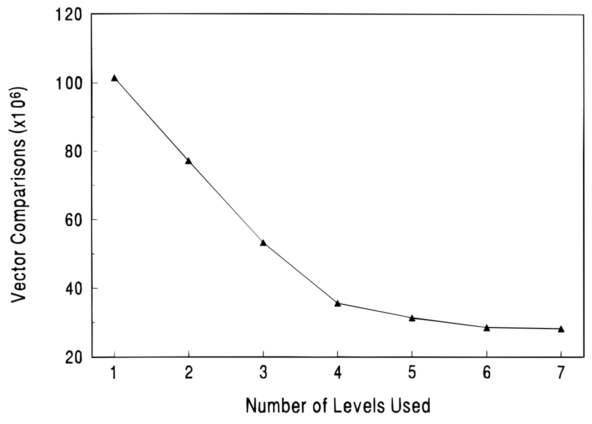

Figure 4. Plot of number of vector comparisons required to find the 200 nearest matches for a subset of 3900 test vectors vs the number of levels used in the partitioned data space. Without partitioning, 5.208×10^8 vector comparisons were required.

speed increase was a factor of 5.1, directly analogous to the results obtained in the experiment based on five nearest matches. However, comparison of Figures 3 and 4 reveals that, for levels 2−7, the relative speed increase for the case of 200 matches was not as good as that obtained in the search for the top five matches. For example, with seven levels of boxes, the speed increased by a factor of 18.5, compared to a factor of 34.1 for the previous search. The key here is that since a higher threshold is required for deciding when to search smaller neighboring boxes, fewer of these boxes will be searched. More often, the larger boxes at the same level as the target box will be searched, resulting in more vector comparisons and a smaller speed increase.

Significantly more mismatches were also encountered in the search for the 200 nearest matches. For the searches based on one or two levels of partitioning, 46 of the 3900 test vectors produced one or more mismatches when the 200 nearest matches obtained with and without partitioning were compared. For the searches based on 3−7 levels of partitioning, 49 of the test vectors produced mismatches. Of the 49 vectors, 46 corresponded to cases in which fewer than 200 library vectors were found in the target box at the top level. This is the case described previously in which there is no guarantee that the 200 nearest matches will be located in the target box and its immediate neighboring boxes. For 47 of the 49 vectors, the nearest matching library vectors had Euclidean distances of greater than 27.0. For the other two cases, the Euclidean distances to the nearest library vector were approximately 6.7. It has been empirically observed that for the seven-dimensional environment vectors, Euclidean distances greater than 1.0 indicate chemical environments that are so different from the environment of the target carbon that the retrieved chemical shift is often of little use.

For the three cases in which the required 200 or more library vectors were found in the target box, two corresponded to discrepancies in only the last several matches of the top 200 and represented differences in Euclidean distances on the order of the precision of the calculation ($\leq 10^{-4}$). The remaining case arose due to the searching of a neighboring box smaller than the target box and represents a failure of the strategy described above in which a minimum number of vectors were required to fall within the smaller neighboring box in order for it to be searched.

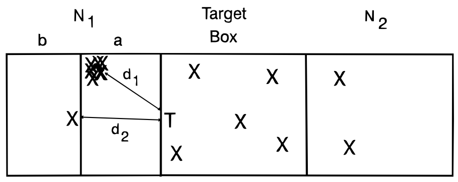

Figure 5. Illustration of one scenario in which nearest matching library vectors can be overlooked during the search of neighboring boxes smaller than the target box. The smaller neighboring box, N_{1a}, meets the criterion of having more vectors than the number of nearest matches sought. However, the location of the target vector near the left box boundary and the clustering of vectors in the upper left corner of box N_{1a} dictates that a vector in box N_{1b} can be a nearest match and therefore be overlooked if box N_{1b} is not searched.

This last mismatch represents an example of how a closest match can be missed due to an unusual distribution of data points in the data space. Figure 5 illustrates in two dimensions a likely cause of this mismatch. The test vector is marked in the target box by a T. The neighboring box, N_1, has sufficient vectors to be split into boxes N_{1a} and N_{1b}. There are sufficient vectors in N_{1a} to allow it to be searched rather than the larger box, N_1. However, the majority of points in N_{1a} are located in a tight cluster in the upper left corner and lie at a distance, d_1, from T. The data point just across the boundary in box N_{2b} (distance d_2) can actually lie closer to T than the majority of points in N_{1a} due to the diagonal of an isosceles right triangle being a factor of $2^{1/2} \approx 1.4$ times greater than the length of the side. Under the rules of the algorithm, box N_{2b} is not searched, however, because it is not adjacent to the target box, and the condition has been met in which the adjacent box has more vectors than the number of nearest matches sought. This failure of the algorithm is a result of mapping a continuum of points (library vectors) into discrete boxes. It should be noted, however, that only one such event occurred out of 3900 test vectors (probability = 0.026%), and the match missed had a large Euclidean distance and was consequently of no value in generating a predicted chemical shift.

The second set of experiments performed studied the effect of varying the number of vectors required to classify a box as dense. The seven-dimensional case was used with seven levels of boxes, the top 200 matches were found, and neighboring boxes of smaller sizes than the target box were searched if they contained greater than 200 vectors. The subset of 3900 test vectors was used for these searches. The density threshold was varied over seven settings from 2000−200, and the results are displayed in Figure 6. The speed increase ranged from a factor of 13.0 for a density limit of 2000 to a factor of 19.5 for a density limit of 200. There were 49 test vectors with mismatches for all of the data points except the points at 1500 and 2000, which each had only 48 mismatches. The reasons for the mismatches are identical to those described previously in the discussion of Figure 4.

The third set of experiments performed studied the effect of changing the minimum limit of vectors required in a neighboring box smaller than the target box to allow that neighboring box to be searched. The experiments performed used the seven-dimensional data space with seven levels of boxes and a density threshold for the target box of both 200 and 500. The set of 3900 test vectors was again used for these searches. The neighboring box limits were varied over eight settings in the range from 25−500, and the results are

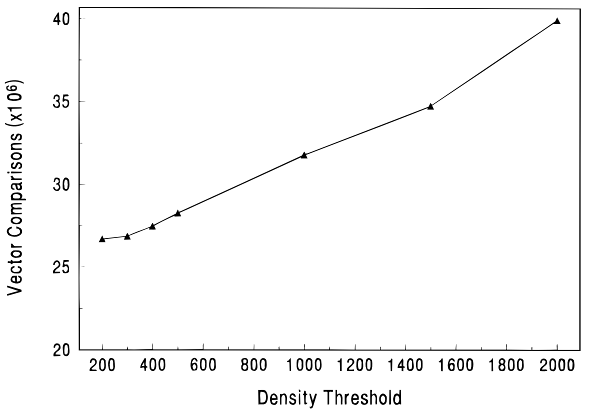

Figure 6. Plot of number of vector comparisons required to find the 200 nearest matches to the subset of 3900 test vectors vs the density threshold used to determine if a box is dense and should be partitioned further.

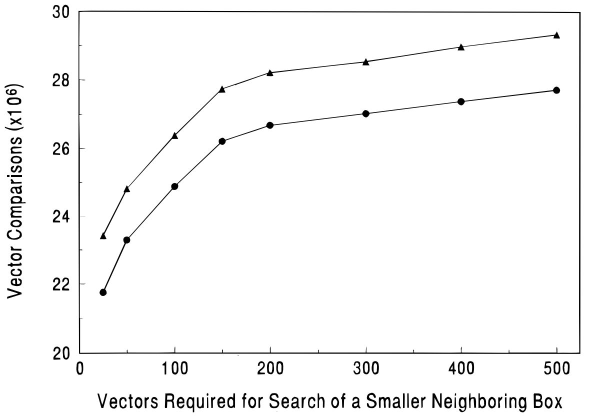

Figure 7. Plot of number of vector comparisons required to find the 200 nearest matches to the subset of 3900 test vectors vs the number of vectors required to lie within a neighboring box smaller than the target box in order for that neighboring box to be searched. The curves marked by triangles and circles correspond to the use of a density threshold of 500 and 200, respectively.

displayed in Figure 7. In the figure, the lines marked by the solid triangles and solid circles represent the experiments performed with density thresholds of 500 and 200, respectively. The results for the density threshold of 200 are slightly better than those obtained with a setting of 500. When no searching of neighboring boxes smaller than the target box was allowed, the number of vector comparisons required for density thresholds of 200 and 500 was 3.10×10^8 and 3.20×10^8, respectively. In both cases, the results improved as the minimum number of vectors required to search a smaller neighboring box was decreased. For the density threshold of 500, the number of vector comparisons decreased from 2.93×10^7 to 2.34×10^7 as the neighboring box limit was varied from 500 to 25. This represents an improvement in the speed increase from a factor of 17.8 to a factor of 22.3, relative to the case in which no partitioning was performed. The number of mismatches, however, increased significantly from the neighbor limit of 150 to 25. At a value of 150, both experiments produced no mismatches with Euclidean distances less than 6.6. At a value of 100, the data set with a density threshold of 200 had one mismatched test vector with a Euclidean distance of 1.0, while the data set with a density threshold of 500 still had no mismatches with a Euclidean distance of less than 6.6. At a value of 50, both data sets had several mismatches with

Euclidean distances less than 0.5, while at a value of 25, the data sets corresponding to density thresholds of 200 and 500 had 20 and 10 such mismatches, respectively. It is essential that the value used for the neighboring box limit results in no mismatches of library vectors with small Euclidean distances. From these data, we have concluded that the best variable settings for obtaining the 200 nearest matches are a density threshold of 200 with a neighboring box limit of 150. The use of seven levels of boxes is also recommended. These settings produced a speed increase of a factor of 19.9 relative to the case in which no partitioning was performed.

CONCLUSION

The partitioning method described in this paper meets the goal of significantly increasing the speed of retrieving vectors from a database of encoded carbon atom environments. This speed increase was obtained while retrieving the same significant matching vectors as would be obtained in a full database search. The only mismatching vectors encountered represented carbons so dissimilar to the target carbon to be of no practical use. The speed increase obtained for the retrieval of the top five matches based on the relative number of vector comparisons was a factor of approximately 34. The analogous speed increase obtained for the retrieval of the top 200 matches was approximately 20, again based on the relative number of vector comparisons.

The benefits of partitioning can be seen by way of a practical example. Using the amount of elapsed computational time required to find the top five matches for the 39 074 carbon atoms in the test set, one can calculate that a 20-carbon structure would require approximately 2.4 s with partitioning and 73.7 s without partitioning to perform the chemical shift retrieval. This is a significant savings in time for a user of the software, particularly if multiple chemical structures are to be investigated.

It is important to emphasize that this partitioning method can be applied to any database retrieval application which involves the representation of the database entries as vectors in a *n*-dimensional Euclidean space. For example, the speed of retrievals of infrared or mass spectral data could be similarly enhanced.

It should be noted, however, that for some applications to which the partitioning algorithm might be applied, memory usage may be a critical factor. If the application has a high dimensionality, memory requirements can become restrictive. The upper limit for the ^{13}C NMR application described here appears to be in the range of 13 dimensions, given a target memory utilization of 15 Mb. The upper limit depends to some extent on the density of the data space, as this will determine the number of levels and the number of dense boxes needed to obtain a partitioning which results in a large increase in search speed. Applications in which the dimensionality is much higher could still use the partitioning scheme, but the dimensionality of the data space would need to be reduced first by deresolving the data or employing data compression methods such as the PCA technique used in generating Figure 1. The partitioning and identification of vectors to be compared to the test vector would then be performed in the reduced dimensionality space. Actual comparisons of the test vector with the library vectors could still be performed in the original data space, however.

ACKNOWLEDGMENT

This research was supported by the Shell Development Co., Houston, TX. The Sadtler Division of Bio-Rad Laboratories, Inc. is acknowledged for providing the spectral and chemical structural data used in this work. Portions of this research were presented at the meeting of the Federation of Analytical Chemistry and Spectroscopy Societies, St. Louis, MO, October 1994.

Supporting Information Available: Description of the software implementation of the partitioned data space, table of parameters describing storage of selected boxes in the partitioned data space, and figures of storage in the array of linked lists and mapping of target vector 11 into each dimension of the partitioned data space (13 pages). This material is contained in many libraries on microfiche, immediately follows this article in the microfilm version of the journal, can be ordered from the ACS, and can be downloaded from the Internet; see any current masthead page for ordering information and Internet access instructions.

REFERENCES AND NOTES

(1) Small, G. W. Automated Spectral Interpretation *Anal. Chem.* **1987**, *59*, 535A-545A.

(2) Bremser, W. HOSE—A Novel Substructure Code. *Anal. Chim. Acta* **1978**, *103*, 355−365.

(3) Bremser, W. Expectation Ranges of ^{13}C NMR Chemical Shifts. *Magn. Reson. Chem.* **1985**, *23*, 271−275.

(4) Chen, L.; Robien, W. The CSEARCH-NMR Data Base Approach to Solve Frequent Questions Concerning Substituent Effects on Carbon-13 NMR Chemical Shifts. *Chemom. Intell. Lab. Syst.* **1993**, *19*, 217−223.

(5) Crandell, C. W.; Gray, N. A. B.; Smith, D. H. Structure Evaluation Using Predicted ^{13}C Spectra. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 48−57.

(6) Cheng, H. N.; Kasehagen, L. J. Integrated Approach for ^{13}C Nuclear Magnetic Resonance Shift Prediction, Spectral Simulation and Library Search. *Anal. Chim. Acta* **1994**, *285*, 223−235.

(7) Von der Lieth, C. W.; Seil, J.; Köhler, Opferkuch, H. J. ^{13}C NMR Data Bank Techniques as Analytical Tools. *Magn. Reson. Chem.* **1985**, *23*, 1048−1055.

(8) Chen, L.; Robien, W. OPSI: A Universal Method for Prediction of Carbon-13 NMR Spectra Based on Optimized Additivity Models. *Anal. Chem.* **1993**, *65*, 2282−2287.

(9) Clerc, J. T.; Sommerauer, H. A Minicomputer Program Based on Additivity Rules for the Estimation of ^{13}C-NMR Chemical Shifts. *Anal. Chim. Acta* **1977**, *95*, 33−40.

(10) Jensen, K. L.; Barber, A. S.; Small, G. W. Simulation of Carbon-13 Nuclear Magnetic Resonance Spectra of Polycyclic Aromatic Compounds. *Anal. Chem.* **1991**, *63*, 1082−1090.

(11) Clouser, D. L.; Jurs, P. C. Simulation of ^{13}C Nuclear Magnetic Resonance Spectra of Tetrahydropyrans Using Regression Analysis and Neural Networks. *Anal. Chim. Acta* **1994**, *295*, 221−231.

(12) Jurs, P. C.; Ball, J. W.; Anker, L. S.; Friedman, T. L. Carbon-13 Nuclear Magnetic Resonance Spectrum Simulation. *J. Chem. Inf. Comput. Sci.* **1992,** *32*, 272−278.

(13) Fürst, A.; Pretsch, E.; Robien, W. Comprehensive Parameter Set for the Prediction of the ^{13}C-NMR Chemical Shifts of sp^{3}-hybridized Carbon Atoms in Organic Compounds. *Anal. Chim. Acta.* **1990,** *233*, 213−222.

(14) Pretsch, E; Fürst, A.; Badertscher, M.; Bürgin, R.; Munk, M. E. C13Shift: A Computer Program for the Prediction of ^{13}C NMR Spectra Based on an Open Set of Additivity Rules. *J. Chem. Inf. Comput. Sci.* **1992,** *32*, 291−295.

(15) Small, G. W. Database Retrieval Techniques for Carbon-13 Nuclear Magnetic Resonance Spectrum Simulation. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 279−285.

(16) Zupan, J. A New Approach to Binary Tree-Based Heuristics *Anal. Chim. Acta.* **1980,** *122*, 337−346.

(17) Zupan, J.; Munk, M. E. Hierarchial Tree Based Storage, Retrieval, and Interpretation of Infrared Spectra *Anal. Chem.* **1985,** 57, 1609−1616.

(18) Small, G. W.; Jurs, P. C. Determination of Topological Similarity of Carbon Atoms in the Simulation of Carbon-13 Nuclear Magnetic Resonance Spectra. *Anal. Chem* **1984**, *56*, 1314−1323.

CI9500647