Similarity Searching and Clustering of Chemical-Structure Databases Using Molecular Property Data

Geoffrey M. Downs* and Peter Willett

Krebs Institute for Biomolecular Research and Department of Information Studies, University of Sheffield, Western Bank, Sheffield S10 2TN, U.K.

William Fisanick

Chemical Abstracts Service, 2540 Olentangy River Road, Columbus, Ohio 43210

Received March 4, 1994®

Previous work on the clustering of chemical-structure databases has focused on the use of intermolecular similarity measures that are based on structural features of various kinds. In this paper, we report nearest-neighbor searching and clustering experiments with a set of 5982 molecules, each of which is characterized by 13 calculated global molecular properties. The nearest-neighbor algorithm is an upperbound procedure that uses the triangle inequality to minimize the number of distance calculations that need to be carried out when searching for nearest neighbors in metric spaces. Our experiments suggest that it performs well when small numbers of nearest neighbors are required, but that the basic "brute-force" procedure is best when large numbers are needed, such as when clustering is to be carried out. The clustering methods tested are the Ward and group-average hierarchic agglomerative methods, the minimum-diameter polythetic hierarchic divisive method, and the Jarvis–Patrick nearest-neighbor method. Our experiments suggest that the first three methods, which gave similar results, are the best methods for clustering molecules characterized by property data. The Jarvis–Patrick method, which has been extensively used for clustering molecules characterized by structural fragments, was not as effective as these other methods.

1. INTRODUCTION

There is considerable interest in the use of similarity and clustering methods with databases of chemical structures,[1-4] and the methods are widely used in both in-house chemical information systems, e.g., those at Pfizer Central Research[5] and at the Upjohn Company[6], and in commercially-available chemical-database software packages, e.g., the PowerSearch module of the MACCS software from MDL Information Systems and the Cluster module of the Daylight software from Daylight Chemical Information Systems. These modules calculate the similarity between a pair of molecules using the fragment-incidence data that have traditionally been used for the screening stage of a substructure search. Although such data provide an extremely simple description of a chemical molecule, effective, chemically useful clusterings can be achieved by using a simple association coefficient, the Tanimoto coefficient, and the nearest-neighbor clustering method due to Jarvis and Patrick.[5,7,8] Moreover, fast nearest-neighbor searching algorithms are available that permit the efficient implementation of this method on large files of structures.[9] Much less interest has been given thus far to similarity searching and clustering when the molecules are characterized by molecular-property data. In this paper, we report the results of a project between Chemical Abstracts Service (CAS) and the University of Sheffield to investigate the use of property data for similarity searching and clustering.

The fragment-incidence data that have been used previously for similarity searching and clustering is fundamentally different from the property data considered here. Given a set of N compounds, each characterized by M properties, the dataset may be represented by an $N \times M$ data matrix in which the IJ-th element represents the value of the J-th property for the I-th compound. This data matrix is both real and complete, since every structure has a real value associated with it for each of the M properties; conversely, with fragment bit-strings the data matrices are binary and sparse, since every structure is characterized by the presence of some small number of binary attributes selected from a much larger number (i.e., the total length of the bit string). These differences suggest that a clustering procedure based on the Tanimoto coefficient and the Jarvis–Patrick method may not be the most appropriate for clustering datasets of property values (since association coefficients such as the Tanimoto coefficient are normally defined only in the context of binary attributes, and since the nearest-neighbor algorithm that lies at the heart of our efficient implementation of the Jarvis–Patrick algorithm is only applicable to sparse data matrices). In this paper, we report a series of experiments that has been carried out to test these hypotheses.

2. EXPERIMENTAL DETAILS

2.1. Dataset Characteristics. The CAS property dataset comprised 5982 organic structures, each with 29 physicochemical properties that had been calculated using standard molecular-modeling and quantum-mechanics packages. This dataset, which has been used previously by Fisanick *et al.*[10] in studies of three-dimensional (3-D) similarity searching, will be referred to subsequently as dataset-A. Much of the developmental work in the present project used a subset of this file that contained just the first 500 of these structures: this will be referred to as dataset-B. Many of the properties are highly correlated with each other, and thus a subset of the properties was selected to characterize each of the molecules in the two datasets in our experiments. For instance, the x, y, and z vectors of the dipole moment are given separately and also summed as *dsum_t*, and so only the latter was used. Similarly, the highest occupied and lowest unoccupied

* To whom all correspondence should be addressed.
® Abstract published in *Advance ACS Abstracts*, August 15, 1994.

0095-2338/94/1634-1094$04.50/0

Table 1. Property Value Ranges for Dataset-A

property	min. value	middle value	max. value	std dev
gsnm	0.0000	0.3299	0.5846	0.108
ht_form	-1300.1800	-12.2036	1803.8470	137.646
tot_eng	-17302.9590	-3337.9153	-602.6120	1320.045
ion_pot	-6.3590	9.3238	21.8770	1.002
fill_orb	8.0000	48.7703	144.0000	17.051
molec_wt	42.0370	270.1619	1407.2370	98.900
dsum_t	0.0000	3.5009	43.6630	2.434
del_h_l	-15.0820	-8.8867	-2.4710	1.488
chg_std	0.0460	0.2185	1.9050	1.148
ed_std	0.2410	1.9566	2.9980	0.202
logp	-7.1300	2.6570	13.8100	2.025
mr	0.0000	72.6847	208.4600	26.879
vol	39.6180	238.2251	681.2360	81.704

molecular orbital (HOMO and LUMO) values are listed separately as is their difference (*del_h_l*), and so only the latter was used. From the original 29 properties, the following 13 were chosen: heat of formation (*ht_form*), global simple normalized mean flexibility index (*gsnm*), total energy (*tot_eng*), ionization potential (*ion_pot*), number of filled orbitals (*fill_orb*), molecular weight (*molec_wt*), total dipole moment sum (*dsum_t*), difference between HOMO and LUMO values (*del_h_l*), standard deviation of partial atom charges (*chg_std*), standard deviation of atom electron densities (*ed_std*), octanol-water partition coefficient (*logp*), molar refractivity (*mr*), and van der Waal's volume (*vol*). Table 1 gives the minimum, middle (closest to mean), and maximum values for these 13 properties, along with their standard deviations.

Applications of cluster analysis often preprocess the data by means of weighting (where some attributes are weighted more than others to reflect their relative degrees of importance) and standardization (where all of the attribute values are reduced to a common scale of measurement). Sneath and Sokal[11] argue strongly against the use of weighting, a view confirmed by previous studies at Sheffield that have shown little difference between weighted and unweighted fragment-based descriptors.[1] There thus seemed little point in weighting any of the property descriptors. Similarly, Everitt[12] advises against standardizing data for clustering. However, although standardization has little effect with fragment-occurrence data,[13] the widely differing ranges and orders of magnitude of the selected properties (as shown in Table 1) suggested that the raw data matrix of property values needed to be standardized to ensure that a similar weight was given to each descriptor to avoid some descriptors masking the contributions of others. From the many methods available,[14] two methods of standardization were tested: standardization by standard deviation and standardization by range. For each property value the difference between the mean and actual value is divided by the standard deviation of range, respectively, to give the standardized value. Standardization by range gives a series of new values in the range +1.0 to -1.0, while for standardization by standard deviation the new values have a mean of 0.0 and a standard deviation of 1.0. Standardization by range thus gives known upper and lower bounds to the values, whereas standardization by standard deviation does not.

2.2. Similarity Measure. A definitive account of similarity measures is provided by Sneath and Sokal[11]. They describe four main classes of similarity coefficient: distance, association, probabilistic, and correlation. The most commonly used measure is the Euclidean distance, and many nearest-neighbor searching algorithms assume that this is used to measure the degree of resemblance between pairs of objects (as discussed in more detail in section 3). Research at Sheffield suggested that the cosine, Tanimoto, and correlation coefficients gave better results than distance measures for measuring the similarity between fragment bit-strings and that, of these, the Tanimoto coefficient gave the most intuitively acceptable rankings of compounds.[15] Accordingly, the Tanimoto coefficient has been widely adopted for the clustering of such data. However, this coefficient is not designed for use with nonbinary variables such as the molecular properties considered in the present work, and we have thus used the Euclidean distance for all of our experiments (with the exception of the group-average clustering experiments, in which the cosine coefficient was used for reasons given in section 4.1).

2.3. Selection of Clustering Methods. There are very many different methods that can be used to cluster a dataset,[11,12,16] of which four were chosen for this project. Previous work at Sheffield on the clustering of chemical-structure databases[1] considered the following classes of method: hierarchic agglomerative, hierarchic divisive, relocation, and nearest-neighbor. This work suggested that the best results were given by the Jarvis–Patrick nearest-neighbor method and by Ward's hierarchic agglomerative method,[1] and both of these methods were thus considered in the present investigation. The early hierarchic-divisive experiments considered only monothetic methods, i.e., those in which clusters are subdivided in each iteration of the clustering process according to the values of only a single attribute. Monothetic divisive methods are more common than polythetic divisive methods, in which all of the attributes are involved in each subdivision, since the latter methods tend to be much more demanding of computational resources. However, Guenoche *et al.*[17] have recently described the polythetic minimum-diameter method, which, while slower than the common monothetic methods, is sufficiently fast in operation to be applicable to datasets of nontrivial size, and we have thus considered this method in the present work. Finally, we have included a further hierarchic agglomerative method, the group-average method, because it has performed consistently well in comparisons of hierarchic agglomerative methods in a wide range of application areas.

3. NEAREST-NEIGHBOR SEARCHING

3.1. Background. Given a set, S, of N objects and a given query object, Q, the nearest-neighbor searching problem is to find those objects in S that are most similar to Q using some quantitative measure of interobject similarity such as the Euclidean distance. The nearest neighbors for Q are most simply identified by a linear scan, in which Q is matched against each member of S in turn. This "brute-force" algorithm has a time complexity of order $O(N)$, which makes nearest-neighbor searching extremely demanding of computational resources if large datasets are to be processed.

Nearest-neighbor searching forms the basis for many pattern-recognition methods and also lies at the heart of most clustering algorithms, and there has thus been much interest in the development of algorithms that allow the nearest neighbors in a dataset to be identified more efficiently than is possible using the brute-force algorithm. Previous work at Sheffield[1] has resulted in a very rapid, inverted-file algorithm that allows interactive nearest-neighbor searching of files containing hundreds of thousands of structures. This algorithm can also be used for the clustering of files of comparable size (albeit much less rapidly, since a clustering method normally requires a nearest-neighbor search of the complete database for each of the N compounds in it rather than just the single search that is required for similarity searching). This algorithm, and others that are based on the inverted

file,[18] reduce the number of similarity calculations that need to be carried out by utilizing the extremely sparse nature of fragment bit-strings. No such reductions are possible with a full data matrix, such as a matrix of property values, and it is therefore necessary to consider alternative approaches to nearest-neighbor searching.

There are three main types of nearest-neighbor searching algorithm, these being algorithms based on k-d trees or branch-and-bound, grid procedures that partition the search space so that only part of it needs to be explored, and upperbound calculations that reduce the number of objects that need to be matched against the query object, Q.[1,19-21] However, only the upperbound procedures achieve efficiencies in operation with high-dimensionality data of the sort considered here (since the phrase "high-dimensionality data" seems to be used in the literature to describe data having eight or more dimensions), and we have thus considered only this type of nearest-neighbor searching algorithm for our work.

Upperbound strategies involve the order $O(N)$ comparisons associated with the brute-force algorithm but use a range of criteria to allow unsuitable candidates to be rejected at less cost than the full distance calculation. For example, the Euclidean distance is bounded by the more easily calculated Hamming and Chebyshev distances, so that only points that pass these tests need go forward to the full Euclidean distance calculation. Thus, Yunck[212] rejects all points where the current nearest-neighbor distance is less than the Chebyshev distance; Kittler[23] rejects all points where the current nearest-neighbor distance multiplied by $\sqrt{k}$ is less than the Hamming distance; and Richetin *et al.*[24] apply both criteria. The other main approach that has been studied involves the use of the triangle inequality with respect to reference points to calculate an upperbound between the query object and potential nearest neighbors in the database that is to be searched. The original algorithm described by Burkhard and Keller[25] used just a single reference point, but improved results are obtained if several reference points are employed (as originally advocated by Shapiro[26] and subsequently by Nevalainen and Katajainen[27] and by Shasha and Wang.[21]) The multiple reference point method is relatively simple to implement and was thus evaluated for use in our property-based clustering experiments.

3.2. The Triangle-Inequality Upperbound Algorithm. The triangle-inequality upperbound algorithm proceeds as follows:

1. Choose one or more reference points (i.e., chemical structures in the present context).
2. For each reference point calculate the Euclidean distances to every other structure in the dataset that is to be searched and then store the resulting distances.
3. Choose one of the reference points as the main reference point and sort the distances associated with this reference point into increasing size, moving the distances associated with the other reference points at the same time to keep them together.
4. Use a binary search to locate the query structure in the sorted list.
5. Incrementally move in both directions along the list outwards from the query structure position until the end of the list is encountered or the triangle inequality condition no longer holds (if the difference between the reference point to query distance and reference point to current-position-in-the-list distance is greater than the query to current nearest-neighbor distance, then the triangle inequality condition does not hold).

This algorithm finds just the single nearest neighbor, i.e., top-1 (where we use the notation top-K to denote the K nearest neighbors of a query object, Q), but it is easy to generalize the algorithm to find the top-20 nearest neighbors for each structure in a dataset. To implement step 5 in the algorithm above for finding top-20, it is necessary to start by calculating the full distances for 20 structures immediately adjacent to the query structure in the sorted list. This ensures that the top-20 list is initialized with the closest structures to the query with respect to the reference point. Finding the top-20 nearest neighbors, rather than just the first, thus means that 20 full distances must be calculated before testing the triangle inequality condition. The current nearest-neighbor distance is then set as the 20th in the top-20 list. The use of multiple reference points does not reduce the amount of sorted-list searching that is required, but it does reduce the number of additional full-distance calculations; this, however, incurs further overheads in creating and referencing the additional lists.

The reference points chosen were the structures with the minimum, middle (closest to mean), and maximum standardized property values. One or more properties were selected, each providing a triplet of reference points. From these, one was selected as the main reference point on which to sort the list in step 3 of the algorithm above.

3.3. Experimental Results. Most of the results were obtained using dataset-B; a few runs with the larger dataset-A confirmed the general conclusions from the experiments with the subset. Initial tests on dataset-B showed that the minimum number of distance calculations occurred when using all 39 possible reference points (i.e., all 13 properties, each with minimum, middle, and maximum points). The program was run using each reference point as the main reference point for sorting and for both range-standardized and standard-deviation-standardized data, a total of 78 runs. A summary of the results for the standard-deviation-standardized data is given in Table 2, which gives the total number of distance calculations, the total number of times the triangle-inequality cutoff was applied, and the total number of times that the end of the list was reached in either direction in step 5 of the algorithm.

It can be seen from the results in Table 2 that the average number of full distance calculations was about 45 000. The results for the range-standardized data were very similar, with the average number of full distance calculations of about 40 000 and very little variation. These figures compare favorably with the 124 750 full distance calculations required for processing dataset-B by the brute-force algorithm. Unfortunately, the latter was approximately three times as fast as the reference-points program, i.e., the establishment and scanning of the distance lists reduced the number of full distance calculations by two-thirds but tripled the processing time. One further observation is the consistency of the results irrespective of which reference point was used as the main reference point, which seems to suggest that both standardization methods are performing well.

Processing the 5982 molecules in dataset-A required 17 889 171 full distance calculations by the brute-force method and took about 22 min on a 33 MHz 80486 PC using a Sheffield Pascal/386 compiler. Using all 39 reference points, with the first property middle value as main reference point, reduced the number of distance calculations to 2 288 768 but took about 60 min. If just the first property's triplet of reference points was used then 9 211 089 distances were calculated in 25 min. Hence, use of the triangle-inequality method approximately halved the number of full distance calculations even with just three reference points and reduced it by almost seven-eighths using all 39 reference points: the efficiency of

Table 2. Results Using Each of 39 Reference Points as the Main Reference Point for Dataset-B, Standard-Deviation-Standardized Data

	standard-deviation-standardized data								
	minimum value			middle value			maximum value		
property	distances	cutoffs	ends	distances	cutoffs	ends	distances	cutoffs	ends
gsnm	44 907	977	21	46 007	978	20	45 274	976	22
ht_form	45 179	983	15	46 114	978	20	45 276	981	17
tot_eng	45 179	983	15	45 470	966	32	44 891	981	17
ion_pot	46 230	972	26	45 463	976	22	45 819	979	19
fill_orb	44 891	981	17	46 383	974	24	45 123	982	16
molec_wt	44 891	981	17	45 486	962	36	45 123	982	16
dsum_t	44 954	979	19	45 171	969	29	46 136	950	48
del_h_l	45 954	979	19	45 895	977	21	45 503	966	32
chg_std	45 436	981	17	45 569	973	25	45 055	981	17
ed_std	45 338	977	21	45 783	980	18	45 055	981	17
logp	46 230	972	26	45 592	966	32	45 123	982	16
mr	46 230	972	26	45 669	965	33	45 276	981	17
vol	44 891	981	17	45 539	980	18	45 276	981	17

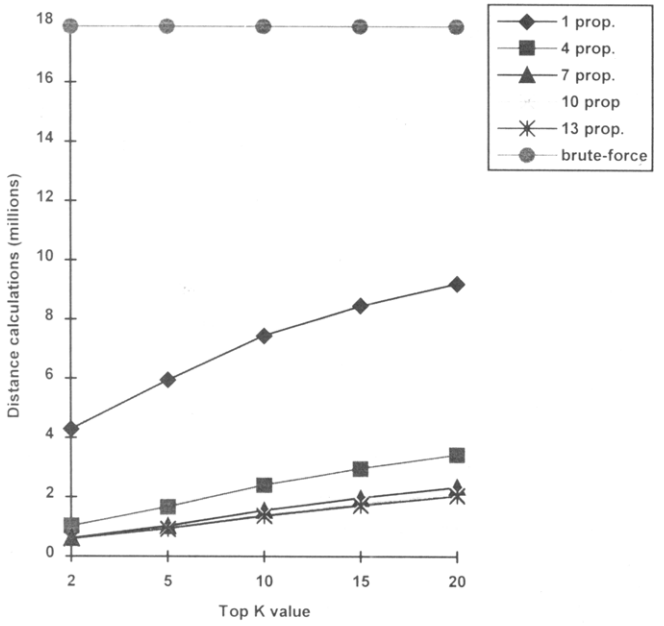

Figure 1. Comparison of the number of distance calculations performed using the brute-force and multiple reference points methods.

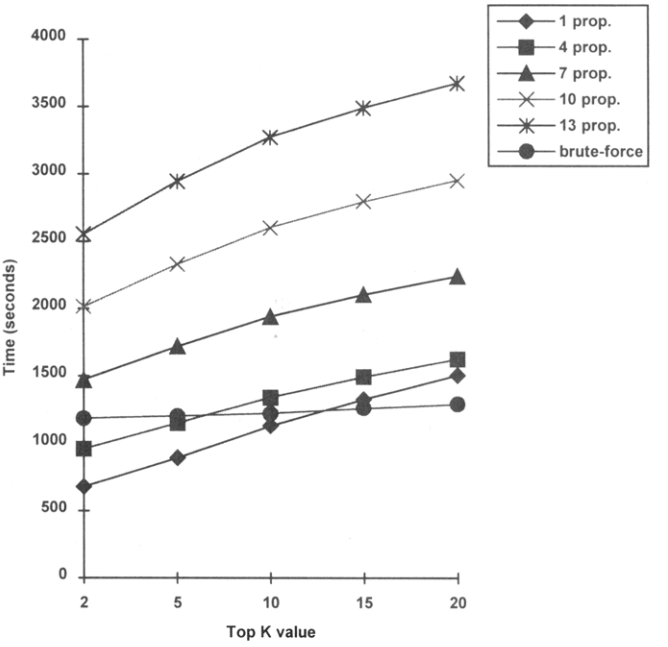

Figure 2. Comparison of the time taken for the brute-force and multiple reference points methods.

this algorithm (in reducing the number of full distance calculations) is thus greater with dataset-A than with the smaller dataset-B. However, the brute-force algorithm is still substantially less time-consuming than the reference-points algorithm owing to the latter's overheads. We believe that this is mainly due to the need to identify the top-20, rather than just the top-1 as in Shapiro's investigations.[26] The identification of the top-20 by the brute-force method requires only a negligible amount of bookkeeping to check and to maintain top-20 lists instead of top-1 lists. However, the time taken for the triangle-inequality method may increase several-fold because more of the sorted main reference-point list needs to be scanned before the triangle-inequality condition is satisfied.

To test this supposition, a further series of experiments were conducted on dataset-A. In these, the top-2, top-5, top-10, top-15, and top-20 lists were generated using the brute-force method and using 1, 4, 7, 10, and 13 properties (i.e., 3, 12, 21, 30, and 39 reference points) for the reference-points method. The results are shown graphically in Figures 1 and 2. In Figure 1, the number of distance calculations is plotted against the top-K value. It can be seen that the reference-points method consistently requires far fewer distance calculations than the brute-force method. The number of distance calculations becomes less as the number of reference points increases, but at a rapidly decreasing rate, so that there is little gain when more than about seven properties (21 reference points) are used. In Figure 2, the time taken is plotted against the top-K value. This shows that the brute-force method is faster except when generating less than about 13 nearest-neighbors using one property (three reference points) and about six nearest-neighbors using four properties (12 reference points). Optimization of the implementation of the reference-points algorithm may move the crossover points between brute-force and reference-points methods but is unlikely to alter the conclusion from the general trend observed here; namely that the reference-points method is unsuitable for high-dimensionality applications where more than one nearest-neighbor is required.

In view of the unsatisfactory results obtained with the triangle-inequality upperbound algorithm, all of the similarities required by the various clustering methods were computed using the brute-force algorithm.

4. COMPARISON OF CLUSTERING METHODS

4.1. Implementation of Methods. 4.1.1. Ward and Group-Average Methods. Hierarchic agglomerative methods are reviewed by Murtagh,[28] who shows that the traditional implementation using the Lance–Williams recurrence algo-

rithm[29] is far more time-consuming than an approach based on Reciprocal Nearest Neighbors, or RNNs. The RNN algorithm can be summarized as follows:

1. Mark all structures, I, as "unused".
2. Starting at an unused I, trace a chain of unused nearest neighbors until a pair of reciprocal nearest neighbors is encountered; i.e., trace $J = \mathrm{NN}(I)$, $K = \mathrm{NN}(J)$,... until $Q = \mathrm{NN}(P)$ and $P = \mathrm{NN}(Q)$.
3. Add RNNs P and Q to the list of RNNs along with the distance between them, mark Q as "used", and replace the centroid of P with the combined centroid of P and Q.
4. Continue the NN-chain from the point in the chain prior to P or choose another unused starting point if P was a starting point.
5. Continue until only one unused point remains, which involves a total of $N-1$ iterations for a dataset containing N objects.

Individual methods differ in the manner in which the closest pair of points is defined and the representation of the merged pair as a single point. The RNN algorithm is applicable to clustering methods in which the closet pair is defined as a distance, i.e., geometrically. In Ward's method, the Euclidean distance is used to determine distances between points and hence to define a cluster centroid. Thus, the Ward cluster hierarchy is obtained if the Euclidean distance is used in the RNN algorithm. The clusters are obtained by sorting the list of $N-1$ RNNs in order of increasing distance between them, so that the first entry in the list is the first pair to be merged and so on.

Group-average clusters can be obtained using the same algorithm if the Cosine coefficient is used instead of the Euclidean distance.[30] This changes the normally nongeometric (graph-theoretic) group-average method into a geometric method, and the RNN approach is valid. Since a coefficient is used instead of a distance, the closest points are those with the largest coefficient, and the RNN list needs to be sorted in order of decreasing coefficient (rather than increasing distance).

4.1.2. Minimum-Diameter Method. The minimum-diameter method is a polythetic hierarchic-divisive method that proceeds by recursively selecting the cluster with the largest diameter and partitioning it into two clusters such that the larger cluster has the smallest possible diameter (Guenoche *et al.*[17]). The diameter of a cluster is defined as the largest dissimilarity between any two of its members and does not necessarily reflect the size (number of members) of the cluster; the diameter of a singleton cluster is defined to be zero.

The minimum-diameter algorithm is a refinement of that due to Rao[31] for simple bipartitioning of a cluster and has a worst-case complexity of $O(N^2 \log N)$. The algorithm can be summarized as follows:

1. For the set of N entities to be clustered, produce an input list of all $N(N-1)/2$ dissimilarities sorted into decreasing order of magnitude.
2. Take the top two entities in the sorted dissimilarity list; these become the focus of the first bipartition of the dataset. Assign all other entities to the least dissimilar of these initial cluster centers.
3. Recursively select the cluster with the largest diameter and partition it into two clusters such that the larger cluster has the smallest possible diameter.
4. Continue for a maximum of $N-1$ bipartitions.

On completion, the first partitioning level shows the bipartitioning for the complete set of objects; each subsequent partitioning level may contain bicolorations for several separate maximum spanning trees, each representing a cluster formed at a previous partitioning level. In the worst case, there will be $N-1$ partitioning levels, where each bicoloration produces a singleton; in the best case there will be $\log_2 N$ partitioning levels, where each bicoloration splits the clusters evenly and a perfectly-balanced binary tree is created.

4.1.3. Jarvis–Patrick Method. The Jarvis–Patrick method involves the use of a list of the top-K nearest neighbors for each of the N structures. For this project, top-20 lists were generated and the nearest neighbors identified using the Euclidean distance. Two structures, X and Y, are placed in the same cluster if all of the following conditions are true.

1. X is the top-K nearest neighbor list of Y.
2. Y is the top-K nearest-neighbor list of X.
3. X and Y have at least $K_{\min}$ of their top-K nearest neighbors in common, where $K_{\min}$ is a user-defined parameter.

Processing the top-20 lists to produce the clusters involves the use of an array of size N, the elements of which are initialized to their array position. The top-20 lists are then scanned linearly for all pairs of structures I and J in N. If the clustering conditions are met for any IJ, then the array entry for the lower element of the two becomes the array entry for the higher element (and similarly so for any other element in the array with an entry equal to the higher element). When all of the top-20 lists have been processed, the array contains the clusters for that $K_{\min}$ value. Any elements with entries equal to their position in the array are cluster start-points and a scan from such a position to the end of the array looking for any other entries equal to that position will give the remaining members of that cluster. As with the hierarchic methods, this process is not dependent on the order in which a dataset is processed. However, instead of choosing a required number of clusters, as done with the hierarchic methods, the partition is governed by the choice of $K_{\min}$, so that the emphasis here is on partitioning by degree of similarity between structures rather than by a predefined number of clusters. It is necessary, therefore, to input a range of $K_{\min}$ values to obtain roughly the range of required clusters.

4.2. Efficiencies of the Methods. The four methods were implemented in Pascal and run on dataset-A using a 33 MHz 80486 PC running code compiled under the Sheffield Pascal/386 compiler with the debugging switches on.

For the hierarchic methods, 12 partitions of 50, 100, 250, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, and 5000 clusters were extracted from the hierarchies. For the Jarvis–Patrick method, $K_{\min}$ values of 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14 were used to generate 10 partitions which produced a range of clusters roughly equivalent to the 12 partitions of the hierarchic methods (since it is not possible to generate an exact number of clusters by this method). For standard-deviation-standardized data these values of $K_{\min}$ gave 118, 179, 345, 682, 1298, 2170, 3207, 4159, 4870, and 5364 clusters, respectively. For range-standardized data they gave 115, 159, 239, 410, 810, 1505, 2494, 3454, 4369, and 5058 clusters, respectively. The range of number of clusters produced is deliberately wide so that the analysis examines the performance of each method as fully as possible.

The Ward and group-average methods took 48 min to produce the list of $N-1$ reciprocal nearest neighbors and 3 min to produce the 12 cluster partitions.

The minimum-diameter method took 78 min to produce the 17 889 171 distances (with most of the time being spent writing the distances to disk), 60 h to produce the sorted list (using an unoptimized sort routine), and 248 min to produce the partition levels and the 12 cluster partitions. Twenty-two partition levels were created, which is close to the minimum

of 13 for a balanced binary tree for 5982 structures.

The Jarvis–Patrick method took 22 min to produce the top-20 nearest-neighbor lists (which were held in RAM and written to disk when completed, hence the difference in time compared with the first stage of the minimum-diameter method) and 13 min to produce the 10 partitions.

4.3. Measurement of Effectiveness. As in our previous work,[1,5] we have evalulated the effectiveness of the various clustering methods by using them for simulated property prediction and then comparing the observed and predicted property values.

Given a molecule, I, in a cluster, J, the predicted property value for I can be estimated as the mean of the property values for all the other structures in the cluster J. The procedure is repeated for each of the N compounds in a dataset and then the correlation established between the sets of N predicted and N observed property values. This "leave-one-out" approach is simple to implement, albeit time-consuming for the dataset sizes considered here, but sensible results will be obtained only for those molecules in a dataset that occur in a cluster that contains at least two other molecules, which means that molecules that occur in singleton or doubleton clusters are ignored.

The correlation between the sets of observed (x) and predicted (y) values were calculated by means of the product moment correlation coefficient (PMCC). This is given by

$$\frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

where $\bar{x}$ and $\bar{y}$ are the means of the observed and predicted values and where the summations are over all of the molecules occurring in clusters containing at least three members. PMCC values range from −1, denoting a perfect inverse correlation, through 0, meaning no correlation, to +1, denoting perfect correlation.

4.4. Results. The PMCC results for each partition and property with standard-deviation-standardized data for the Ward, group-average, minimum-diameter, and Jarvis–Patrick clustering methods are given in Table 3 sections 1–4, respectively. The number of property values used to calculate the PMCC value for each partition is given on the bottom row of each section. This number falls as the number of clusters rises, owing to the elimination of the singleton and doubleton clusters mentioned previously: the relationship between these two numbers is shown graphically in Figure 3. From the balanced hierarchy, or series of partitions, resulting from the clustering of an evenly-distributed set of points, a nonlinear relationship would be expected, as large clusters gradually diminish in size until, at the end, all singletons are produced. A dataset of 6000 evenly-distributed points would be expected to give doubleton clusters for a partition of 3000 clusters and mostly singleton clusters for a partition of 5000 clusters. Instead, Figure 3 shows a near-linear relationship; the Jarvis–Patrick results are almost perfectly linear, Ward and group-average are slightly less so, and minimum-diameter is slightly sigmoid. This implies that the cluster sizes are not evenly distributed; rather, that singletons and doublets are produced constantly throughout the hierarchy or series of partitions. The minimum-diameter sigmoid curve approximates best to the expectation of an even division of clusters and is a direct result of the way the method operates by divisive bipartitioning. Table 4 summarizes the minimum and maximum PMCC values obtained for each clustering method using standard deviation and range standardized data.

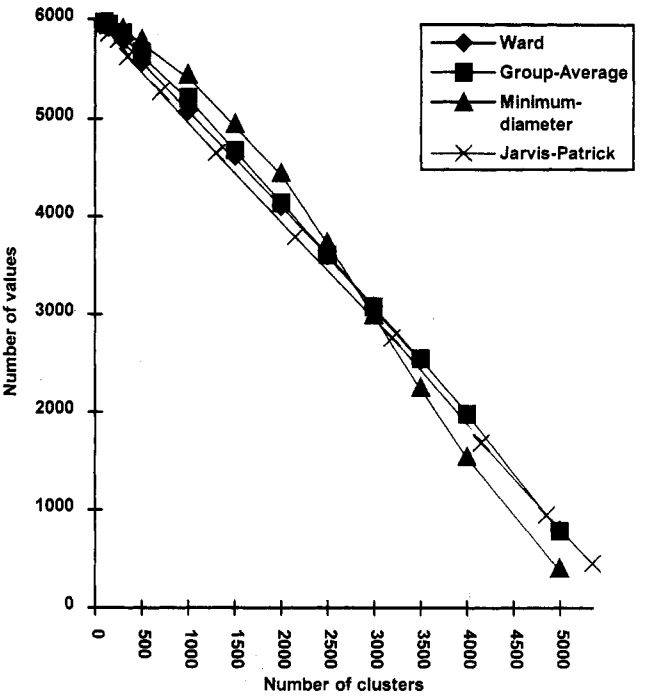

Figure 3. Comparison of the number of property values used in PMCC calculations for the four clustering methods.

The figures in Table 3 (sections 1–4) show a consistent failure of the Jarvis–Patrick method to give reasonable PMCC values (i.e., above about 0.8) for partitions with less than 3000 clusters, at which point the hierarchic methods have achieved PMCC values of 0.9 and above. As the number of clusters approaches 5000, the Jarvis–Patrick PMCC values equal and then slightly exceed those obtained from the hierarchic methods. This behavior is illustrated graphically in Figure 4, which shows the relationship between the PMCC and number of clusters for *tot_eng*; comparable relationships are obtained with all of the other 12 properties. The difference in shape between the Jarvis–Patrick and hierarchic method curves is remarkable and is directly related to the sizes of the clusters produced. Below 3000 clusters, Jarvis–Patrick consistently produces one or two very large clusters, with the remainder being very small clusters, mostly singletons. These large clusters will contain a diverse set of structures, resulting in low PMCC values. At around 3000 clusters, which results from a K_{min} value of 11, the very large cluster(s) starts to fragment into smaller, tighter clusters. At around 5000 clusters, resulting from K_{min} values of 13 and 14, the PMCC values for these smaller clusters are as good as, if not better than, those produced by hierarchic methods. This behavior is characteristic of Jarvis–Patrick clustering and has been observed in other studies using 2-D fragment-occurrence data.

The three hierarchic methods all provide a high level of performance, with very similar PMCC values being obtained for the same numbers of clusters produced. Minimum-diameter tends to have higher PMCC values when less than 1000 clusters are produced, after which Ward and group-average tend to have higher values. This pattern of behavior is shown for *ht_form*, *tot_eng*, *ion_pot* (just), *fill_orb*, *molec_wt*, *dsum_t*, *del_h_l*, and *ed_std*. For *gsnm*, minimum-diameter is in between the Ward and group-average values for up to 500 clusters, after which it is lower than both. For *chg_std*, minimum-diameter has lower values throughout than both Ward and group-average. For *logp*, minimum-diameter and Ward values are almost identical, and both show the above pattern with respect to group-average. For *mr*, minimum-diameter is consistently better than Ward, and for

Table 3. Section 1, PMCC Values for Ward Clustering of Dataset-A, Standard-Deviation Standardized; Section 2, PMCC Values for Group-Average Clustering of Dataset-A, Standard-Deviation Standardized; Section 3, PMCC Values for Minimum-Diameter Clustering of Dataset-A, Standard-Deviation Standardized; and Section 4, PMCC Values for Jarvis–Patrick Clustering of Dataset-A, Standard-Deviation Standardized

	number of clusters formed											
property	50	100	250	500	1000	1500	2000	2500	3000	3500	4000	5000
						Section 1						
gsnm	0.369	0.477	0.705	0.798	0.842	0.885	0.906	0.921	0.932	0.940	0.943	0.956
ht_form	0.560	0.592	0.728	0.753	0.809	0.836	0.860	0.879	0.890	0.902	0.914	0.931
tot_eng	0.781	0.795	0.836	0.876	0.924	0.943	0.953	0.960	0.964	0.970	0.974	0.977
ion_pot	0.525	0.622	0.731	0.769	0.832	0.861	0.887	0.903	0.911	0.919	0.924	0.944
fill_orb	0.728	0.809	0.846	0.896	0.935	0.957	0.966	0.971	0.974	0.979	0.982	0.985
molec_wt	0.772	0.803	0.843	0.891	0.929	0.952	0.961	0.968	0.970	0.975	0.979	0.981
dsum_t	0.534	0.624	0.670	0.702	0.827	0.859	0.887	0.902	0.908	0.911	0.923	0.935
del_h_l	0.557	0.621	0.792	0.836	0.888	0.912	0.928	0.942	0.943	0.949	0.950	0.958
chg_std	0.798	0.825	0.925	0.952	0.968	0.971	0.969	0.970	0.970	0.968	0.969	0.954
ed_std	0.557	0.624	0.714	0.798	0.868	0.888	0.907	0.924	0.930	0.936	0.947	0.958
logp	0.563	0.620	0.707	0.765	0.839	0.878	0.904	0.914	0.924	0.928	0.939	0.937
mr	0.775	0.819	0.848	0.904	0.939	0.961	0.971	0.975	0.977	0.980	0.983	0.983
vol	0.763	0.807	0.846	0.899	0.937	0.960	0.969	0.974	0.976	0.980	0.984	0.986
values	5950	5916	5789	5565	5084	4615	4100	3601	3083	2531	1964	796
						Section 2						
gsnm	0.699	0.741	0.793	0.825	0.866	0.880	0.897	0.912	0.924	0.937	0.941	0.963
ht_form	0.650	0.667	0.721	0.794	0.834	0.863	0.876	0.895	0.913	0.931	0.934	0.963
tot_eng	0.796	0.813	0.852	0.886	0.905	0.920	0.926	0.930	0.939	0.950	0.955	0.976
ion_pot	0.685	0.725	0.775	0.816	0.863	0.882	0.899	0.910	0.921	0.931	0.944	0.956
fill_orb	0.815	0.840	0.870	0.898	0.917	0.931	0.936	0.942	0.948	0.953	0.959	0.970
molec_wt	0.812	0.832	0.856	0.883	0.905	0.919	0.925	0.928	0.936	0.948	0.954	0.967
dsum_t	0.530	0.631	0.712	0.793	0.853	0.878	0.898	0.909	0.916	0.926	0.934	0.954
del_h_l	0.679	0.731	0.795	0.835	0.868	0.889	0.903	0.917	0.927	0.935	0.942	0.959
chg_std	0.834	0.847	0.861	0.932	0.943	0.948	0.950	0.951	0.964	0.967	0.973	0.982
ed_std	0.652	0.683	0.777	0.824	0.861	0.893	0.902	0.912	0.930	0.942	0.949	0.963
logp	0.677	0.704	0.746	0.801	0.843	0.862	0.877	0.889	0.906	0.919	0.931	0.952
mr	0.827	0.852	0.873	0.898	0.921	0.933	0.938	0.944	0.952	0.956	0.962	0.970
vol	0.819	0.848	0.874	0.900	0.920	0.933	0.938	0.944	0.950	0.955	0.961	0.968
values	5967	5948	5866	5665	5209	4683	4138	3608	3074	2549	1975	787
						Section 3						
gsnm	0.538	0.641	0.737	0.794	0.847	0.871	0.884	0.888	0.900	0.913	0.923	0.951
ht_form	0.758	0.795	0.829	0.826	0.836	0.840	0.844	0.849	0.854	0.870	0.881	0.875
tot_eng	0.843	0.882	0.911	0.924	0.935	0.941	0.946	0.952	0.954	0.961	0.969	0.977
ion_pot	0.702	0.740	0.793	0.811	0.835	0.854	0.865	0.875	0.844	0.898	0.901	0.933
$fill_{orb}$	0.859	0.902	0.931	0.944	0.954	0.959	0.962	0.968	0.969	0.973	0.977	0.984
molec_wt	0.850	0.883	0.914	0.928	0.943	0.946	0.952	0.955	0.958	0.964	0.970	0.979
dsum_t	0.611	0.683	0.749	0.804	0.839	0.862	0.878	0.878	0.890	0.903	0.911	0.907
del_h_l	0.756	0.787	0.834	0.853	0.874	0.895	0.905	0.912	0.920	0.928	0.932	0.951
chg_std	0.553	0.770	0.837	0.883	0.926	0.948	0.955	0.958	0.956	0.958	0.956	0.939
ed_std	0.571	0.710	0.817	0.857	0.889	0.898	0.902	0.908	0.911	0.920	0.926	0.930
logp	0.613	0.718	0.767	0.803	0.833	0.855	0.869	0.887	0.897	0.909	0.920	0.917
mr	0.867	0.912	0.939	0.951	0.960	0.964	0.967	0.971	0.973	0.974	0.977	0.982
vol	0.848	0.901	0.933	0.946	0.957	0.961	0.965	0.970	0.973	0.975	0.979	0.985
values	5963	5946	5905	5798	5445	4957	4455	3734	2997	2247	1546	411

	number of clusters formed									
property	118	179	345	682	1298	2170	3207	4159	4870	5364
					Section 4					
gsnm	0.043	0.054	0.075	0.180	0.297	0.443	0.875	0.924	0.955	0.964
ht_form	0.076	0.110	0.091	0.122	0.217	0.565	0.809	0.901	0.963	0.977
tot_eng	0.058	0.059	0.075	0.134	0.230	0.543	0.831	0.902	0.969	0.975
ion_pot	0.332	0.334	0.371	0.400	0.470	0.690	0.868	0.926	0.950	0.962
fill_orb	0.052	0.058	0.075	0.133	0.248	0.527	0.877	0.935	0.971	0.973
molec_wt	0.051	0.052	0.076	0.122	0.228	0.534	0.873	0.931	0.967	0.967
dsum_t	0.154	0.158	0.322	0.333	0.412	0.630	0.883	0.913	0.936	0.956
del_h_l	0.130	0.132	0.145	0.208	0.257	0.437	0.861	0.920	0.957	0.963
chg_std	0.052	0.053	0.070	0.098	0.138	0.809	0.955	0.971	0.979	0.985
ed_std	0.052	0.053	0.070	0.098	0.138	0.809	0.955	0.971	0.979	0.985
ed_std	0.041	0.068	0.101	0.111	0.219	0.759	0.867	0.929	0.956	0.959
logp	0.072	0.074	0.062	0.126	0.332	0.563	0.842	0.898	0.939	0.951
mr	0.097	0.101	0.116	0.154	0.255	0.536	0.907	0.969	0.980	0.980
vol	0.039	0.049	0.067	0.116	0.240	0.540	0.901	0.956	0.974	0.976
values	5860	5796	5620	5272	4655	3787	2755	1688	949	460

more than a thousand clusters is almost identical to group-average. In contrast, for *vol*, minimum-diameter is consistently better than group-average, and for more than a thousand clusters is almost identical to Ward.

An entirely comparable series of experiments was carried out using the range-standardized data, with very similar results to those reported above for the standard-deviation-standardized data. Jarvis–Patrick is again consistently inferior to the three hierarchic methods, all of which have very similar PMCC values for the same numbers of clusters produced. Minimum-diameter again tends to give higher PMCC values when less than 1000 clusters are produced, after which Ward and group-

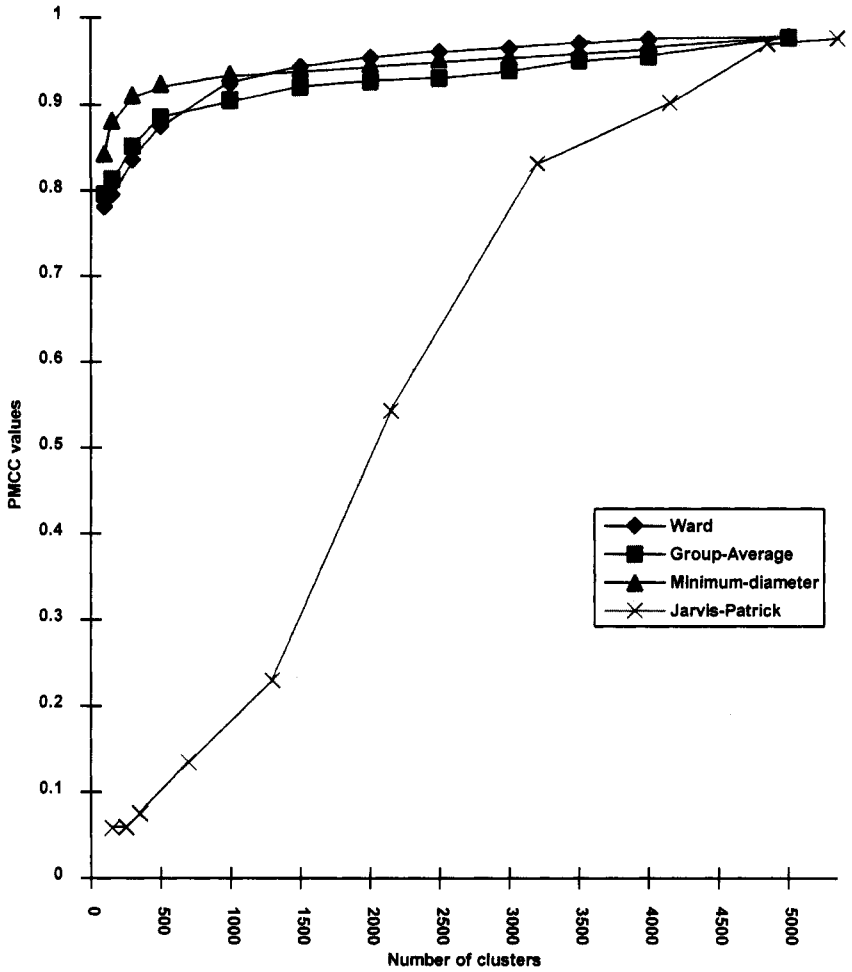

Figure 4. Comparison of the PMCC values using *tot_eng* property values for the four clustering methods.

Table 4. Summary of PMCC Value Ranges for Dataset-A

	std-deviation standardized		range standardized	
	min.	max.	min.	max.
Ward	0.369	0.986	0.395	0.991
group-average	0.530	0.982	0.469	0.970
minimum-diameter	0.538	0.985	0.465	0.992
Jarvis–Patrick	0.039	0.985	−0.027	0.990

[a] Summary of PMCC Value Ranges for Dataset-A

average tend to have higher values. There are some differences in that some properties give slightly higher PMCC values with one standardization method than with the other. Nontrivial differences were observed with only two of the properties, these being *ht_form* and *dsum_t*. In the former case, above 250 clusters the minimum-diameter PMCC values decline to a low point at about 1000 clusters after which they gradually increase again. The values of Ward and group-average are similarly erratic, but they maintain a general upward trend. However, there is a large difference between them, with group-average having consistently higher values. For *dsum_t*, the values are significantly lower and more erratic than for any other property, and yet Jarvis–Patrick produces less wide-ranging values. Minimum-diameter does not produce reasonable PMCC values at any stage, and, although much higher, the Ward and group-average values are not good. The reason for this behavior is not clear.

5. SUMMARY AND CONCLUSIONS

In this paper, we have discussed nearest-neighbor searching algorithms and clustering methods that can be used to process sets of molecules characterized by global molecular properties.

A literature review of nearest-neighbor searching algorithms suggests that the triangle-inequality algorithm is the most suitable for a large dataset with many descriptors. Our implementation of this algorithm demonstrates that it can bring about substantial reductions in the number of distance calculations required in a nearest-neighbor search using the Euclidean distance when compared with the simpler brute-force algorithm. However, when the top-20 nearest neighbors are required (rather than just the first nearest neighbor), the overheads involved result in the triangle-inequality algorithm being slower than the brute-force algorithm. Our results thus indicate that the triangle-inequality algorithm may be inappropriate where more than one nearest neighbor is required and where the full distance calculation involves only a few tens of descriptors (making it relatively quick to calculate).

Clustering experiments were carried out with the Jarvis–Patrick, Ward, group-average, and minimum-diameter clustering methods, with the classifications resulting from these methods being evalulated by means of simulated property prediction experiments. Our experiments suggest that the Jarvis–Patrick method, which is widely used for clustering molecules characterized by fragment-incidence data, is much less appropriate for use with molecules characterized by property data. Specifically, the range of partitions producing reasonable PMCC values is much narrower than for the other three methods and limited to those producing very large numbers of small clusters (and consequently using few property values in the PMCC calculation). In the most important lower-number-of-clusters range (up to 1000 clusters), the Jarvis–Patrick method performs particularly badly. Examination of the cluster sizes shows the cluster size distributions to be very

uneven in this range, with typically a few very large clusters and a lot of very small ones. Thus, although this method is the fastest, the results are poor and restrictive.

The other three methods all give similar ranges of PMCC values. The minimum-diameter method performed particularly well at the lower end of the number-of-clusters range, generally producing better PMCC values for up to 1000 clusters than Ward or group-average. Examination of the cluster sizes shows the cluster-size distribution to be more even in this range than for Ward and group-average, with slightly fewer singletons and doublets. Unfortunately, the minimum-diameter method does not benefit from an equivalent to the RNN algorithm. It is thus extremely time-consuming, and the requirement for a sorted list of all dissimilarities makes the method inappropriate for large numbers of structures. It is not known whether a modified version of the method could reduce this requirement. The Ward and group-average methods perform equally well, and the availability of the RNN technique means that they are only slightly more time-consuming in operation than the Jarvis–Patrick method. These methods generally perform less well than the minimum-diameter method for partitions up to 1000 clusters and better than it thereafter.

This work was undertaken as part of an ongoing project at CAS to develop new ways of providing access to CAS Registry data.[10,32] One area of current interest is in investigation of 3-D substructure searching techniques.[32] Such searches are known to result in search outputs that are often much larger than is common in 2-D substructure searching, especially if the methods are to be extended in the future to encompass flexible 3-D searching.[33,34] Accordingly, we need to provide searchers with tools that can assist them in obtaining an overview of the range of structural types represented in the output from a search. Hierarchic clustering methods are very appropriate for such an application since they enable a user to inspect the search output at whatever level of detail seems appropriate given the size of the output that has been produced. The results obtained in this work suggest that such methods can provide an effective means of grouping related structures.

ACKNOWLEDGMENT

We gratefully acknowledge funding and support from Chemical Abstracts Service, and we thank Kevin Cross for assistance in the generation of property data, John Barnard for assistance with the implementation of the minimum-diameter method, and Andrew Poirrette for a careful reading of an initial draft of this manuscript. The Krebs Institute for Biomolecular Research is a recognized center for the Molecular Recognition Initiative of the Science and Engineering Research Council.

REFERENCES AND NOTES

(1) Willett, P. *Similarity and Clustering in Chemical Information Systems*; Letchworth: Research Studies Press, 1987.
(2) Johnson, M. A.; Maggiora, G. *Concepts and Applications of Molecular Similarity*; New York: John Wiley, 1990.
(3) Barnard, J. M.; Downs, G. M. Clustering of Chemical Structures on the Basis of Two-Dimensional Similarity Measures. *J. Chem. Inform. Comput. Sci.* **1992**, *32*, 644–649.
(4) Downs, G. M.; Willett, P. Clustering of Chemical-Structure Databases for Compound Selection. In *Chemometric Methods in Molecular Design*; van de Waterbeemd, H., Ed.; in press.
(5) Willett, P.; Winterman, V.; Bawden, D. Implementation of Non-Hierarchic Clustering Methods in Chemical Information Systems: Selection of Chemical Compounds for Biological Testing and Clustering of Substructure Search Output. *J. Chem. Inform. Comput. Sci.* **1986**, *26*, 109–118.
(6) Lajiness, M. S.; Johnson, M. A.; Maggiora, G. Implementing Drug Screening Programmes Using Molecular Similarity Methods. In *QSAR: Quantitative Structure-Activity Relatioships in Drug Design*; Fauchere, J. L., Ed.; Alan, R. Liss Inc.: New York, 1989; pp 173–176.
(7) Jarvis, R. A.; Patrick, E. A. Clustering Using a Similarity Measure Based on Shared Nearest Neighbors. *IEEE Trans. Comput.* **1973**, *C-22*, 1025–1034.
(8) Willett, P.; Winterman, V.; Bawden, D. Implementation of Nearest Neighbor Searching in an Online Chemical Structure Search System. *J. Chem. Inform. Comput. Sci.* **1986**, *26*, 36–41.
(9) Willett, P. The Calculation of Inter-Molecular Similarity Coefficients Using an Inverted-File Algorithm. *Anal. Chim. Acta* **1982**, *138*, 339–342.
(10) Fisanick, W.; Cross, K. P.; Rusinko, A., III, Characteristics of Computer-Generated 3D and Related Molecular Property Data for CAS Registry Substances. *Tetrahedron Comput. Methodol.* **1990**, *3*, 635–652.
(11) Sneath, P. H. A.; Sokal, R. R. *Numerical Taxonomy*; W. H. Freeman: San Francisco, 1973.
(12) Everitt, B. S. *Cluster Analysis*; Edward Arnold: London, 1993.
(13) Bath, P. A.; Morris, C. A.; Willett, P. Effect of Standardization of Fragment-based Measures of Structural Similarity. *J. Chemometrics* **1993**, *7*, 543–550.
(14) Milligan, G. W.; Cooper, M. C. A Study of Standardization of Variables in Cluster Analysis. *J. Classification* **1988**, *5*, 181–204.
(15) Willett, P.; Winterman, V. A. Comparison of some Measures for the Determination of Inter-Molecular Structural Similarity. *Quant. Struct.-Act. Relat.* **1986**, *5*, 18–25.
(16) Gordon, A. D. *Classification*; Chapman and Hall: London, 1981.
(17) Guenoche, A.; Hansen, P.; Jaumard, B. Efficient Algorithms for Divisive Hierarchical Clustering with the Diameter Criterion. *J. Classification* **1991**, *8*, 5–30.
(18) Willett, P. Some Heuristics for Nearest-Neighbor Searching in Chemical Structure Files. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 22–25.
(19) Murtagh, F. *Multidimensional Clustering* Algorithms. *COMPSTAT Lectures*, *4*; Physica-Verlag: Vienna, 1985.
(20) Murtagh, F. Search Algorithms for Numeric and Quantitative Data. In: *Intelligent Information Retrieval: the Case of Astronomy and Related Space Sciences*; Heck, A., Murtagh, F., Eds.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1993; pp 29–48.
(21) Shasha, D.; Wang, T.-L. New Techniques for Best-Match Retrieval. *ACM Trans. Inform. Syst.* **1990**, *8*, 140–158.
(22) Yunck, T. P. A Technique to Identify Nearest Neighbors. *IEEE Trans. Syst. Man Cybernet.* **1976**, *SMC-6*, 678–683.
(23) Kittler, J. A. Method for Determining K-Nearest Neighbors. *Kybernetes* **1978**, *7*, 313–315.
(24) Richetin, M.; Rives, G.; Naranjo, M. Rapid Algorithm for the Determination of K-Nearest Neighbors. *RAIRO Informatique/Computer Science* **1980**, *14*, 369–378.
(25) Burkhard, W. A.; Keller, R. M. Some Approaches to Best-Match File Searching. *Comm. ACM* **1973**, *16*, 230–236.
(26) Shapiro, M. The Choice of Reference Points in Best-Match File Searching. *Comm. ACM* **1977**, *20*, 330–343.
(27) Nevalainen, O.; Katajainen, J. Experiments with a Closest Point Algorithm in Hamming Space. *Angew. Informatik* **1982**, *5*, 277–281.
(28) Murtagh, F. A. Survey of Recent Advances in Hierarchical Clustering Algorithms. *Comput. J.* **1983**, *26*, 354–359.
(29) Lance, G. N.; Williams, W. T. A General Theory of Classifactory Sorting Systems. I. Hierarchical systems. *Comput. J.* **1967**, *9*, 373–380.
(30) Voorhees, E. M. Implementing Agglomerative Hierachcial Clustering Algorithms for Use in Document Retrieval. *Inf. Process. Manage.* **1986**, *22*, 465–476.
(31) Rao, M. R. Cluster Analysis and Mathematical Programming. *J. Amer. Stat. Assoc.* **1971**, *66*, 622–626.
(32) Fisanick, W.; Cross, K. P.; Forman, J. C.; Rusinko, A., III Experimental System for Similarity and 3D Searching of CAS Registry Substances. 1. 3D Substructure Searching. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 548–559.
(33) Clark, D. E.; Willett, P.; Kenny, P. W. Pharmacophoric Pattern Matching in Files of Three-Dimensional Chemical Structures: Representation and Searching of Conformationally-Flexible Molecules. *J. Mol. Graph.* **1992**, *10*, 194–204.
(34) Clark, D. E.; Jones, G.; Willett, P.; Kenny, P. W.; Glen, R. C. Pharmacophoric Pattern Matching in Files of Three-Dimensional Chemical Structures: Comparison of Conformational-Searching Algorithms. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 197–206.