Algorithms for Clustering Molecular Dynamics Configurations

ANDREW E. TORDA* and WILFRED F. VAN GUNSTEREN

Computational Chemistry (Physical Chemistry), ETH Zentrum, Zürich, CH-8092, Switzerland

Received 3 November 1993; accepted 24 February 1994

ABSTRACT

Two traditional clustering algorithms are applied to configurations from a long molecular dynamics trajectory and compared using two sets of test data. First, a subset of atoms was chosen to present conformations which naturally fall into a number of clusters. Second, a subset of atoms was selected to span a relatively continuous region of conformational space rather than form discrete conformational classes. Of the two algorithms used, the single linkage method is inappropriate for this kind of data. The divisive hierarchical method, based on minimizing the difference between cluster centroids and extrema, is successful but also prone to imposing clustering hierarchy where none can be justified.

Introduction

A typical molecular dynamics (MD) simulation may generate thousands of configurations of a system separated by regular steps in time. If, however, one is interested in structural (rather than dynamic) properties, this implicit time axis will serve more as a nuisance than as a useful property. In this case, one wants to identify states which are frequently and repeatedly populated, regardless of when they occur in a simulation. By definition, this amounts to a requirement for traditional statistical cluster analysis.

Regardless of the clustering algorithm used, any cluster analysis procedure will require a similarity matrix in which each element represents the structural difference between a pair of structures. It is also often convenient to regard this matrix as the distance matrix for a high-dimensional polygon in which each point represents a single structure. In 1983, Levitt[1] calculated such matrices and their projections into two-dimensional Cartesian coordinates. These projections could then be used to sketch out the path of a trajectory and suggest the presence of clusters in conformational space. The problem with this approach is that the high-dimensional polygon corresponding to the original distance matrix may be poorly represented in two-dimensional space and, for example, points close in the low-dimensional space may actually

* Author to whom all correspondence should be addressed.

CCC 0192-8651 / 94 / 121331-10

be separated by larger distances before the projection.

True automatic cluster analysis has been applied to conformations, although not necessarily from MD trajectories. Unger et al.[2] looked at more than 80,000 hexamer fragments taken from literature protein structures. After an initial assessment of the data, a 1 Å cutoff of root mean square (rms) difference in coordinates was used for both the formation of initial clusters and the centers of new clusters during an iterative process. This procedure effectively reduced the mass of data to about 100 important structural units. Rooman et al.[3] clustered short peptide fragments into a hierarchical scheme based on the sum of the squared distances of individual elements to the cluster's center of mass. This was extremely successful in identifying broad conformational classes and recurring motifs. Karpen et al.[4] used an iterative approach to cluster 15,000 configurations from a trajectory of a pentapeptide based on differences in dihedral angles. This method strongly relied on the selection of an arbitrary cluster cutoff distance based on arguments about what constituted a distinct conformation as compared to thermal fluctuations.

Finally, a novel approach to cluster analysis of trajectories was applied by Gordon and Somorjai.[5] Fuzzy cluster analysis was used to group parathyroid hormone fragment conformations into a predetermined number of clusters. This method has the advantage that it retains the idea of reducing a mass of structures down to a manageable number of representative cluster centres while conceding that cluster membership is not absolute. A transition state, for example, may well be best treated by giving it fuzzy membership of more than one cluster. Unlike the clustering methods which required the choice of a distance cutoff for cluster membership, this implementation required a predetermined number of clusters.

In contrast to previous work, our aim was to apply more than one clustering algorithm in an attempt to determine what type of procedure best suits the particular nature of MD configurations. We also wished to assess the utility of cluster analysis for systems larger than the small peptides typically used by previous workers. Furthermore, it was of interest to use more than one data set, because some data will form clusters naturally, whereas some will have relatively little structure. Finally, we were interested in algorithms which did not require advance selection of cutoffs for cluster size or the number of clusters.

Methods

SIMILARITY MEASURES

Regardless of the cluster analysis algorithm, one needs a measure of how each element differs from every other. Root mean square differences of Cartesian coordinates have been used,[2,5] but there are two possible disadvantages. First, until very recently, there was no evidence that this measure obeyed the triangle inequality. Second, we wished to avoid relying on optimal superposition of coordinates, although this is only a minor technical point. This problem could be avoided, as well as the necessity for coordinate superposition, by clustering based on rms differences of dihedral angles.[4] Unfortunately, structural similarity is not necessarily well correlated with the difference in internal angles. A small change in a single angle may lead to a large structural change due to leverage effects, whereas two large changes in dihedral angles may lead to a minimal structural change. For example, a peptide plane flip involves two large dihedral angle changes, but possibly little overall structural change.

For our analysis, we follow Rooman et al.[3] and Levitt,[1] who constructed similarity matrices based on the rms deviation of intramolecular distances, also referred to as the distance matrix error.[6] We define the difference D_{ab} between two conformations, a and b, as

$$D_{ab} = \left(\frac{2}{N(N-1)} \sum_{i<j}^{N} \left(d_{ij}^{a} - d_{ij}^{b} \right)^2 \right)^{1/2} \qquad (1)$$

where the summation runs over all pairs ij of the N atoms being considered in the configurations a and b. d_{ij} is the three-dimensional distance between atoms i and j. This measure has the advantage that D_{ab} is well correlated with the difference between structures a and b. It has the clear disadvantage that it is not sensitive to chirality, so in the extreme case, $D_{ab} = 0$ for mirror image structures. Fortunately, we do not expect chiral inversions or changes in overall fold during a MD trajectory. The measure does obey the triangle inequality, so if one know the high-dimensional distance d_{ab} between conformers a and b and the distance d_{bc} between conformers b and c, this does put bounds on the distance d_{ac} between conformers a and c. This is more than a mathematical curiosity if we wish to apply a clustering algorithm which relies

knowing the shortest path between two conformers.

GENERATION OF CONFIGURATIONS

Configurations were taken 0.5 ps from a 1 ns trajectory *in vacuo* of the 64-residue structured domain of barley serine protease inhibitor. This yielded 2000 structures to be clustered. Because this trajectory was solely to serve as a source of structures, the MD simulation is described only briefly. The calculations were carried out using the GROMOS simulation package,[7] with a timestep of 2 fs and weakly coupled to a temperature bath,[8] at 300 K with a relaxation time of $\tau_T = 0.1$ ps. The SHAKE algorithm was used to constrain bond lengths.[9] Starting coordinates were taken from a published, nuclear magnetic resonance (NMR) derived structure of barley serine protease inhibitor.[10] Time-averaged distance restraints[11] and *J*-coupling restraints[12] were both imposed using a relaxation time of 20 ps. Experimental distance restraints were taken from Poulsen[13] and *J*-coupling restraints from published data.[14] An initial 30 ps of simulation were used to equilibrate the system before the 1 ns used for analysis.

HIERARCHICAL DIVISIVE ALGORITHM

The first clustering algorithm applied to the data was of the divisive hierarchical type.[15] First we define some properties of a cluster using the analogy of a graph. From the similarity matrix, the distance from each configuration to every other configuration is known. The largest such distance for a configuration is called the eccentricity of that configuration. The configuration of smallest eccentricity is known as the cluster centroid. This point is marked by a cross in Figure 1. We can also consider the largest single distance within a cluster and call this the diameter of the cluster. This is marked by the arrowed line in Figure 1. The pair of points separated by this distance are called the extrema and are circled in the figure.

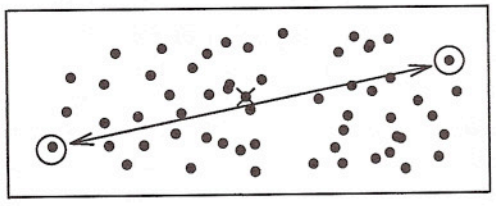

FIGURE 1. Definitions of terms applied to clusters. The crossed point is the cluster centroid, the circled points are the extrema, and the arrow shows the cluster diameter.

The actual divisive algorithm is best explained by Figure 2. Initially, all points are part of one large cluster (A). After calculating the eccentricity of each point, the two extrema (circled points) are taken as the initial centroids for two new clusters. Each point in the parent cluster is then apportioned to one of the new clusters depending on which centroid it is closest to. This division, shown at stage B, results in some points falling on the wrong side of the natural clustering line. The centroid of each of the new clusters is then calculated (stage C) and the points are reapportioned (stage D). The centers of the new clusters are recalculated and the points reapportioned until the centroid points are stable or a maximum number of iterations is reached. We set the maximum number of iterations to 10, but in practice, convergence was usually achieved in two or three cycles of centroid calculation and reapportioning.

Although we did not observe any convergence problems, conceptually, one can construct a set of points with a number of outliers. If the outlying points are sufficiently removed, they will form their own clusters. More likely, they would have the effect of distorting the final clusters. Because they affect the selection of cluster centroid, they

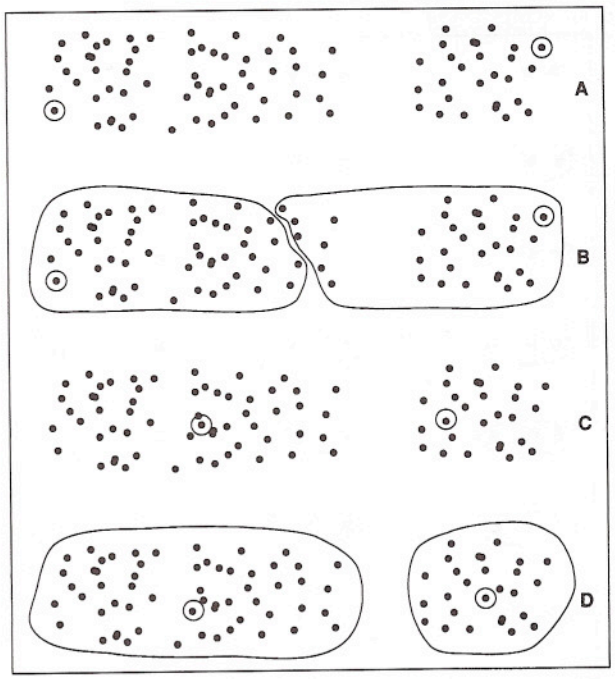

FIGURE 2. The divisive hierarchical clustering algorithm applied to a set of two-dimensional points.

could also add to the number of iterations needed for convergence. The use of a graph definition of cluster centroid has an important side effect. If one were to use a center of mass approach to calculating cluster centroid, the exact distance to an outlier determines the degree to which the cluster centroid is skewed. With the graph definition, it only matters that a point is of high eccentricity, not exactly how high.

SINGLE LINKAGE ALGORITHM

The second clustering algorithm applied to the data was based on the single linkage method.[15] In this procedure, the entire data set is considered as a weighted undirected graph with the configurations as vertices and the elements of the similarity matrix as weighted edges. Kruskal's algorithm[16] is then used to find a minimal spanning tree. At each stage, each connected subgraph is a potentially interesting cluster.

This can be understood by considering the simple example shown in Figure 3. This shows a set of points scattered in only one dimension and with the distance between every pair of points (configurations) known. The distances are first sorted and then considered, in turn, starting from the shortest distance. The top line of Figure 3 shows such a set of points with no initial clusters. In the first step (a), the first distance is considered, two points are joined, and the first cluster of only two configurations is formed. In step (b), the next distance is considered, adding a third point to the first cluster. In step (c), the next shortest distance joins two previously unclustered points, forming a new cluster. In step (d), a third cluster is formed and all the points are members of some cluster. This still, however, does not constitute a minimal spanning tree, so the algorithm is not finished. In steps (e) and (f), there is a merging of clusters until all the points form a single grouping.

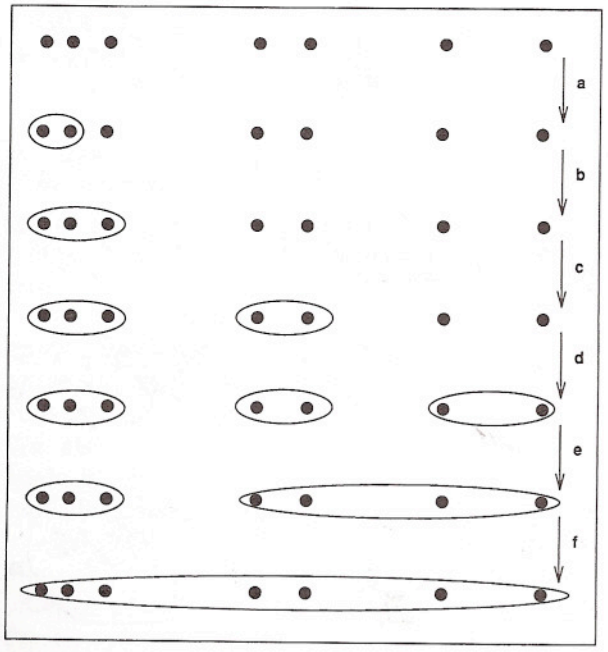

FIGURE 3. The single linkage clustering algorithm applied to a set of one-dimensional points.

The algorithm has several attractive aspects. First, it is intuitively appealing to form clusters by joining the most closely related structures, regardless of when they occur in a trajectory. Second, the progress of the algorithm is readily monitored. At each step, one can count and plot the number of clusters formed and the number of structures clustered as a function of distance. If there is a sudden change in the number of clusters as the distance increases, it suggests a natural structure in the data. Finally, if one is willing to discard outlying points, there is no need to continue until all points are clustered.

Results

To compare clustering algorithms, it is worth trying more than one kind of data set. Ideally, there should be one set in which configurations fall into clear and natural clusters. Any algorithm should work with this data. At the same time, it is useful to have a data set with less structure. One would like to know if an algorithm has a tendency to produce misleading clusters where they are not physically justified. By choosing subsets of atoms from the barley serine protease inhibitor simulation, we attempted to generate two such data sets. To this end, we first selected set (a): all backbone heavy atoms (carbonyl carbon, amide nitrogen, and C^{α}) from residues 46–51 and 60–64. These atoms form two strands of a four-stranded beta sheet,[17] and an examination of the NMR distance information[13] suggested the presence of distinct conformational states. For the data set with less structure, set (b), we simply took the backbone heavy atoms from all 64 residues of the protein. Although individual regions may hop between conformational substates, the combination of all such transitions in different parts of the molecule will tend to distribute the conformations more evenly through conformational space.

The justification for the atom selections can be seen by considering the distribution of similarities between configurations. There are $N_c = 2000$ trajectory snapshots and thus 1,999,000 ($N_c(N_c - 1)/2$) unique nonzero entries in the similarity matrix for either atom subset. The histograms in Figure 4 show the distribution of the similarities. In the upper panel (a), the distribution is plotted for atom set (a), the small section from the beta strands. The most common difference between structures is about 0.8 Å; but, more importantly, there is a distinct asymmetry to the plot. This is in contrast to the lower panel, which shows the same distribution when all backbone atoms are considered. The most common difference between structures is 1.5 Å, but the distribution is symmetric about the median. This reflects the relatively unstructured uniform distribution of configurations in the high-dimensional space.

APPLICATION OF THE DIVISIVE HIERARCHICAL CLUSTERING ALGORITHM

When run to its conclusion, the result of the hierarchical clustering algorithm is to spread the trajectory across a binary tree. At the root of the tree is the initial cluster consisting of all configurations. Going down, in a hierarchical fashion, are child nodes where each is a subcluster. At the leaves of the tree are individual configurations. This way of viewing structures is different from the usual picture of a trajectory as a function of time. For N_c configurations, the tree will have N_c, leaves, but because it is not a balanced binary tree, only a lower limit can be specified for the number of levels in the tree ($\log_2(N_c)$). For the 2000 structures considered in each of the data sets here, this means at least 11 levels. In practice, transition states and small energetic mimima lead to the formation of many small clusters and a distinctly unbalanced and weedy binary tree.

If one simply wanted to divide a trajectory's conformational space, the binary tree could be taken as a final result, but this would not be very informative. More usefully, one can begin to look for relatively populated conformational states. These should be collections of structures confined to small conformational volumes. With the binary tree representation, such clusters are easily located by a recursive traversal of the tree, starting from the root and selecting each node farthest from the

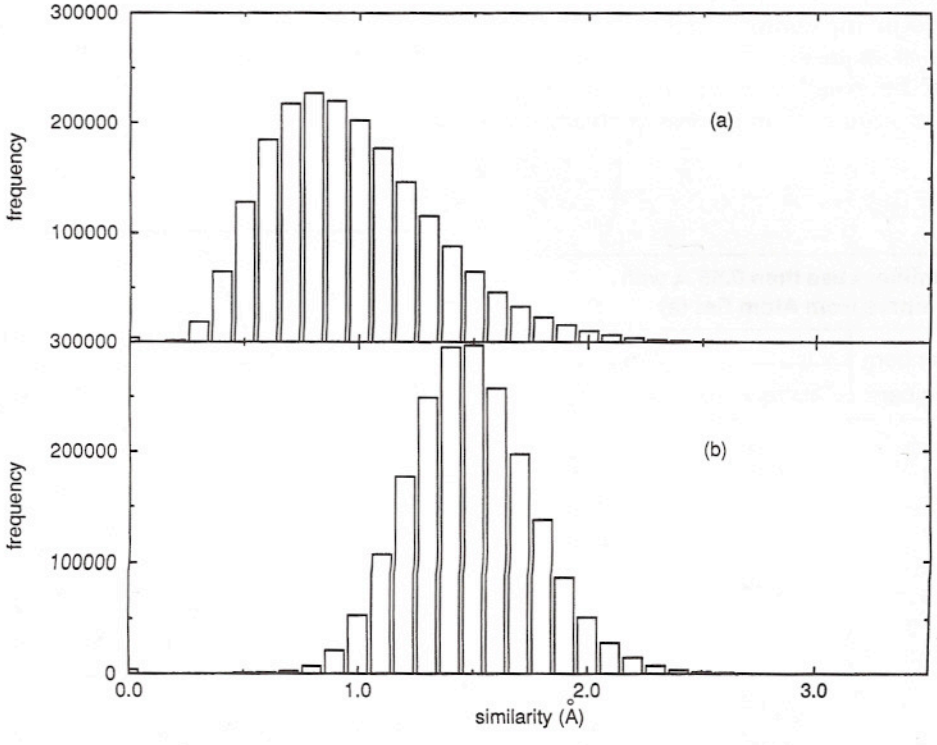

FIGURE 4. Histograms of interconfiguration similarities calculated from eq. (1) applied to 2000 barley serine protease MD configurations. (a) The distribution of distances between configurations based on the subset of backbone atoms from the two strands of beta sheet. (b) The distribution of similarities based on all backbone atoms.

root with a minimum number of members and with a diameter less than some arbitrary size. Selecting the node farthest from the root means that a node is not selected if one of its children meets the selection criteria.

This process was first applied to atom set (a), the two backbone fragments from the beta sheet of barley serine protease. Figure 4a shows that the most common interconfiguration distance is about 0.8 Å, so true clusters should be of a smaller diameter than that. To search for important narrow energetic wells, one can then pick a small maximum cluster diameter of, for example, 0.55 Å and only those clusters with 10 or more members. This leads to 13 clusters, as summarized in Table I. Although this group of clusters accounts for only 138 of the original 2000 structures, the selection criteria ensure that they are representative of populated energetic minima. The table also shows another important property of the clusters. This is the time, within the original trajectory, spanned by each cluster. The whole trajectory spans 1000 ps, and the timespan of a cluster is calculated by subtracting the time of the first from the last cluster member. This is of interest because structures close to each other in sequence will tend to be structurally similar. If the members of a cluster cover a significant amount of time, it suggests that they are true recurring conformations.

In the case of atom set (a), the success of the algorithm can be roughly gauged from simple plots of the structures. Plotting the centroid and the two extrema of a cluster gives a good indication of its spread while presenting a fairly clear picture. For example, Figure 5 shows the first four clusters from Table I. Picking any of the other clusters gives a similar picture. The structures in each panel were all fitted to the centroid of the first cluster of Table I and are all shown in the same orientation within their box. The four clusters shown all have the expected internal similarity but differ from each other in aspects like the orientation of one end of one of the beta strands or the backbone angles in another region.

The divisive algorithm was then applied to atom set (b) (all heavy backbone atoms). The entire set of conformations spans 3.47 Å, and Figure 4 shows that the most common interconfiguration distance is near 1.5 Å. Again, to search for low-energy (highly populated) recurring conformations, one might search for clusters of diameter less than 1.2 Å with more than 10 members. This leads to 10 clusters described in Table II. There are two main

TABLE I.
Clusters of Diameter Less than 0.55 Å with 10 or More Members from Atom Set (a).

Cluster	Number members	Diameter (Å)	Time spanned (ps)
0	12	0.51	726
1	11	0.52	581
2	10	0.53	843
3	10	0.48	933
4	10	0.50	498
5	11	0.53	469
6	10	0.54	761
7	11	0.51	611
8	10	0.54	612
9	10	0.52	782
10	11	0.54	720
11	11	0.53	741
12	11	0.51	682

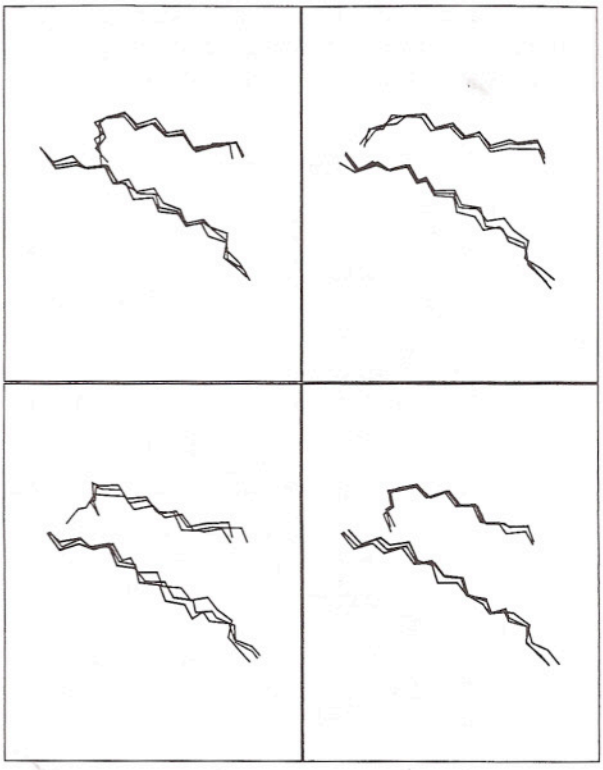

FIGURE 5. Example clusters formed from atom set (a) using the divisive algorithm. Each panel shows the centroid and two extrema from one cluster. Each individual structure consists of two strands of backbone heavy atoms. All structures in all panels were fitted to the centroid of the upper right cluster, based on the shown subset of atoms.

TABLE II.
Clusters of Diameter Less Than 1.2 Å with 10 or More Members from Atom Set (b).

Cluster	Number members	Diameter (Å)	Time spanned (ps)
0	12	1.19	789
1	10	1.18	244
2	10	1.14	5
3	11	1.15	842
4	11	1.08	357
5	11	1.12	7
6	10	1.10	221
7	10	1.15	642
8	11	1.15	6
9	10	1.15	5

differences when compared to the data in Table I. First, to locate a reasonable number of clusters with at least 10 members, one has to use a cluster diameter cutoff of about 1.2 Å rather than 0.55 Å, as for the previous atom set. Whether one regards this spread of structures as structurally similar is a matter of arbitrary judgment. Second, the last column of Table II shows that four of the clusters consist almost entirely of sequential structures. These clusters are simply reflecting time proximity of structures rather than real recurring conformations. Although it is not shown in the table, clusters 4 and 6 have more than nine members which are also sequential in the original trajectory.

APPLICATION OF THE SINGLE LINKAGE CLUSTERING ALGORITHM

In contrast to the divisive approach, which considers the distance from each configuration to a cluster centroid, the single linkage algorithm is based on the shortest distance between any pair of structures. As described earlier, the progress of the algorithm can be monitored at each step by plotting the number of clusters that have been formed and the number of structures that have joined an existing cluster. These plots are shown in Figure 6 for both atom sets. The left-hand panels show the number of clusters as the range of interconfiguration distances increases, and the right-hand panels show the number of structures which have been clustered. Initially, when few configurations have joined a cluster, additional distances will often lead to the formation of a new, but smaller cluster. For atom set (a), the number of clusters continues to rise until 0.27 Å, when 191 clusters have been

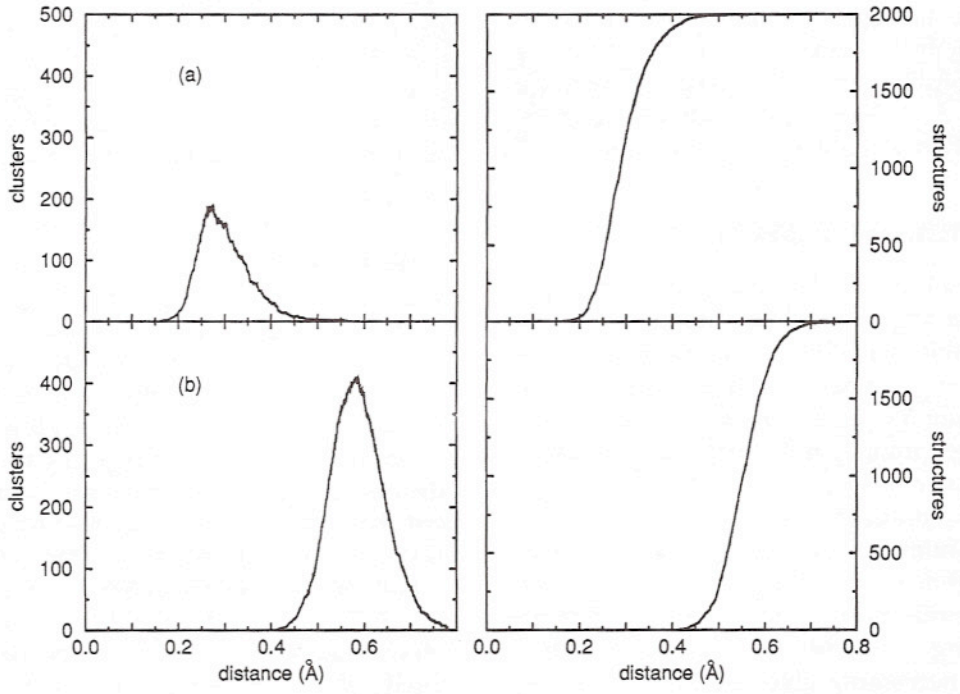

FIGURE 6. Application of the single linkage algorithm. The upper panels are for data set (a), the lower panels for data set (b). Left panels show the number of clusters formed as the distance between structures increases. Right panels show the number of configurations clustered up to that point in the algorithm.

formed, and then starts to fall as the number of cluster merge operations exceeds the number of cluster formation operations. For atom set (b), the corresponding point occurs at 0.58 Å after 409 clusters have been formed.

To attempt to extract useful information from the single linkage algorithm, one must look at a plot like Figure 6, pick a cutoff distance, and consider the clusters formed at that point. For example, with atom set (a), a cutoff distance of 0.30 Å results in 160 clusters. The right-hand panel of Figure 6 shows that 924 of 2000 structures have joined clusters at this point. Applying this cutoff leads to one cluster with 450 members, one of 21 members, and 158 clusters of 6 or less members. All of the tiny clusters consist of structures which are sequential in the original trajectory, and most have only two members. Clearly this is not a satisfactory result. In case the problem is that cluster merging has removed most of the information, one could look at 0.25 Å, where there are also 160 clusters. This leads to one cluster of 31 members, one of 28, and 158 clusters of 6 or less members. Later in the algorithm's progress, when more structures have been included, a cutoff of 0.34 Å results in 1646 structures in 100 clusters. This leads to one cluster of 1274 members, one of 107 members, and 98 clusters of 4 or less members. Applying the single linkage algorithm to atom set (b) results in exactly the same kind of disappointing behavior. Typically one or two large clusters are formed with a plethora of tiny clusters usually consisting of two sequential configurations.

RUNNING TIME OF ALGORITHMS

Aside from judging the results of each algorithm, one can also assess their running time. The hierarchical divisive method is bounded by $O(N_s^2)$, where N_s is the number of structures to be clustered. The running time for the single linkage method is the running time for Kruskal's minimum spanning tree algorithm.[16] This is $O(e \log e)$, where e is the number of edges in the graph. In our case, the number of edges is $(N_s^2(N_s^2 - 1)/2$, so the algorithm is bounded by $O(N_s^2 \log N_c^2)$. Although algorithms exist which will find a minimum spanning tree with $O(N_s^2)$ running time, these do not necessarily give minimum spanning trees of subgraphs at each step. In other words, they do not provide clusters at each stage of the calculation.

In practice, these running times are not critical. The time-consuming step is the calculation of the initial similarity matrix. Before two configurations can be compared using eq. (1), one needs a matrix of internal distances for each configuration. This operation is $O(N_a^2)$, where N_a^2 is the number of atoms considered in each conformation. The comparison of these matrices to generate the final similarity matrix requires quadratic time with respect to the number of structures. Although the final result is bounded by $O(N_c^2 N_a^2)$, this does not account for a potential memory problem.

A single distance matrix has $(N_a^2(N_a^2 - 1))/2$ entries. If one has, for example, 100 atoms, this results in 4950 entries. With 2000 structures and using 4-byte floating-point arithmetic, storing all the distance matrices would require just under 40 Mb of memory. This means that for realistic sized calculations, the calculation is often going to be bound by input/output operations. The only relief is provided by the fact that the similarity matrix need only be calculated once for a specified set of atoms in a particular trajectory.

In contrast, the second step of clustering, applying an algorithm such as the hierarchical divisive method or single linkage method, can be conducted entirely in memory and is independent of the original number of atoms. Using the data here as an example, we began with 2000 structures and a similarity matrix of 1,999,000 entries. Using the same machine assumptions as before, this requires 8 Mb of memory. Of course, these figures are somewhat arbitrary and performance depends on available machines and the size of trajectories.

Discussion

The clearest result from these calculations is that the hierarchical divisive algorithm seems to produce useful, structurally significant results, but the single linkage algorithm produces nothing of value. Although this is surprising considering the ideal picture presented in Figure 3, the problem is that the points in a well-sampled trajectory do not have such a clear structure. This can be seen by considering the exaggerated situation shown in Figure 7. There are clearly two clusters present which should be automatically identified. Unfortunately, a line of closely spaced, less interesting points joins the two clusters. In the case of an MD trajectory, this might represent two conformational states joined by a transition pathway. The single

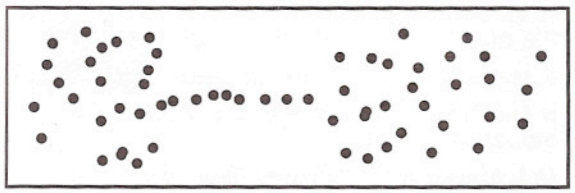

FIGURE 7. Distribution of points not suited to the single linkage algorithm.

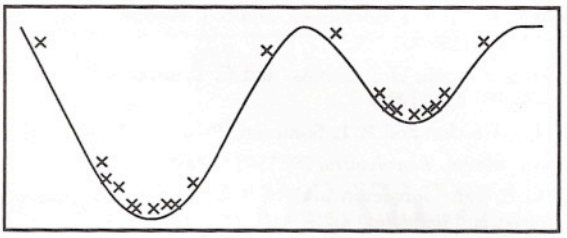

FIGURE 8. Distribution of structures over a simple energy surface.

linkage algorithm fails because it is based on the shortest distance between points. In this case, the diagram is constructed so that the most closely spaced points are in the transition region, and attempting to cluster this data would lead to a single cluster forming near the center and expanding to encompass the two interesting clusters.

Although the hierarchical divisive method produced apparently satisfactory results, it may be potentially misleading. One can imagine a set of points distributed uniformly on a regular grid with no natural clustering. The divisive method will take such data and simply carve it into even pieces, producing meaningless clusters. To some extent, this is what happens when the method is applied to atom set (b). If a trajectory's configurations are evenly distributed across conformational space, then the divisive algorithm will tend to produce no more then time slices, as seen in the last column of Table II.

This behavior could be seen as a weakness of the divisive algorithm, but it really reflects the nature of the data. Atom set (a) was chosen because we expected the small set of atoms to adopt a relatively small set of conformations. When considering the whole backbone, however [atom set (b)], the difference between any pair of structures will rarely reflect a single conformational property. Instead, it will reflect changes in many simultaneous and partly independent regions of the system.

It is of some interest to compare our results with the approaches of previous workers. Karpen et al.[4] took 15,000 structures spanning 2.2 ns and divided them into six clusters. In this work, no attempt was made to force all structures into a small number of clusters. Instead, we discussed only what we defined to be interesting clusters, even when they accounted for only a small fraction of the structures. The reason for this is shown in Figure 8, which shows a set of configurations scattered over a simplified one-dimensional energy surface. There are two populated energetic minima which should be identified as clusters as well as a number of structures occupying high-energy states. If the trajectory is sampled frequently enough, these conformers should spread themselves according to a Boltzmann distribution over the energy surface. This will include a number of structures in unlikely high-energy states. To obtain the most structurally important conformers, these outlying points should probably be ignored. Our means of selecting clusters tended to pick out configurations sitting in the bottom of energetic wells, whereas the approach of Karpen et al.[4] would tend to include even the high-energy structures. The fuzzy clustering approach of Gordon and Somorjai[5] would give these structures fuzzy membership of more than one cluster. The obvious way to sidestep this question would be to energy minimize all structures before cluster analysis and thus push each configuration toward the nearest energetic minimum.

As computers become faster, more data are produced and the need for data reduction becomes clearer. Already, trajectory simulations routinely generate 10^3 or 10^4 snapshots, and even NMR structure calculations may produce collections of 10^2 configurations. Cluster analysis is obviously one way to condense the structural information, albeit at the expense of dynamic properties. The only caveat is that not all algorithms are useful and, when applied to the wrong data, the results can be misleading.

Note Added in Proof. Since submitting this article, similar work,[18] in press, has been brought to our attention addressing almost identical issues. It is the intention of all parties to extend the work with a comparison of clustering algorithms and common data sets.

References

1. M. Levitt, *J. Mol. Biol.*, **168,** 621 (1983).
2. R. Unger, D. Harel, S. Wherland, and J. L. Susman, *Proteins*, **5,** 355 (1989).

3. M. J. Rooman, J. Rodriguez, and S. J. Wodak, *J. Mol. Biol.*, **213,** 327 (1990).
4. M. E. Karpen, D. J. Tobias, and C. L. Brooks III, *Biochemistry*, **32,** 412 (1993).
5. H. L. Gordon and R. L. Somorjai, *Proteins*, **14,** 249 (1992).
6. T. F. Havel, *Biopolymers*, **29,** 1565 (1990).
7. W. F. van Gunsteren and H. J. C. Berendsen, *Groningen Molecular Simulation (GROMOS) Library Manual*, Biomos, Groningen, The Netherlands, 1987.
8. H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, *J. Chem. Phys.*, **81,** 3684 (1984).
9. J.-P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen, *J. Comp. Phys.*, **23,** 327 (1977).
10. G. M. Clore, A. M. Gronenborn, M. Kjaer, and F. M. Poulsen, *Prot. Eng.*, **1,** 305 (1987).
11. A. E. Torda, R. M. Scheek, and W. F. van Gunsteren, *J. Mol. Biol.*, **214,** 223 (1990).
12. A. E. Torda, R. M. Brunne, T. Huber, H. Kessler, and W. F. van Gunsteren, *J. Biomol. NMR.*, **3,** 55 (1993).
13. F. M. Poulsen, personal communication.
14. S. Ludvigsen, K. V. Andersen, and F. M. Poulsen, *J. Mol. Biol.*, **217,** 731 (1991).
15. D. L. Massart and L. Kaufman, *The Interpretation of Analytical Chemical Data by the Use of Cluster Analysis*, John Wiley & Sons, New York, 1983.
16. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, MA, 1987.
17. C. A. McPhalen, I. Svendsen, I. Jonassen, and M. N. G. James, *Proc. Natl. Acad. Sci.*, **82,** 7242 (1985).
18. P. S. Shenkin and D. Q. McDonald, *J. Comp. Chem.*, in press (1994).