Rapid geometric searching in protein structures

Andrew T. Brint, Hazel M. Davies, Eleanor M. Mitchell and Peter Willett

Department of Information Studies, University of Sheffield, Western Bank, Sheffield, UK

This paper discusses the implementation of geometric searching in protein structures. A comparison of geometric searching algorithms that have been used for substructure searching in small 3D molecules suggests that the algorithm due to Lesk is most appropriate for searching for patterns of atoms in proteins; however, the computational requirements of this algorithm are considerable. An improved, two-stage procedure is described in which the Lesk algorithm is used as a precursor to the subgraph isomorphism algorithm of Ullman, and it is demonstrated that geometric searching can be implemented with a reasonable degree of efficiency.

Keywords: graph matching; Lesk algorithm; protein data bank; subgraph isomorphism algorithm; Ullman algorithm

INTRODUCTION

The development of X-ray crystallographic techniques has led to the increasing availability of three-dimensional (3D) coordinate data for proteins and other macromolecules. Much of this data is available publicly via the Protein Data Bank (PDB),[1,2,4] which contains 3D structural information for about 300 macromolecules. The PDB has long been used as an archive for the deposition of coordinates and as a data source for molecular graphics and related programs. To date, however, there have been only a few reports of attempts to exploit the coordinate data in the PDB by providing routines for the automatic identification of patterns contained within the 3D structures. Several studies have been reported of database searching systems for small 3D molecules, such as those in the Cambridge Crystallographic Databank (CCDB).[8,12,15,17,18] These studies have involved the specification of query patterns in terms of the distances separating the constituent atoms, and in this report we discuss the use of such information for searching the macromolecular structures in the PDB. Specifically, we describe algorithms that can be used for implementing geometric searching, i.e., the automatic identification of a user-defined pattern of atoms in a 3D chemical molecule,[8] in protein structures that are represented by the coordinates of the Cα atoms that form the protein backbone. The restriction to Cα atoms acts as a useful data reduction technique since it avoids the need to carry out an exhaustive search of all the atoms in the structure.

Address reprint requests to Dr. Willett at the Department of Information Studies, University of Sheffield, Western Bank, Sheffield S10 2TN, UK.

Received 6 July 1988; revised 19 December 1988

GEOMETRIC SEARCHING ALGORITHMS

Assume that the query pattern that is to be searched for in the PDB is represented by the set of N/Q constituent atoms and the associated $NQ(NQ-1)/2$ distinct interatomic distances (and that an analogous representation is used for each of the proteins in the database). Such a representation can be considered as a fully connected, labeled graph, in which the atoms and distances in the pattern or structure correspond to the nodes and edges of a graph. Geometric searching involves identifying such a query graph in a larger graph (i.e., a protein molecule in the PDB) and is thus an instance of the *subgraph isomorphism* problem that has been extensively studied in graph theory.[8,20,21]

A simple subgraph isomorphism algorithm for detecting the presence of all occurrences of a query pattern containing NQ atoms in a database structure containing NS atoms is as follows:

(1) Generate a combination of NQ of the NS atoms in the database structure.
(2) Check this combination for isomorphism (i.e., 3D structural equivalence) with the query pattern.
(3) Go to Step 1 if all of the $NS!/(NQ!(NS - NQ)!)$ combinations have not been generated yet.

This algorithm is extremely demanding of computational resources for two reasons. First, the factorial number of combinations that needs to be generated from the database structure and then tested for isophorism with the query pattern. In the case of protein structures represented by the Cα atoms, NS is likely to be of the order of some tens or a few hundreds and the number of combinations correspondingly huge. Second, the isomorphism test in Step 2 requires, at most $O(NQ!)$ query atom-to-database atom comparisons to confirm or deny the presence of the pattern.[20,21] Geometric searching in proteins is thus likely to be computationally feasible only if ways can be found to overcome these two problems.

The first published algorithm for geometric searching

is that due to Lesk, which was designed specifically for the processing of protein structures.[16] This algorithm is as follows:

(1) Calculate the interatomic distance for each distinct pair of atoms in the query pattern, and sort these distances into increasing order.
(2) A bit string is associated with each query atom, in which the Ith bit is set if that atom has another atom at the Ith distinct distance in the sorted list of query interatomic distances.
(3) Repeat Step 2 for each of the NS database atoms, so that each of them has an analogous bit string associated with it, with the Ith bit being set if that database atom has another database atom at the Ith distinct distance in the sorted list of query interatomic distances.
(4) Check the bit string associated with each of the database atoms to determine whether that database atom can be matched with one of the query atoms; if not, that database atom is eliminated from further consideration in the subsequent stages of the algorithm.
(5) If any database atoms have been eliminated from consideration, the bit strings corresponding to the remaining database atoms are regenerated (so as to reflect the atoms that have been eliminated in the previous iteration) and Steps 3 and 4 repeated until no further deletions can be achieved.
(6) Form a list for each of the query atoms of all of the remaining database atoms that are a possible match for it.
(7) Form a combination of NQ database atoms, one from each of the lists formed in Step 6, and test to see whether it is a match by trying to rotate it onto the query pattern.
(8) Repeat Step 7 for all of the possible combinations.

Brint and Willett[8] present a worked example of the algorithm when it is used for searching for a query pattern in a small molecule (rather than in a macromolecule).

The Lesk algorithm achieves efficiencies in operation, when compared with the previous, brute-force algorithm, by means of the bit string operations, which permit the identification of those database atoms that have atoms at the same distance (to within any allowed tolerance), as do the atoms in the query substructure. These sets of possible matches are iteratively refined by reducing the number of query atom-to-database atom equivalences; since there are fewer potential matches to be considered, there is a reduction in the number of combinations that need to be considered in the final, costly stage of the algorithm. Even so, Lesk's experiments, using a five-atom pattern that recurred in the α-helix of polyalanine, demonstrate clearly the long running times that result from the huge numbers of matches that still need to be considered in the final stage.

Although originally designed specifically for the processing of macromolecular structures, Lesk's algorithm can be used with any type of 3D coordinate data. It was accordingly one of the algorithms considered by Brint and Willett in their comparative evaluation of geometric searching algorithms that can be used with the small 3D structures in the CCDB; in addition to the Lesk algorithm, Brint and Willett tested three others, which they refer to as Ullman, clique detection and set reduction.[8] The results of this comparison suggest that the algorithm originally described by Ullman[22] is by far the most efficient for geometric searching applications.

Unlike the Lesk algorithm, the Ullman algorithm does not involved an explicit stage for generating large numbers of combinations of atoms. Instead, it operates by means of a backtracking tree search in which database atoms are tentatively assigned to query atoms and the match extended in a depth-first manner until a complete match is obtained or until a mismatch is detected; in this case, the search then backtracks to the previous assignment. Thus, combinations are generated one atom at a time and the generation terminated if a match is not possible (whereas the Lesk algorithm explicitly generates all possible combinations consistent with the bit string assignments). Backtracking search is a common technique for improving the efficiency of graph matching operations; Ullman's contribution was to note the drastic increases in efficiency that can be obtained by the use of a *refinement procedure* that limits the number of levels of the search tree that have to be investigated before a mismatch is identified. Specifically, the algorithm makes use of the fact that if some query atom Q_x has another query atom Q_w at some specific distance and if some database atom S_z matches with Q_w, then there must be some atom S_y at the appropriate distance from S_z which also matches with Q_x: this is a necessary, but not sufficient, condition for a subgraph isomorphism to be present. The refinement procedure is called before each possible assignment of a database atom to a query atom: the matched substructure is increased by one atom only if the condition holds for all atoms w, x, y, z. Ullman advocated this particular refinement technique in the context of general graphs, without any specific application context. The algorithm seems to work extremely well with 3D chemical structures since the graphs considered here are fully connected; each atom is related to all of the other atoms by the interatomic distance. Thus, there is a large amount of information available to the refinement procedure and mismatches are detected very rapidly; generally, graphs are not fully connected, but instead contain nodes that are connected to only some of the other nodes, so that the refinement procedure is less able to detect mismatches and much more backtracking needs to take place. A worked example of the use of the Ullman algorithm for geometric searching in small molecules is presented by Brint and Willett.[8]

Despite the great efficiency of the Ullman algorithm for geometric searching in databases of small molecules, there are very substantial storage overheads when it is used for searching macromolecules. Specifically, for a query pattern and a database structure containing NQ and NS atoms respectively, the algorithm takes as input a pair of adjacency matrices of size $NQ \times NQ$ and $NS \times NS$ elements, respectively; in addition, each node of

the backtracking tree search requires the generation and processing of a matrix of size $NQ \times NS$ elements. In the case of protein structures, NS is large and thus the protein adjacency matrix is extremely demanding of storage. The clique detection algorithm described by Brint and Willett is also not suitable for use with protein structures since the number of cliques generated from a graph rises drastically with the size of the graph that is being processed, a fact that was apparent even with the small structures used in their evaluation.[8] Thus, only the remaining two algorithms, the set reduction procedure of Jakes *et al.*[18] and the Lesk algorithm, appear to be suitable for geometric searching in macromolecules. Brint presents results of searches which suggest that the Lesk algorithm is the more efficient of these two; even so, searches of structures from the PDB were found to require run times of hundreds of CPU seconds on a Prime 9950 when large patterns or large structures need to be processed.[11] Such a time requirement is unlikely to be acceptable if there is a need to search for a query pattern in the entire PDB (rather than in a single protein structure, as in Brint's experiments).

The discussion in the last paragraph might suggest that geometric searching in proteins cannot be carried out very efficiently. However, a practicable strategy does appear possible by combining the Lesk algorithm with the Ullman algorithm to give a two-stage matching procedure. Specifically, the bit string matching component of the Lesk algorithm can be used to identify possible query atom-to-database atom equivalences. Then, rather than using these equivalences as the input to the combinatorial generation procedure (Steps 7 and 8 in the algorithm presented previously), they should instead act as the input to the Ullman algorithm. Thus, if the Lesk procedure has been successful in reducing the number of possible equivalences, only some (and we hope very few) of the NS database atoms need be considered in the construction of the matrices required by the Ullman algorithm, which can accordingly be used even with very large macromolecules.[13]

Having described the proposed, two-stage geometric searching algorithm, we now summarize its complexity. The number of distinct interatomic distances in the query pattern will not be greater than $NQ(NQ-1)/2$ (and can be less if some of the distances are equal) and thus the input matrices required for the initialization stage of the Lesk algorithm are of size $O(NQ^3)$ and $O(NQ^2 \times NS)$ elements. Since NS is likely to be very much greater than NQ, the query matrix can be ignored so that the storage requirement of the Lesk algorithm is of order $O(NQ^2 \times NS)$. For comparison, the storage requirements of the Ullman algorithm are dominated by the need for random access to the interatomic distance matrix, which contains order $O(NS^2)$ elements in the absence of any initial step that can eliminate some of the database atoms from consideration. If a fraction, $f(0 \leqslant f \leqslant 1)$, of the database atoms remains after the Lesk bit string matching, the storage requirement is of order $O(f^2 \times NS^2)$ and thus drops very rapidly with an increase in the number of database atoms eliminated in the first stage. There are also, of course, concomitant reductions in the size of the intermediate matrices, processed during the depth-first search. In addition to reducing the memory requirements of the Ullman algorithm, the bit string matching stage can also be expected to decrease the CPU requirements, owing to the very much smaller number of database atoms that need to be considered for matching against each of the NQ query atoms during the backtracking stage.

Our actual implementation of this two-stage geometric searching algorithm used Steps 1–5 of the Lesk algorithm as described above, with the exception that Steps 3 and 4 were not repeated (i.e., only a single iteration of these steps was used) since it was found that the bulk of the mismatches were detected during the first iteration of the algorithm.[11] Those database and query atoms that have been identified as possible matches are then passed on for further processing by the Ullman algorithm.

EXPERIMENTAL RESULTS AND DISCUSSION

An investigation of the efficiency of the proposed two-stage algorithm requires the availability of a large file of variegated queries that can be searched for in the PDB. However, it is not clear where such a file might be obtained, and we have accordingly elected to use artificial patterns, as in our previous study of geometric searching algorithms.[8] Specifically, patterns containing 4,7 and 10 atoms were created using a Numerical Algorithms Group (NAG) Fortran 77 random number generator, with the generator primed to produce interatomic distances in the range 5–20Å. The resulting patterns were then searched for in each of the following six structures from the PDB:

- 1MBD (Myoglobin (deoxy, at pH 8.4) from whale resolved at 1.4Å)[19]
- 2ACT (Actinidin from kiwi fruit resolved at 2.0Å)[3]
- 2EBX (Erabutoxin B from sea snake resolved at 1.4Å)[14]
- 3CNA (Concanavalin A from jack bean resolved at 2.4Å)[14]
- 2GN5 (Gene 5/DNA binding protein from filamentous bacteriophage resolved at 2.3Å)[7]
- 3DFR (Dehydrofolate reductase from Lactobacillus casei resolved at 1.7Å)[5]

The geometric searching algorithm was encoded in Fortran 77 and run on the University of Sheffield IBM 3083 BX under VM/CMS, with calls to the system clock being included to allow the time requirements of each stage of the algorithm to be identified. The algorithm was implemented so as to allow a database interatomic distance to be accepted as a match for a query interatomic distance if they matched to within 0.5, 1.5 or 2.5Å. Ten different patterns of each size — 4, 7 or 10 atoms — were searched for in each of the six proteins at each of the three distance tolerances, for a total of 180 geometric searches. The results of these runs are presented in Table 1. The figures quoted are the median times (in CPU seconds on the IBM 3083 BX), averaged over

Table 1. Median run times (in CPU seconds on an IBM 3083 RX) for geometric searching in proteins. The average is calculated over sets of 10 randomly generated patterns of size 4, 7 or 10 atoms and with a distance tolerance of 0.5, 1.5 or 2.5Å. The first and second figure in each element of the table corresponds to the time for the Lesk bit string matching and for the Ullman backtracking, respectively

		1MBD		2ACT		2EBX		2GN5		3CNA		3DFR	
	4	0.3	15.0	0.6	30.2	0.1	0.2	0.1	1.4	0.6	22.0	0.2	11.9
0.5	7	0.5	2.5	0.8	5.2	0.1	0.1	0.2	0.6	0.8	4.6	0.6	3.4
	10	0.6	2.7	0.9	5.1	0.1	0.1	0.2	0.5	1.0	2.7	0.7	2.3
	4	0.4	22.1	0.5	30.2	0.1	0.2	0.1	2.1	0.6	33.1	0.4	21.
1.5	7	0.6	2.2	0.9	5.2	0.1	0.2	0.2	0.5	1.0	5.7	0.6	3.0
	10	0.8	1.7	1.4	5.1	0.2	0.1	0.4	0.5	1.3	3.7	1.0	2.7
	4	0.4	17.9	0.6	35.5	0.1	0.2	0.2	2.2	0.7	23.1	0.4	18.0
2.5	7	1.7	3.1	1.2	4.8	0.1	0.1	0.3	0.4	1.1	3.8	0.8	2.6
	10	1.2	2.7	1.8	4.3	0.2	0.1	0.5	0.4	1.9	4.5	1.3	2.9

the set of 10 patterns in each case, for the bit string matching stage of the Lesk algorithm (one iteration only as described previously) and the time taken by the Ullman algorithm to run to completion (i.e., until the presence of the query pattern was confirmed or until a query atom was identified for which there was no matching database atom). The listed times do not include the time required by the Lesk algorithm to process each query pattern to identify the set of distinct interatomic distances; this time was insignificant when compared with the requirements of the other two stages. More extended results are presented by Davies.[13]

Several trends are evident from the figures in Table 1. The Lesk component of the processing involves the generation of the bit strings that represent the interatomic distance in the query pattern, followed by the comparison of the bit strings representing atoms in the query pattern and in the database structure to determine possible atomic equivalences. The time requirement hence increases in line with NQ, the size of the query pattern, as can be seen from an inspection of the run times as NQ is increased from 4 to 7 to 10. In the case of the Ullman component, two factors are at work. An increase in the size of the query pattern results in an increase in the depth of the search tree, since there are more query atoms for which equivalences must be found. However, the refinement technique used by the algorithm involves consideration of the distances of all of the other $NQ-1$ atoms from each specified query atom; thus, an increase in the size of the query pattern also results in an increase in the effectiveness of the refinement (i.e., in rapid detection of mismatched atoms). This second factor tends to predominate so that the run times are generally longer for smaller queries. There is no obvious consistency in the effect of variations in the error tolerance on the run times for query patterns of a given size NS, the number of residues in a database structure, strongly affects the performance of both parts of the combined algorithm. Thus, all of the listed figures for the smallest protein, 2EBX with just 62 residues, are not more than 0.2 CPU seconds; these figures may be compared, e.g., with those for the largest structure, the 237-residue 3CNA. There is not, however, a completely monotonic relationship between run time and protein size since many of the largest times in Table 1 are for searches of the 218-residue 2ACT, which is only the second largest protein in the test set.

It will be clear from the results presented in Table 1 that the combined algorithm allows substantial increases in the efficiency of geometric searching in macromolecules when compared with the basic Lesk algorithm. Further increases in performance can be achieved if additional information is available as to the types of the individual Cα atoms since it is then possible to restrict the number of database atoms that need to be considered for matching with each of the query atoms in the two stages of the combined algorithm; indeed, many of the possible equivalences can be eliminated from further consideration even before the start of the Lesk bit string matching. Two types of information are readily available from the PDB files that can be used to characterize the Cα atoms — the residue type, which can be defined as either polar or nonpolar, and the type of secondary structure in which a residue occurs, which can be defined as helix, strand, turn random coil or not specified. In graph theoretic terms, the availability of such information means that the graphs used to represent the query patterns and the database structures not only have labeled edges (i.e., the interatomic distances) but also labeled nodes (i.e., the atom types).

A second set of 180 searches was carried out in which the polar and secondary structure types of the Cα atoms were specified. These searches were implemented by assigning database atoms a pair of descriptors, with the values 1–2 (for polar or nonpolar) and 1–5 (for helix, strand, turn, random coil or unspecified). This information was extracted from the PDB records. The atoms in the query patterns had analogous descriptors assigned by a NAG routine, and the secondary structure descriptors were checked to ensure that only valid ones were generated for the protein that was to be searched (e.g., if the PDB record for some structure contained no helices, then none of the secondary structure descriptors for any of the query atoms were given the value of 1). The results of these runs are listed in Table 2.

A comparison of the figures in Tables 1 and 2 demonstrates the gains in efficiency that can be obtained in both the Lesk and Ullman parts of the combined algorithm if atomic type information is available. The most obvious example of this reduction in computational requirements is with 3DFR. Here, 92% of the residues have been assigned secondary structure classes by the depositors and a considerable reduction in the number of matches is thus obtained prior to graph matching; on average, only 35% of the residues needed to be processed by the Lesk algorithm. In general, as would be expected, the reduction in computation increases in line with the amount of secondary structure information available. There is no such information in the PDB for 2GN5; only the polar/nonpolar information can

Table 2. Median run times as for Table 1 but with the specification of residue type information

		1MBD		2ACT		2EBX		2GN5		3CNA		3DFR	
	4	0.1	8.6	0.3	0.1	0.0	0.2	0.1	0.4	0.2	4.6	0.0	0.1
0.5	7	0.4	0.8	0.1	0.1	0.1	0.1	0.1	0.3	0.2	0.6	0.2	0.6
	10	0.2	1.2	0.5	3.3	0.1	0.2	0.2	0.3	1.0	0.6	0.1	0.1
	4	0.3	14.4	0.1	0.1	0.0	0.1	0.1	0.4	0.2	0.7	0.0	0.1
1.5	7	0.2	0.6	0.1	0.1	0.1	0.1	0.2	0.3	0.6	2.9	0.0	0.0
	10	0.2	0.3	0.8	3.3	0.1	0.1	0.3	0.3	0.5	0.6	0.4	0.8
	4	0.1	0.4	0.3	28.7	0.0	0.0	0.1	0.4	0.1	0.7	0.0	0.1
2.5	7	0.6	1.4	0.1	0.1	0.1	0.1	0.2	0.4	0.4	0.9	0.0	0.0
	10	0.8	1.0	1.0	1.7	0.2	0.1	0.4	0.4	1.3	2.5	0.1	0.0

thus be used, and there are only very slight increases in speed when compared with the corresponding results in Table 1.

CONCLUSIONS

This paper has studied the use of interatomic distance information for the automatic identification of patterns in protein tertiary structures, specifically in the sets of coordinates representing the Cα atoms. Two algorithms for this purpose are considered, and it is shown that a combination of the two approaches is sufficiently fast in operation to allow geometric searching in the PDB on a routine basis. The computational requirements can be substantially reduced if residue and secondary structural types can be specified to eliminate some of the query atom-to-database atom equivalences that need to be considered by the graph matching algorithms.

There are several limitations in the work that should be emphasized. First, the figures quoted are median values, and this disguises the occasional pathological search requiring many tens of seconds of CPU time for the Ullman algorithm to run to completion. Second, the figures relate to randomly generated patterns, only a very few of which actually occurred in any of the patterns that were being searched. Finally, we have considered only the Cαs comprising the protein backbone. The inclusion of all the atoms, even if these were restricted to the non-hydrogen atoms, would obviously increase the scale of the processing. That said, the inclusion of different atomic types would also substantially increase the information available at each stage of the algorithm.

There is clear scope for further work to investigate these points more fully. However, we feel that the results presented here suggest that geometric searching in proteins can be carried out at an acceptable computational cost.

ACKNOWLEDGMENTS

We thank the British Library Research and Development Department, the Department of Education and Science and the Science and Engineering Research Council for funding. We also thank Peter Artymiuk, Geoffrey Ford and David Rice from the Department of Biochemistry, University of Sheffield, for their helpful advice on various aspects of this work and an anonymous referee for comments on an earlier draft of this manuscript.

REFERENCES

1 Abola, E.E., *et al.* The Protein Data Bank, in Glaeser, P.S. (ed.), *The Role of Data in Scientific Progress.* Elsevier Science Publishers B.V., New York, 1985

2 Abola, E.E., *et al.* Protein Data Bank, in Allen, F.H. *et al.* (eds.), *Crystallographic Databases: Information Content, Software Systems, Scientific Applications* Data Commission of the International Union of Crystallography, Cambridge, 1987

3 Baker, E.N., and Dodson, E.J. Crystallographic refinement of the structure of actinidin at 1.7 Angstroms resolution by fast Fourier least-squares methods. *Acta Crystallographica, Section A*, **36**, 1980, 559–572

4 Bernstein, F.C., *et al.* The Protein Data Bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.* 1977, **122**, 535–542

5 Bolin, J.T., *et al.* Crystal structure of Escherichia coli and Lactobacillus casei dihydrofolate reductase refined at 1.7 Angstroms resolution. I. General features and binding of methotrexate. *J. Biol. Chem.* 1982, **257**, 3650–3662

6 Bourne, P.E., *et al.* Erabutoxin-B-initial protein refinement and sequence analysis at 0.140-NM resolution. *Eur. J. Biochem.* 1985, **153**, 521–527

7 Brayer, G.D., and McPherson, A. Refined structure of the gene 5/DNA binding protein from Bacteriophage FD. *J. Mol. Biol.* 1983, **169**, 565–596

8 Brint, A.T., and Willett, P. Pharmacophoric pattern matching in files of 3D chemical structures: comparison of geometric searching algorithms. *J. Mol. Graphics* 1987, **5**, 49–56

9 Brint, A.T., and Willett, P. Algorithms for the identification of three-dimensional maximal common substructures. *J. Chem. Info. and Comp. Sci.* 1987, **27**, 152–158

10 Brint, A.T., and Willett, P. Identifying 3D maximal common substructures using transputer networks. *J. Mol. Graphics* 1987, **5**, 200–207

11 Brint, A.T. *Matching Algorithms for Handling Three-Dimensional Molecular Coordinate Data*, PhD thesis, University of Sheffield, 1987

12 Crandell, C.W., and Smith, D.H. Computer-assisted examination of compounds for common three-dimensional substructures. *J. Chem. Info. and Comp. Sci.* 1983, **23**, 186–197

13 Davies, H. *An Evaluation of Two Algorithms for Three-Dimensional Substructure Searching in Files of Protein Structures from the Brookhaven Protein Data Bank*, MSc dissertaion, University of Sheffield, 1987

14 Hardman, K.D., and Ainsworth, C.F. *Private communication*, 1976

15 Gund, P. Three-dimensional pharmacophoric pat-

tern searching. *Progress in Molecular and Sub-cellular Biology* 1977, **5**, 117–143
16 Lesk, A.M. Detection of 3D patterns of atoms in chemical structures. *Comm. ACM* 1979, **22**, 219–224
17 Jakes, S.E., and Willett, P. Pharmacophoric pattern matching in files of 3D chemical structures: selection of interatomic distance screens. *J. Mol. Graphics* 1986, **4**, 12–20
18 Jakes, S.E., *et al.* Pharmacophoric pattern matching in files of 3D chemical structures: evaluation of search performance. *J. Mol. Graphics* 1987, **5**, 41–48
19 Phillips, S.E.V. *Private communication*, 1981
20 Read, R.C., and Corneil, D.G. The graph isomorphism disease. *J. Graph Theory* 1977, **1**, 339–363
21 Tarjan, R.E. Graph algorithms in chemical computation. *ACS Symposium Series* 1977, **46**, 1–20
22 Ullman, J.R. An algorithm for subgraph isomorphism. *J. ACM* 1976, **16**, 31–42