Journal of Computer-Aided Molecular Design, 9 (1995) 351–358
ESCOM

J-CAMD 298

The atom assignment problem in automated de novo drug design. 2. A method for molecular graph and fragment perception

M.T. Barakat and P.M. Dean*

Drug Design Group, Department of Pharmacology, University of Cambridge, Tennis Court Road, Cambridge CB2 1QJ, U.K.

Received 1 February 1995
Accepted 23 May 1995

Keywords: De novo drug design; Molecular electrostatic potential; Molecular graphs; Molecular similarity

Summary

If atom assignment onto 3D molecular graphs is to be optimized, an efficient scheme for placement must be developed. The strategy adopted in this paper is to analyze the molecular graphs in terms of cyclical and non-cyclical nodes; the latter are further divided into terminal and non-terminal nodes. Molecular fragments, from a fragments database, are described in a similar way. A canonical numbering scheme for the fragments and the local subgraph of the molecular graph enables fragments to be placed efficiently onto the molecular graph. Further optimization is achieved by placing similar fragments into bins using a hashing scheme based on the canonical numbering. The graph perception algorithm is illustrated in detail.

Introduction

A general strategy for reducing the combinatorial problems of atom assignment to 3D molecular graphs, placed within a drug-binding site, has been presented in the preceding paper [1]. Molecular structures may be partitioned into small molecular fragments. In many cases, the properties of the fragment may be transferred to a larger evolving molecule in a manner acceptable for optimization of the atom assignment problem.

The purpose of assignment by fragments is to reduce the need for constant recalculation of properties as the assignment progresses. Single-atom assignment, as opposed to fragment placement, would necessitate a large amount of computation. Since atom placement is a combinatorial problem (the size of the problem is often of the order of 10^{25} possible placements for a molecule with 40–50 atoms), a minimization in the amount of computation is essential to make the problem tractable. Fragments strongly reduce, but do not eliminate, the combinatorial problem.

Thus, given a molecular graph, a method is required for perceiving its features as a collection of fragments and expressing them in a way which will facilitate the atom assignment using fragment-by-fragment placement. This paper outlines the procedures used; the strategy is based primarily on the connectivity pattern of the graph. Four main steps are considered: (1) molecular graphs are characterized and perceived in terms of nodes and rings; (2) the fragment library is characterized in a similar way, wild cards are added to the library for special cases and the possibility of fragment rotation in the placement procedure is incorporated into the library; (3) a canonicalized numbering scheme is presented for all types of node; and (4) an efficient searching procedure for the fragment library is achieved by a hashing scheme for all nodes. The procedures are illustrated briefly in an example of a molecular graph extracted from the Cambridge Structural Database (CSD) [2]. The following paper reports a method for optimizing the atom assignment onto a molecular graph according to a property mapped onto an approximate surface envelope of the graph [3].

Theory and Methods

Characterization of molecular graphs

The molecular graph is made up of vertices and edges, as illustrated in Fig. 1. Vertices can be subdivided into nodes (three-, four- and five-atom non-cyclical nodes) and termini. This distinction is important in fragment place-

*To whom correspondence should be addressed.

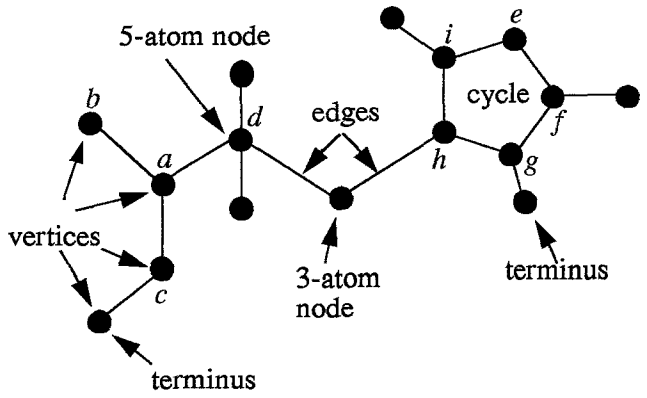


Fig. 1. A schematic representation of a molecular graph. The node indicated with *a* has connecting atoms *b*, *c*, *d*; the cyclical node has atoms labelled *e*–*i*.

ment, since there is no need to place fragments at termini; fragment placement should occur only at nodes. Similarly, edges, which represent bonds, can be subdivided into chains and cycles. For the purpose of fragment placement, the presence of a planar cycle indicates that an aromatic ring fragment should be placed onto that cycle, and it is pointless trying to place an aromatic ring at a node which is not a cycle. Therefore, in the perception of the graph, the presence and characteristics of non-cyclical and cyclical nodes must be discerned.

Perception of nodes

A node can be considered as a centre where a molecular fragment could be placed. Both non-cyclical and cyclical nodes occur, corresponding to the aliphatic and aromatic fragments in the fragment library. In the fragment placement algorithm, the perception of nodes is made at the two-dimensional level, i.e., in terms of connectivity. Thus, all three-, four- and five-atom non-cyclical nodes are noted, as well as any five-, six-, nine- and 10-edged cyclical nodes. Each node must correspond to at least one fragment from the library. If an unusual connectivity occurs (for instance, a non-cyclical node with a connectivity of 5), the program exits with an error message about that node. However, if there is no fragment which can fit a node, and the atoms of that node can be assigned from the overlap of fragments at adjacent nodes, then there is no need to exit, and execution continues.

In addition to the connectivity of nodes, certain nodes carry extra information, which is outlined below.

(1) Fixed atom type at a certain vertex. This is an option in the program which is particularly useful when an existing lead is present and only slight modifications are required to the atoms on the graph.

(2) Hydrogen-bonding information. A drug designer may require placement of a hydrogen-bond acceptor (or donor) at a certain node.

(3) Three-atom non-cyclical nodes have a central bond angle (see Fig. 2). Depending on the $sp/sp^2/sp^3$ cutoff criteria (160°, 122°, 114°), the three-atom node could be treated as an sp, sp^2 or sp^3 node. If the bond angle lies between 114° and 122°, then the node adopts both an sp^2 and an sp^3 status, since the resolution of the graph is not assumed to be perfect.

The bond angle for the three-atom node is important, since it is pointless to attempt placement of an sp fragment (e.g. a cyano group) onto a three-atom node which has a bond angle of 112°. The geometry for the four- and five-atom non-cyclical nodes and the cyclical nodes is treated more flexibly, and there are no geometrical constraints for these nodes when it comes to choosing a fragment.

Perception of rings

There are many ring perception algorithms for use on chemical graphs. They can be classified according to the initial approach used to find the required set of rings for each structure. The initial approaches are itemized below [4]:

(1) find all possible cycles and then select those required;

(2) find all possible simple cycles and then select those required;

(3) generate a fundamental basis of cycles from which all other cycles can be derived as necessary;

(4) determine directly the smallest fundamental basis, known as the smallest set of smallest rings (SSSR).

The algorithm chosen in the graph perception part of the fragment placement program follows the scheme outlined by Wipke and Dyott [5], and falls into the third category. It employs the Welch-assembly-Gibbs algorithm, which firstly finds the fundamental basis set of rings by Welch's stage 1 algorithm [6]; secondly, ring assemblies are explored by grouping those fundamental cycles which have edges in common; and thirdly, all rings are found by the Gibb's algorithm [7]. The perception

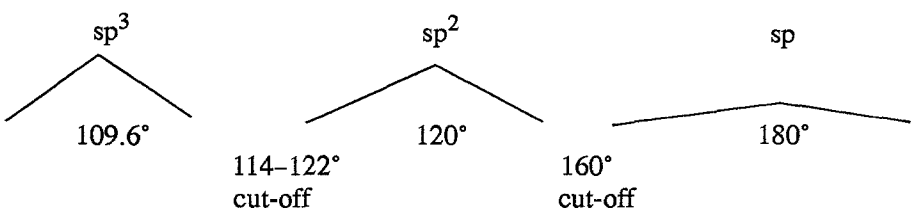


Fig. 2. Diagram illustrating the bond angle cutoffs for different hybridization states for various three-atom nodes.

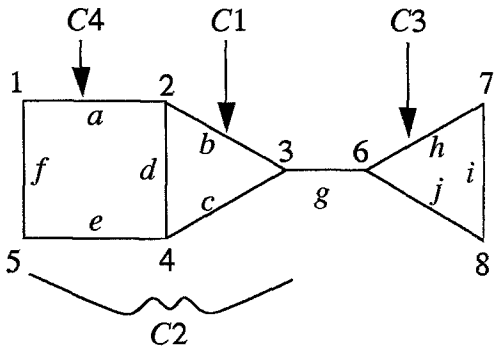


Fig. 3. A sample structure for the ring-finding algorithm, containing several cycles. Integers denote vertex numbers, lower-case letters represent edges, and cycle numbers are prefixed with *C*.

method is derived from graph theory and uses vertex-by-edge incidence matrices. The advantage of the Wipke and Dyott approach is that they use binary sets and supersets, enabling rapid and simple bit-vector matching to test for intersection. All rings are found by taking the ring sum of all combinations of the associated cycle vectors.

The following is an illustration of Welch's algorithm for finding the fundamental basis set of rings. Consider the structure given in Fig. 3. The vertex-by-edge ($V \times E$) incidence matrix for the structure in Fig. 3 is generated (Table 1). Then for each column, n, choose preferably an unused row, m, with a 1 in the nth column. Label the top of the nth column with the integer m and then replace any other row having a 1 in the nth column by its sum, modulo 2, with the mth row. The resulting label matrix is shown in Table 2.

The label matrix contains one null vector (row 4). Welch [6] showed that this is always the case. To obtain the fundamental cycles, each unlabelled edge, n, is combined with edges labelled by integers k, where $a_{k,n}=1$. This gives three fundamental cycles: (*d,b,c*), (*f,a,b,c,e*), and (*j,h,i*), which have the basis cycle matrix shown in Table 3.

The ring sum operation $C1+C2$ gives another basis cycle (*a,d,e,f*), while the other two ring sums produce

TABLE 1
VERTEX-BY-EDGE ($V \times E$) INCIDENCE MATRIX FOR THE STRUCTURE GIVEN IN FIG. 3

Vertex	Edge									
	a	b	c	d	e	f	g	h	i	j
1	1	0	0	0	0	1	0	0	0	0
2	1	1	0	1	0	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0
4	0	0	1	1	1	0	0	0	0	0
5	0	0	0	0	1	1	0	0	0	0
6	0	0	0	0	0	0	1	1	0	1
7	0	0	0	0	0	0	0	1	1	0
8	0	0	0	0	0	0	0	0	1	1

TABLE 2
VERTEX-BY-EDGE LABEL MATRIX FOR THE STRUCTURE GIVEN IN FIG. 3

Vertex	Label									
	1	2	3		5		6	7	8	
	a	b	c	d	e	f	g	h	i	j
1	1	0	0	0	0	1	0	0	0	0
2	0	1	0	1	0	1	0	0	0	0
3	0	0	1	1	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	1
8	0	0	0	0	0	0	0	0	1	1

unconnected unions of cycles. All combinations of basis cycle vectors are generated. Welch's omission of disjoint cycle unions is corrected by Gibbs algorithm at this stage, since they were required in some cases for the perception of *all* rings.

After finding the basis set of rings, the efficiency of the Welch algorithm can be improved by introducing a stage where ring assemblies are detected [5]. A ring assembly is a collection of rings which have bonds in common. Use is made of rapid and simple bit-vector matching to test for intersection. Finally, the Gibb's algorithm is applied to each ring assembly. This stage progressively takes the ring sums of pairs of cycles. The two smallest of each triplet set are kept. This is repeated with all possible pairings, until no further alterations are made to the set. A reduced set of basis cycles is thus isolated.

In the fragment placement program, all rings are found by this ring perception algorithm, but only five-, six-, nine- and 10-edged rings can be overlaid by aromatic fragments from the fragment library. Even within these rings, if the connectivity does not match any of the aromatic rings in the database, the cyclical node is broken down into its constituent aliphatic nodes, and atom assignment results from an aliphatic fragment placement.

Characterization of fragments in the library

The fragments in the fragment library must be defined by the same conventions as the molecular graph. In this way, both the fragment search procedure and the transfer of the fragment properties to the node of the molecular graph will be facilitated. A further complication with fragment placement onto molecular graphs is that fragments must be able to rotate at nodes. All possible placements of a fragment at a node must be accounted for.

Fragment perception

As with the molecular graph, each three-, four- and five-atom aliphatic fragment can be treated as a non-cyclical node, with a central atom and two, three or four

TABLE 3
BASIS CYCLE MATRIX FOR THE STRUCTURE GIVEN IN FIG. 3

Rings	Edge									
	a	b	c	d	e	f	g	h	i	j
C1	0	1	1	1	0	0	0	0	0	0
C2	1	1	1	0	1	1	0	0	0	0
C3	0	0	0	0	0	0	0	1	1	1
C1+C2	1	0	0	1	1	1	0	0	0	0
C1+C3	0	1	1	1	0	0	0	1	1	1
C2+C3	1	1	1	0	1	1	0	1	1	1

outer atoms. The aromatic fragments will each form cyclical nodes, where the ordering of the atoms is sequential; in the case of the fused 9- and 10-edged rings, the first atom in the sequence is one of the atoms at the intersection. The properties of each fragment (atomic residual charge, hydrogen-bonding, atomic hydrophobic parameters, intrafragment bond order and bond angles) are read into the program.

Wild cards for aliphatic nitrogens and carbons

The valency of the central atom of all aliphatic fragments is complete; that of the outer atoms depends on the fragment. In cases where the outer atom of the fragment is nitrogen or carbon, the fragment could be placed on more than one type of graph node: an outer nitrogen atom could be placed at a vertex which has a connectivity of 2 or 3, while an outer carbon atom could be placed at a vertex which has a connectivity of 2, 3 or 4. This is shown in Fig. 4. To overcome the many connectivities which outer-atom nitrogen and carbon atoms can adopt, wild cards are introduced for these atoms, so that their connectivity will adopt that of the node at which they are being placed.

Identification of fragment rotations

The molecular graph is fixed; the fragments are allowed to rotate on the graph. When only fragment connectivities are considered, a particular fragment may be placed in more than one way onto the molecular graph. In the example illustrated in Fig. 4, fragment m040 could be placed onto the molecular graph as drawn, or with the O and H swapped round. So, throughout the fragment library, where repeat patterns of connectivity exist (excluding those rotations in which the atom types and bond orders of the atoms to be swapped are identical), all possible rotations of the fragments are calculated (including stereoisomers for the five-atom aliphatic fragments). Each rotation gives the identical connectivity pattern of the original orientation of the fragment. The largest number of rotations for each aliphatic fragment is *n!*, where *n* is the number of outer atoms in that fragment. So, for a four-atom fragment, the maximum number of rotations is 6, while for a five-atom fragment this value is 24 (including stereoisomers). For the aromatic fragments, the maximum number of rotations is 2 for the five-, six- and nine-edged rings, and 4 for the 10-edged duplex.

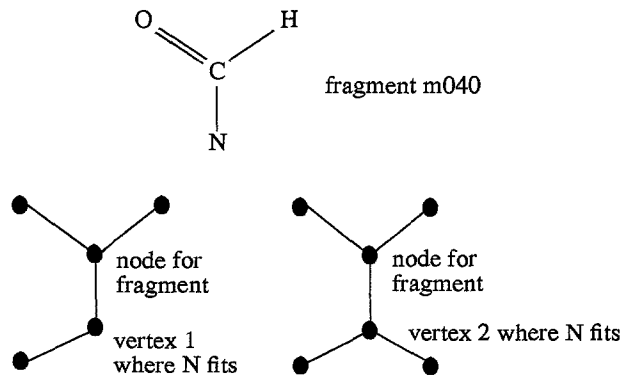


Fig. 4. Illustration of how a nitrogen in a fragment can be placed at more than one graph vertex.

Canonicalization

To improve the efficiency of the searching procedure, each graph node needs to be related in some unique way to the list of possible fragments which could overlay it. Underlying this is the standardization of the numbering of the outer atoms of each node and fragment. This standard form would have to be defined by encoded rules that determine a standard, or 'canonical' numbering of each node or fragment [8]. In this way, graph nodes would be numbered according to the same set of rules governing fragment numbering. Since atom type information is absent in the molecular graph, this cannot be considered in the set of rules. The rules must rely on connectivity.

Numbering of non-cyclical nodes and aliphatic fragments

Consider node a in the molecular graph shown in Fig. 1. It is a four-vertex node, having the connecting vertices *b, c, d*. The node 'ensemble' (*a,b,c,d*) has the following connectivity: 3, 1, 2, 4. The following rules can be applied to the numbering of all non-cyclical nodes:

(1) the central vertex becomes the first atom of each node;

(2) outer vertices are sorted into ascending connectivity order.

Using these rules, one can see that the numbering for node *a* is as shown: *a,b,c,d*.

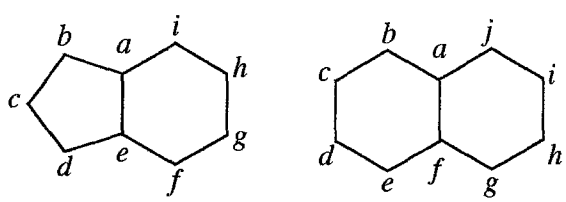


Fig. 5. Diagrammatic representation of core atoms from two duplex ring fragments.

Connectivity pattern (key) for node *a,b,c,d* is: 3124
Bin address = base 5 to base 10 conversion of outer atoms = 39
Fragments located in bin 39:

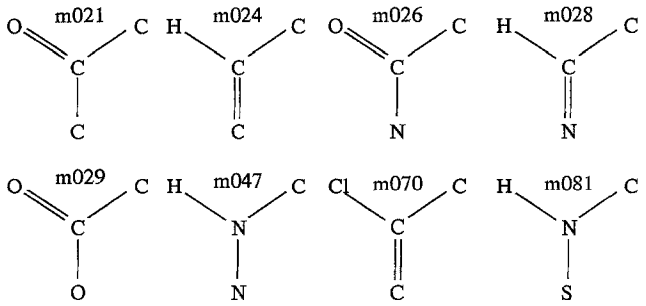


Fig. 6. Application of the aliphatic function, *f*, on an aliphatic key to gain the bin address of fragments shown in Fig. 1.

The same rules can be applied to the numbering of aliphatic fragments. In the case where there is a repeated connectivity for a fragment (e.g., connectivity 3,1,1,4), a rotation is allowed if the atom types and bond orders of the rotated fragment are not identical to those of the original orientation of the fragment (so, for the 3,1,1,4 connectivity pattern, the two possible rotations are: *a,b,c,d* and *a,c,b,d*). When a wild card is encountered (e.g. for an outer N or C atom), it is systematically replaced by the connectivities allowed (2 and 3 for N; 2, 3 and 4 for C), and the numbering rules given above are applied to each possibility.

Numbering of cyclical nodes and aromatic ring fragments

Since cycles are not composed of a central vertex with outer vertices, their canonicalization must utilize different rules.

Single cycles

Consider the cyclical node *e,f,g,h,i* shown in Fig. 1. The ordering of the vertices of cycles, in contrast to non-cyclical nodes, is a sequential numbering round the cycle.

The first vertex, *e*, in the cycle is preferably one which will result in a symmetry giving the same connectivity pattern, irrespective of whether the direction is clockwise or anticlockwise; the handling of the possible rotations of aromatic fragments is then only a matter of flipping the ring fragment about the first atom (possible fragment rotations are *e,f,g,h,i* and *e,i,h,g,f*). If no symmetry is present, then any vertex could be the first, but the reverse orientation for a ring fragment must be treated separately, since it will have a different connectivity pattern.

In the case where a certain numbering of the cycle in the graph does not correspond to the numbering of any aromatic fragment in the library, the first vertex of the graph cycle is changed to another vertex, and the database is searched again for an aromatic fragment with the same new numbering. If, after all numbering possibilities have been tried, no fragment is found, the cyclical node is ignored and is broken down into its constituent non-cyclical nodes.

Duplex cycles

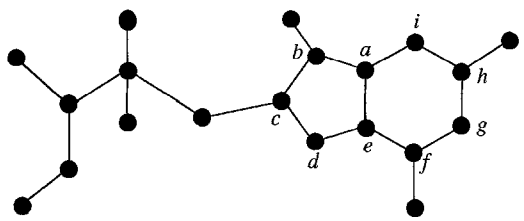
The numbering scheme for duplices is similar to that of single cycles, except that the first vertex must be one of the vertices at the intersection (see Fig. 5). In the case of nine-edged cycles, the ordering proceeds through the five-edged subcycle first. With fragment duplex rotations, the maximum possible for nine-edged ring fragments is two (*a,b,c,d,e,f,g,h,i* and *e,d,c,b,a,i,h,g,f*), whereas there is a maximum of four ways of numbering 10-edged ring fragments (*a,b,c,d,e,f,g,h,i,j*, *f,e,d,c,b,a,j,i,h,g*, *a,j,i,h,g,f,e,d,c,b* and *f,g,h,i,j,a,b,c,d,e*). Again, fragment rotations which give identical fragments in terms of atom type are ignored.

Fragment searching in the library

The fragment library can be considered as a dictionary, or 'symbol table'. The connectivity (or connectivities in the case of fragments with wild cards) of each fragment has been calculated. The fragments (records) which could fit the connectivity (key) of a particular node on the molecular graph need to be found. The advantage of using keys is that they provide a rapid prescreen to the records [9]. Each key could form a bin ('key address') in which all fragments having the connectivity pattern of the key are kept. Because of the use of wild cards, the system is degenerate, and the same fragment could exist in more than one bin. There are many methods for searching [10]; hashing was the technique chosen here for fragment placement. This procedure is more efficient than sequential or binary searching techniques.

Hashing involves the direct accessing of keys by per-

Connectivity pattern (key) for node *a,b,c,d,e,f,g,h,i* is: 0 1 1 0 0 1 0 1 0
Bin address = base 2 to base 10 conversion = 202



Fragment located in bin 202:

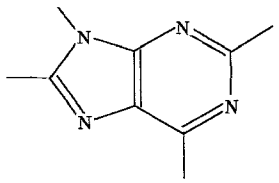


Fig. 7. Application of the aromatic function, *f*, on an aromatic key to gain the bin address of fragments.

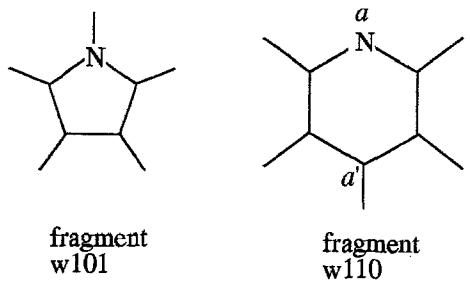


Fig. 8. Illustration of the numbering scheme of fragments w101 and w110. The first atom for fragment w110 actually chosen was atom *a'*.

forming an arithmetic transformation on the keys to obtain their symbol table addresses. An arithmetic function, *f*, is needed so that:

$$address = f(key) \quad (1)$$

With fragment placement, some function is required so that, for any given graph node key (i.e., pattern of connectivity), the bin address could be calculated where all the fragments which fit the node (in terms of connectivity) would be found.

Non-cyclical nodes

The canonicalized numbering of a particular node or fragment can be treated as a unique code (key) which describes the connectivity pattern. All non-cyclical keys are essentially strings containing integers 1, 2, 3 or 4. The function *f* was chosen to be the base 5 to base 10 conversion of the outer atoms of the string. Base 5 was chosen, since there was no fragment in the library with a connectivity greater than 4. The conversion to base 10 was used, since this could form the basis of the bin address where the fragments with a certain key would be located (see Fig. 6).

Cyclical nodes

The connectivity pattern for cyclical nodes and aromatic fragments can be broken down into a string of integers 0 and 1, i.e., a binary string. The pattern is the connectivity of the vertices involved from which a value of 2 has been subtracted. Furthermore, the vertices at an intersection are assigned a value of 0 in the code. The function, *f*, chosen for the cyclical nodes was the conversion of the whole binary string to a base 10 bin address (see Fig. 7).

Care was taken so that the first atom chosen of the aromatic fragments did not give a binary code equal to that from a fragment with different numbers of atoms, e.g. five-edged fragment w101 (11111; address = 31) and six-edged fragment w110 (011111; address = 31); to resolve this problem, the first atom for fragment w110 was changed so that the new connectivity pattern for w110 was (111011; address = 59), thus ensuring that only five-atom ring fragments were placed in the bin where w101 was located, and only six-atom ring fragments were placed in the bin of w110 (see Fig. 8). The basic steps in the algorithm for molecular graph and fragment perception are the following:

Read in fragment library
  apply canonicalized numbering scheme to the fragments
  find fragment rotations
  calculate connectivity keys of fragments
  get hash function (key address) of each fragment
  place each fragment in the bin whose number is the key address
Perceive all nodes in the target molecular graph
  perceive all non-cyclical and cyclical nodes
  apply canonicalized numbering scheme to the nodes
  calculate connectivity keys of nodes
  apply hash functions to get fragment bin addresses by conversion to base 10

Results

We give one example of how the algorithm handles the molecule and its associated molecular graph, illustrated in Fig. 9. This structure was taken from the CSD (reference

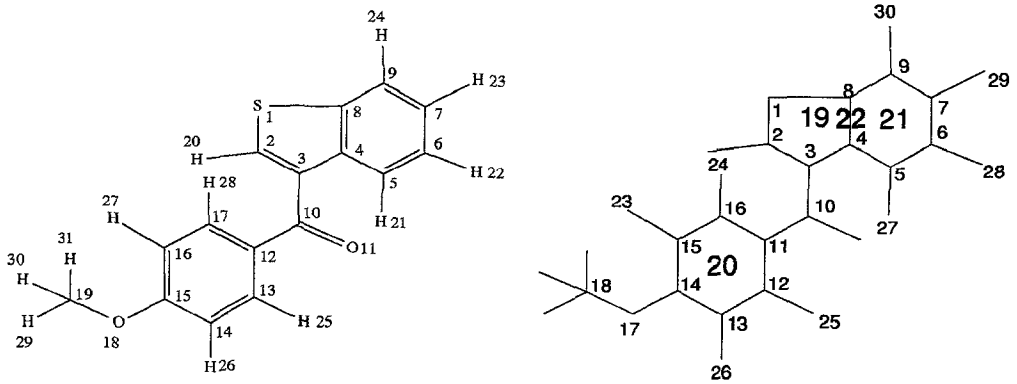


Fig. 9. The structure of CSD reference number 71678. Non-carbon atoms are labelled and numbered; the corresponding molecular graph with numbered nodes is drawn on the right. Nodes 19, 20 and 21 are rings; node 22 is a duplex of nodes 19 and 21.

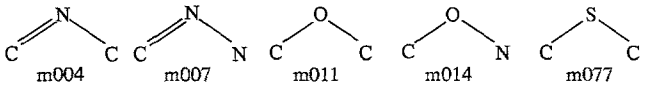


Fig. 10. Fragments stored at hash address 18. These can be placed at node 1 of Fig. 9.

code: JEDEUS; reference number: 71678). All atoms are numbered sequentially, with the hydrogen atoms numbered last.

The ring-searching algorithm finds three rings, and two of them form a duplex. Since the example is so similar to that given in Fig. 3, it will not be discussed further. Instead, we focus on fragment perception; Table 4 gives the fragment perception data. We show how fragment perception works by considering atoms 1 and 2 for brevity. The rest of the data is too voluminous to illustrate; problems with an incomplete database are also revealed. Atom 1 is connected to atoms 2 and 8 and therefore has a connectivity of 2; atoms 2 and 8 each have a connectivity of 3.

TABLE 4
FRAGMENT PERCEPTION DATA FOR ALL THE NODES OF STRUCTURE JEDEUS, SHOWN IN FIG. 9

Node	Central atom	Skeleton atom number	Hash function	Number of fragments at hash key	Number of rotations
1	1	2, 8	18	5	8
2	2	20, 1, 3	38	14	25
3	3	2, 4, 10	93	5	16
4	4	3, 5, 8	93	5	16
5	5	21, 4, 6	43	9	14
6	6	22, 5, 7	43	9	14
7	7	23, 6, 9	43	9	14
8	8	1, 4, 9	68	11	43
9	9	24, 7, 8	43	9	14
10	10	11, 3, 12	43	9	14
11	12	10, 13, 17	93	5	16
12	13	25, 12, 14	43	9	14
13	14	26, 13, 15	43	9	14
14	15	18, 14, 16	68	11	43
15	16	27, 15, 17	43	9	14
16	17	28, 12, 16	43	9	14
17	18	15, 19	19	4	6
18	19	29, 30, 31, 18	157	11	17
19	1	2, 3, 4, 8	15	3	5
20	16	17, 12, 13, 14, 15	63	1	1
21	4	5, 6, 7, 8, 9	63	1	1
22	8	1, 2, 3, 4, 5, 6, 7, 9	111	1	1
23	27	16	1	2	4
24	28	17	1	2	4
25	25	13	1	2	4
26	26	14	1	2	4
27	21	5	1	2	4
28	22	6	1	2	4
29	23	7	1	2	4
30	24	9	1	2	4

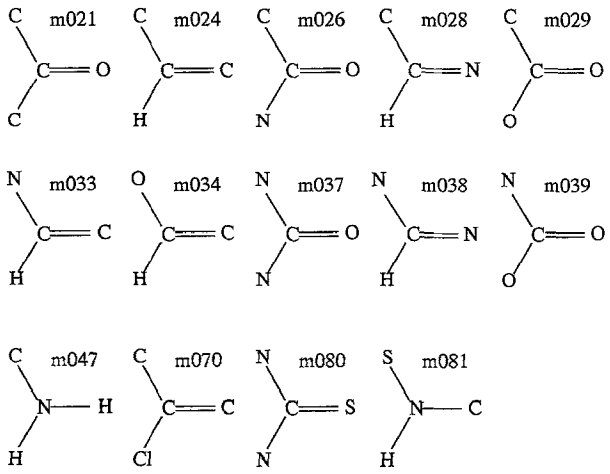


Fig. 11. Fragments stored at hash address 38. These can be placed at node 2 of Fig. 9.

Thus, for atom 1 the hash address is obtained from the base 5 to base 10 conversion (2)33 = 15 + 3 = 18. As illustrated in Fig. 10, five fragments have the hash address of 18; all have connectivities of 3 associated with both outer atoms of the fragment. The central atom of each fragment can be placed at node 1. Fragments m004, m007 and m014 can have two orientations if the central atom is placed at node 1. Fragments m011 and m077 have only one orientation. Thus, the total sum of orientational placements is 8. Note that fragment m077, when placed at node 1, corresponds exactly to the atoms 1, 2 and 8.

Atom 2, which is located at node 2 of the molecular graph, has a connectivity of 3; it is connected to atoms 1 (connectivity 2), 3 (connectivity 3) and 20 (connectivity 1). The hash address is given by conversion of the connectivities (3)123 = 25 + 10 + 3 = 38. As shown in Fig. 11, 14 fragments have the hash address of 38.

Fragments m021, m037 and m080 have only one rotation, whereas the others are capable of being placed in two ways. Thus, the total number of rotations of these fragments is 25. Atom 20, being a terminal atom, is part of the fragment centred on node 2 and does not need to be handled separately. Note that there is no fragment at hash address 38 that fits the atoms in the original molecule found at node 2.

There are three rings in the molecule. Node 19 is a five-membered ring; its hash address is computed by subtracting 2 from the connectivity at each node. Thus the connectivity key is obtained for vertices 1, 2, 3, 4 and 8 as 0 1 1 1 1; the bin address can be obtained from a base 2 to base 10 conversion 8 + 4 + 2 + 1 = 15 to give an address of 15. This node can be placed in five ways. Three fragments are found at this address (Fig. 12), and none contains a sulphur atom. Similarly, nodes 20 and 21 are six-membered rings; they are cyclical nodes with a connectivity pattern of 1 1 1 1 1 1, corresponding to a hash address of 63. There is only one fragment in this bin

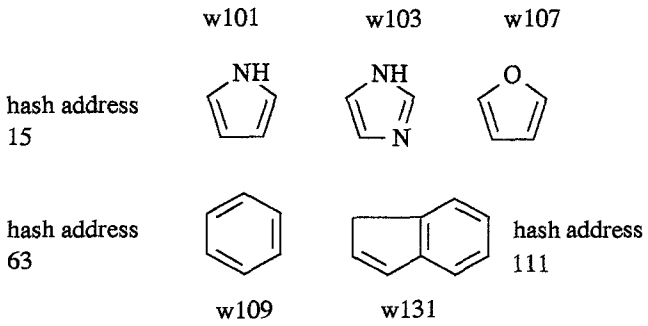


Fig. 12. Some ring fragments stored at different hash addresses.

and only one placement is possible (Fig. 12). Node 22 is a duplex composed of a five- and a six-membered ring. Its hash address is 111 and it has only one fragment, with one possible rotation (Fig. 12). This fragment does not contain the sulphur atom.

The size of the placement problem, N, that an algorithm would face in trying to perform an atom assignment at each node is given by Eq. 2:

$$N = \prod_{i=1}^{j} \sum_{f=1}^{g_i} r_f \qquad (2)$$

where the node i can be assigned a fragment f from a set containing g_i members and there are r_f orientations of the fragment f at that node. If all atoms were to be placed on JEDEUS by an aliphatic placement, there would be 18 nodes (the first 18 lines of Table 4) with 316 rotations, giving a placement problem of about 10^{21} possibilities. If the cyclical nodes can be treated by single placements, the number of nodes is reduced to 15, with a total of 77 rotations. In this case the problem size for atom placement is reduced to 4.7×10^8. However, there are no fragments in the library which could fit nodes 19 and 22 with the appropriate atoms. Where ring fragments do not contain the correct atoms, they can be made from an aliphatic placement. Thus, the actual size of the combinatorial problem lies between these limits.

We find that, by storing fragments with identical connectivity patterns in appropriate bins, it should be possible to retrieve them efficiently for placement onto a defined molecular graph.

Discussion and Conclusions

This paper has described how, by using graph theory, a molecular graph can be perceived in terms of non-cyclical and cyclical nodes. The same method has been applied to the fragments in the fragment library, which are treated as individual nodes. In fragment placement onto molecular graphs, the nodes of the graph need to be overlaid by fragments from the library which, primarily, have the same connectivity pattern. This is facilitated by canonicalization of the connectivity patterns of the nodes of the molecular graph and of the fragments (including their possible rotations). The canonicalization procedure entails sorting the outer vertices of non-cyclical nodes into ascending connectivity order, and arranging the vertices of cyclical nodes into consecutive connectivity order. These processed connectivity patterns can then be converted to unique codes, corresponding to the correct bin addresses of the fragments having the required connectivity patterns. The conversion is via a hash function which translates aliphatic connectivity patterns from base 5 to base 10, and aromatic connectivity patterns from base 2 to base 10.

The hashing procedure allows direct accessing of the correct fragments for the molecular graph in terms of connectivity, and should significantly reduce the time needed to search the fragment library. Fragment searching, however, is only one part of the fragment placement procedure. The total number of placements results in a combinatorial explosion, and powerful optimization techniques are required to tackle it. These techniques are discussed in the following paper [3].

Acknowledgements

We wish to thank the Wellcome Trust for a Wellcome Prize Studentship (M.T.B.) and for a Principal Research Fellowship (P.M.D.). Part of this work was carried out in the Cambridge Centre for Molecular Recognition.

References

1 Barakat, M.T. and Dean, P.M., J. Comput.-Aided Mol. Design, 9 (1995) 341.
2 Allen, F.H., Bellard, S., Brice, M.D., Cartwright, B.A., Doubleday, A., Higgs, H., Hummelink, T., Hummelink-Peters, B.G., Kennard, O., Motherwell, W.D.S., Rogers, J.R. and Watson, D.G., Acta Crystallogr., B35 (1979) 2331.
3 Barakat, M.T. and Dean, P.M., J. Comput.-Aided Mol. Design, 9 (1995) 359.
4 Downs, G.M., Gillet, V.J., Holliday, J.D. and Lynch, M.F., J. Chem. Inf. Comput. Sci., 29 (1989) 172.
5 Wipke, W.T. and Dyott, T.M., J. Chem. Inf. Comput. Sci., 15 (1975) 140.
6 Welch, J., J. Assoc. Comput. Machinery, 13 (1966) 205.
7 Gibbs, H., J. Assoc. Comput. Machinery, 16 (1969) 564.
8 Gray, N.A.B., Computer Assisted Structure Elucidation, Wiley, New York, NY, 1986.
9 Sheridan, R.P., Nilakantan, R., Rusinko III, A., Bauman, N., Haraki, K.S. and Venkatarghavan, R., J. Chem. Inf. Comput. Sci., 29 (1989) 255.
10 Sedgewick, R., Algorithms, Addison Wesley, Reading, MA, 1983.