Journal of Computer-Aided Molecular Design, 8 (1994) 51–66
ESCOM

J-CAMD 229

CAVEAT: A program to facilitate the design of organic molecules*

Georges Lauri and Paul A. Bartlett**

Department of Chemistry, University of California, Berkeley, CA 94720, U.S.A.

Received 15 September 1993
Accepted 1 October 1993

Key words: Drug design; Database searching; CAVEAT; Ligand design

SUMMARY

A frequently encountered problem in the design of enzyme inhibitors and other biologically active molecules is the identification of molecular frameworks to serve as templates or linking units that can position functional groups in specific relative orientations. The program CAVEAT was designed to address this problem by searching 3D databases for such molecular fragments. Key innovations introduced in CAVEAT are a focus on relationships between bonds and the provision of automated methods to identify and classify structural frameworks. Performance has been a particular concern in formulating CAVEAT, since it is intended to be used in an interactive manner. The focus in this report is the design and implementation of the principal algorithms and the performance achieved.

INTRODUCTION

The program CAVEAT was devised to facilitate the structure-based design of enzyme inhibitors and related biologically active molecules [1,2]. There are a number of problems that fall under the definition of 'structure-based design', and even more conceptual approaches to their solution. Frequently, either to devise a mimic of a known ligand or a de novo design for a molecule to complement a defined binding site, molecular frameworks are sought as templates or linking units to position functional groups in a specific orientation relative to one another. We recognized that subjective, hit-or-miss approaches to identifying these frameworks are a slow aspect of the design phase in such projects, hence we were stimulated to automate the process. Furthermore, in considering the variety of problems encountered in structure-based design, we identified a unifying theme to these searches, namely a focus on the *orientation of bonds* rather

*CAVEAT is available from the Office of Technology Licensing, University of California, Berkeley, CA, and from Molecular Simulations, Inc., Burlington, MA; further information is available from P.A. Bartlett.
**To whom correspondence should be addressed.

than the *location of atoms*; this concept became the foundation of the program CAVEAT. In the course of implementing these ideas, we found considerable redundancy in the structural information represented by the templates identified in a typical search, hence we developed algorithms to automate the *identification* of connecting fragments and their *classification* based on similarity. The bond-based organization of its databases and the methods used to identify and compare structural frameworks represent the key technical contributions from CAVEAT to the field of 3D database searching for drug design (recently reviewed in Ref. 3; see also Refs. 4–7).

Since CAVEAT was first described [1], we have made a number of refinements in the search algorithms, extended and adapted it to other design problems that can be expressed conveniently in terms of bond-vector relationships, and implemented the classification module. In this report, the design and implementation of the key components of CAVEAT are addressed specifically; applications are discussed elsewhere [8]. Many of the methods described are graph-theoretical; discussions of the underlying theory may be found in Refs. 9 and 10, and fundamental algorithms and associated data structures are discussed in Refs. 11–14.

CAVEAT PROGRAM ORGANIZATION

The organization of the programs embodied in CAVEAT reflects our concern with a number of issues. Foremost is the issue of speed. Since CAVEAT is to be used as an interactive 'idea generator', i.e., to assist the chemist in coming up with useful candidate structures for further refinement in the design process, it has to operate in real-time on the desktop workstations typically used for molecular modeling; even 30 minutes can be too long to wait for an answer in many cases. Second, CAVEAT has to be compatible with very large 3D databases, incorporating a variety of molecular types, from experimental structures (e.g., the Cambridge Structural Database [15]) to computed (e.g., CAST-3D [4,16] and proprietary databases of CONCORD-predicted [17] coordinates) or macromolecular structures (e.g., the Protein Data Bank [18,19]). And third, CAVEAT has to be adaptable to a variety of problems, not only those for which we originally developed the approach, but others in which more elaborate searches are required. These requirements dictated a modular approach, in which a very rapid search within a large database is conducted first, using a match to specific bond vectors as the sole criterion for retrieval, followed by more time-intensive evaluation and classification of relatively few hit structures against additional geometric or other criteria.

CAVEAT is organized around three separate, primary programs: VPREP, VSRCH and CLASS. VPREP processes a database of 3D structures (a 'source database') to determine the relative orientation of all the relevant pairwise combinations of bonds in the individual molecules, as selected by the user. From this information, VPREP creates a CAVEAT database, which is essentially an index to the source database. VSRCH searches this index for molecules that match a particular query; because the desired information has already been computed in generating the CAVEAT database, this search is inherently fast. CLASS further processes the resulting oriented set of matches, implementing the more sophisticated second tier evaluation and classification.

CAVEAT also contains a number of programs that perform more mundane tasks, such as storing and retrieving 3D structures in various file formats, superimposing retrieved structures to query targets, managing multiple-structure files, and writing CLASS command files from interactive input. All of these constituent programs interact by reading and writing structure files

in a variety of commonly used formats under the control of a set of scripts that spare the end user the complexities involved. While these routines are necessary to streamline the use of CAVEAT, the methods used are not novel and are not discussed in this report.

The search engine

The CAVEAT search engine is designed to retrieve from a database of 3D structures molecules with specific bonds that match a vector relationship specified in the query. For most design problems, the matching bonds in the hit structures will be replaced with a substituent or other part of the design target, hence the geometrical attributes of the bonds, rather than their chemical or electronic character, are the primary issue. The representation of specific bonds in the database structures as vectors, and the organization of a database around precomputed pairs of such vectors ('vector pairs') are the key features of the CAVEAT search process.

A CAVEAT database is organized by representing each molecule from the 3D source database by a set of pairwise combinations of specific bonds, each of which is treated as a unit vector (Fig. 1). Which subset of bonds and pairs of bonds in the molecules are selected in this manner can be varied, and depends on the kind of problem for which the CAVEAT database is intended, as described below. The location of each vector is defined by a base or **b** atom, and its orientation is determined by the bond between the base and a tip or **t** atom. This representation intentionally

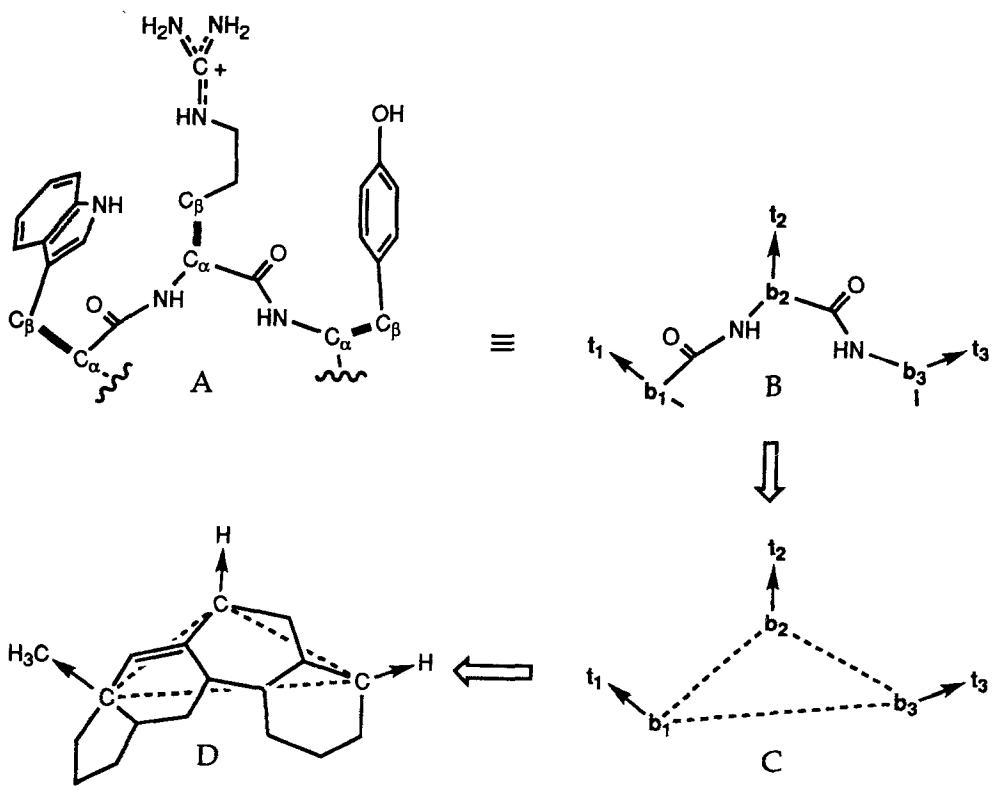

Fig. 1. To devise a peptide mimic, the C_α-C_β bonds of the peptide (A) are represented as vectors (B); the relationship between these vectors is determined by the three vector-pair combinations (C), which must also be present in a hit structure (D) if it is to serve as a template for the design.

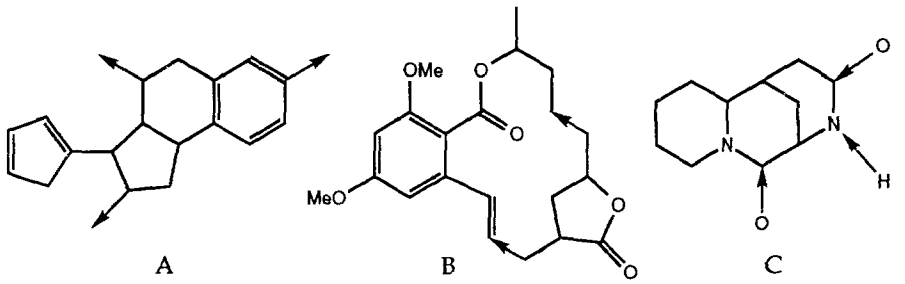

Fig. 2. Selected bond vectors, appropriate for different problem types: (A) a rigid template between vector base atoms; (B) a linking unit between vector tip atoms; (C) potential sites for placement of hydrogen-bond donor and acceptor moieties for de novo design.

ignores the length of the bond, since for the purposes of most CAVEAT searches, a hydrogen, a methyl, or an iodo substituent would all satisfy a search query. This organization lends itself well to rapid searching, because a set of ($\mathbf{b_1}$-$\mathbf{t_1}$, $\mathbf{b_2}$-$\mathbf{t_2}$) vector pairs forms an internal-coordinate representation of a molecule, and can embody arbitrary connectivity and rigidity criteria that need to be computed only at the time the CAVEAT database is created.

Queries that involve specific relationships among three or more vectors are defined as a combination of vector pairs. For three vectors, A, B and C, the three pairs A-B, B-C and A-C in almost all cases determine the relationship between the vectors uniquely. For larger sets, the problem becomes overdetermined if all vector pairs are chosen for the search criteria; under these circumstances, physically relevant triangulations of vectors serve to define the overall relationship adequately.

Choice of vector pairs

The type of problem for which the CAVEAT database is intended determines which subset of bonds in the source molecules is utilized by VPREP in computing the vector-pair information to be stored in the database, based on graph-theoretical criteria. As a consequence, a given source database may be processed into several different CAVEAT databases. For example, a database of vector pairs comprising substituents on cyclic or polycyclic molecules would be useful in identifying rigid or semirigid templates for peptide mimics or for de novo design (Fig. 2); in turn, bonds within macrocyclic rings or acyclic molecules yield a database useful for identifying chains of atoms that can serve as bridging or constraining units for stabilization of a specific peptide conformation. Flexibility also exists in the polarity chosen for the vector and in other criteria for selection; whereas in most CAVEAT databases the vectors are oriented such that the base atoms correspond to backbone atoms in the query, a database more appropriate for de novo design has this orientation reversed: vector pairs defined by substituents on sp^2-hybridized atoms in which the template atom is the tip and the substituent atom is the base can be used to identify molecules that may satisfy hydrogen-bonding constraints within an active site.

In a database of templates, the vectors represent possible positions on the source structures where substituents could be attached in a conformationally defined relationship, i.e., to a rigid system (Fig. 3). 'Ring systems' in a molecule are identified as inclusive sets of rings which have one or more atoms in common; they correspond to the edge-biconnected components of the mol-

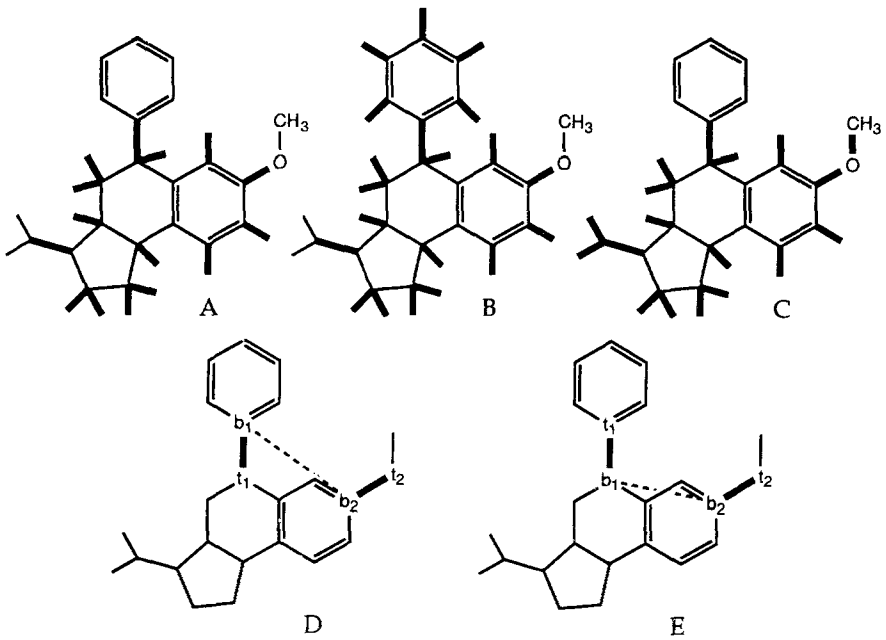

Fig. 3. Vectors for various definitions of 'rigid' systems: (A) simple ring system; (B) joined ring system; (C) one-vector-extended ring system. (D) Example of a vector pair that is disallowed because the base atoms do not span a molecular fragment; (E) an allowed vector pair.

ecular graph. A 'rigid system' can then be taken as a ring system itself, expanded to include ring systems joined by one or two bonds, or even extended to include atoms one bond removed. And, if conformational constraints are relaxed entirely, the entire heavy-atom framework of a molecule can be taken as the rigid system. Our definition of 'substituent' is a bond in which the base atom is part of the rigid system and the tip atom is not part of the same *ring* system as the base (however, the base and tip may be part of the same *rigid* system, if this is defined to include ring systems that have no atom in common). The definition of substituent in this context differs from that traditionally used in organic chemistry, since hydrogens are accepted as substituents as well. Pairs of vectors are stored in the database only if their bases are on the same rigid system and they span a molecular fragment; formally, this means that they must be connected in the molecular graph by a path not containing the tip atoms, which is determined from the traversal obtained during computation of the ring systems. For reverse databases, the definitions of the base and tip atoms are simply swapped.

A second fundamental type of database is constructed from vectors that represent endocyclic bonds of macrocyclic molecules, as illustrated in Fig. 4. In this case, to be included as a vector pair, the two vectors must be part of the same ring system, with the tip atoms separated by more than two bonds and the base atoms separated by more than one bond; furthermore, cleavage of the two bonds must fragment the parent structure, placing the tips in one fragment and the bases in the other. These vector pairs are defined formally as two-edge cuts of the biconnected components of the molecular graph; a precomputed characterization of the triconnected components allows this condition to be verified quickly for any candidate vector pair. In a further refinement

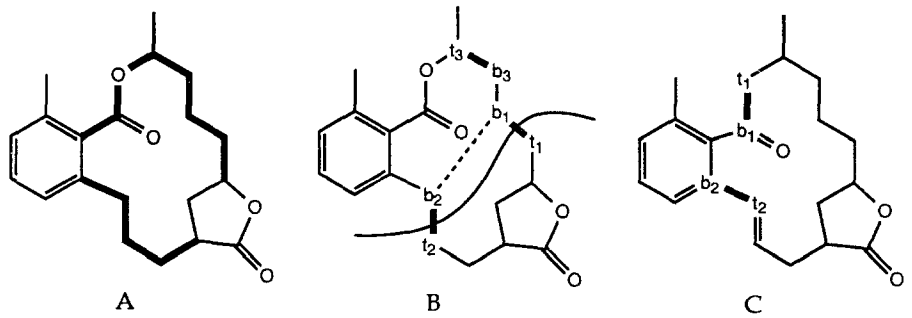

Fig. 4. Vectors from bonds that are endocyclic to a macrocyclic ring system: (A) the set of bonds from which vector pairs may be constructed; (B) examples of valid, ($\mathbf{b_1}$-$\mathbf{t_1}$, $\mathbf{b_2}$-$\mathbf{t_2}$), and invalid, ($\mathbf{b_1}$-$\mathbf{t_1}$, $\mathbf{b_3}$-$\mathbf{t_3}$) and ($\mathbf{b_2}$-$\mathbf{t_2}$, $\mathbf{b_3}$-$\mathbf{t_3}$), vector pairs constructed from these bonds; ($\mathbf{b_1}$-$\mathbf{t_1}$, $\mathbf{b_3}$-$\mathbf{t_3}$) is invalid because $\mathbf{b_1}$ and $\mathbf{b_3}$ are not far enough apart, and ($\mathbf{b_2}$-$\mathbf{t_2}$, $\mathbf{b_3}$-$\mathbf{t_3}$) is disallowed because there is no path connecting $\mathbf{t_2}$ and $\mathbf{t_3}$ that does not pass through $\mathbf{b_2}$ or $\mathbf{b_3}$; (C) example of a vector pair that is disallowed because the path ($\mathbf{t_1}$-$\mathbf{t_2}$) is long compared to distance ($\mathbf{b_1}$-$\mathbf{b_2}$).

of this approach, vector pairs may be excluded if the ratio of the shortest path length between the tip atoms to the distance between the base atoms exceeds a specified threshold; this constraint is made available so that fragments in which a short distance is spanned by an excessively long chain are excluded from consideration.

Representation of the database

The following terminology is used in defining the parameters of a vector pair ($\mathbf{b_1}$-$\mathbf{t_1}$, $\mathbf{b_2}$-$\mathbf{t_2}$): the distance **d** ($\mathbf{b_1}$-$\mathbf{b_2}$), the dihedral angle δ ($\mathbf{t_1}$-$\mathbf{b_1}$-$\mathbf{b_2}$-$\mathbf{t_2}$) and the angles α_1 ($\mathbf{t_1}$-$\mathbf{b_1}$-$\mathbf{b_2}$) and α_2 ($\mathbf{t_2}$-$\mathbf{b_2}$-$\mathbf{b_1}$).

After identification of the vector pairs for a given molecule and a given database definition, the parameters that characterize each vector pair must be stored in the database in such a way as to make fast retrieval possible. To this end, the continuous parameter space of the vector pairs is partitioned into discrete bins, and it is the specific bin location for a vector pair, rather than the exact value of a parameter, that is used in the CAVEAT database. Moreover, the database is inverted, associating a list of pair and molecule identifiers with each bin in the parameter space.

An efficient binning scheme requires that no replication exists (i.e., vector pairs that are equivalent for the purpose of searching and retrieval should be mapped to the same bin) and that each bin represents a roughly similar proportion of the parameter space. Furthermore, a search should identify all vector pairs that match the query under rotation or reflection, that is, molecules for which it is the enantiomer of the source structure that is an acceptable hit. For this reason, a vector pair is first given a canonical orientation so that $|\alpha_1 - \pi/2| \leq |\alpha_2 - \pi/2|$ and reflected so that $\delta \geq 0$; the pair is flagged in these operations so that both transformations may be undone if necessary. Next, the distance **d** is assigned to a discrete bin, based on a user-determined bin distribution, and α_1 is binned uniformly from 0 to π radians, again under user control. The remaining parameters α_2 and δ define the surface of a sphere (Fig. 5); α_2 is also binned uniformly from 0 to π radians, with the same number of partitions as α_1, and for each of these bins, δ is partitioned in a fashion that assigns a roughly uniform area to each (α_2, δ) bin on the

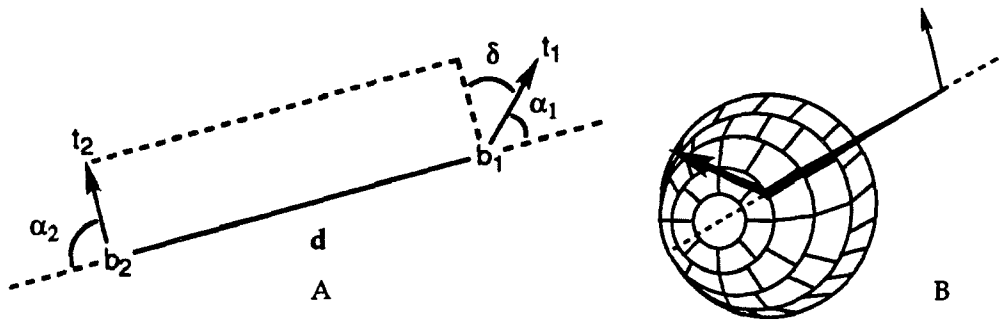

Fig. 5. Schematic representation of the binning scheme: (A) the four principle vector-pair parameters; (B) illustration of the dihedral binning sphere.

surface of the sphere. As a result, there are many more δ bins at the equator ($\alpha_2 \approx \pi/2$) than near the poles ($\alpha_2 \approx 0$ or π). Because of the condition $|\alpha_1 - \pi/2| \leq |\alpha_2 - \pi/2|$, this approach ensures that δ is sampled with a resolution appropriate to the degree to which it is well defined.

This method of parametrizing the vector pairs was chosen in preference to one based on interatomic distances, because of difficulties that arise in the latter scheme as a result of bonds of different length that represent the same vector, and as a result of the fact that angular relationships do not scale uniformly with distance. Although a distance-based vector definition can be implemented readily with conventional database structures, considerable complexity is introduced if these inherent problems are to be corrected for vector-pair searches [4].

As constructed initially, the CAVEAT database thus consists of a set of bins, one for each point in the discretized vector-pair parameter space; each bin contains a linked list of pair entries, ordered by molecule index (in the sequence of database construction). A pair entry indicates the molecule from which it was derived, the base and tip atoms for both vectors, and whether it was inverted or reflected. The database is stored on disk, with one file containing the pair entries and list pointers as they were written for each molecule processed, and one file containing the index of the first bin for each list.

A completed database is then optimized for searching by sorting the pair entries and eliminating the list pointers; pairs in the same bin are thus contiguous and in the original order by molecule index. The sorting process is implemented efficiently by an in-core sort of file sections small enough to fit in memory, followed by a merge of the sorted sections.

Database searching

A CAVEAT query is expressed as a set of target vectors and a pairing scheme for the search; a tolerance specification is assigned to each pair, giving the allowed error in the base–base distance, **d**, and the solid angle subtended by an error cone representing the angular tolerances. The search then proceeds in three steps: (1) identification of the bins that fall within the error tolerances for each query pair, and of all the vector pairs in these bins; (2) identification of those molecules that contain at least one matching vector pair for each query pair; and (3) for each of these molecules, identification of the ways in which all of the conditions in the query can be satisfied simultaneously.

In the first step, finding all of the bins containing vector pairs that match a query pair is

straightforward. The allowed tolerance in the distance parameter **d** defines neighboring bins that are also acceptable matches for the query pair; to accommodate tolerance in the angular parameters, neighboring bins in α_1 and on the (α_2, δ) binning sphere are included as necessary to approximate cones around the specified vectors. Under certain circumstances, a given parameter bin may match a query pair in more than one way; these bins are marked so that all the relevant rotations and reflections will be applied to pairs retrieved from them.

Having identified the relevant bins for each query pair, the search engine goes through all the pair lists for all bins simultaneously, in order of molecule index, until a molecule is found with a vector pair in each list. For these molecules, all the pair entries matching each query pair are retrieved in turn, and passed on to the final step. To make this process efficient in terms of both CPU and disk usage, the pair lists are organized into a heap data structure according to molecule index and a large number of entries for each list are cached.

In the final stage of the search, each molecule obtained from the preceding step is examined to determine if there exists a matching of molecule pairs to query pairs that satisfies all the connectivity constraints of the search query. This constraint is formalized as follows: a pair graph is defined in which the vertices represent the pairs and in which the edges connect pairs that have a vector in common; the edges are labeled to identify the position in each pair of the shared vector. Acceptable matchings are the subgraph isomorphisms between query and molecule pair graphs in which matched edges have identical labels and query pairs match only to molecule pairs from the corresponding pair lists. These isomorphisms are enumerated using a breadth-first search, in which each query pair and its associated pair list is evaluated in turn. The final output is a list of the molecules in the database that satisfy the query, along with the atom numbers of the base and tip atoms in the hit that correspond to the query vectors for each acceptable matching.

If a set of molecule pairs match the query in both their original and reflected forms, which enantiomer is the one that matches cannot be determined at this stage. Later, when the molecules are retrieved and overlapped to the query vectors, an excessive orientation error is observed for the incorrect enantiomer and it can be eliminated.

The screening and clustering engine

The CAVEAT screening and clustering engine CLASS is designed to reduce the amount of work that needs to be performed by the user in evaluating the results of a search. This task can be substantial, since search queries with physically reasonable tolerances on problems of practical interest can generate thousands of hits. The situation can be remedied to some extent by automating portions of the screening process so that certain kinds of frameworks can be eliminated, according to criteria specified by the user. Such query refinement can improve the quality of the retrieved structures markedly; however, any redundancy in the structural frameworks present in the database will still be reflected in the output. While it may be informative to examine all the representatives at a later time, their inclusion in the list presented to the user for initial evaluation is unnecessary. Hence, while CLASS can perform a variety of useful screening tasks, its main purpose is to identify the connecting framework of hit structures, classify them based on the structural motif they embody, and generate output in which each pattern is represented by only one structure.

CLASS thus receives as its input a set of oriented molecules and a list of the atoms that

define the matching vectors, and, following the instructions in a command file, applies the appropriate identification, screening and classification methods to its input structures.

The input to CLASS consists of a set of oriented molecules and a list of the atoms that define the matching vectors, and a command file with instructions as to the appropriate identification, screening and classification methods to be used. The output consists of lists of the resulting groups with all of their members, along with a file containing a representative structure for each group, in which the parent framework is highlighted for use with modeling programs.

Identification of the rigid framework

CAVEAT is designed to provide structural ideas rather than complete molecules. Hence, in the CLASS screening and clustering algorithms only the key structural elements are considered; extraneous substituents, rings judged to be unimportant for maintenance of the relative conformation of the matching bonds, etc., may be pruned by the chemist in further design steps and are therefore ignored. The rigid framework, or 'core', of a set of vectors in a structure is intended to correspond roughly to that part of the molecule necessary to obtain a rigid linkage between the base or tip atoms of the matching bonds, depending on the search type. This analysis can be more precise than the one performed when the database is created, because it is tailored to a specific set of vectors and because the core structural elements being evaluated are generally smaller than the rigid system identified in VPREP. CLASS gives the user the choice of two definitions of core, the conservative 'whole core', which includes all the whole-cycle systems that connect the vectors, and the 'reduced core', from which those elements of polycyclic systems are deleted that do not themselves support the vectors or influence the conformation. Examples of whole-core and reduced-core definitions are shown in Fig. 6.

The following procedure gives a precise definition of the two types of cores. Initially, the whole core is identified. The first step is to remove the matching bonds from the molecule. The structure is then decomposed into its basic ring systems, defined as inclusive sets of rings that have one or more bonds in common and identified as vertex-biconnected components of the

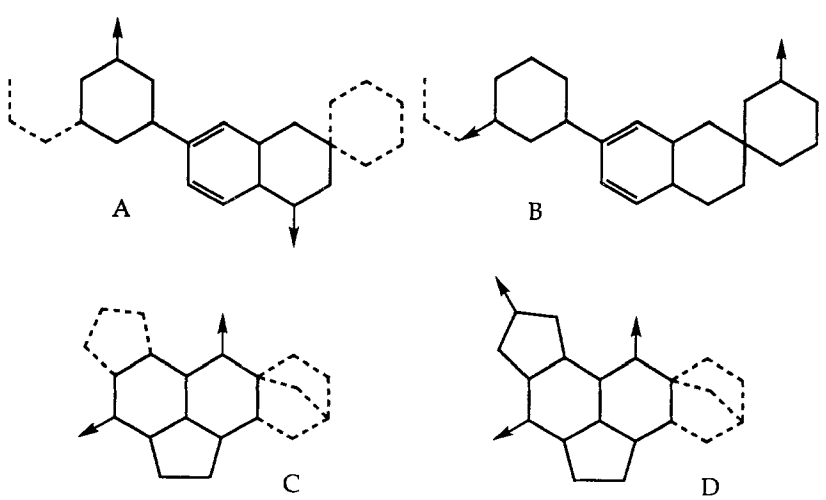

Fig. 6. Illustration of cores (bold) and extraneous bonds (dashed): (A, B) whole cores; (C, D) reduced cores.

molecular graph. Finally, the whole core is identified as those basic ring systems that lie on a direct path between vectors in any pair. To identify the reduced core, each basic ring system of the whole core is searched in turn for bridging bonds, defined as bonds whose removal would disconnect the ring system into a set of fragments such that only one fragment is connected to other parts of the whole core or to the matching bonds; these bonds are identified by a brute-force, bond-by-bond search of the whole ring system. Any disconnected fragment found in this process that does not form a three- or four-membered ring with the bridging bond is removed from the ring system; once this procedure is complete, the remaining parts of the ring systems form the reduced core.

These definitions capture chemical intuition about rigidity fairly well. In particular, spiro-linked rings that do not connect the vectors directly are unlikely to influence the conformation of the rest of the ring system, and are thus always removed. The pruning of fused rings is more problematic, because significant strain may be introduced at the ring fusions, thus influencing the overall conformation. Gauging ring strain by the presence of three- and four-membered rings is far from perfect, and therefore this pruning is performed only as an option. In any case, the comparison process in CLASS can refine this selection further by choosing only the smallest cores within a family (see below).

Scoring and screening

CLASS implements a number of straightforward chemical screens according to size, presence or absence of a SMILES-like pattern of atoms or atom types, bond type of vectors, hybridization or number of substituents of the core connecting atoms, number of atoms at ring fusions in the core, and relative energy of a conformation, if available. CLASS can also screen structures based on proximity to a surface (of a site or an ensemble of conformations) or on possible bad steric interaction with another molecule. A numerical score can also be computed to estimate the shape complementarity of the framework to another molecule, using the method in Ref. 20; this score can be used as a cutoff filter or to rank the resulting hits. Because only a small number of molecules in fixed orientations need to be tested, relatively naïve algorithms are adequate for all of these tests.

Two further screening modes have been designed to eliminate hits that would engender eclipsing interactions with appended substituents. For example, in searching for a template to serve as a peptide mimic, conformations of the amino acid side chains that are staggered in the peptide, and therefore likely to be the low-energy conformation, must not result in eclipsing interactions when these side chains are grafted onto the template. To avoid this situation, CLASS can screen hits directly against a set of substituents for each vector and assess the presence or absence of eclipsing interactions, using standard bond angles appropriate for the hybridization of the atoms involved. The program can also screen the core according to its ability to trace the backbone of another structure, as in the design of a peptide mimic, which can also minimize the possibility of eclipsing interactions. A simple non-backtracking search is used, growing the substructure isomorphism from the vector bases and guiding the search by trying to minimize the distance between matched atoms at each step.

Classification

CLASS provides a number of ways to measure the similarity of the cores of the hits for use

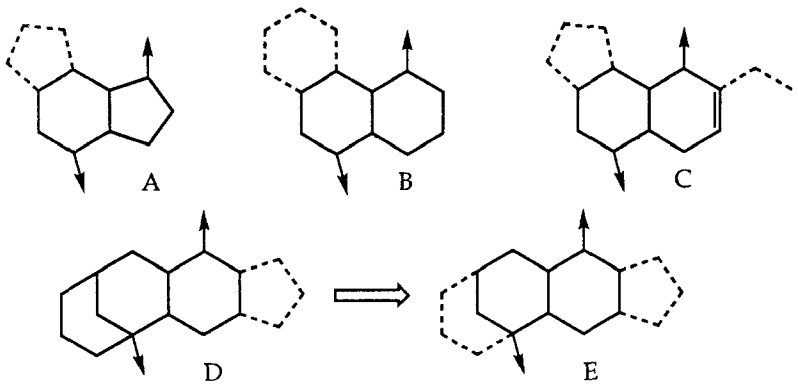

Fig. 7. Determination of similarity between core structures: (A) and (B) are not similar, but (B) and (C) are; (B) and (C) are one-way similar to (D), so (D) can be interpreted as (E).

in classifying the structures, see Fig. 7. We distinguish two kinds of similarity: structures of the same size can be similar to each other, while a smaller structure can be one-way similar to a larger structure if it is similar to some substructure of the latter. The basis for assessing similarity is always the identification of a substructure isomorphism of the cores of the two structures being compared; if no such mapping can be found, the structures are considered to be dissimilar. If an isomorphism is found, the result depends on the method chosen: the structures may be deemed similar without further checking, the mapping obtained may be used to distinguish structures based on additional criteria (e.g., the bond orders of matching bonds, the presence and orientation of stereocenters at bridgeheads or at vector attachment points), or to quantitate the similarity based on the rms fit of matching atoms.

Three different clustering methods [21] are implemented in CLASS to group the structures, based on their similarity to each other. In 'partial comparison' mode, the groups are obtained by a single-pass clustering algorithm, which limits the number of similarity calculations required. In 'full comparison' mode, all structures are first compared in a pairwise fashion to give a distance matrix; the groups are then computed using either single-linkage parametric clustering or an iterative version of Jarvis–Patrick clustering [22; J.M. Blaney, personal communication]. All three methods are automated and only require the user to define a maximum acceptable intracluster distance.

If a smaller core is one-way similar to a larger core, the smaller structure is said to subsume the larger one. When this occurs, it means that the method used to determine the core of the larger structure was too conservative, failing to remove some part of the structure that turned out to be unimportant for defining its conformation. Thus, as mentioned above, the similarity measure can decode information implicit in the searched database to refine the earlier, approximate determination of the core. To take advantage of this information, CLASS first extends the subsumption relationship to the groups that the structures are in: a group subsumes another one if any structure in the first group subsumes some structure in the second group. The groups that are not subsumed by any others are then collected into a main group list, the remainder making up the subsidiary group list; CLASS output records these two separate lists as well as the subsuming relationships between all the groups.

In devising a fast algorithm to compute the initial subgraph isomorphism on which all similarity measures between hits are based, we took advantage of three aspects of the problem. First, there is almost always great variety among the hit structures, hence the initial graph-isomorphism computation fails for most pairs of structures. Second, the structures are oriented in space to overlap the vectors, so geometrical orientation can be used to guide the search. Third, users expect structures to be grouped together only if they are close in space, thus it is appropriate for the algorithm to distinguish structures that are isomorphic if they are distant spatially.

We start by precomputing, for each core graph, a set of graph-theoretical invariants consisting of the number of vertices of each order and the number of simple circuits of each length up to seven; these invariants have the property that one core graph can be subgraph isomorphic to another one only if all its invariants are smaller. The check for subgraph isomorphism applies this condition as an initial screen, rejecting all pairs of structures for which it does not hold. If this test is inconclusive, a subset isomorphism is sought using a non-backtracking search. The search first assigns the mappings required by the vectors, then simultaneously traverses the two molecules, matching each atom of the smaller molecule in turn. The traversal order and the atoms to be matched are determined by the number of assigned and unassigned neighbors and the spatial separation of the candidates available at each step.

PERFORMANCE

Measurements of program performance indicate that the two potential bottlenecks in a CAVEAT search are the initial database search and the pairwise comparison of hits required for clustering; accordingly, these two areas received the most attention in design and implementation of the algorithms. In this section, we describe the performance achieved in the critical sections

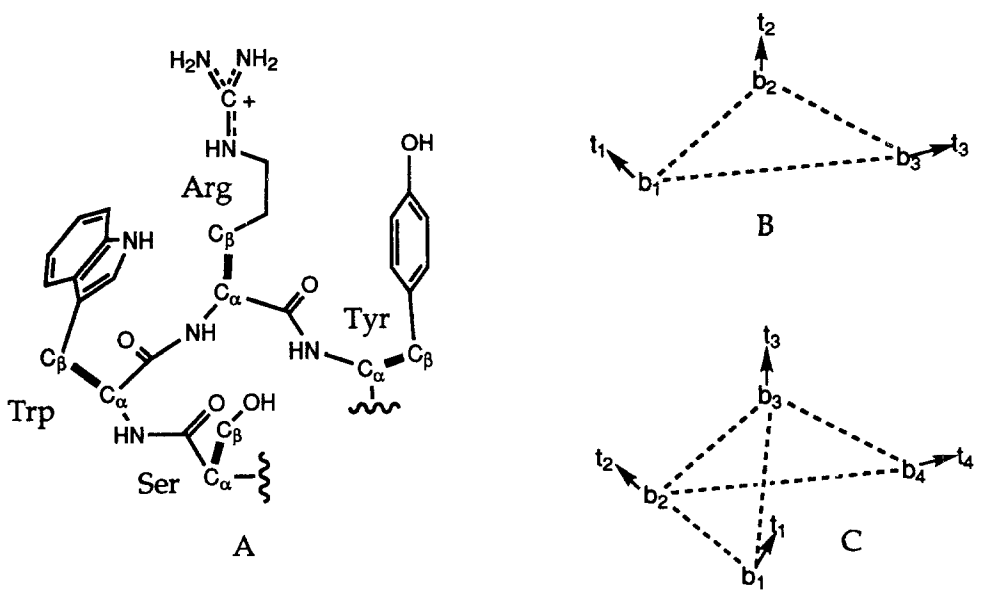

Fig. 8. Vector pairs for searches for a tendamistat template: (A) Ser^{17}-Trp^{18}-Arg^{19}-Tyr^{20} of tendamistat; (B) pairings for a three-vector search: Trp-Arg-Tyr; (C) pairings for a four-vector search: Ser-Trp-Arg-Tyr.

TABLE 1
QUERIES USED FOR PERFORMANCE MEASUREMENTS

Query	Vector pairs	Tolerance in **d** (Å)	Tolerance in α, δ (rad)
A	Trp-Arg,Trp-Tyr,Arg-Tyr	0.2	0.15
B	Trp-Arg,Trp-Tyr,Arg-Tyr	0.3	0.2
C	Trp-Arg,Trp-Tyr,Arg-Tyr	0.35	0.25
D	Trp-Arg,Trp-Tyr,Arg-Tyr	0.4	0.3
E[a]	Ser-Trp,Trp-Arg,Arg-Tyr	0.3	0.2
	Ser-Arg,Trp-Tyr	0.3	0.3

[a] This four-vector query spans a distance larger than the size of the molecules in TRIAD; note that different tolerances are applied for the more distant vector pairs (Ser-Arg and Trp-Tyr) in this case.

of the program, as well as for the system as a whole.

Timing measurements were obtained on an SGI R3000 Indigo Entry with 48 MB of RAM and SCSI-2 disks giving 1.6 MB/s sequential and 90 kB/s 4K half-disk random throughput, running IRIX 4.0.5F. All tests were run from a remote telnet session to an otherwise idle machine with disk caches flushed before each measurement. We used the large-memory binaries in CAVEAT (2.1d), modified only to collect the necessary timing information. All measurements reflect elapsed time.

The TRIAD, CAST and CSD databases were used for these benchmarks. All three databases were processed in VPREP's template mode, using the options to link ring systems removed by one or two bonds and to extend ring systems by one bond outward. TRIAD is a database of minimized tricyclic hydrocarbons, generated in-house [23]; CAST is a subset of rigid structures generated with CONCORD [17], taken from the CAST-3D database [4,16]; CSD is a subset of organic structures from the Cambridge Structural Database v. 5.0 release [15].

The benchmark queries are based on the C_α-C_β vectors of the Ser^{17}-Trp^{18}-Arg^{19}-Tyr^{20} tetrad

TABLE 2
SEARCH PERFORMANCE

Database[a]	No. of structures[b]	No. of pairs[c]	Query	All pairs[d]	Matches[e]	Time (s)
TRIAD	403926	67585636	A	43335	153	24
			B	134494	749	60
			C	218003	5485	126
CAST	387721	52174877	A	10930	41	15
			B	30259	144	30
			C	67222	931	51
			D	111107	2560	108
			E	23390	0	83
CSD	42550	5172301	D	9295	472	20
			E	3404	1	16

[a] TRIAD is an in-house database of computed tricyclic hydrocarbon structures [23]; CAST is a subset of the CAST-3D database [4,16]; CSD is a subset of the Cambridge Structural Database [15].
[b] Number of molecules contributing at least one vector pair to the database.
[c] Total number of pairs in the database.
[d] Number of molecules that contain all the pairs in the query.
[e] Number of molecules that match all the vectors in at least one way; symmetrical molecules typically match in more than one way.

TABLE 3
COMPARISON PERFORMANCE

Query	Database	Hits[a]	Total[b]	Full[c]	Successful[d]	Time (s)
B	TRIAD	748	536712	72034	22186	7
C	CAST	1297	1663211	282070	27239	31
D	CSD	592	343667	67161	7930	12

[a] Number of oriented molecules for input to CLASS; these numbers are larger than those of Table 2, because multiple matching orientations are included.
[b] Total number of comparisons performed.
[c] Number of invocations of the full isomorphism algorithm, after invariant screen.
[d] Number of comparisons in which a subgraph isomorphism was found.

in the α-amylase inhibitor tendamistat [24], as illustrated in Fig. 8 and defined, with tolerances, in Table 1.

Specific measurements are summarized in Table 2 for the search engine, in Table 3 for the all-pairs subgraph isomorphism, and in Table 4 for the time required for complete searches, including retrieval of 3D structures from the source database, overlap of structures to vectors, and reading and writing of intermediate and output files. In all three tables, only results indicative of normal usage are shown. Note that the benchmarks of Tables 3 and 4 do not make use of any screening capabilities; incorporation of these filters in a real application of CLASS would typically decrease the number of groups as well as the elapsed time.

DISCUSSION

Among the keys to CAVEAT's usefulness as a tool for generating ideas are its speed and its accessibility, since it facilitates interactive exploration of a large number of different possibilities and it runs on the same desktop workstations used for molecular modeling. The search strategy was devised to supply high performance in a limited environment, and was tailored to the attributes of framework searches: small number of query vectors, relatively tight tolerances, no chemical constraints. These characteristics result in a low potential for screenout based on individual search criteria. With over 500 000 bins, the parameter space of vector pairs in CAVEAT is the most precise one available for framework searches; even so, screenout efficiencies for reasonable searches are often below 60%, as illustrated in Table 2. Alternative 3D search methods [3,25,26] do not use such a large parameter space and thus would get even smaller screenout

TABLE 4
TOTAL ELAPSED TIME FOR CAVEAT SEARCHES WITH CLASSIFICATION

Query	Database	Hits[a]	Groups[b]	Search[c] (s)	Class[d] (s)	Total (s)
B	TRIAD	748	52/18	60	18	123
C	CAST	1297	153/377	51	61	228
D	CSD	592	135/159	20	39	140

[a] Number of oriented molecules retrieved and passed through CLASS.
[b] Total number of main/subsidiary groups.
[c] Total time spent in VSRCH.
[d] Total time spent in CLASS.

efficiencies; moreover, these methods require retrieval of all the structures that are not screened out. As long as databases must be maintained in secondary storage, this retrieval step is expensive; the key advantage of the CAVEAT approach is that enough information is present in the pair lists, as opposed to conventional bit screens, so that it is not necessary to retrieve the 3D structures to identify those that can satisfy all the criteria together. Query B on TRIAD in Table 2 illustrates this benefit: TRIAD structures averaging 1 kB would need to be randomly accessed at a rate of 2.2 MB/s to obtain comparable performance with conventional methods, whereas CAVEAT only uses 150 kB/s and is generally CPU-bound.

However, the ability to search large databases quickly is wasted if evaluation of the results is itself time consuming. Surprisingly, little attention [7] has been paid to this aspect of the problem. CLASS represents a first effort to reduce the time spent by the user evaluating hits, without unduly lengthening the computational time required or introducing unreasonable artifacts. Indeed, CLASS was designed to imitate as closely as possible the process that chemists were observed to be using at this stage. The fast comparison technique presented here was implemented after we found that single-pass clustering alone introduced artifacts that users found objectionable. In order to compute quickly the all-pairs distance matrix used for more sophisticated clustering methods, we had to speed up the comparison of structures substantially; the graph invariants described above proved quite effective, eliminating about 80% of the pairs, as illustrated in Table 3, and thereby reducing the total classification time so it is comparable to the search time.

These optimizations have reduced the two potential bottlenecks identified during the development of CAVEAT, to the extent that more than half of the elapsed time in a search can be attributed to the disk I/O and file format interpretation and conversion overhead that is unavoidable in a file-based program, as illustrated in Table 4. The impact of this overhead is reduced by the typical usage pattern of CAVEAT, in which several passes through CLASS with different parameters are made for each full database search.

CONCLUSIONS

In order for a database search strategy to be used by chemists in the design of new molecules, the program must be fast enough to be used interactively. The key technical contributions from CAVEAT in the area of database searching are in the methods used to achieve the necessary speed. With VSRCH, careful choices of the way in which the problem is formulated and the database is represented, along with the specific retrieval algorithms, make 3D database searching relatively fast, even on desktop workstations. Rapid screening on a geometric basis then makes it possible to implement the relatively more complex structure evaluation and framework identification methods found in CLASS. These in turn allow oriented hit molecules to be compared quickly enough, using subgraph-isomorphism-based methods, that reliable all-pairs clustering algorithms can be employed.

ACKNOWLEDGEMENTS

The original design and implementation of the search engine are due to S.J. Telfer, G.T. Shea and S. Waterman. We thank J.W. Macko and Chemical Abstract Services for making the CAST-

3D database available to us, R. Huber (Max-Planck-Institut für Biochemie, Munich) for the coordinates of tendamistat, J.M. Blaney for discussions on clustering methods and their implementation, and all the users of CAVEAT for their suggestions.

REFERENCES

1 Bartlett, P.A., Shea, G.T., Telfer, S.J. and Waterman, S., In Roberts, S.M. (Ed.) Molecular Recognition: Chemical and Biological Problems, Royal Society of Chemistry, London, 1989, pp. 182–196.
2 Bartlett, P.A., Etzkorn, F.A., Guo, T., Lauri, G., Liu, K., Lipton, M., Morgan, B.P., Shea, G.T., Shrader, W.D. and Waterman, S., In Chemistry at the Frontiers of Medicine, Proceedings of the Robert A. Welch Foundation Conference on Chemical Research XXXV, Houston, TX, 1992, pp. 45–68.
3 Martin, Y.C., J. Med. Chem., 35 (1992) 2145.
4 Fisanick, W., Cross, K.P., Forman, J.C. and Rusinko III, A., J. Chem. Inf. Comput. Sci., 33 (1993) 548.
5 Ho, C.M.W. and Marshall, G.R., J. Comput.-Aided Mol. Design, 7 (1993) 3.
6 Good, A.C., Hodgkin, E.E. and Richards, W.G., J. Comput.-Aided Mol. Design, 6 (1992) 513.
7 Willett, P., Winterman, V. and Bawden, D., J. Chem. Inf. Comput. Sci., 26 (1986) 109.
8 Bartlett, P.A. et al., manuscript in preparation.
9 Wilson, R.J., Introduction to Graph Theory, 2nd ed., Academic Press, New York, NY, 1979.
10 Bondy, J.A. and Murthy, U.S.R., Graph Theory with Applications, North Holland, New York, NY, 1976.
11 Aho, A.V., Hopcroft, J.E. and Ullman, J.D., Data Structures and Algorithms, Addison-Wesley, Reading, MA, 1983.
12 Aho, A.V., Hopcroft, J.E. and Ullman, J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.
13 Tarjan, R.E., Data Structures and Network Algorithms, Vol. 44, SIAM, Philadelphia, PA, 1983.
14 Mehlhorn, K., Data Structures and Algorithms, Vol. 2, Springer Verlag, Berlin, 1984.
15 Allen, F.H., Davies, J.E., Galloy, J.J., Johnson, O., Kennard, O., Macrae, C.F., Mitchell, E.M., Mitchell, G.F., Smith, J.M. and Watson, D.G., J. Chem. Inf. Comput. Sci., 31 (1991) 187.
16 Contact J.W. Macko at Chemical Abstract Services, New Product Development, for information about CAST-3D.
17 Pearlman, R.S., Balducci, R., Rusinko, A., Skell, J.M. and Smith, K.M., CONCORD, Tripos Associates, St. Louis, MO.
18 Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., Meyer Jr., E.F., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T. and Tasumi, M., J. Mol. Biol., 112 (1977) 535.
19 Abola, E.E., Bernstein, F.C., Bryant, S.H., Koetzle, T.F. and Weng, J., In Allen, F.H., Bergerhoff, G. and Sievers, R. (Eds.) Crystallographic Databases – Information Content, Software Systems, Scientific Applications, Data Commission of the International Union of Crystallography, Bonn/Cambridge/Chester, 1987, pp. 107–132.
20 Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R. and Ferrin, T.E., J. Mol. Biol., 161 (1982) 269.
21 Willett, P., Similarity and Clustering in Chemical Information Systems, Research Studies Press, Letchworth, 1987.
22 Jarvis, R.A. and Patrick, E.A., IEEE Trans. Comput., C-22 (1973) 1025.
23 Bartlett, P., Weiss, G. and Lauri, G., unpublished work.
24 Pflugrath, J.W., Wiegand, G., Huber, R. and Vértesy, L., J. Mol. Biol., 189 (1986) 383; PDB structure 1HOE.
25 Willett, P., Three-Dimensional Chemical Structure Handling, Research Studies Press, Taunton, 1991.
26 Sheridan, R.P., Nilakantan, R., Rusinko III, A., Bauman, N., Haraki, K.S. and Venkataraghavan, R., J. Chem. Inf. Comput. Sci., 29 (1989) 255.