Journal of Computer-Aided Molecular Design, 9 (1995) 13–32
ESCOM

J-CAMD 275

PRO_LIGAND: An approach to de novo molecular design. 1. Application to the design of organic molecules

David E. Clark, David Frenkel, Stephen A. Levy, Jin Li, Christopher W. Murray, Barry Robson*, Bohdan Waszkowycz and David R. Westhead

Proteus Molecular Design Ltd., Proteus House, Lyme Green Business Park, Macclesfield, Cheshire SK11 0JL, U.K.

Received 17 June 1994
Accepted 28 August 1994

Keywords: Drug design; De novo design; Enzyme inhibitors; Graph theory

Summary

An approach to de novo molecular design, PRO_LIGAND, has been developed that, in the environment of a large, integrated molecular design and simulation system, provides a unified framework for the generation of novel molecules which are either similar or complementary to a specified target. The approach is based on a methodology that has proved to be effective in other studies – placing molecular fragments upon target interaction sites – but incorporates many novel features such as the use of a rapid graph-theoretical algorithm for fragment placing, a generalised driver for structure generation which offers a large variety of fragment assembly strategies to the user and the pre-screening of library fragments. After a detailed description of the relevant modules of the package, PRO_LIGAND's efficacy in aiding rational drug design is demonstrated by its ability to design mimics of methotrexate and potential inhibitors for dihydrofolate reductase and HIV-1 protease.

Introduction

The number of protein structures determined by X-ray crystallography and NMR is ever-increasing and likely to grow rapidly within the foreseeable future [1,2]. This state of affairs has led to the growing acceptance of Structure-Based Drug Design (SBDD) as a paradigm for the design and development of novel pharmaceutical agents [3–9]. In cases where the proposed target molecule is not amenable to experimental study, computational techniques such as homology modelling [10], pharmacophore mapping [11–13] and Comparative Molecular Field Analysis (CoMFA) [14] can provide helpful insights to guide the design process.

There is currently intense interest in the development of computational methods which can make use of such information to suggest novel structures which may either prove to be useful lead compounds or, as is more likely, act as a stimulus to the creativity of the designer. Ideally, these techniques should be fast, objective and produce a set of diverse yet chemically sensible structures. A number of these de novo design programs have been reported [15–34] and the field has been recently reviewed [28].

Our in-house expert system for molecular design and simulation, PROMETHEUS, has been enhanced by the incorporation of a method for de novo molecular design which we have called PRO_LIGAND (**PRO**metheus' **L**ogically **I**ntegrated **G**eneration of **A**ctive **N**ovel **D**rugs). By means of this integration, the fifth-generation capabilities of the GLOBAL language may be used to automate and control the de novo design process. Furthermore, the reproducibility of each design experiment can be ensured by the employment of protocols or 'super-algorithms' which may also incorporate human expertise (e.g., concerning toxicity prediction) in the form of expert system rules or 'fuzzy logic'. For details on the origins of the GLOBAL language, the reader is referred to Refs. 35 and 36. In this paper, the first of a series, the underlying philosophy of PRO_LIGAND and its design will be described together with test cases demonstrating its utility in the process of rational drug design.

Program description

De novo design programs can be divided into two categories, according to the nature of the fundamental

*To whom correspondence should be addressed.

0920-654X/$ 6.00 + 1.00 © 1995 ESCOM Science Publishers B.V.

'building block' employed in the structure-generation process. One class of program seeks to build structures in an *atom-by-atom* manner [19,20,25,29]; this approach has some theoretical advantages in terms of the diversity of structures that can be produced but in actuality there are difficulties both in implementation and in execution [20, 25,26,29]. Alternatively, structures can be generated by assembling pre-stored 3D molecular fragments, usually small moieties of limited conformational flexibility. In practice, it seems that such *fragment-based* methods constitute the best compromise between the diversity of structures produced and the speed of program execution. For this reason, PRO_LIGAND has been designed in accordance with this approach.

The main tenet of the philosophy underpinning the design of PRO_LIGAND is that of the need for *flexibility*; both in the types of input information accepted and in the mode of structure assembly. In particular, PRO_LIGAND is constructed in such a way that an input structure can be used as a basis for the generation of structures either *similar* or *complementary* to it. Thus, given a small organic molecule or peptide, PRO_LIGAND can design either another small molecule with similar chemical characteristics or a pseudoreceptor. Alternatively, if presented with a receptor structure and knowledge of the position of the active site, PRO_LIGAND can design an organic molecule or a peptide to bind to the active site, or an analogous receptor site.

Here, we describe the operation of PRO_LIGAND for the design of small organic molecules; peptide design will be the subject of a subsequent paper [37]. A further paper will detail the use of PRO_LIGAND with input data derived from a series of active structures or a 3D QSAR study [38].

The following sections will describe three modules of PRO_LIGAND, each of which has a unique function in the design process. These modules are listed below, together with a brief description of their function; their relationship to each other is illustrated in Fig. 1.

(1) **Design-base Generation** – unifies the format of the input information and creates a *design base* consisting of all the atoms of the input structure which are of relevance to the design process, e.g., those which comprise the active site of a macromolecular target;

(2) **Design-model Generation** – transforms the design base into a *design model* by the rule-based conversion of atom types and positions to *interaction sites*. These interaction sites indicate the spatial and physicochemical characteristics desired in the molecule(s) to be generated by the Structure Generation module [39];

(3) **Structure Generation** – builds novel structures consistent with the design model. PRO_LIGAND follows a similar approach to that of the LUDI program [21–23] in choosing to build structures by joining molecular fragments fitted directly onto the interaction sites in the design model. Such an approach involves only geometrical calculations and is therefore rapid compared with methods requiring the evaluation of energy functions and their derivatives. It is believed that avoiding the latter does not introduce serious errors into the design process [22]. However, as Böhm points out [23], force-field calculations should be carried out after the design procedure to verify energetic aspects of the designed molecules. For example, it would be necessary to check that a putative ligand does indeed fit into the protein binding site in a low-energy conformation.

A fourth module – **Structure Refinement** – can use the initial output from the Structure Generation module as the basis for generation of further novel structures which have improved design properties. To accomplish this, an approach based upon a *genetic algorithm* [40] has been developed. This module will be reported in a subsequent paper [41].

The modules are driven by keyword-based command files and data is passed internally by means of a flexible data structure known as the Generalised Molecular Structure Descriptor (GMSD). Since the GMSD data structure plays a central role in the operation of PRO_LIGAND, it will be described first. Thereafter, each of the three operational modules will be described in more detail.

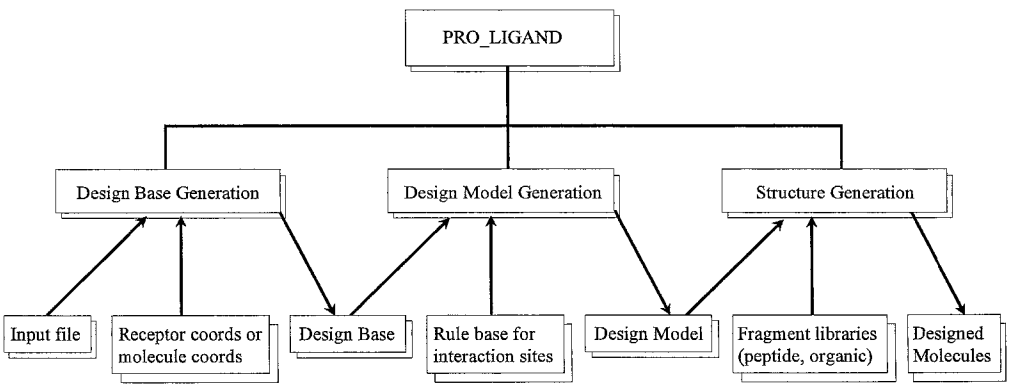


Fig. 1. Overview of the PRO_LIGAND architecture.

Generalised Molecular Structure Descriptor

The GMSD is a file structure designed to describe a set of labelled points in 3D space. The points may constitute a real molecule, a pharmacophore or any of the abstractions used by PRO_LIGAND in the design process, such as the design base or design model. The GMSD contains entries of the format '**KEYWORD** value_list'.

Typical keywords are: '**ATOM**', whose value list contains atom-based information such as label, type, and partial charge; '**CARTESIAN**', whose value list holds x, y, z co-ordinates for each of the atoms; and '**INTERACT_SITE**', describing the labels assigned to the atoms of a molecular fragment.

Conceptually, the GMSD is divided into two sections: the *basic information* block and the *additional information* block. As the names suggest, the former carries simple molecular structure information such as atom types, connection lists and Cartesian co-ordinates, while the latter contains extra information pertaining to the structure, such as its interaction sites.

The structure of the GMSD makes it extremely flexible and very easy to extend if it is decided that new keywords are needed. By storing the GMSD in binary form, very fast access can be achieved.

The **KEYWORD** value_list format is also used in the creation of the job and command files and rule bases, further contributing to the flexible and unified approach embodied in PRO_LIGAND.

Design-base Generation module

The initial module of PRO_LIGAND is the Design-base Generation module whose purpose is to take the input information as given and transform it into a single, unified form (the *design base*) which can be operated upon by the Design-model Generation module according to whether a similarity or complementarity approach to structure generation is desired. For the purposes of the current work, the input information can be in one of two forms: the co-ordinates of a macromolecular receptor together with an active-site specification or a single small organic molecule.

In the case where the user wishes to define an active site in connection with the input, the **ACTIVE_SITE** command is employed. This takes the form

ACTIVE_SITE atom_i atom_j Δx Δy Δz r

The **ACTIVE_SITE** keyword permits three modes of defining the position and spatial extent of an active site:

(1) If atom_i is different from atom_j, then the centre of the active site is taken to be the centre of mass of all atoms lying between (and including) i and j. Δx, Δy, Δz are user-definable translations from this centre, but in this instance will normally be zero.

(2) If atom_i and atom_j are the same then this position is taken as the active-site centre. Δx, Δy, Δz are user-definable translations from this centre.

(3) If either or both of atom_i and atom_j are zero, then Δx, Δy, Δz are assumed to be absolute co-ordinates for the active-site centre.

In all cases r is the user-specified radius of the active site.

Note that the **ACTIVE_SITE** keyword can be repeated as many times as desired. This enables a precise representation of the active site to be built up by spheres placed at the user's discretion in a manner somewhat analogous to that employed in the DOCK program of Kuntz et al. [42]. In practice, we have found that a simple approach to defining the active site is to employ the third option mentioned above, using the co-ordinates of selected heavy atoms of a co-crystallised inhibitor to specify the centres of the spheres delineating the active site. The radius chosen will obviously depend on the example in hand, but we have found values of 5–7 Å to be sufficient in the cases we have examined. If no inhibitor positions are available, then options (1) or (2) may be more appropriate.

The exact form of the design base will depend on the input information. If an active site is specified by the user, then the design base will consist of all the atoms falling within the limits of the active-site definition(s). On the other hand, if no active site is specified, the design base simply becomes a copy of the input molecule. In either case, the design base is output in GMSD format together with a command file containing the appropriate commands to run the next module, i.e., Design-model Generation.

Design-model Generation module

The Design-model Generation module operates on the design base to create a design model. The design model is a GMSD file containing a set of *interaction sites* which serve to define desired physicochemical properties at specific points in space [39]. At present, the types of interaction site employed by PRO_LIGAND are similar to those in the LUDI program [21–23], although the set may be readily adapted or extended. These interaction sites and their labels are the following:

(1) Hydrogen bond acceptor: denoted by the vector **A-Y** where, for example, A is a carbonyl oxygen atom and Y the attached carbon atom;

(2) Hydrogen bond donor: denoted by the vector **D-X** where, for example, D is a hydrogen atom in a primary amine and X the attached nitrogen atom;

(3) Lipophilic aromatic: denoted by R where, for example, R is a carbon atom in a benzene ring;

(4) Lipophilic aliphatic: denoted by L where, for example, L is a carbon atom in a methyl group.

The construction and positioning of these sites will be described below.

The module has two modes of operation: the production of a design model *similar* to the design base, or the production of one which is *complementary* to it. In the 'similar-design' mode, the module produces a design model which has steric and chemical features similar to the input molecule. The Structure Generation module will use this to create possible analogues of the input structure. By contrast, in the 'complementary-design' mode, a design model is produced which will result in the production of a complementary molecule, for instance a molecule to fit in the active site of a receptor, or to bind to DNA. It is worth noting at this point that after design-model generation, the concepts of 'similar design' and 'complementary design' are unified, and the operation of the Structure Generation module is independent of which type of problem is to be solved.

The interaction sites in the design model are created from the design base using rules which are specified in the *rule-base* file. These rules relate the *type* and *spatial position* of interaction sites to be generated to the atom types of the design-base atoms. The format and operation of the rule bases are given in detail in Appendix 1 and are described briefly below.

When operating in 'similar design' mode, interaction sites are created in the same position as the design-base atoms, as illustrated in Fig. 2. The type of the interaction site created is related to the design-base atom type through a rule; for instance, it is possible to create a hydrogen bond acceptor site in the position of the oxygen atom of each carbonyl group in the design base. When operating in complementary-design mode, a rule again relates the type of site generated to a design-base atom type, but this time the rule has additional components which specify the position of the site with respect to the design-base atom (see Fig. 3). The position of the site is described in the most general possible way. Sites are

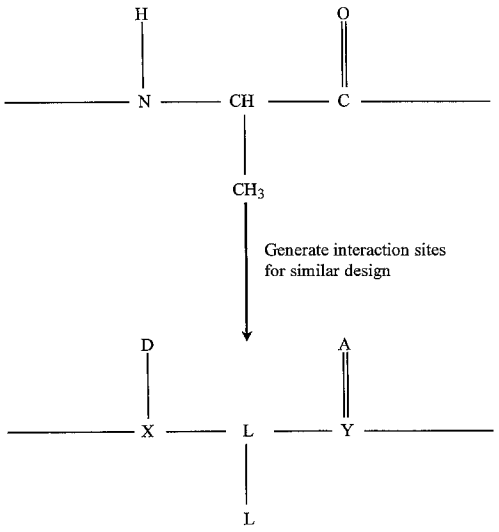


Fig. 2. Generation of interaction sites in similar-design mode.

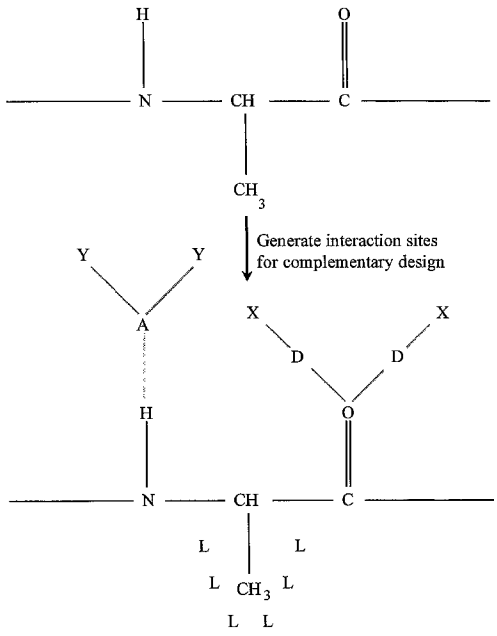


Fig. 3. Generation of interaction sites in complementary-design mode.

positioned by means of a rule type at a given bond length, valence angle and torsion angle from specified design-base atoms, and an alternative rule type is provided to give straightforward positioning of interaction sites with respect to atoms which are members of planar rings. In addition to single-point sites, both rule types allow the generation of vector interaction sites consisting of two points. This enables the generation of vector hydrogen bond sites which specify the complete geometry of groups forming hydrogen bonds with the receptor. When operating in complementary-design mode, sites which form steric clashes with the receptor are deleted from the design model.

As well as allowing the positioning of individual sites, the rule base allows sites to be positioned at equal spatial separation in ranges of bond lengths, valence angles and torsion angles. This allows for the creation of design models in which, for instance, hydrogen bond contacts with non-ideal geometry are allowed.

There are many advantages of the general and flexible nature of the rule base employed by the Design-model Generation module. Firstly, rules and atom types do not appear in the program code and hence the same code can be used for a variety of molecular systems and force fields. All that is required is construction of an appropriate rule base. Currently, we have standard rule bases in use for complementary design to protein receptors and DNA and for similar design to general organic molecules and peptides. Three sets of atom types are catered for at the present time: AMBER [43,44], POLY [45] and COSMIC [46]. Secondly, the rule base can be edited by the user. This allows quantities such as the number of sites to be generated and the extent of non-ideality of the allowed

hydrogen bond contacts inter alia to be varied in a straightforward manner. We have found that this type of flexibility can be very useful as part of a feedback loop, using information generated by a run of the Structure Generation module to optimise the design process for a particular application. Taken together, these features permit easy and rapid 'prototyping' of rule bases and thus prevent the user from being trapped by hard-coded and perhaps over-specialised rules.

Structure Generation module

The Structure Generation module builds structures using fragment libraries. The structures are consistent with the constraints inherent in the design model and the building process is controlled by the user input to the module.

Initially, the module reads in the user-supplied data, together with information concerning vital files such as the design-model files and the fragment library files. These data are passed through to the rest of the module using a keyword-controlled command file in a manner similar to that employed in the other modules. In what follows, the first subsection will describe the structure and constitution of the fragment libraries used in structure generation, the second will discuss the structure building process per se, the third will detail some miscellaneous options and features available in structure generation, and the final subsection will outline the scoring algorithm used to rank generated structures.

Fragment libraries

In constructing a fragment library, two alternative strategies are available. Some workers have used a small library (e.g., GroupBuild uses only 14 fragments [26]), hoping to take advantage of the fact that a certain diversity of structures can be constructed from a small set of relatively simple primitive units. The second approach defines many more fragments (e.g., the LUDI program [21,22]), intending to build each structure from a small selection of these fragments. PRO_LIGAND leans towards the small-library approach, with each library having a maximum of about 50 fragments, although there is no actual limit on this number. Small libraries have the advantage of being easy to manipulate so that the user is able to organize and rank the fragments to give more effective building strategies for particular problems.

The PRO_LIGAND libraries have the structure shown in Fig. 4. The peptide library contains conformations of the commonly occurring amino acids which can be used to build peptide structures, as will be described in a future paper. For the present work, it is the organic libraries which are of interest. The four parent libraries shown are accessed during the four different phases of building (i.e., the *place*, *place-join*, *place-bridge* and *bridge* phases). Each of these parent libraries is divided into various sublibraries according to the chemical nature of the fragments they contain. For example, an 'aro_polar' sublibrary will contain aromatic fragments with hydrogen bonding capability and an 'aliphatic' sublibrary will hold aliphatic lipophilic fragments. If the user wishes to exert some control over the order in which various fragment types are accessed, these sublibraries can be assigned an individual rank. The ranking directs the algorithm to search through higher ranking fragments before attempting to fit lower ranking ones. For instance, if the user wishes the hydrogen bonding features in the design model to be filled in preference to lipophilic features, the 'aro_polar' sublibrary would be assigned a higher rank than the 'aliphatic' sublibrary. By default, all sublibraries are ranked equally. The exact constitution of each of them is by no means fixed, and the user can add or delete fragments to any sublibrary or create new sublibraries tailored for a particular purpose.

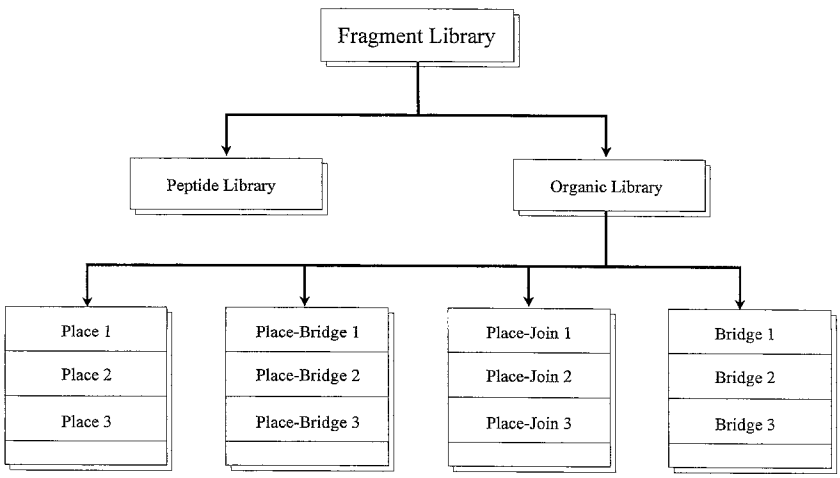


Fig. 4. Overview of the PRO_LIGAND fragment-library architecture.

TABLE 1
TYPICAL FRAGMENTS CONTAINED IN THE PLACE-JOIN LIBRARY

Sublibrary	Fragments
Ali_polar1	Acetaldehyde, butyrolactone, cyclohexanone, epoxypropane, formamide, formic acid, methyl acetate, acetamide, urea
Ali_polar2	Acetone, propenal, cycloheptanone, formaldehyde, cyanomethane, methyl amine, methyl acetamide, formimine
Ali_polar3	Dimethyl sulphoxide, guanidine, methyl carbamate, formamidine, *N*-methyl hydroxylamine, formaldehyde oxime
Ali_polar4	Ammonia, dimethyl ether, dimethyl amine, methanol, water
Aliphatic1	Propene, cyclohexane, cyclopropane, *cis* and *trans* 2,3-butene, *trans* butadiene, ethene, methyl cyclopropane, norbornane
Aliphatic2	1,3-Dithiane, dimethyl sulphide, ethane, ethyne, isobutane, propyne, methane, propane
Aro_polar1	Aniline, pyridine, pyrrole, thiazole
Aro_polar2	1,3-Oxazole, 1,3,5-triazine, imidazole, pyrazine, pyrimidine, tetrazole
Aromatic	Benzene, naphthalene, *N*-dimethyl analine, thiophene

As mentioned above, PRO_LIGAND tends towards the use of small libraries containing fairly basic chemical moieties which can be simply constructed and energy minimised. Table 1 lists some of the fragments which constitute the place-join library and which are typical of those employed by PRO_LIGAND. Each of the fragments in the library is stored in a GMSD file which contains Cartesian coordinates, atom types etc., and, where appropriate, multiple conformations for conformationally flexible fragments such as amino acid residues. Each fragment file also contains labels indicating which atoms (or *fragment interaction sites*) are to be tested for fits onto the design-model interaction sites. These interaction site labels may be added directly to the GMSD file by hand, or via a graphical user interface incorporated in our in-house molecular graphics package (PROMETHEUS EYE). The latter allows the user to pick atoms in a fragment structure and select appropriate labels which will be saved automatically with the structure. Typical examples of labels for each type of fragment are shown in Fig. 5. As can be seen, in addition to the interaction-site types described in the previous section, Structure Generation also makes use of the '*Join sites*' type of interaction site during the fragment assembly process, which is denoted by vectors **J-K**. These indicate X-H bonds on each fragment which are available for forming connections to other fragments in the manner of the CAVEAT program [47,48].

It should be noted that there may be several different labellings for each fragment and these may be assigned ranks by the user to reflect their anticipated importance in the building process. In the fitting process each of these representations is considered, with higher ranking representations examined first, since these generally have more interaction sites. Representations with the same rank are searched through in a random order during the structure-generation calculation. It is unnecessary to specify X and Y sites on the fragments, since these are generated automatically at run-time. During Structure Generation, PRO_LIGAND also performs methyl rotations at increments controllable by the user to ensure that, for example, all the hydrogens in the methyl group of methanol are considered as join sites.

Building structures

After reading the user-supplied data and design-model files, the module enters a series of drivers. The most important of these drivers controls the overall building strategy and passes through the four phases of Structure Generation, viz.

(1) placement of fragments
(2) place-bridging of fragments
(3) place-joining of fragments
(4) bridging of fragments

For each phase there are separate fragment libraries, each potentially with its own set of ranked sublibraries. It is important to note that the underlying algorithm of each phase is identical. The distinction made between them is mainly to allow better understanding and control of the structure-generation process.

The amount of building in each phase and the order in which the different modes of building are triggered is controlled by the user. The placement phase simply places library fragments onto the target sites specified in 3D

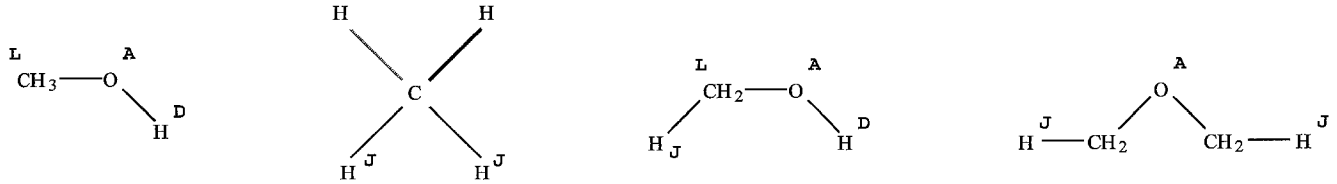


Fig. 5. Labelling of library fragments.

coordinate space by the design model. The place-joining phase attaches the fragment to fragments already placed onto the design model, whilst the place-bridging phase tries to bridge across two or more already placed fragments. In both these cases the new fragments hit design-model target sites at the same time. The bridging phase simply fits bridges between placed fragments, without seeking to satisfy any interaction sites in the process. These four modes of operation are schematically represented in Fig. 6. As a consequence of PRO_LIGAND's general driver structure, it is possible to construct molecules by sequential linking of fragments in the spirit of programs such as GROW [15,16] or by an exhaustive placement strategy with a final linking stage in a manner akin to LUDI [21–23]. An advantage of our method is that alternative strategies in between these two extremes are also available. More details about the structure-generation driver are given in Appendix 2.

After the outer driver has made decisions about the mode of building and the libraries used, an inner driver is called which searches through a library until it locates (or fails to locate) a hit. This driver then controls the placement of the new fragment and performs a number of checks on the quality of the placement. A flow diagram illustrating the operation of this part of the module is given in Fig. 7 and is explained in detail below. Note that the same driver and operations are employed independent of the mode of placement used.

The module first reads a fragment from the library according to a randomly ordered list. For this fragment, the module loops over the different conformations of the fragment in a random order, together with the different sets of fragment interaction sites (also in a random order, but respecting any ranking that may be present); these loops are not shown in Fig. 7 for clarity. The randomisation is important because PRO_LIGAND uses a depth-first strategy to circumvent the combinatorial problem of building a ligand – in other words, the first acceptable fit of a fragment is automatically accepted. This means that the order of relevant lists and arrays must be continuously updated to prevent biases being introduced into the procedure.

The next operation is to establish whether the fragment can be placed onto the design model. The detection of a possible fit is performed by means of the subgraph isomorphism algorithm of Ullmann [49], using an implementation similar to that of Brint and Willett [50]. This algorithm has the advantages of quickly detecting fragments that do not fit and being able to enumerate all possible hits if required. One slight disadvantage is that the algorithm compares distance matrices for the design-model interaction sites and the fragment interaction sites and so cannot detect differences in chirality. A subsequent check is thus required when more than three interaction sites are being fitted. This chirality check is accomplished in a similar manner to that described by Golender and Vorpagel [51]. User-supplied tolerances control how tightly the fragment and the design model must match. After locating a hit the algorithm automatically passes the fragment on for further processing, but the graph matching code can be re-entered to generate new hits should the current one be rejected subsequently. Note that the Ull-

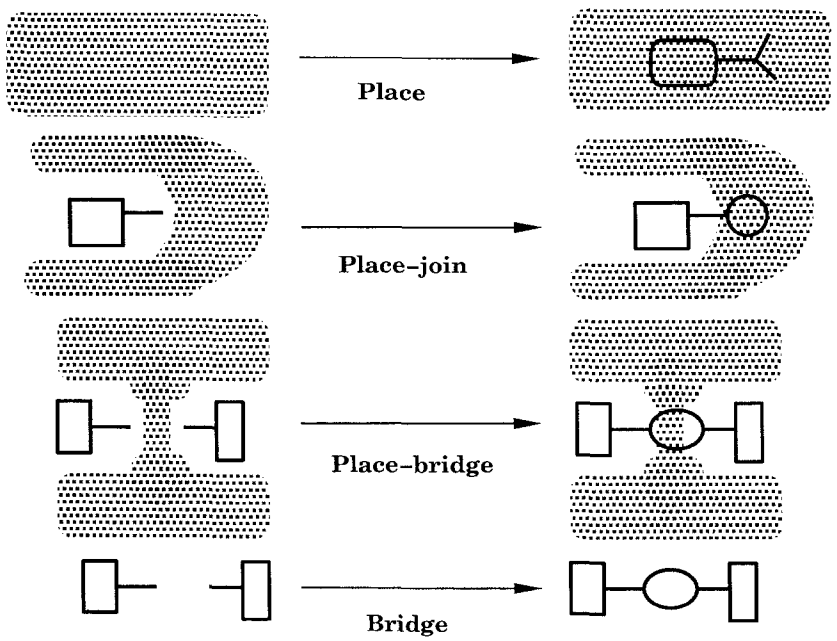


Fig. 6. Available building modes in structure generation. The dots represent interaction sites which may be hit by a placed fragment. Note that interaction sites are deleted from around a fragment once it is placed.

mann method constitutes an *exact* matching procedure, i.e., *all* fragment interaction sites must be matched by corresponding *distinct* design-model interaction sites. It is possible to envisage a *partial* matching procedure using a clique detection algorithm [52–54] in which the user specifies a minimum number of interaction sites that must be matched for each fragment. Such an approach has not been adopted here for at least two reasons. Firstly, we are concerned that the fitted fragments do not stray too far from the edges of the design model, since this implies a likely clash with the receptor wall. For this reason, as far as possible we attempt to label the fragments so that the distribution of interaction sites represents the spatial extent of the fragment. By then insisting that all of these sites be hit, we reduce the likelihood of fragments being placed in positions giving rise to undesirable steric contact with the receptor. Secondly, previous studies have shown that clique detection is rather less efficient for the purpose of graph matching than the Ullmann algorithm [50], particularly when the likelihood of a mismatch is high. Furthermore, the clique detection algorithm of Bron and Kerbosch, which is often used in molecular structure applications [13,53,55–57], can be very demanding of virtual memory since it requires the manipulation of a matrix which is potentially $(NF \times ND)^2$ in size, where NF is the number of interaction sites on the fragment and ND is the number of interaction sites on the design model.

The fragment is next fitted onto the design model using an rms superposition algorithm. This fitting is weighted, so that some interaction sites can be fitted more tightly than others if the user so desires. For example, hydrogen bonding features will normally be weighted more strongly than lipophilic features. Once the fragment has been placed on the design model, some tests are performed to check the suitability of the fit. The first test searches for clashes between already placed fragments and the new fragment. It detects whether clashes exist between heavy (i.e., non-hydrogen) atoms and if they do, decides whether they can be resolved through the formation of bonds. Of course, resolvable clashes will be present for any kind of bridge or join fragment, but even with a simple placed fragment the algorithm may detect resolvable clashes and subsequently form new bonds. When searching for resolvable clashes, a pair of (say) C-H bonds which are suitably oriented for bond formation will be sought. If such a pair does not exist in the current conformation, the algorithm will check to see if rotation of available methyl groups will allow bond formation. If the bonds concerned are too far apart for direct bond formation, the possibility of inserting a methylene bridge between the fragments is automatically considered.

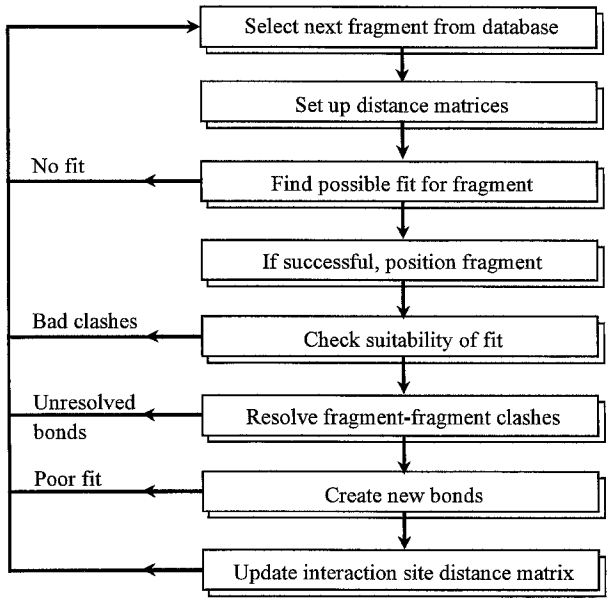


Fig. 7. Overview of the structure-generation process.

TABLE 2
LIBRARY FRAGMENTS USED IN THE CONSTRUCTION OF THE MTX MIMIC

Fragment	Number of times used
Pyrazine	1
$H_2CNCHNH$ (a specialised ring-fusion fragment)	1
Dimethyl aniline	1
Planar NH_3	2
Methanol	2
Propane	1
Ethane	1
Formaldehyde	1

The next test involves a check to ensure that any proposed bonds do not involve the joining of an unsuitable pair of atoms, e.g., O-O, N-N and so forth. The final test looks for van der Waals clashes between heavy atoms in the active site and fragment. If any of these tests fail, the fitted fragment is rejected and the graph matching algorithm is restarted to look for new fits.

The fragment is then added to the ligand structure and new bonds are created where required. Subsequently, a geometry correction is performed to create an ideal geometry around the bonds just formed. To ensure that these corrections have not caused a gross distortion in the ligand, it is refitted on the design model. If this refitting fails, the newly joined fragment is rejected and the algorithm backtracks to the previous partial structure.

The fitting process is now complete, but it is necessary to update the design model. Interaction sites that have been hit are removed from the list, as are interaction sites that clash (determined by a user-defined tolerance) with heavy atoms on the new fragment. Additionally, the partial ligand structure is examined and most of its hydrogen atoms are used to generate new join sites which are included in the design model.

The driver is exited when either the library has been exhausted or one fragment has been placed. It should be noted that no attempt is ever made to differentiate between high-scoring or low-scoring fits. Any fit that satisfies user-definable constraints is immediately accepted. This constitutes a depth-first search through solution space and solutions of varying quality and great diversity are quickly generated. Since one of the primary functions of tools such as PRO_LIGAND is to generate ideas by 'molecular brainstorming', such a strategy is well justified. Subsequent refinement can be used to weed out or improve solutions of lower quality.

Other options and features

The Structure Generation module can also be instructed to increase the frequency of ring fusion during the building process. Fusion can be thought of as place-bridging, bridging already joined fragments in such a way that a ring is created. The place-bridge library contains fragments such as butadiene and butane which are in the correct conformations to form fused rings. If these fragments are placed in a separate high-ranking library, a fusion phase can be included in the strategy. This method has been successfully used to reproduce the framework of steroids during building.

Options are present that allow the pre-screening of library fragments and their conformations in a manner analogous to a 3D database search [58]. This option creates a file containing information on whether a particular fragment or conformation can ever be fitted onto the design model. The program uses this information to avoid searching through useless files or conformations and thus speeds up the building process. The job can be restarted using an existing screen file and the screen file can also be edited by hand to exclude fragments that are known to be detrimental to a particular building task.

PRO_LIGAND also has an option for the inclusion of a *seed fragment*, i.e., a structural moiety that forms some portion of the ligand and is used as a starting point in structure generation. More interestingly, previously generated structures or parts of previous structures can be used as seeds. The seed option can thus be thought of as trapping information about important or useful parts of the tree-search problem, and allowing future runs to focus on these areas. Another approach to this problem that has been implemented allows the automatic formation of application-dependent fragment libraries. If requested, the Structure Generation module can produce libraries of structures of size n, where n is the number of original fragments in each of the new library fragments and is user-definable. The algorithm guarantees that growth can occur from each of the fragments it generates. This strategy allows one to store knowledge about which fragments can join to each other and may be used again to reduce computational effort in future structure-generation runs.

Scoring algorithm

Once a structure has been built by PRO_LIGAND, it is assigned a score to reflect how well it has met the constraints and features of the design model and also other intrinsic structural features specified by the user. The score is a weighted sum of the number of interaction sites hit together with terms for the number of asymmetric carbon atoms, number of rotatable bonds and degree of structural disjointness (which may result from a 'place-and-join' building procedure).

More specifically, a structure's score, S, may be calculated as follows:

$$S = \sum_{1}^{NA} W_A + \sum_{1}^{ND} W_D + \sum_{1}^{NAl} W_{Al} + \sum_{1}^{NAr} W_{Ar} + \sum_{1}^{NRot} W_{Rot} + \sum_{1}^{NAsym} W_{Asym} + \sum_{1}^{NComp-1} W_{Disj}$$

where NA and ND are the numbers of hydrogen bond acceptor and donor sites hit by the structure, NAl and NAr are the numbers of lipophilic aliphatic and aromatic interaction sites hit by the structure, NRot and NAsym are the number of rotatable bonds and the number of asymmetric carbon atoms in the structure and NComp is the number of disjoint components in the structure. W_A is the contribution to the score for each hydrogen bond acceptor site hit and the other weights refer to their respective features.

All the weights mentioned may be specified by the user so that those structures which best meet the user's requirements, both in terms of the design-model constraints and intrinsic structural features, will be assigned the top scores. The precise values for each of the weights will obviously vary from one design problem to another, but in general, one would want to weight hydrogen bonding features more strongly than lipophilic features and to give small penalties (i.e., negative weights) to rotatable bonds and asymmetric carbon centres. In general, disjoint structures will not be of interest and this can be reflected by the assignment of a large penalty for those structures with more than one distinct component. In the absence of user-specified values, the program assigns defaults of –0.1 to W_{Rot} and W_{Asym}, +1.0 to W_A and W_D, +0.25 to W_{Al} and W_{Ar} and –2.0 to W_{Disj}.

While the above score function is simple, it has proved to be quite adequate in practice as an initial 'screen' to be applied to the generated structures. Obviously, no scoring function, however complex, is going to be accurate or flexible enough to replace the experienced judgment of a synthetic or medicinal chemist. The best that can be hoped for is some gross ranking of the output to give the user some assistance in extracting the solutions worthy of further consideration. At present, we are also implementing a set of clustering and ranking procedures [58,60] to allow the user to gain a rapid overview of the structural

classes generated. This will be particularly useful when the output consists of several hundreds or thousands of structures. Such tools are commonly used in conjunction with 3D database searching systems, where the problem of voluminous output, especially from 'flexible' searches, is well documented [61–63].

Output and graphical analysis

The output from a PRO_LIGAND run consists of the required number of structures in GMSD format, together with a log file containing their scores and the history of the building process. PROMETHEUS EYE, the graphical display package of PROMETHEUS, enables viewing of the design base and design model, a dynamic replay of each molecule's construction and it offers facilities to aid the construction of fragment libraries.

Results

As an indication of the capabilities of PRO_LIGAND, we have tested the approach using well-characterised examples. Below we illustrate the use of the program in similar-design mode to design a mimic of the anti-cancer agent methotrexate, and in complementary-design mode to design potential inhibitors for dihydrofolate reductase and HIV-1 protease. It is worth noting that in all these examples, the results presented represent the culmination of several runs of the structure generation module, which is usually instructed to generate in the order of 100 structures per run. Such iteration is necessary both to tune the various parameters and score weights for a particular example and also permits the use of a 'seeding' strategy, as exemplified by the HIV-1 protease case.

Similar design to methotrexate

As is well known, the enzyme dihydrofolate reductase (DHFR) catalyses the NADPH-dependent reduction of dihydrofolate (FH_2) to tetrahydrofolate (FH_4). FH_4 plays a vital role in the biosynthesis of thymidylate (dTMP), which is an essential building block of DNA. Inhibition of DHFR interrupts the supply of FH_4, causing disruption in the synthesis of purine and pyrimidine bases and, eventually, cell death. The discovery of inhibitors of

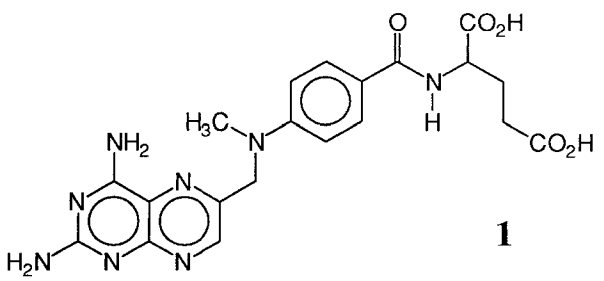


Fig. 8. The structure of methotrexate (**1**).

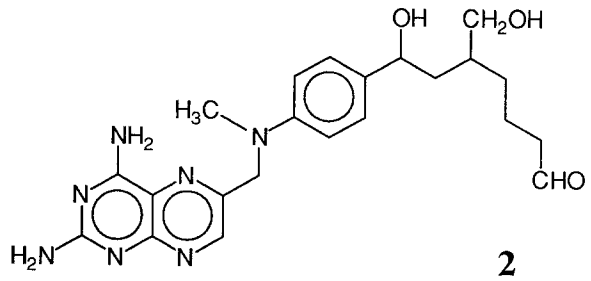


Fig. 9. A methotrexate mimic (**2**) designed by PRO_LIGAND.

DHFR has led to several useful drugs for the treatment of cancer, bacterial infections and malaria [64].

One such inhibitor is methotrexate (**1**), shown in Fig. 8, which has been studied for over 30 years and remains an important therapeutic agent for cancer and a variety of other indications, such as severe cases of rheumatoid arthritis and psoriasis. Although methotrexate is of definite utility against some forms of cancer, it has shown limited activity against other types of malignancy and is also rather toxic. For these reasons, it would be desirable to develop a drug which possesses a broader range of applications and which has less severe side effects [65].

To test PRO_LIGAND in similar-design mode, we constructed a design base from all the atoms of methotrexate. The heavy-atom positions were taken directly from the crystal structure of the inhibitor as crystallised with *E. coli* DHFR (Brookhaven Databank code 4DFR) [66] and appropriate hydrogens were added afterwards. The Design-model Generation module was then run on this design base to yield a design model consisting of 48 interaction sites (16 R, 12 L, 3 X, 5 D, 3 Y, 9 A).

This design model formed the input for the Structure Generation module. In order to encourage the production of ring systems analogous to methotrexate in the designed structures, a 'GROW_AND_FUSE' strategy was chosen. The choice of this strategy simply instructs the Structure Generation module to search the library of fragments suitable for fusion before any other during the place-bridge phase of building.

The structure of one of the analogues (**2**) designed by PRO_LIGAND is shown in Fig. 9. As can be seen, the structure closely mimics that of methotrexate, particularly in the reproduction of the pteridine ring and the para-substituted benzene. The main differences lie in the oxidation states of the hydrogen bond acceptors which form the analogue to the glutamate moiety of methotrexate. In the molecule designed by PRO_LIGAND, the carboxylic acid groups are replaced by an alcohol and an aldehyde. In addition, there is an extra rotatable bond in the chain linking the two acceptor groups. Finally, the backbone amide has disappeared to be replaced by another alcohol group. Table 2 lists the fragments that were used by PRO_LIGAND to create the designed molecule.

The typical CPU time required to generate such a

structure is in the order of 35 s on an SGI Indy workstation.

Complementary design to DHFR

Rather than designing molecules which are similar to a known lead, another approach to inhibitor design is to build structures complementary to the enzyme active site. This is a rather less constrained problem and so promises to generate a greater diversity of structures.

For complementary design to DHFR, we again began with the 4DFR crystal structure, specifically, the B chain. The atoms comprising the active site were defined as those falling within spheres of 5.0 Å radius centred on the crystallographic positions of the methotrexate atoms. This was found to be sufficient to include the residues important for inhibitor binding. Two water molecules known to be important for binding (B603 and B639) were added to give a total of 179 design-base atoms.

This design base was then operated upon by the Design-model Generation module which resulted in a design model consisting of 415 interaction sites, shown in Fig. 10. In addition to the underlying swathe of lipophilic sites which are produced by various backbone and side-chain carbon atoms, there are four distinct groups of hydrogen bonding interaction sites. Group A is a set of donor sites arising partly from the guanidinium group of Arg^{57}, but mainly from that of Arg^{52}, which also gives rise to Group B. The Group C donor sites are produced by the water molecule which mediates between Asp^{27} and the N5 atom of methotrexate. The set of acceptor sites, Group D, results from the carboxyl group of Asp^{27}. Finally, two vector acceptor sites appear between Groups B and C. These arise from the carbonyl group of Ile^{94}. Other such vectors from this residue and also from Ile^{5} have been removed by the clash-checking algorithm, probably because of a bad contact with a hydrogen on the receptor.

Once again, a GROW_AND_FUSE building strategy was employed in the structure generation process. One of the structures (**3**) produced by PRO_LIGAND is shown in Fig. 11. In this example, PRO_LIGAND has chosen to satisfy the Group A sites with a pyrido[3,4-*b*]-pyridine fragment (constructed from the fusion of pyridine with CH_2NCHCH_2). While this fragment is reminiscent of the pteridine ring system of methotrexate, it lacks the necessary hydrogen bond-donating substituents that would permit it to bind to Groups C and D. The lipophilic linker group is provided by dihydrophenanthrene which is attractive both for its rigidity and ready availability. However, while PRO_LIGAND has met the Group C

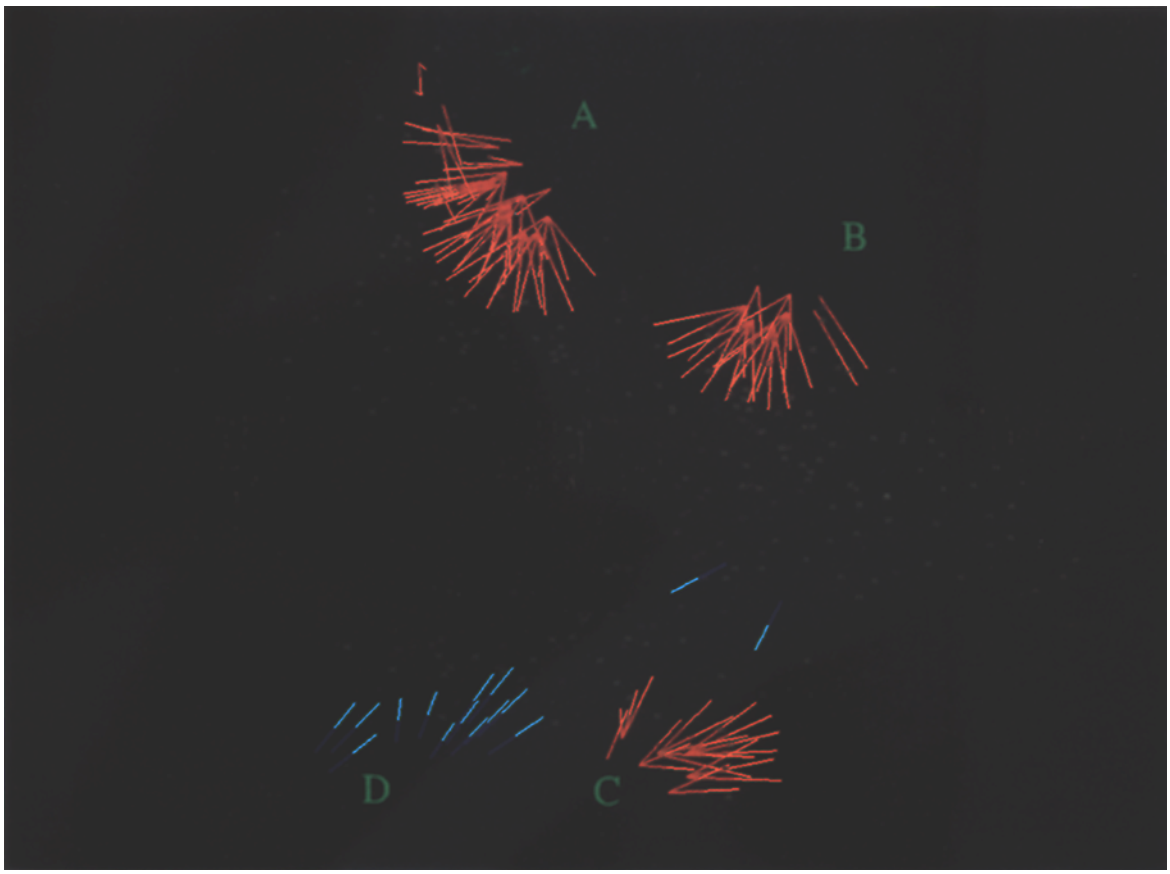


Fig. 10. The design model for complementary design to DHFR.

and D sites, the aliphatic ether-containing chain linking the tricyclic lipophilic group and the phenolic moiety is perhaps rather too flexible.

In an attempt to reduce this flexibility, the structure was put back into PRO_LIGAND as a seed fragment and Structure Generation was run with a GENERAL building strategy (one in which the user can choose the precise number and sequence of fragment placing and bridging operations) equivalent to a 'ring-bracing' mode [32]. One structure (**4**) resulting from this run is shown in Fig. 12. As can be seen, the chain linking the pyrido[3,4-*b*]-pyridine and dihydro-phenanthrene groups has been braced by the formation of a seven-membered ring containing a sulphur atom. A number of other bridges of this nature were also observed, suggesting that there is a sizeable region of lipophilic space that could be filled in this area. Interestingly, PRO_LIGAND failed to bridge the chain linking the dihydrophenanthrene and phenolic moieties. The reason for this is believed to be that any bridging fragments placed at this end of the molecule caused unacceptable clashes with the receptor.

Another structure (**5**) generated by PRO_LIGAND is shown in Fig. 13. This structure, containing a lipophilic backbone of five fused rings, is rather more rigid than the previous example. Again, all the groups of hydrogen bond sites are met by this structure. This is illustrated in Fig. 14 which shows **5** superimposed upon the hydrogen binding sites of the design model.

It should be noted that all of the above structures were built from fragments containing no more than one ring – a clear indication of PRO_LIGAND's ability to construct complex cyclic structures. The typical CPU time required to build a structure complementary to the DHFR binding site was approximately 5 min on an SGI Indigo workstation. The time taken for each attempt to ring-brace a structure was about 10 s on the same machine.

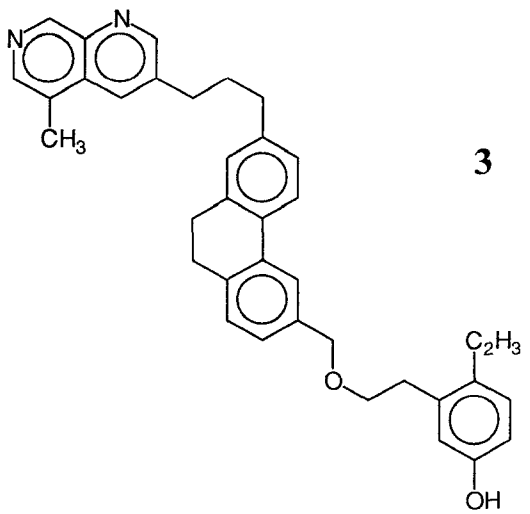


Fig. 11. A potential DHFR inhibitor (**3**) designed by PRO_LIGAND.

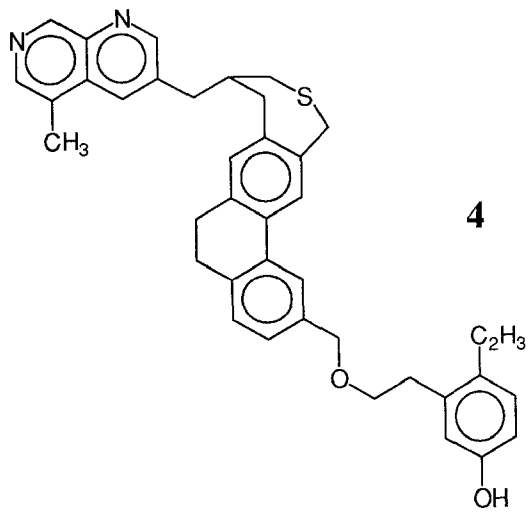


Fig. 12. Structure (**4**), illustrating PRO_LIGAND in ring-bracing mode.

Complementary design to HIV-1 protease

In the search for therapeutics and vaccines to treat and prevent acquired immunodeficiency syndrome (AIDS), much attention has focussed upon its causative agent – human immunodeficiency virus (HIV). In the life-cycle of HIV, the processing of *gag* and *gag-pol* polyproteins by the enzyme HIV protease has been shown to be essential for viral replication. Thus, it is generally believed that if the activity of the protease can be inhibited, the spread of viral infection can be attenuated [67,68]. The protease has thus become a popular target for rational drug design efforts and a number of novel inhibitors have been designed using SBDD methods [69–71].

As a further illustration of PRO_LIGAND in complementary-design mode, the approach was used to design a ligand to fit the binding site of HIV-1 protease. The crystal structure used in this example was that of HIV-1 protease complexed with the inhibitor acetyl pepstatin (PDB entry 5HVP) [72]. As in the previous example, the active site was defined using the positions of selected inhibitor atoms as centres for spheres, in this case of 7.5 Å radius. With the addition of a water molecule necessary for mediation in the contact between Val^3 and Sta^4 in the inhibitor and Ile^{50} and Ile^{250} in the protease [72], a total of 491 atoms were present in the final design base.

After the operation of the Design-model Generation module, a design model consisting of 855 interaction sites was produced. Figure 15 shows a cut through this design model and the design base from which it was derived. The picture clearly shows how the hydrogen bond interaction sites represent the position and nature of the desired functionality in the ligand to be generated. For instance, the water molecule mentioned above gives rise to two sets of orange-red vector sites where acceptor groups should be situated and carbonyl groups in the active site can be seen to spawn blue vector sites for the positioning of

donor groups. The design model also reflects the approximate C_2 symmetry of the active site – a feature often exploited by effective inhibitors of the enzyme [69,70].

To build structures to fit this model, a GROW strategy for structure generation was employed. In this strategy, a single fragment is placed and then subsequent fragments are joined to the current partial ligand. One of the resulting structures (**6**) is shown in Fig. 16 and is superimposed upon the design model in Fig. 17. The structure shown is actually the result of several PRO_LIGAND runs, during which observed desirable features were trapped as 'seeds' for subsequent structure generation runs. This iterative design process is illustrated in Fig. 18. The figure shows firstly the initial thiazole ring seed (**7**). This was selected because it had been seen to fit snugly on the design model, while making favourable hydrogen bond contact with the mediating water molecule mentioned above and being well positioned for the growth of side chains to form the backbone of the inhibitor. A subsequent run added a reduced amide functionality to this seed. The resulting structure (**8**) was in turn taken as a seed and a further structure generation run produced the structure (**9**) which formed the seed fragment for the final structure.

As can be seen from Fig. 17, the designed molecule can be expected to have many favourable hydrogen bond interactions with the enzyme active site. Of particular interest is the effective 'trapping' of the mediating water by the thiazole ring nitrogen and a carbonyl oxygen. It should be noted that the main aim of our study was to illustrate the power of PRO_LIGAND by producing a thought-provoking solution for the backbone of an HIV-1 protease inhibitor. The simpler design problem of selecting optimal substituents for the S and P subsites has been considered by other workers [22,25,26] and could also be undertaken by PRO_LIGAND.

Discussion

The three examples presented above demonstrate PRO_LIGAND's ability to generate, with considerable rapidity, diverse and novel chemical structures which satisfy the constraints imposed by a design model. These structures may be either similar to the input molecule (as in the case of the methotrexate mimic) or complementary to it (as in the examples of inhibitor design for DHFR and HIV-1 protease).

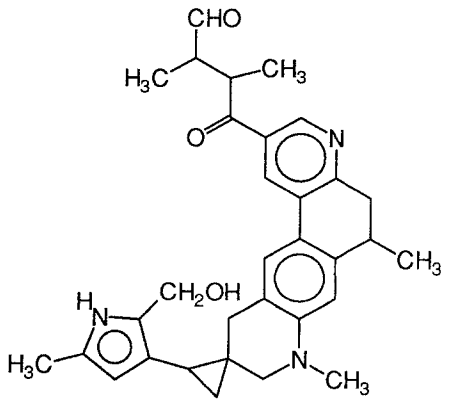
Of the de novo design programs already described in the literature, PRO_LIGAND is perhaps most similar to LUDI [21,22] in that structure generation is effected by joining molecular fragments placed upon target interaction sites. However, there are several points concerning the design philosophy and implementation of PRO_LIGAND that we feel represent significant and novel advances in methodology.

Firstly, in designing PRO_LIGAND, the desire for a unified and general approach to de novo ligand design has been at the forefront of our thinking. We have also sought to allow the user as much control as possible over the design process. This is reflected in many aspects of the approach, such as the flexible and readily extendable nature of the GMSD interface, the multi-sphere definition of the active site, the almost limitless variety of strategies available for fragment assembly via the generalised Structure Generation driver, the ease with which fragment libraries can be assembled and manipulated for particular tasks, the development of user-definable rule bases governing the nature and position of the interaction sites created during design-model generation and the ability to weight the terms in the scoring function to give high scores to the structures considered desirable by the user. The approach taken is sufficiently general to permit PRO_LIGAND to design not only organic molecules but also peptides and, furthermore, to build structures to satisfy pharmacophore maps and CoMFA models. These latter applications will be the subjects of future papers.

A further and fundamental concept is the unification of the concepts of 'similar' and 'complementary' design by means of the design model. That is, the task of structure generation is oblivious to the type of design it is carrying out – the placing and joining of fragments is the same in either case. This considerably extends the range of design problems to which PRO_LIGAND may be applied.

Other features which we consider to be unique to PRO_LIGAND are the use of a proven molecular graph theory procedure for fragment placing which we believe to be more efficient than that employed in LUDI, the pre-screening of library fragments which further increases efficiency by immediately removing those fragments which cannot fit the design model, and the use of sophisticated geometry-correction routines in an attempt to ensure reasonable geometries in generated structures.

This last point obviously highlights the fact that in developing PRO_LIGAND we have chosen not to use



5

Fig. 13. Structure of another potential DHFR inhibitor (**5**) designed by PRO_LIGAND.

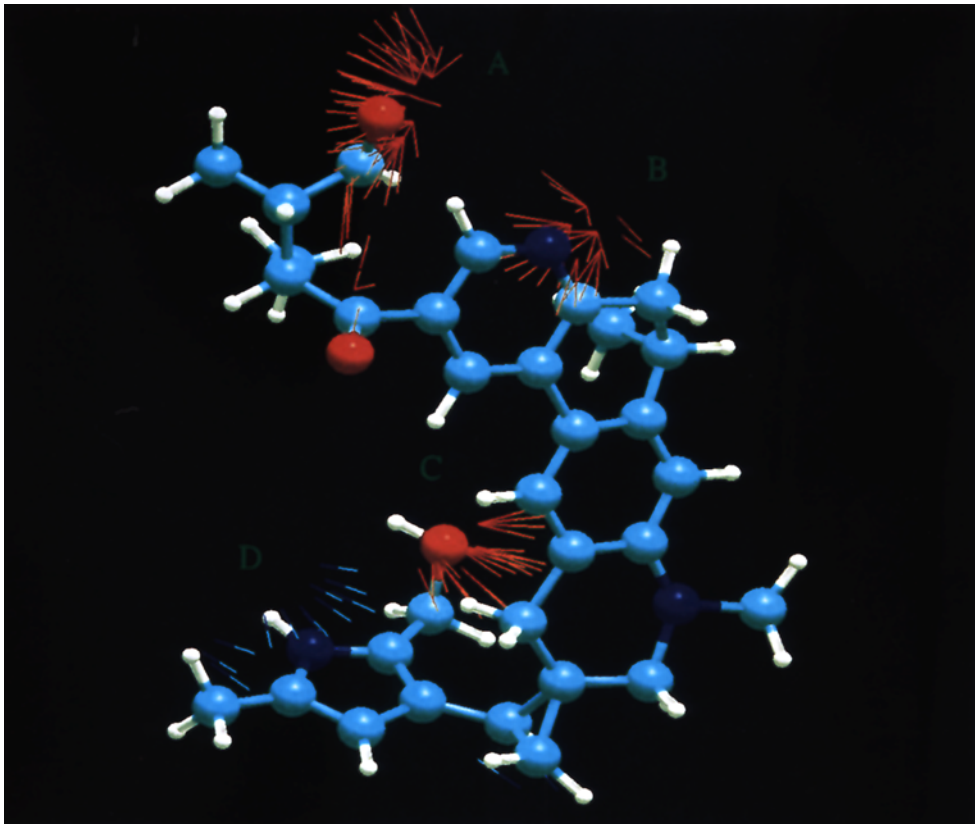


Fig. 14. Potential inhibitor (**5**) designed by PRO_LIGAND, showing a fit to the design model.

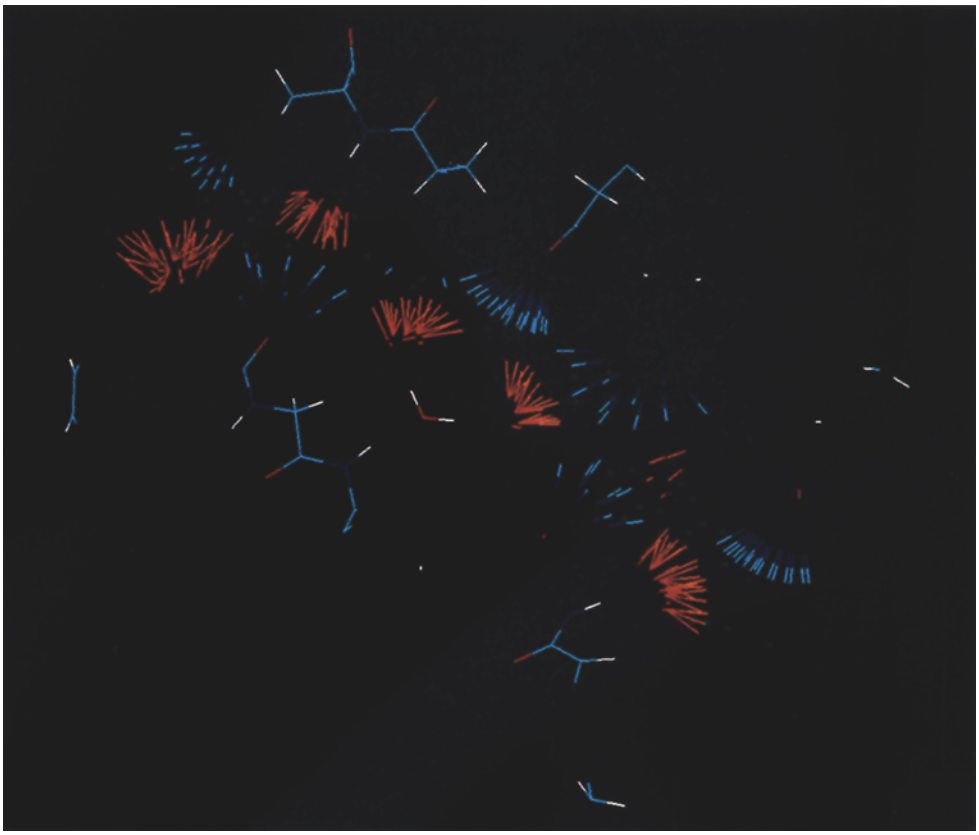
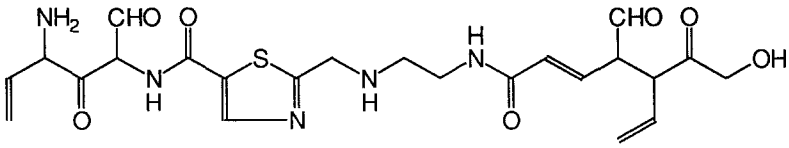


Fig. 15. A cut through the design model for complementary design to HIV-1 protease, showing some of the parent features from the enzyme active site.



6

Fig. 16. A potential HIV-1 protease inhibitor (**6**) designed by PRO_LIGAND.

force-field methods to guide the structure-generation process or to rank the generated structures. There are several reasons for this. Firstly, as Böhm has pointed out [22], traditional force-field methods are prone to the multiple-minimum problem, i.e., a considerable number of calculations could be required to determine the 'optimal' position of the ligand. Secondly, use of force-field calculations is likely to add significantly to the time taken to generate a ligand, particularly if the minimisation of the growing ligand is carried out with respect to the receptor. Finally, a further difficulty arises from the lack of a force field which is suitably parameterised to treat both macromolecular and small organic molecules simultaneously. Our experience with PRO_LIGAND has shown that useful solutions may be obtained using the purely geometrical approach advocated by Böhm. It should always be borne in mind that PRO_LIGAND is primarily intended as an 'idea generator' and not as a substitute for the experience and creativity of medicinal chemists and molecular modellers.

We are, however, actively considering the use of limited force-field calculations in the geometry-correction process. As it is impossible for the current set of correction rules to take account of every case that may be encountered, there are still occasions when the bond lengths or angles in generated structures show deviations from their ideal values. The use of a rapid molecular mechanics 'clean-up' procedure prior to refitting the partial structure back onto the design model would address this problem, without adding significantly to the CPU time required for structure generation.

As mentioned earlier, in PRO_LIGAND we have chosen to use a fairly small (although readily extendable) library of fragments from which to assemble structures. In this respect, our method differs significantly from LUDI which uses about 1000 fragments. The ability of PRO_LIGAND to construct fused ring systems means that only primitive cyclic structures need to be included. At present, the default libraries contain a total of less than 200 fragments although, of course, there is significant redundancy in terms of the chemical diversity since a given fragment (e.g., methanol) may appear in all four libraries but can be labelled differently in each. In fact, the total number of chemically distinct fragments is only 68. Even with this small number, we found no shortage of diversity in the structures generated. Of course, invoking the screening option can reduce this number for any particular system. It is our belief that increasing the num-

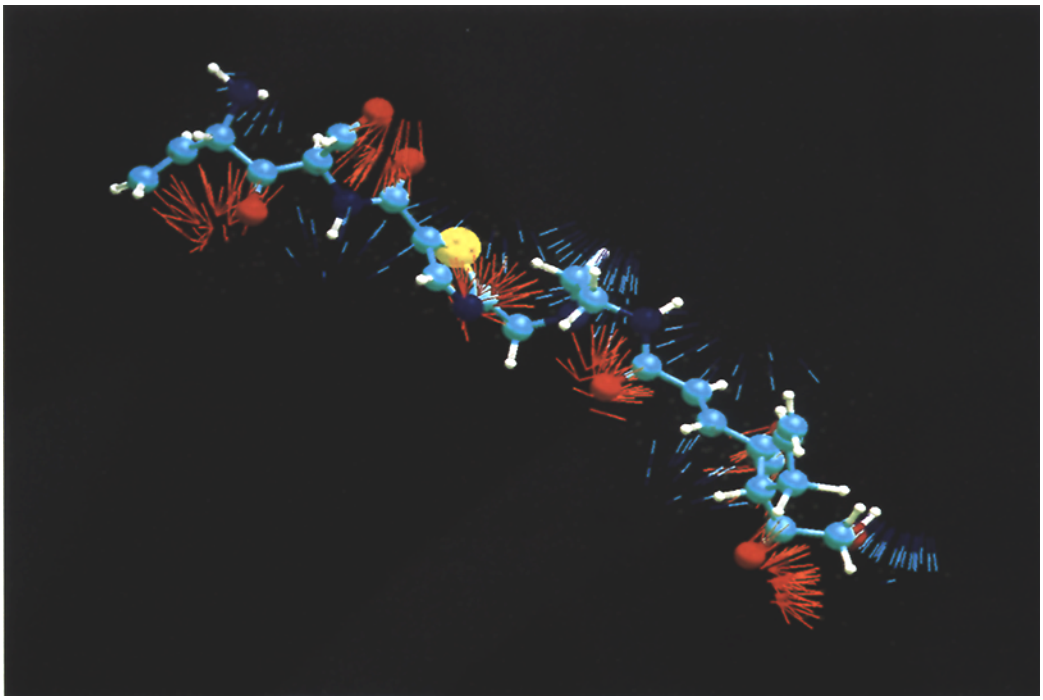


Fig. 17. Potential inhibitor (**6**) designed by PRO_LIGAND, showing a fit to the design model.

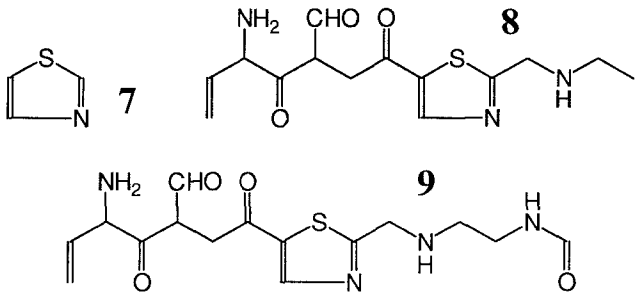


Fig. 18. Seed fragments used in the generation of (**6**).

ber of fragments substantially will generally have a deleterious effect on the execution time of the program, while not significantly increasing the novelty or interest of the designed structures.

Conclusions

We have described an approach to de novo molecular design called PRO_LIGAND which we believe embodies some significant and novel features both in its design and implementation. The utility of PRO_LIGAND has been demonstrated by three examples showing its ability to build diverse and novel structures which are either similar or complementary to a specified molecular target. The use and development of PRO_LIGAND within the expert system PROMETHEUS remains an active research area. Future papers will detail PRO_LIGAND's methodology for peptide design, the design of novel molecules from pharmacophore models and 3D QSAR information and the genetic algorithm-based structure refinement module [41], as well as the application of PRO_LIGAND to the design of DNA-binding drugs.

Acknowledgements

We thank Richard Sykes for the development of the PROMETHEUS EYE package which has facilitated and augmented the development and presentation of PRO_LIGAND.

References

1 Walkinshaw, M.D., Med. Res. Rev., 12 (1992) 317.
2 Fesik, S.W., J. Biomol. NMR, 3 (1993) 261.
3 Navia, M.A. and Murcko, M.A., Curr. Opin. Struct. Biol., 2 (1992) 202.
4 Kuntz, I.D., Science, 257 (1992) 1078.
5 Bugg, C.E., Carson, W.M. and Montgomery, J.A., Sci. Am., 269 (1993) 60.
6 Ealick, S.E. and Armstrong, S.R., Curr. Opin. Struct. Biol., 3 (1993) 861.
7 Reich, S.H. and Webber, S.E., Perspect. Drug Discov. Design, 1 (1993) 371.
8 Greer, J., Erickson, J.W., Baldwin, J.J. and Varney, M.D., J. Med. Chem., 37 (1994) 1035.
9 Verlinde, C.L.M.J. and Hol, W.G.J., Structure, 2 (1994) 577.
10 Blundell, T.L., Sibanda, B.L., Sternberg, M.J.E. and Thornton, J.M., Nature, 326 (1987) 347.
11 Mayer, D., Naylor, C.B., Motoc, I. and Marshall, G.R., J. Comput.-Aided Mol. Design, 1 (1987) 3.
12 Sheridan, R.P., Nilakantan, R., Dixon, J.S. and Venkataraghavan, R., J. Med. Chem., 29 (1986) 899.
13 Martin, Y.C., Bures, M.G., Danaher, E.A., DeLazzar, J., Lico, I. and Pavlik, P.A., J. Comput.-Aided Mol. Design, 7 (1993) 83.
14 Cramer, R.D., Patterson, D.E. and Bunce, J.D., J. Am. Chem. Soc., 10 (1988) 5959.
15 Moon, J.B. and Howe, W.J., Protein Struct. Funct. Genet., 11 (1991) 314.
16 Moon, J.B. and Howe, W.J., In Wermuth, C.G. (Ed.) Trends in QSAR and Molecular Modelling 92 (Proceedings of the 9th European Symposium on Structure–Activity Relationships: QSAR and Molecular Modelling), ESCOM, Leiden, 1993, pp. 11–19.
17 Miranker, A. and Karplus, M., Protein Struct. Funct. Genet., 11 (1991) 29.
18 Caflisch, A., Miranker, A. and Karplus, M., J. Med. Chem., 36 (1993) 2142.
19 Nishibata, Y. and Itai, A., Tetrahedron, 47 (1991) 8985.
20 Nishibata, Y. and Itai, A., J. Med. Chem., 36 (1993) 2921.
21 Böhm, H.-J., J. Comput.-Aided Mol. Design, 6 (1992) 61.
22 Böhm, H.-J., J. Comput.-Aided Mol. Design, 6 (1992) 593.
23 Böhm, H.-J., In Kubinyi, H. (Ed.) 3D QSAR in Drug Design: Theory, Methods and Applications, ESCOM, Leiden, 1993, pp. 386–405.
24 Lewis, R.A., Roe, D.C., Huang, C., Ferrin, T.E., Langridge, R. and Kuntz, I.D., J. Mol. Graph., 10 (1992) 66.
25 Rotstein, S.H. and Murcko, M.A., J. Comput.-Aided Mol. Design, 7 (1993) 23.
26 Rotstein, S.H. and Murcko, M.A., J. Med. Chem., 36 (1993) 1700.
27 Gillet, V.J., Johnson, A.P., Mata, P., Sike, S. and Williams, P., J. Comput.-Aided Mol. Design, 7 (1993) 127.
28 Gillet, V.J., Newell, W., Mata, P., Myatt, G., Sike, S., Zsoldos, Z. and Johnson, A.P., J. Chem. Inf. Comput. Sci., 34 (1994) 207.
29 Pearlman, D.A. and Murcko, M.A., J. Comput. Chem., 14 (1993) 1184.
30 Tschinke, V. and Cohen, N.C., J. Med. Chem., 36 (1993) 3863.
31 Ho, C.W.M. and Marshall, G.R., J. Comput.-Aided Mol. Design, 7 (1993) 623.
32 Leach, A.R. and Lewis, R.A., J. Comput. Chem., 15 (1994) 233.
33 Leach, A.R. and Kilvington, S.R., J. Comput.-Aided Mol. Design, 8 (1994) 283.
34 Eisen, M.B., Wiley, D.C., Karplus, M. and Hubbard, R.E., Protein Struct. Funct. Genet., 19 (1994) 199.
35 Ball, J., Fishleigh, R.V., Greaney, P., Li, J., Marsden, A., Platt, E., Pool, J.L. and Robson, B., In Bawden, D. and Mitchell, E.M. (Eds.) Chemical Structure Information Systems: Beyond the Structure Diagram, Ellis Horwood, Chichester, 1990, pp. 107–123.
36 Robson, B., Ball, J., Fishleigh, R.V., Greaney, P., Li, J., Marsden, A., Platt, E. and Pool, J.L., Biochem. Soc. Symp., 57 (1991) 91.
37 Frenkel, D., Clark, D.E., Li, J., Murray, C.W., Robson, B., Waszkowycz, B. and Westhead, D.R., J. Comput.-Aided Mol. Design, submitted for publication.
38 Waszkowycz, B., Clark, D.E., Frenkel, D., Li, J., Murray, C.W., Robson, B. and Westhead, D.R., J. Med. Chem., 37 (1994) 3994.
39 Klebe, G., J. Mol. Biol., 237 (1994) 212.
40 Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

41 Westhead, D.R., Clark, D.E., Frenkel, D., Li, J., Murray, C.W., Robson, B. and Waszkowycz, B., J. Comput.-Aided Mol. Design, 9 (1995) in press.
42 Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R. and Ferrin, T.E., J. Mol. Biol., 161 (1982) 269.
43 Weiner, S.J., Kollman, P.A., Case, D.A., Singh, U.C., Ghio, C., Alagona, G., Profeta Jr., S. and Weiner, P., J. Am. Chem. Soc., 106 (1984) 765.
44 Weiner, S.J., Kollman, P.A., Nguyen, D.T. and Case, D.A., J. Comput. Chem., 7 (1986) 230.
45 Robson, B. and Platt, E., J. Mol. Biol., 188 (1986) 258.
46 Morley, S.D., Abraham, R.J., Haworth, I.S., Jackson, D.E., Saunders, M.R. and Vinter, J.G., J. Comput.-Aided Mol. Design, 5 (1991) 475.
47 Bartlett, P.A., Shea, G.T., Telfer, S.J. and Waterman, S., In Roberts, S.M., Ley, S.V. and Campbell, M.M. (Eds.) Chemical and Biological Problems in Molecular Recognition, Royal Society of Chemistry, Cambridge, 1989, pp. 182–196.
48 Lauri, G. and Bartlett, P.A., J. Comput.-Aided Mol. Design, 8 (1994) 51.
49 Ullmann, J.R., J. Assoc. Comput. Machinery, 23 (1976) 31.
50 Brint, A.T. and Willett, P., J. Mol. Graph., 5 (1987) 49.
51 Golender, V.E. and Vorpagel, E.R., In Kubinyi, H. (Ed.) 3D QSAR in Drug Design: Theory, Methods and Applications, ESCOM, Leiden, 1993, pp. 137–149.
52 Bron, C. and Kerbosch, J., Commun. Assoc. Comput. Machinery, 16 (1973) 575.
53 Brint, A.T. and Willett, P., J. Chem. Inf. Comput. Sci., 27 (1987) 152.
54 Ho, C.W.M. and Marshall, G.R., J. Comput.-Aided Mol. Design, 7 (1993) 3.
55 Kuhl, F.S., Crippen, G.M. and Friesen, D.K., J. Comput. Chem., 5 (1984) 24.
56 Smellie, A.S., Crippen, G.M. and Richards, W.G., J. Chem. Inf. Comput. Sci., 31 (1991) 386.
57 Grindley, H.M., Artymiuk, P.J., Rice, D.W. and Willett, P., J. Mol. Biol., 229 (1993) 707.
58 Jakes, S.E. and Willett, P., J. Mol. Graph., 4 (1986) 12.
59 Willett, P., Similarity and Clustering in Chemical Information Systems, Research Studies Press, Letchworth, 1987.
60 Bawden, D., In Warr, W.A. (Ed.) Chemical Structures 2: The International Language of Chemistry, Springer, Heidelberg, 1993, pp. 383–388.
61 Martin, Y.C., Bures, M.G. and Willett, P., In Lipkowitz, K.B. and Boyd, D.B. (Eds.) Reviews in Computational Chemistry, Vol. 1, VCH, New York, NY, 1990, pp. 213–263.
62 Moock, T.E., Henry, D.R., Ozkabak, A.G. and Alamgir, M., J. Chem. Inf. Comput. Sci., 34 (1994) 184.
63 Clark, D.E., Jones, G., Willett, P., Kenny, P.W. and Glen, R.C., J. Chem. Inf. Comput. Sci., 34 (1994) 197.
64 Kuyper, L.F., In Perun, T.J. and Propst, C.L. (Eds.) Computer-Aided Drug Design, Marcel Dekker, New York, NY, 1989, pp. 327–369.
65 Ramnarayan, K., Hausheer, F.H. and Singh, U.C., CDA News, 8 (1993) 18.
66 Bolin, J.T., Filman, D.J., Matthews, D.A., Hamlin, R.C. and Kraut, J., J. Biol. Chem., 257 (1982) 13650.
67 Kohl, N.E., Emini, E.A., Schlief, W.A., David, L.J., Heimbach, J.C., Dixon, R.A.F., Scolnick, E.M. and Sigal, I.S., Proc. Natl. Acad. Sci. USA, 85 (1988) 4686.
68 McQuade, T.J., Tomaselli, A.G., Liu, L., Karacostas, V., Moss, B., Sawyer, T.K., Heinrikson, R.L. and Tarpley, W.G., Science, 247 (1990) 454.
69 Appelt, K., Perspect. Drug Discov. Design, 1 (1993) 23.
70 Fitzgerald, P.M.D., Curr. Opin. Struct. Biol., 3 (1993) 868.
71 Redshaw, S., Exp. Opin. Invest. Drugs, 3 (1994) 273.
72 Fitzgerald, P.M.D., McKeever, B.M., VanMiddlesworth, J.F., Springer, J.P., Heimbach, J.C., Leu, C.-T., Herber, W.K., Dixon, R.A.F. and Darke, P.L., J. Biol. Chem., 265 (1990) 14209.

Appendix 1

Rule bases for design-model generation

The flow of the Design-model Generation module is illustrated in Figs. 19 and 20. When operating in similar-design mode, the Design-model Generation module generates interaction sites by the processes of atom-type conversion and atom copying. While type conversion simply changes atom types of the design-base atoms, atom copying adds new atoms to the design-base list with identical attributes except that the new atoms have different atom types. Using atom copying it is possible to create more than one interaction site at the position of a given design-base atom.

The main rule for atom type conversion is expressed in the rule-base file as follows:

ATOM_RULE generic_type db_type(s)

which reduces a list of design-base atom types to a single generic type. For instance

ATOM_RULE R CA CB CW C* CN CD CE CF CG CP

reduces all the AMBER aromatic carbon atom types to type R.

The following auxiliary rules control atom-type conversion:

ATOM_BOND db_type1 db_type2(s)

which specifies that an atom of type db_type1 should only be retyped by an **ATOM_RULE** rule if it is bonded to an atom of type db_type2. Replacing the **ATOM_BOND** by **ATOM_NOT_BOND** ensures that the retyping only occurs if an atom of type db_type1 is *not* bonded to an atom of type db_type2.

ATOM_ENV db_type db_residue_label(s)

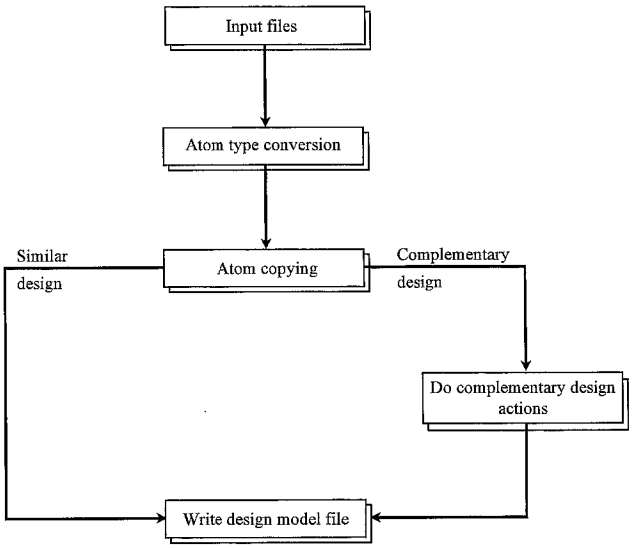


Fig. 19. Overview of the operation of the Design-model Generation module.

which insists that an atom of type db_type should only be retyped if it is part of a residue defined by db_residue_label. For instance, **ATOM_ENV** NB HIS says that the AMBER type NB should only be retyped if it is part of a histidine residue.

The **ATOM_BOND** and **ATOM_NOT_BOND** rules allow nearest neighbours to influence the type conversion process. The **ATOM_ENV** rule allows wider environmental influence. However, for some applications, a direct influence of next-nearest neighbours is required for the atom-type conversions. If this is the case, the user is provided with the option of two passes of atom-type conversion. This option is chosen with the **TWO_PASS_TYPE_CONV** keyword in the rule-base file.

The rule for atom copying has the format **ATOM_COPY** db_type1 new_type(s) and creates copies of all atoms with db_type1 with types new_type(s).

When operating in similar-design mode, the module simply writes to the design-model file those members of the list of new atom types which were actually changed during atom-type conversion or created by atom copying. Atoms which retain their original design-base atom type are ignored.

When operating in complementary-design mode the module again uses atom-type conversion and copying as the first step. This is to allow the user to manipulate the design-base atom types before the complementary interaction sites are generated. This can be useful when a number of distinct atom types can be considered equivalent as far as interaction-site generation is concerned (an example here might be the hydrogen atoms in OH and NH groups). After this, the module uses 'complementary rules' to generate interaction sites.

The flow of the Design-model Generation module for complementary design after the atom-typing phase is illustrated in Fig. 20. The outer loop runs over all the atoms in the design base, and the inner one over all the complementary rules in the rule base. The module takes the design-base atoms one by one and then goes through the rule base. Each time a rule is found which applies to the current atom type, interaction sites are generated according to instructions in the rule.

There are two types of complementary rule. The first is called a 'linear rule'. Here, the position of the site is defined by specifying a bond length, a valence angle and a torsion angle from the atom giving rise to the site, as illustrated in Fig. 21. By default, the second atom (parent) required to define the valence angle, and the third atom (grandparent) required to define the torsion angle, are selected randomly by the algorithm from atoms with appropriate connections. It is possible, however, to define specific atom types for which the algorithm should search when assigning parents and grandparents using the **RULE_PARENT** and **RULE_GRANDPARENT** keywords.

The second type of complementary rule recognised by the module is termed a 'ring rule'. Again, the position of the site is given by a bond length, a valence angle and a torsion angle from the atom giving rise to the site. However, in this case the valence angle is measured from a dummy atom which is generated on the perpendicular bisector drawn from the atom giving rise to the site to the opposite side of the triangle formed by this atom and two atoms connected to it, as illustrated in Fig. 22. The two 'parent' atoms are currently randomly selected by the module from those atoms with appropriate connections. In most situations the atom in question is connected to only two atoms (for instance the unprotonated nitrogen in histidine) and the random selection makes no difference.

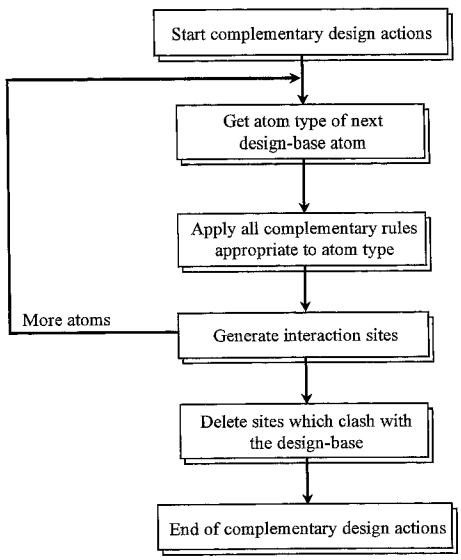


Fig. 20. Operation of the Design-model Generation module in complementary-design mode.

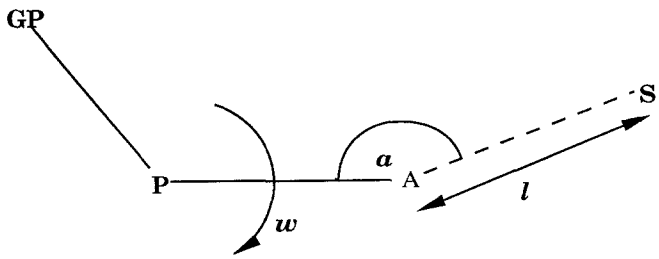


Fig. 21. Positioning of interaction sites by a linear rule. The site **S** is placed with respect to the atom **A**, parent atom **P** and grandparent atom **GP** at a bond length l, a valence angle a and a torsion angle w.

When using either of the above complementary rule types, it is possible to put sites in specific positions, or to populate ranges of bond length, valence and torsion angles with evenly spaced sites.

The following keyword entries in the rule-base file define the complementary rules:

CMP_LIN_RULE atom_type site_type bndl bndu sbnd vall valu sval torl toru stor (site_type bndl bndu sbnd vall ...)

This rule generates sites for the 'linear' type rule defined above and it means that for each atom in the design base typed to atom_type the algorithm should generate a site of type site_type, using parent and grandparent atoms which are derived as explained above. The records bndl and bndu give lower and upper limits for the distance between the site and the atom giving rise to it. Sites are then generated by the algorithm at spacing sbnd within this range of distances. If sbnd is greater than bndl – bndu then two sites are generated at the extrema of the range. If sbnd = 0.0 or bndl = bndu then one site is generated at bndl. All distances are in Å. The records vall, valu and sval work in the same way, placing sites in the range of valence angles between vall and valu; and similarly torl, toru and stor define the torsional position and spacing of sites. It is worth noting at this point that, for the angular records, upper and lower limits are in degrees, but the spacing of the sites is still in Å. The algorithm internally generates appropriate angle increments to give sites which are approximately evenly spaced on a spherical surface of radius bndl at the user-defined spacing value.

As an example of this rule

CMP_LIN_RULE LAL L 4.0 4.0 0.0 0.0 180.0 2.0 0.0 360.0 2.0

places sites of type L on the surface of a sphere of radius 4.0 Å at a spacing of 2.0 Å around all atoms of type LAL.

Some interaction sites (e.g., the D-X hydrogen bond donor sites) require the placing of two points. This can be done by simply stating the second site type and its position with respect to the first site directly after the specification of the first point. Now the parent atom becomes the original design-base atom and the grandparent atom the original parent atom. For example

CMP_LIN_RULE OCB D 1.9 1.9 0.0 110.0 180.0 1.0 0.0 360.0 1.0 X 1.0 1.0 0.0 180.0 180.0 0.0 0.0 0.0 0.0

places sites of type D 1.9 Å from atoms of type OCB in the range of valence angles 110°–180° and in the full 0°–360° range of torsions. For each of these sites a further site of type X is 1.0 Å from the D site, so that OCB-D-X is linear.

The 'ring' type rule is specified using the following keyword:

CMP_RING_RULE ludi_type site_type bndl bndu sbnd vall valu sval torl toru stor (site_type bndl bndu sbnd vall...)

This rule works in the same way as the **CMP_LIN_RULE**, except that the 'parent' and 'grandparent' atoms are chosen as for a 'ring' rule, as defined in Fig. 22.

When in complementary mode, the final step is deletion of the interaction sites which make van der Waals clashes with the receptor. This requires specification of radii for all interaction sites and atoms in the source molecule, which is done using the parameter file specified by the **ATOM_RAD_FILE** keyword in the input file. This parameter file contains the following keywords:

ATOM_RAD_INDX integer_index atom_type(s)

which attaches an index to a set of atom types, and

ATOM_RAD_VAL integer_index real_radius

which associates a radius with each of the atom types previously assigned the integer_index.

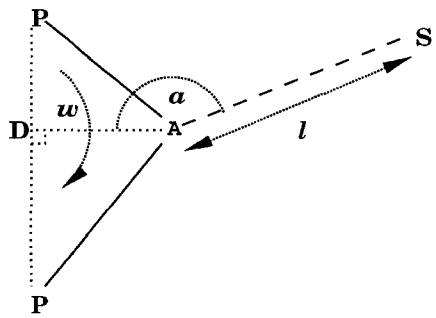


Fig. 22. Positioning of interaction sites by a ring rule. The site **S** is placed at a bond length l, a valence angle a and a torsion angle w with respect to the atom **A**, a dummy atom **D** and one of the parent atoms **P**.

Appendix 2

A generalised driver for fragment-based structure generation

The driver is designed to allow as much flexibility as possible in the building process. In the following, the strings in bold capitals represent keywords to which the user assigns values in the input file to structure generation. For example, **NFRAG_PLACE** may be assigned the value of 10 and **PLACE_BRIDGE_LIBRARY** may be assigned the value '/usr/people/pro_ligand/pb_lib/aro_polar1 1', where '1' indicates the rank assigned to the specified library. A diagram of the loop structure of the driver is given in Fig. 23.

The loops labelled a, b, c and d represent different modes of building. The first loop, a, places fragments onto the design-model interaction sites. Up to **NFRAG_PLACE** fragments are placed every time this loop is traversed. The algorithm searches through all the **PLACE_LIBRARY**s in the order of their respective ranks until it has exhausted the libraries or exceeded **NFRAG_PLACE**. The second loop, b, places fragments onto design-model interaction sites whilst also bridging between already placed fragments. The algorithm searches through the **PLACE_BRIDGE_LIBRARY**s, each with its associated rank, until it has exhausted the libraries or fitted **NFRAG_PLACE_BRIDGE** fragments. Additional flexibility can be obtained by using purely bridging libraries instead of true bridge-place libraries during this phase of structure generation. The c loop places fragments onto interaction sites whilst joining the new fragment onto already placed fragments. As above, the **PLACE_JOIN_LIBRARY**s are searched through in order of rank until **NFRAG_PLACE_JOIN** fragments have been fitted. The d loop represents a final bridging phase and has similar options associated with it, although they are less useful in this case.

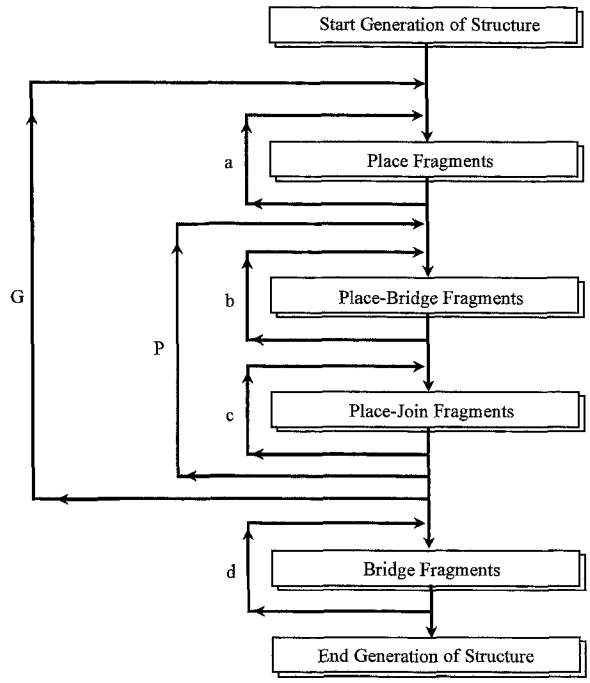


Fig. 23. Program loop structure of generalised driver for structure generation.

The G loop is traversed up to **MAXIT_GRAND_LOOP** times. This allows one to go through a restricted place-join and/or bridge-join phase before placing more fragments. The P loop is traversed up to **MAXIT_PETIT_LOOP** times. It allows one (amongst other things) to exhaust the place-joining and place-bridging phase before resorting to a pure placement phase.

By specifying a GENERAL strategy, the user is free to choose the values for each of the keywords mentioned above. In this way, a large variety of types of building are available. Alternatively, a selection of specific strategies have been 'hard-wired' for easy use. For instance, if the LUDI mode of building is selected, the following sequence of commands will be invoked, producing an exhaustive placement of fragments followed by a final bridging phase.

NFRAG_PLACE 1
NFRAG_BRIDGE_JOIN 0
NFRAG_PLACE_JOIN 0
NFRAG_BRIDGE 1000
MAXIT_GRAND_LOOP 1000
MAXIT_PETIT_LOOP 0

Alternatively, a GROW strategy may be invoked which will trigger the following commands and result in a sequential 'build-up' approach to structure generation:

NFRAG_PLACE 1
NFRAG_BRIDGE_JOIN 0
NFRAG_PLACE_JOIN 1
NFRAG_BRIDGE 0
MAXIT_GRAND_LOOP 1
MAXIT_PETIT_LOOP 1000