Journal of Computer-Aided Molecular Design, 9 (1995) 269–282
ESCOM

J-CAMD 288

BUILDER v.2: Improving the chemistry of a de novo design strategy

Diana C. Roe and Irwin D. Kuntz*

Department of Pharmaceutical Chemistry, University of California, San Francisco, CA 94143-0446, U.S.A.

Received 12 September 1994
Accepted 14 November 1994

Keywords: Structure-based design; Fragment joining; Ligand design; Molecular modelling; Enzyme inhibitors

Summary

Significant improvements have been made to the de novo drug design program BUILDER. The BUILDER strategy is to find molecule templates that bind tightly to 'hot spots' in the target receptor, and then generate bridges to join these templates. In this paper, the bridging algorithm has been further developed to improve the chemical sense and diversity of the bridges, as well as the robustness of the technique. The improved algorithm is then applied to rebuild known bridges in methotrexate and HIV protease. Finally, the entire BUILDER approach is tested by rebuilding methotrexate de novo.

Introduction

Structure-based drug design is becoming an increasingly powerful approach for finding lead compounds [1–7]. The challenge is to find bioactive molecules with novel scaffolds that are suitable as lead drug compounds. Towards this purpose, the three-dimensional structure of the target receptor can be exploited to find molecules which complement its shape and functionality, and thus bind tightly. The structure is usually provided by X-ray crystallography, although homology-built structures [2] have also been used effectively. Once a lead compound is found, it is often possible to generate co-crystal structures of the lead compound with the target receptor, and to use the additional structural information this provides for further development. This cycle can be repeated until a potent potential drug is found [1,4]. Thus, the structure-based design strategy is a powerful tool in two stages of drug design, i.e., lead discovery and lead refinement.

Many programs have been developed to help at either stage of drug design. The current methodologies fall into two major categories: database searching and de novo design. The database searching protocols mainly follow a 'docking' strategy. Each compound in the database is placed in the target site in a variety of orientations, and each orientation is scored by some measure of the quality of fit. The best scoring orientation for each molecule is then saved and compared to all other molecules in the database. Finally, the overall top rankings are saved. Current docking programs include DOCK [8,9], CLIX [10], FLOG [11] and the work of Bacon and Moult [12].

One strong advantage of the database searching approach is that it can be run on a database of available compounds, and so the top results can be either retrieved from inventory or purchased, rather than synthesized, for testing. This is also a limitation, as it is confined to existing molecules. Another difficulty is that the conformational space of each potential ligand is not well examined (some procedures use only one conformation per compound), and thus compounds can be missed because the correct conformer was never tested. Methods have been developed to improve conformational sampling [11,13,14], but these are not yet practical for a large degree of sampling over an entire database.

In contrast to the 'docking' approach, which takes a compound and fits it into a receptor site, the de novo design approach builds a compound directly into the site. This addresses some of the weaknesses of database searching. The range of molecules that can be formed is limited only by the heuristics of the de novo design program, rather than by the set of existing molecules. Also, it allows searching over a larger conformational space, as the potential lead compound is built adaptively to best fit the site, and therefore is not limited to a set of pre-existing database conformations. Finally, de novo strategies can also be used to refine existing lead compounds, as

*To whom correspondence should be addressed.

well as to discover new lead compounds. Thus, there is significant potential for this approach as a tool in drug design.

De novo design approaches

The de novo design programs can be further divided into two approaches: sequential buildup and fragment connection. The first method starts with a seed atom or chemical group, and then adds atoms or groups in positions and orientations that interact well with the target receptor. Several existing programs employ this strategy: Legend [13,14] and Genstar [15], which grow structures one atom at a time; GrowMol [16], which uses single atoms and functional groups as building blocks; GROW [17], which builds peptides one amino acid at a time; and GroupBuild [18] and SPROUT [19,20], which use organic fragments to grow molecules.

In contrast, the fragment connection method starts by placing atoms or groups in the target site in 'hot spots' or places of strong interaction with the receptor. These fragments are then connected via a bridging group of atoms to form a composite molecule (see Fig. 1). The advantage of this approach over sequential buildup is that the resultant composite will involve all the key interaction sites, whereas the sequential approach always goes to places of immediate energetic advantage and may miss critical features that can only be reached by proceeding through areas of poor interaction [18]. The disadvantage is that there may not be chemically feasible bridging groups to join all the fragments in their most favored locations.

There are many ways to place functional groups or larger fragments in areas of strong interaction. One is to use Goodford's GRID [21,22] program to locate 'hot spots' for functional groups. Another is to use the 'HSITE' [23,24] program to find hydrogen bonding locations. The program LUDI [25,26] uses a rule-based system to define interaction site points. The programs MCSS [27] and Concepts [28] have been developed to place small functional groups into areas of good interaction, using molecular dynamics. Larger groups are found with FOUNDATION [29], which performs a search on a 3D database to find appropriate small molecules. Docking programs can also be used to place several smaller molecules in areas of strong interaction.

Once the 'hot spot' regions have been filled, the next step is to find bridging groups to join the atoms or groups. LUDI [25] and CAVEAT [30] both use a database searching technique with a special bridging group library for compounds whose ends match the fragments in end-to-end distances and angles. This approach is useful, but limited by the database. NEWLEAD [31] uses a similar technique, but it has a database of smaller bridging groups (which are labeled spacers) which can be combined (no more than three in a row) to form a bridging group. SPLICE [32] joins fragments with overlapping bonds, which works well when such overlaps exist. Leach [33] has developed an algorithm that generates chains and then uses the 'tweak' algorithm [34,35] to match the ends of the chains to the templates. Lewis et al. have developed several methods to generate connectors, including the use of spacer skeletons [36,37], a diamond lattice [38], and an irregular lattice, which was the original method in BUILDER v.1 [39].

BUILDER overview

The original BUILDER v.1 software consisted of an interactive program, run from within a graphics environment provided by MidasPlus [40]. It used DOCK to initially place molecules into the active site, prior to the BUILDER session. Within the interactive session, the user could specify a zone of interest, and would be shown a series of top-ranking DOCK molecules in that zone one at a time. Any interesting molecule or parts of a molecule could then be chosen to serve as a starting fragment. The user would specify which fragments to join, and BUILDER would give the user a series of bridges from which to choose. The bridges were generated by performing a breadth-first search on an irregular lattice, to find a path through the lattice that joined the two fragments. The irregular lattice was created from a series of top-

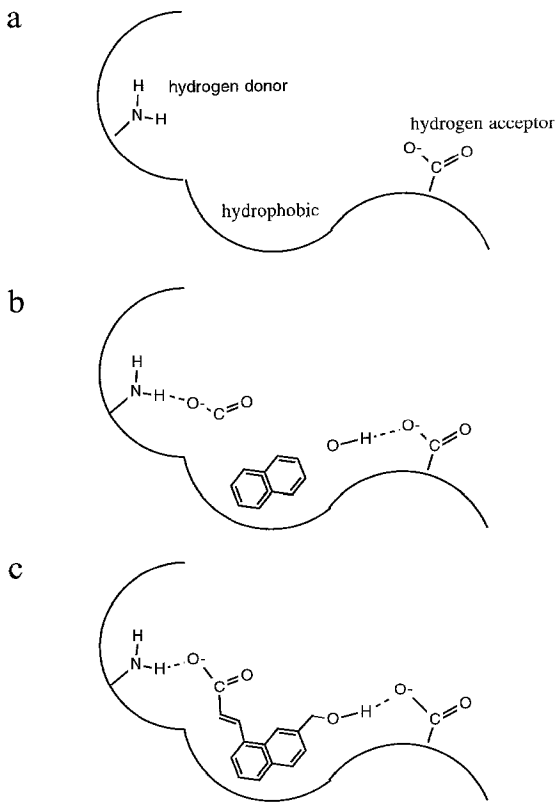

Fig. 1. Overview of BUILDER's design strategy. (a) Characterize zones of interest in the receptor. (b) Fill zones of interest with fragments. (c) Join fragments with bridging groups to form a composite molecule.

scoring docked molecules in the receptor site (around 300 molecules for a site). Pruning strategies were employed during the search to ensure that the path followed chemical rules (for example, that it had proper bond angles and torsion angles, and consisted of a sensible combination of atoms). The advantage of using such an algorithm to generate paths, as opposed to searching a bridge library, is twofold. First, it allows for more possibilities than searching a small database. Second, it allows for a better torsional sampling of bridge groups.

But how good are the resultant generated bridges? To answer this question, it is necessary to examine the desired properties of bridges. It is assumed that the fragments will provide most of the binding for the composite molecule. In contrast, the bridging groups are in areas not expected to contribute strongly to binding. Thus, the bridging groups should be chosen based on synthetic and geometric considerations for the resultant composite compound. These considerations can be seen in the following specific geometric and chemical goals for bridges.

Desired properties for bridging groups

The following are geometric goals for bridges. (i) Bridges should be short (2–5 bond lengths). If there is more distance between key interaction groups, it might be more advantageous to place another fragment at some point in-between the two to maximize the interaction energy, rather than forming a long connector with no apparent addition to the binding energy. (ii) A relatively large number of chemically distinct bridges is sought, so the chemist can choose amongst them for those satisfying certain synthetic or other requirements. (iii) The bridging groups should reflect a good sampling of torsional space. A 10° variation in torsion angle will not greatly raise the conformational energy of a proposed bridge, but it may determine whether that bridge can join two fragments. (iv) Finally, a scheme to generate rigid linkers is desirable. The more rigidity, the lower the conformational entropy loss upon binding. This must be balanced, however, with synthetic accessibility, as multiple ring systems may complicate the synthesis.

The goals for chemical properties are simpler. (i) The proposed bridges should be chemically sensible. They should consist of atom type combinations that exist in nature. For example, three oxygens in a row do not make chemical sense. (ii) They should have realistic bond lengths and angles for the atom types. (iii) There should be a chemically diverse set of bridges. This provides the chemist with a nice variety to choose from when designing a molecule.

How did the bridges from the original BUILDER v.1 compare with our stated geometric and chemical goals? They stand up to most of the geometric criteria – the linkers are short, many in number and represent a full sampling of torsional space. However, the bridges all consist of linear chains of atoms, which can be floppy. Therefore, an additional rigidification scheme would be desirable. By the chemical criteria, the BUILDER v.1 results had many shortcomings. The bridges were not required to make much chemical sense. There was no constraint on the angle the path had to make to the end fragments. Thus, the bond and dihedral angles from the path to the fragment could be very distorted. Also, the resultant bridges did not consist of specific atom types (i.e., an element combined with its hybridization state), but only of element types. Hybridization information was treated separately, resulting in quite loose tolerances. The bond lengths could be off by as much as 0.5 Å, since only element types were used to judge allowed bond lengths. Similarly, bond angles were chosen from a small set of angles (109.5°, 120° and 180°) with a 5° tolerance, permitting considerable sp^3, sp^2 and sp distortion. This also led to chemically unreasonable atom type combinations, as only the element type, and not its hybridization, was used to judge its feasibility. For example, since C-O-C is an allowed bridge when elements only are considered, a path in which the middle O was sp^2-hybridized was allowed. Thus, clearly there were many areas to improve the original approach.

Changes made in BUILDER v.2

In this paper we report a variety of changes to the original BUILDER approach to address the shortcomings mentioned above. These changes include: (i) modifications of the original breadth-first search code to improve the searching strategy; (ii) a new series of post-processing steps after the initial paths are found, to add specific atom types to each path, and to adjust bond lengths and angles accordingly; (iii) additional heuristics to prevent chemically unfeasible atom type combinations; (iv) a ring generator to help rigidify the results; (v) a modification in the lattice generation strategy, from using a lattice formed from DOCK molecules to a random lattice; and (vi) finally, many implementation improvements to facilitate the use of BUILDER.

Methodology

Overview

BUILDER v.2 uses the same general approach as in the original version for designing compounds inside a target receptor (see the BUILDER overview section for more details). The main change in BUILDER v.2 compared to v.1 is how the bridging groups are built. The BUILDER v.2 approach to building connectors can be described as a five-step process (see Fig. 2), starting with two fragments in the active site that are to be joined, where the user has specified which atom in each fragment is to be used in the connection. (i) Generate a random lattice in the active site. This step has to be performed

only once for each active site. The lattice can be reused for any new fragments to be joined later in that site. (ii) Find paths through the lattice that connect the two atoms in the fragments, via a breadth-first search. This is similar to the original breadth-first search, with one key change: for each point in a path, the element type is ignored, and instead the hybridization state is stored. The resulting paths are then termed 'generic paths', for they represent only information about hybridization states and do not specify any atom type. (iii) Generate specific atom types. Many specific atom type combinations are generated for each generic path, following heuristic chemical rules to reduce the number of combinations. The result of this step is a linear chain of specific atoms in specific states of hybridization. (iv) Use the SHAKE algorithm [41,42] to adjust bond lengths and angles. (v) Perform final adjustments, adding atoms, such as oxygens on carbonyl carbons, and rings if possible. The end result is a linker with specific atom types which is not necessarily linear.

Creation of the random lattice

While BUILDER v.2 can search any lattice or grid to join fragments, the original 'irregular lattice' had several problems, mainly due to the very uneven density that existed throughout. The lattice could be particularly sparse in some areas, often in large areas between key pockets, exactly where the connecting groups are needed to join key fragments in these pockets. We elected to explore a random lattice instead, to provide a more even distribution of points while avoiding the poor mapping of diverse molecular conformations that occurs with regularly spaced lattices.

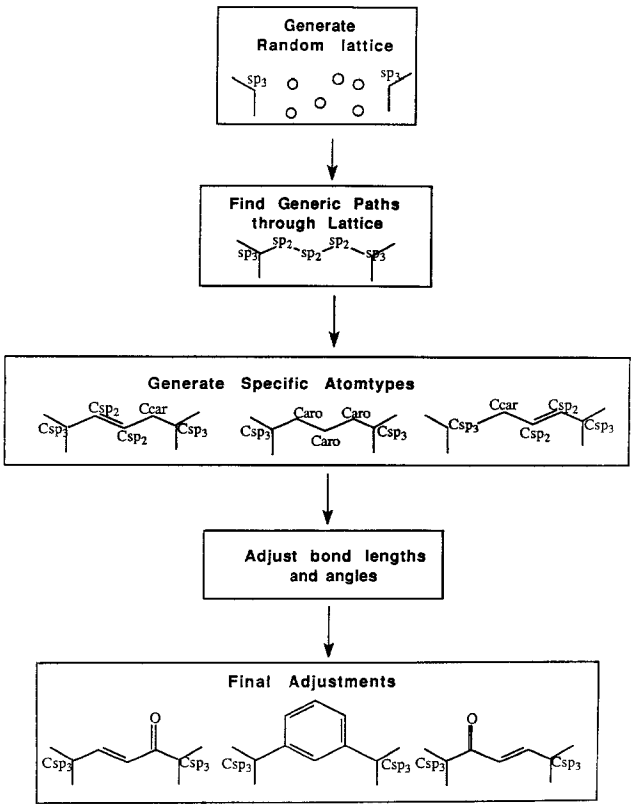

Fig. 2. Flow chart of the BUILDER strategy for generating bridging groups.

To create the lattice, the user specifies box dimensions for the proposed lattice location. Atoms are placed into the box using a pseudo-random number generator. The user also specifies the average spacing between the atoms (i.e., the density of atoms in the box). There is a trade-off here between the quality of the space sampling, which improves with a finer spacing, and speed, which improves with a wider spacing. In practice, we have found an average spacing of about 0.4 Å to be the best balance between these factors. Once positioned, each atom is then checked for intersections with the target receptor. This confines the breadth-first search to the allowed space within the pocket.

Generic path search

The algorithm for finding the generic paths follows the original breadth-first search algorithm, with the following changes: (i) as mentioned above, atom types are treated as generic; (ii) the hybridization states for each point in the path are examined in more depth; (iii) the handling of end points has been expanded to take into account the bond and torsion angles between the path atoms and the end fragments; (iv) finally, there have been several optimizations to improve search efficiency.

The main change in our treatment of hybridization is how the hybridization state for each atom in a path is determined. Once three atoms are found on a path, their bond angle can be used to define the hybridization of the middle atom. Originally, these hybridization states were stored by lattice point, so that once a lattice point was added to a path in a particular hybridization state, it could only be added to another path in that same state. Now the hybridizations themselves are stored with each path, so the choices for one path do not affect those for another. Along with this change, we allow a point in a path to have more than one hybridization state. This is especially important when the bond angle tolerances are loose, and the point in the path can fall into two (or more) possible hybridization states. In such cases, two (or more) paths are created to represent each hybridization state, rather than arbitrarily choosing one over the other. Finally, the pruning algorithms have been changed so as to handle the hybridizations better. Originally, the torsion test only examined atoms of the same hybridization state (i.e. sp^2-sp^2 or sp^3-sp^3). Now it can handle mixed hybridization definitions (i.e. sp^2-sp^3). In addition, a hybridization prune has been added to remove unwanted combinations (for example sp^3-sp-sp^3).

The third major change to the breadth-first search algorithm is in the handling of the fragment ends. The

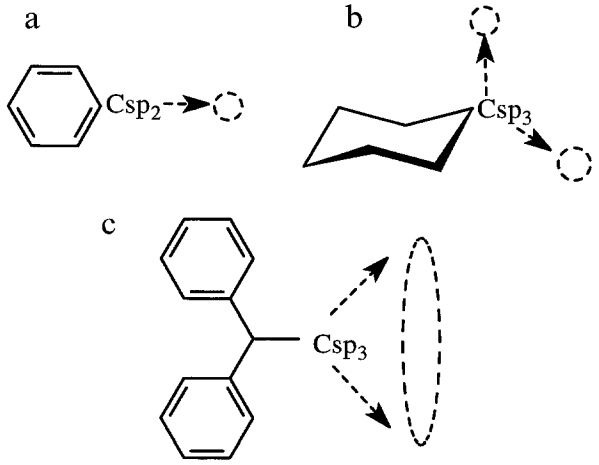

Fig. 3. In some cases the end points tightly define the positions for any connecting atoms. This is the case for (a) and (b), where only the circled positions are acceptable. In case (c) this location is not tightly defined.

angles and torsions defined by fragment ends are now used to prune the breadth-first search paths, to ensure that the paths approach the ends from chemically acceptable vectors. The angle and torsion tolerances for the end atoms are separated from the tolerances along the path, so the user can make them looser or tighter as desired.

The last set of changes to the breadth-first search algorithm involves optimizing its search performance. Often the geometry of a fragment will specify a unique location in space, or a choice of locations, for the first atom in any connection path from that fragment. For example, in Fig. 3a, when connecting from benzene, the first atom in the path must be near the circled location in order to pass the angle and torsion constraints of benzene. Similarly, in Fig. 3b, with cyclohexane it must be in one of the two circled locations. However, in Fig. 3c the location is not tightly defined and there is a torus of acceptable atom positions. In tightly defined cases such as Figs. 3a and 3b, a lattice point is generated into each of the circled regions depicted in Fig. 3, to ensure that at least one lattice point exists in these regions. We also use these locations in space to terminate searches, again as shown in Fig. 3. A search is essentially completed as soon as one of these points is found, as it is known that they are the only acceptable lattice points to the fragment end point. To continue past these points in a breadth-first search, all the paths of the next generation would have to be examined, which would be proportional to the square of the number of nodes in the current generation, increasing the total search time substantially. Therefore, to save time an option was added to move the end point of the search up from the fragment end to the set of acceptable lattice points.

Generating specific atom types

Once the generic paths are formed, a set of paths containing specific atom types must be generated for each path. This stage involves substituting all appropriate specific atom types for each generic atom, and using a 'GOODLIST' of allowed three-atom combinations to limit the number of possibilities to those which are chemically reasonable. The algorithm used to generate the specific atom types is a breadth-first search. The first atom in the generic path is set to the original atom type for that end point. Each subsequent atom in the generic path is replaced in turn by all atom types that could be bonded to the previous atom type. This is pruned to remove all atom types that do not have the correct hybridization for that point in the generic path. Each atom type is further pruned for chemical sense, by examining its combination with the previous two atom types, to see if that combination of three atom types is in the GOODLIST of allowed combinations. A separate GOODLIST is used for combinations involving end atoms, to allow more choices at the ends.

The GOODLIST can be established or modified by each user. For our tests, the GOODLIST consisted of a set of 18 different functional groups (see Fig. 4). No other combinations were allowed. In order to keep the resulting paths very simple, we further restricted the GOODLIST

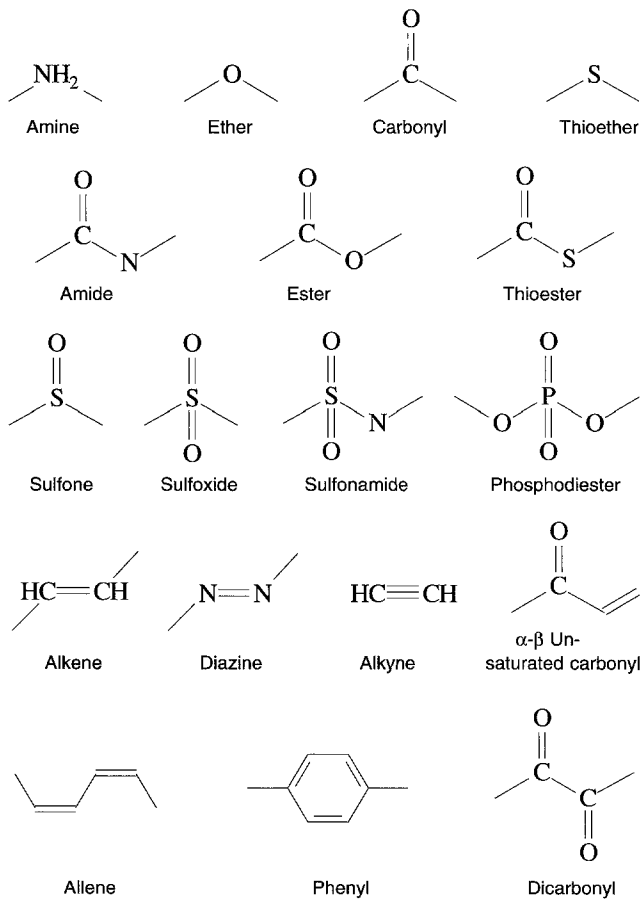

Fig. 4. Set of allowed functional groups.

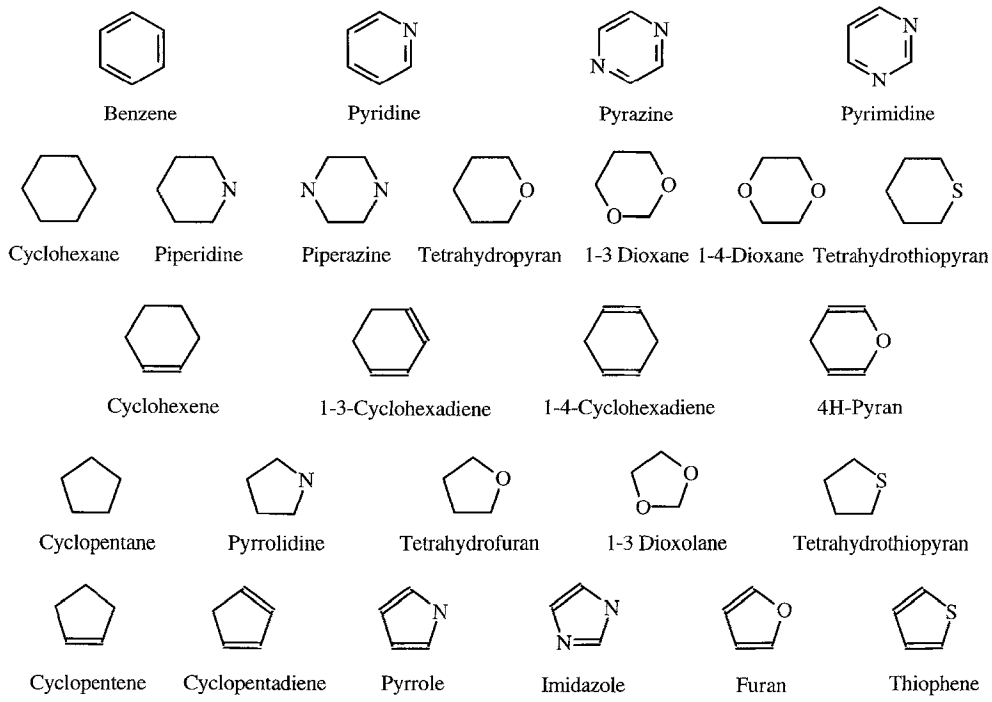

Fig. 5. Rings used in BUILDER test cases.

so that none of the defined functional groups could adjoin one another. They had to be separated by at least one sp^3 carbon. This restriction was not added to the GOODLIST defining the fragment ends, because the end points might not always be sp^3 carbons.

Adjusting bond lengths and angles

Now that specific atom types have been added, the bond lengths and bond angles are adjusted appropriately. The SHAKE algorithm [41,42], which was originally developed to constrain bond lengths and angles to within certain tolerances during molecular dynamics simulations, was used to adjust the bond lengths and angles of the paths. SHAKE is an iterative algorithm, which goes through a set of distance constraints for bond lengths and angles and adjusts them one at a time. It continues until all distance residuals are less than a user-set tolerance (shake_tol), or until a maximum number of iterations have been tried. We use a shake_tol of 0.02 and a maximum number of iterations of 500. If a path cannot be 'shaken down', it is removed, although this rarely happens.

In practice, SHAKE works very well at adjusting bond lengths and angles, correcting to within 0.1 Å of ideal for bond lengths and to within 3° of ideal for bond angles. The only exception is the presence of 180° angles, which are not well corrected. For this case, we use a second angle correction routine, where the angles are straightened in a series of rotations so as not to change already corrected bond lengths. The end result is a set of paths with very accurate bond lengths and bond angles. Since SHAKE may have altered the torsion angles of the paths, a torsion screen is performed at this time, and those outside the accepted range are removed. Similarly, since SHAKE may have moved the end points of the linker slightly, the resulting linkers are quickly translated back to the original end points. In cases where the vector from the end point is tightly defined (see Fig. 3), the Kabsch algorithm [43,44] is used to translate both the end points and end vectors onto the original end points and vectors.

Final adjustments

Final adjustments are made to turn the paths into bridging groups. First, certain atoms need to be added. For the functional groups we used, all carbonyl carbons, sulfone, sulfoxide, and phosphate atoms need sp^2-oxygens added to them. Then the paths need to be examined to

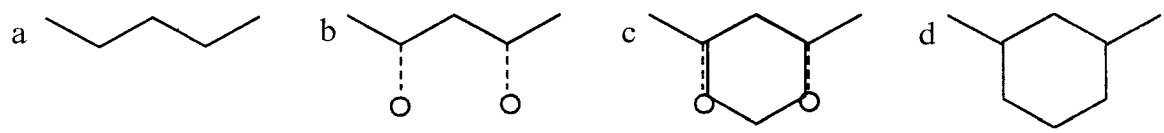

Fig. 6. Steps involved in generating rings onto a bridge. (a) The initial bridge. (b) Addition of dummy atoms. (c) Matching of bridge and dummy atoms to a ring. (d) Superimposition of the ring onto the bridge.

TABLE 1
ANGLE AND TORSION PARAMETERS

Hybridization	Ideal angle (°)	Minus tolerance (°)	Plus tolerance (°)
Full sampling[a]			
Angles			
sp	180	20	20
sp^2	120	10	10
sp^3	109.5	9	11
End angles			
sp	180	15	15
sp^2	120	10	10
sp^3	109.5	9	11
Narrow sampling[a]			
Angles			
sp	180	12	12
sp^2	120	5	5
sp^3	109.5	7	7
End angles			
sp	180	10	10
sp^2	120	5	5
sp^3	109.5	7	7
Torsions			
sp-sp	0.0	180.0	180.0
sp^2-sp^2	0.0	12.0	12.0
	180	12.0	12.0
sp^3-sp^3	60.0	15.0	15.0
	180.0	15.0	15.0
	300.0	15.0	15.0

[a] Narrow and full sampling refer to two different choices for parameter sets. See text for further discussion.

see if any portions can be embedded into rings. Next, a bump check against the target receptor is performed. The original lattice is constructed within the context of the receptor, so the original paths will not bump, but the subsequent angle adjustment may have moved an atom sufficiently to bump against the receptor. Also, ring atoms and other added atoms may bump against the receptor. Finally, an intramolecular clash check is performed on the bridge. To do this, the van der Waals radii used for the atoms are reduced from a value corresponding to the lowest energy in the Lennard-Jones potential, to the value corresponding to zero energy. A user-defined tolerance is also added to allow some flexibility, although we usually set this to a very low value (0.1 Å).

The ring-generation algorithm currently used by BUILDER v.2 is straightforward. More sophisticated algorithms [45] have been developed that can generate complicated multi-ring systems. However, since the goal of BUILDER is to try to keep the connecting groups reasonably accessible to synthesis, a simpler plan was employed. For each connector, each ring in a user-defined list is tested to see if it can be embedded into that connector. The ring list we use is shown in Fig. 5. If the atom types in the connector match the atom types in the ring, an attempt is made to match the ring to the connector using the Kabsch algorithm [43,44]. Two additional atoms are generated from the connector to help in this matching (see Fig. 6). If the rms of the overlapping ring and linker atoms is below a certain tolerance, then the matching is considered successful and the ring is added. Fused rings are handled in this algorithm by initially including them in the user-defined list. If more than one ring can match a path, both possibilities are generated and are considered as separate bridging groups.

The final bridging groups are screened with several user-defined values. The first is enddist_tol, which is the rmsd of the end points from the original. The second is endang_tol, i.e., the error in the angle from the bridge atoms to the end points. Finally, the algorithm can lead to several bridges with the same atom type combinations. Each of these represents a slightly different conformation. Some conformations are closer than others, and the user can specify an rmsd tolerance (unique_tol) at which two conformations are considered to be identical. In these cases, the bridge whose end points are nearest the original is retained.

Testing method

Before testing this method for building bridging groups, it is important to define the factors that are used to judge the results. The first is CPU time. Since BUILDER is an interactive program, it is important that the results for each stage are calculated in a reasonable amount of time, on the order of seconds to no more than a few minutes of CPU time. The second criterion is the quality of the bridging groups. This can be examined by looking at the bond lengths, angles and torsions in the bridges and seeing how they compare to ideal values. In addition, in cases where BUILDER v.2 is rebuilding a known bridge, the rms fit to that bridge can also be used to judge quality. Another factor is consistency. How dependent is BUILDER v.2 on the initial placement of the lattice – does moving the lattice change the results? The final factor is the number of bridges produced. As mentioned before, it is important that the user has a reasonably diverse set of bridges to choose from. We examined these factors in the following set of tests.

TABLE 2
PARAMETER DEFAULTS

Parameter name	Default value
Constraint ellipse	0.0, 2.10
Enddist_tol	0.15
Endang_tol	5.0
Shake_tol	0.02
Unique_tol	1.0
Clash_tol	0.2
Ring_tol	0.15

TABLE 3
RESULTS OF REBUILDING 18 DIFFERENT FUNCTIONAL GROUPS

Link	Length of bridge	Best rmsd (Å)	CPU time (s)[b]	Final number of bridges
Carbonyl	4	0.11	19	33
Ester	4	0.16	11	28
Amide	4	0.13	7	14
Thioester	4	0.18	4	8
Dicarbonyl	4	0.14	8	36
Sulfone	4	0.33	10	16
Sulfoxide	4	0.44	8	18
Sulfonamide	4	0.17	19	2
Phosphodiester	4	0.19	6	12
Ether	4	0.18	6	33
Thioether	4	0.16	8	16
Amine	4	0.12	7	40
Alkyne	4	0.11	3	1
Diazine	4	0.12	5	10
Alkene	4	0.15	8	11
α,β Unsaturated carbonyl	4	0.13	7	15
Allene	5	0.14	381	121
Allene[a]	5	0.15	21	81
Benzene	5	0.25	329	58
Benzene[a]	5	0.38	25	24

[a] Redone with narrow sampling.
[b] CPU time on an IRIS Indigo R4000.

Robustness testing

The first test for the bridge-building algorithm was to see if BUILDER v.2 could reproduce known results. For this purpose, a set of known bridges were made from the 18 functional groups depicted in Fig. 3. Each of these bridges consisted of the functional group, with sp^3 carbons added to both sides until the total length of the bridge was at least four bond lengths (five for phenyl and allene groups). Then, BUILDER v.2 was given only the end points for each of these bridges, and it was tested to see if it could reproduce the original bridge. All cases were run with the full sampling parameters listed in Table 1, with the default parameters listed in Table 2, and with the add_ring function on. The results are shown in Table 3. Using the random lattice, BUILDER v.2 can reproduce all bridges with nearly ideal geometries. However, the full sampling is very time-consuming for the two longer bridges of five bond lengths. This time can be reduced by using the lower sampling value for angles. With this lower sampling, these bridges are rebuilt much faster. These results are shown at the bottom of Table 3.

The sampling tolerance is a trade-off between speed and consistency. The higher sampling is strongly overdetermined, meaning that there are many paths that lead to similar results. This leads to more consistency, and helps guarantee that if a path is possible, it will be found. A series of runs was performed to test the consistency of these parameters, by examining how shifting positions within the lattice would affect the results. The 18 bridges mentioned above were run on the lattice, where they were translated by 0.1 Å increments for a total of 0.5 Å in each direction ($5\times5\times5=125$ different positions were tried). With the wider sampling parameters, the original bridges were rebuilt in more than 99% of the tests. With even wider parameters, it is possible to ensure that they are rebuilt 100%, but this more than doubles the CPU time, for diminishing returns. The same test done with narrow parameters rebuilt the paths only 80% of the time. The fact that paths may be missed with this lower sampling is not a serious problem, as the main purpose of BUILDER is to generate a set of acceptable bridges. It is not necessary that all possible bridges in the system are reproduced.

This trade-off between speed and thoroughness is further seen in the results as the length of the bridge is varied. For this test, a series of bridges was constructed containing an amide functional group, and more sp^3 carbons were added sequentially to each end in order to increase the size of the bridge. The results are shown in Table 4. With the full sampling, the bridge is consistently reproduced, until a length of six where it exceeds the maximum allowed number of generic linkers and therefore does not finish. The time taken varies approximately with the square of the path length. The narrow sampling does not produce good results for the short paths. However, for the longer paths it does produce a variety of bridges in a reasonable amount of time, which is the main goal of BUILDER. As expected, since it does not fully sample space, it does not consistently reproduce the starting bridge. Which sampling values are better depends on the individual application and how thoroughly the user wishes to examine the possible bridges.

Another factor to consider is the quality of the bridges created. We measured the deviations in bond angles and torsion angles. The bond lengths are forced by SHAKE to values within 0.1 Å of ideal. The angles have a little more freedom, as they are indirectly fixed in SHAKE

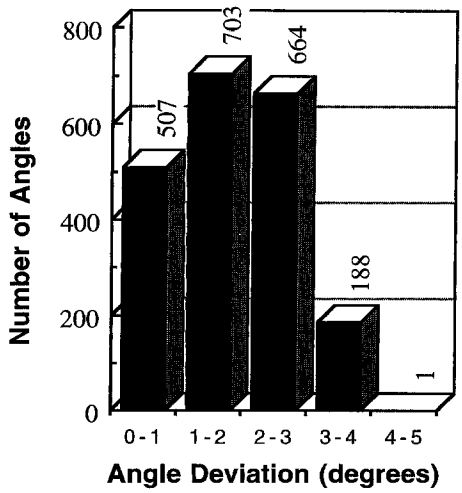

Fig. 7. Angle deviations from ideal for bridge atoms.

TABLE 4
RESULTS OF TESTING ON LENGTH: REBUILDING AMIDES OF VARYING LENGTH

Length of original link	Starting bridge reproduced	Rmsd (Å) from starting bridge	Number of nodes examined	Number of generic paths	Total resulting paths	Time for generic paths (s)	Total time (s)[a]
Full sampling							
2	Yes	0.03	12	7	2	<1	<1
3	Yes	0.03	140	27	8	<1	<1
4	Yes	0.07	11442	201	8	3	6
5	Yes	0.18	23162	8907	43	11	273
6	No	N/A[b]	–	>15000	–	–	–
Narrow sampling							
2	Yes	0.02	9	2	1	<1	<1
3	No	N/A[b]	2932	0	0	5	5
4	No	N/A[b]	3802	13	1	1	1
5	Yes	0.23	14228	360	17	6	13
6	Yes	0.14	23712	981	109	8	91

[a] CPU time on an IRIS Indigo R4000.
[b] Not applicable since the starting bridge was not reproduced.

through the end-to-end distances. The angles tested were taken from the sum of all the results in Table 3. The bond angles and torsion angles from the bridge to the end points were treated separately from the angles within the bridge. The results are shown in Figs. 7 and 8. The resulting angles are all within 4° from ideal for the bridge atoms, and mostly within 4° for the end angles. Similarly, the torsions span a range within the user-defined acceptable values. Thus, in terms of bond and torsion angles the bridges make chemical sense.

Comparison to previous methodology

To validate the new approach, we tested both programs with a series of problems. The first test compares the irregular lattice generated from DOCK hits to a randomly generated lattice. For this case, we started with the crystal structure of dihydrofolate reductase bound to methotrexate [46], using structure 4DFR from the Brookhaven Protein Data Bank (PDB) [47]. For the random lattice, a box was defined in the methotrexate binding site, and a lattice with an average spacing of 0.4 Å was generated using a pseudo-random number generator. A bump check was performed to remove any atoms in the space of the protein. For the irregular lattice, the original lattice of docked molecules described in Ref. 39 was used with some modifications. In order to make a fair comparison, all atoms outside the box used for the random lattice were removed from the irregular lattice. Atoms from docked molecules were then removed from the irregular lattice until both lattices had the same number of atoms. Thus both lattices filled the same space, and had the same average density, but different distributions of lattice points. Two bridging groups of methotrexate, link1 and link2 (shown in Fig. 9), were then rebuilt using each lattice. Originally we thought that the lattice made up of docked molecules would have a better distribution of points. However, as shown in Table 5, the random lattice performs as well, if not better. The linkers found by the random lattice had as good or better rmsd values, and the time used to find the linkers was shorter. A closer look at the comparison showed that the irregular lattice generated more generic paths. However, the fact that the overall rmsd was worse for the irregular lattice in link2 showed that many of these generic paths were redundant, without sampling the space as well as the random lattice paths. This is probably due to the uneven nature of the irregular lattice, being too dense in some places and too sparse in others.

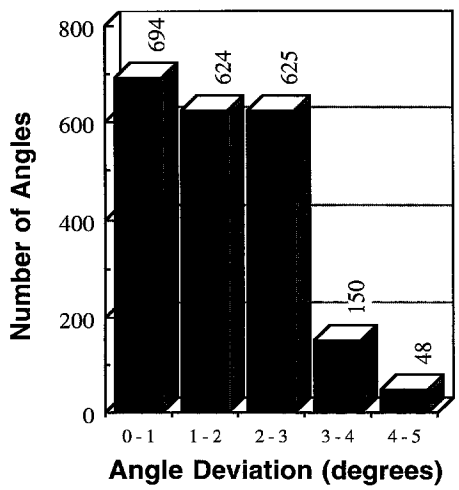

Fig. 8. Angle deviations from ideal for bridge end points.

As a further comparison of methodology, the original BUILDER v.1 was run on the test case described above. The results are summarized in Table 6. As can be seen, BUILDER v.1 is two to three times slower than v.2. This is due to optimization of the newer version. The original

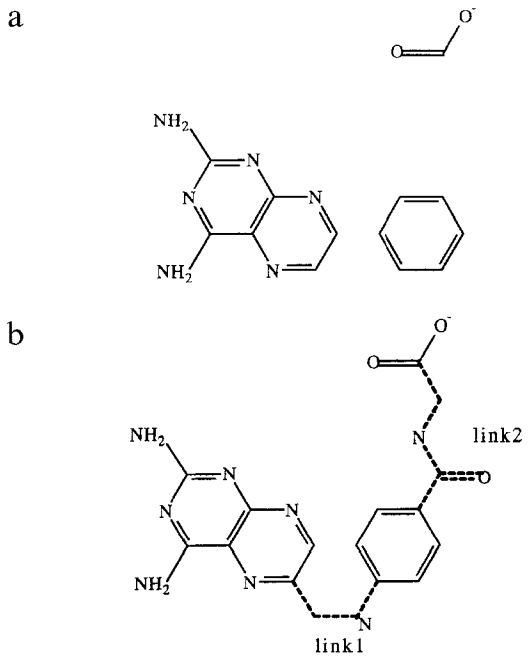

Fig. 9. Methotrexate test case. (a) Methotrexate fragments to be connected with BUILDER. (b) Bridges in methotrexate (link1 and link2) which are regenerated using BUILDER.

version generates more bridges, and all of these are shorter. However, the angles to the end fragments have serious errors. In fact, the program does not produce any paths that are within 10° of the correct angle to the end points. Also, the bridges from BUILDER v.2 are chemically more reasonable. Their bond lengths and angles are much closer to ideal. Also, they have acceptable atom type combinations, whereas two thirds of the bridges generated by BUILDER v.1 for link2 of methotrexate were nonsensical. Examining the resulting bridges, there is also more chemical variety in the new version, which occurs since each lattice point in v.2 represents all possible atom types, as opposed to only a single element type in the original version.

Finally, in comparing results from both versions, the new method is much easier to interpret. The original linkers are only chains of atoms for which the user has to examine the angles themselves to determine the hybridizations, and then check for chemical sense. In contrast, the new method provides this hybridization information, and the bridges are not necessarily linear as they may have carbonyl oxygens, sulfoxide, oxygens or rings added to them, which greatly improve visualization and comprehension of the chemistry represented by the linkers.

Applications

Rebuilding methotrexate

To further validate the new BUILDER v.2 methodology, tests were performed to see if it could rebuild existing linkers in known drug compounds. The first test was to rebuild methotrexate bound to dihydrofolate reductase [46], using 4DFR from the PDB [47]. Methotrexate was split into three key binding fragments, shown in Fig. 9. The first fragment is the pteridine ring, which has strong hydrogen bonds and electrostatic interactions to 4DFR. The second fragment is the phenyl group, which binds to a hydrophobic region in 4DFR. The third fragment is the α-carboxylate group, which is strongly hydrogen bonded to the receptor. Two bridges are required to join these fragments, as shown in Fig. 9. The results of running BUILDER v.2 are shown in Table 7. The program was able to reproduce the known bridges with very good rmsd values. In addition, it was also able to generate a number of acceptable alternative bridges.

Rebuilding HIV-1 protease inhibitor A74704

The second application is more challenging. The inhibitor A74704, bound to HIV-1 protease [48], was used from PDB entry 9HVP. It was split into five fragments, shown in Fig. 10a, identical to those used in NEWLEAD [31]. These fragments comprise the peptide side chains, which can be rejoined by the linkers shown in Fig. 10b, which comprise the backbone. Notice that link1 is required as a starting point for link2, link2 is required for link3 and link3 is required for link4. To reproduce the original inhibitor, these links were generated sequentially, that is, the atoms from each previous link were used as starting points for the next link. This is a test of the robustness of the method, as large rms deviations from the crystal structure could propagate down the set of links. It

TABLE 5
COMPARISON OF IRREGULAR VERSUS RANDOM LATTICE ON 4DFR[a]

	Lattice	CPU time (s)[b]	Number of nodes examined	Number of generic paths	Total resulting bridges	Best rmsd to known bridge
Link1	Random	1	1382	19	16	0.138
	Irregular	2	1183	59	16	0.138
Link2	Random	14	17194	147	24	0.148
	Irregular	108	17156	1775	38	0.216

[a] In order to emulate the actual bond angles to the end fragments and torsions found in methotrexate, an endang_tol of 8° was applied, and final_torsions was loosened to 30° for sp^2-sp^2 and sp^3-sp^3 torsions.
[b] CPU time on an IRIS Indigo R4000.

TABLE 6
COMPARISON OF ORIGINAL BUILDER VERSUS CURRENT BUILDER v.2 ON 4DFR[a]

	Program	CPU time (s)[b]	Total number of bridges	Total number of chemically reasonable bridges	Average angle deviation (°)	Average angle deviation at ends (°)	Average bond length deviation (Å)
Link1	Current	2	16	16	1.9	2.7	0.01
	Original	6	14	12	5.4	31.5	0.31
Link2	Current	14	24	24	2.6	5.1	0.01
	Original	22	537	173	8.6	30.1	0.36

[a] In order to emulate the actual bond angles to the end fragments and torsions found in methotrexate, an endang_tol of 8° was applied, and final_torsions was loosened to 30° for sp^2-sp^2 and sp^3-sp^3 torsions.
[b] CPU time on an IRIS Indigo R4000.

turned out, as can be seen in Table 7, that all the original links were reproduced with good rmsd values. Along with reproducing the peptide backbone, BUILDER v.2 was able to come up with a number of alternative backbone structures at each bridge.

Regenerating methotrexate from DOCKed fragments

As a final application, the entire BUILDER methodology was tested. The purpose of the test was to show that it is possible to generate methotrexate from a series of docked fragments using BUILDER. Since there are many user choices throughout the BUILDER process, it would be very difficult to prove that a particular molecule would or would not be designed. This test case is simply showing that it is within the large realm of molecules a user could design.

First, a DOCK run was performed on the Comprehensive Medicinal Chemistry Database (CMC) [49], which is a database of about 5000 medicinal compounds. In addition, a pteridine ring fragment was DOCKed into 4DFR. Next, the site was characterized and three zones of interest were defined. The first zone, which corresponds to the pteridine ring binding pocket, was identified using Goodford's GRID program with an amino probe. It was also identified with a carbon probe as being a good pocket for shape complementarity, as there are many possible van der Waals interactions. The second zone, which corresponds to the phenyl ring, was identified as a good hydrophobic binding pocket, both by visual inspection and by using the program HINT [50] to highlight the hydrophobic portions of the active site. The third zone, which corresponds to the α-carboxylate binding site, was identified as a good electrostatic binding site using GRID with a carboxylate probe.

Each of these zones was queried with the 'zone' command to find appropriate fragments. Again, it should be noted that since BUILDER is an interactive system, it would be difficult to prove that a user would choose the particular fragments that rebuild methotrexate. It can only be shown that these would be reasonable choices. By 'reasonable' we mean that the particular fragments chosen can be justified by our criteria. Since these fragments all came from top-ranking DOCK results, they all have good electrostatic and van der Waals interactions with the site. We decided amongst these top-scoring molecular fragments on the basis of how well they interacted with the crucial characteristics we identified for each zone of interest.

In the first zone, two of the nitrogens of the pteridine ring make very strong electrostatic interactions in two amino probe minima identified by GRID. The pteridine ring also has very good van der Waals contacts with the site. Thus, the entire pteridine moiety makes strong interactions and is therefore a good starting fragment. In the second zone, chlorandanic acid was the first compound from the 'zone' query that had a ring placed in the hydrophobic pocket. Since the only interaction of interest in this zone was the hydrophobic interaction, chlorandanic acid was paired down to just its phenyl group, which had

TABLE 7
RESULTS ON REBUILDING KNOWN COMPOUNDS

Inhibitor	Link	Link length	Rmsd	Total no. of bridges generated (0.25 Å cutoff)	CPU time[a]
Methotrexate[b]	link1	3	0.271	17	1.1
	link2	4	0.220	39	15.5
HIV protease inhibitor[c]	link1	5	0.370	59	411.0
	link2	2	0.148	3	0.3
	link3	3	0.112	14	0.2
	link4	3	0.328	7	1.5

[a] CPU time on an IRIS Indigo R4000.
[b] To emulate the actual bond angles to the end fragments and torsions found in methotrexate, an endang_tol of 8° was applied, and final_torsions was loosened to 30° for sp^2-sp^2 and sp^3-sp^3 torsions.
[c] To emulate the actual bond angles to the end fragments and torsions found in HIV protease inhibitor, an endang_tol of 8° was applied for all links, and no final_torsions restrictions were used for link2 and link3.

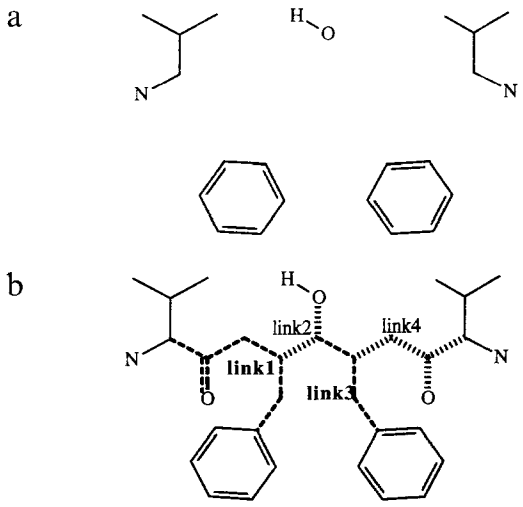

Fig. 10. HIV-1 protease test case. (a) Fragments of HIV-1 protease inhibitor to be connected using BUILDER. (b). Bridges in HIV-1 protease inhibitor (link1–link4) which are regenerated sequentially using BUILDER.

the key interaction in this zone. In the third zone, the second resulting structure from the 'zone' query was phthalysulfamethizole, which contained a carboxylate in the location of the GRID carboxylate 'hot spot'. As in zone 2, since this was the main interaction of interest, the rest of the molecule was deleted. Figure 11 shows each of the molecules chosen, and how they were paired down.

The next step was to find bridges to connect the fragments. Even though the fragments were distorted from the crystal structure, BUILDER was able to reproduce the crystallographic connecting groups. Finally, the composite small molecule was energy minimized with SYBYL [51]. A superposition of the resulting composite structure with methotrexate is shown in Fig. 11. The overall rmsd is 0.506 Å. Thus, with our criteria for characterizing the site, we were able to rebuild methotrexate. Again, with other starting criteria a variety of other structures can be built instead; however, methotrexate is shown to be within the large set of compounds that can be built with this method.

Discussion and Conclusions

This work represents a significant improvement compared to the original BUILDER v.1 strategy. The resulting bridging groups are better, both in chemical sense and in diversity. This is because for every path found in the lattice in the original method, all the atom type combinations that can represent that path (within the user-defined heuristics) are generated, increasing both the number and diversity of the results. The heuristics are another advantage of the new method. The original method used a 'BADLIST' of combinations not to try, whereas the new method uses a 'GOODLIST' to define allowed combinations. There are so many possibilities for combinations, that by restricting only some of them the results can lead to many unnecessarily complicated atom type combinations. It is much easier to define what is allowed and eliminate the rest. This results in reasonably simple bridges, which hopefully would be more synthetically accessible. More sophisticated bridges can be added, if desired, by simply adding more combinations to the 'GOODLIST'.

The applications demonstrate the general robustness of the new bridging method. Given the coordinates of the fragments to bridge, it can reproduce a number of crystal structures. In the HIV-1 protease test case, it was demonstrated that the new algorithm could both reproduce protein backbone bridges and produce novel alternative backbones to link the side chains, which may be useful for designing peptide mimics. This test case also demonstrated robustness, as one bridge could be used to build the next, and the crystal structure was still reproduced. The strongest test for robustness of the bridge generation strategy was the final one of regenerating methotrexate from docked fragments. The docked coordinates differed somewhat from the crystallographic coordinates, being slightly rotated and translated. BUILDER v.2 was still able to reproduce bridges with the same atom type com-

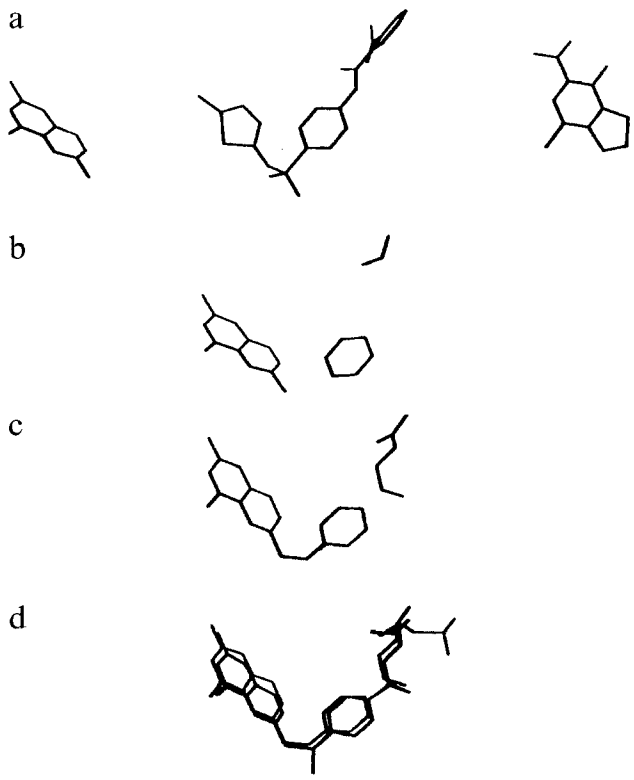

Fig. 11. Rebuilding methotrexate. (a) Starting fragments are pterin from a DOCK single run and molecules from a DOCK search of the CMC database. (b) Pairing down to essential portions of fragments. (c) Connecting fragments and minimization of the structure. (d) Comparison to the crystal structure of methotrexate. The rmsd is 0.512 Å.

binations and similar conformations as the crystallographic ones.

It is difficult to compare directly the results of BUILDER to those of other bridging algorithms, because the goals of each algorithm are slightly different. The programs CAVEAT and NEWLEAD start mainly with single atoms or pharmacophores to combine, and so the goal is to find interesting scaffolds that have key atoms in the pharmacophoric positions. For these purposes, large ring systems and the ability to span a larger area in space are more appropriate. LUDI combines specific site points to a given fragment, and thus requires a smaller bridge library. In contrast, BUILDER and SPLICE start with larger fragments that interact with the site. Simplicity and chemical accessibility are desired in these bridges. Thus, these connectors are simply overlaps of fragment bonds, in the case of SPLICE, and mainly linear bridges or smaller ring systems in the case of BUILDER.

In comparing the different algorithms, the following characteristics can be considered. The database searching approaches (CAVEAT, LUDI and NEWLEAD) are fast and not limited by the size of the desired bridge, but they are limited both by the content of the database and in the sampling of torsion space, as they only look at rigid fragments. NEWLEAD has more flexibility, searching a small library of allowed spacers where any combination of three is acceptable. BUILDER has the most torsional sampling, as it is not limited to only three spacers or rotatable torsions. It has more single atom types to allow more sampling. The trade-off for this increased flexibility is a strong limitation on the length of the resulting bridge. BUILDER is slow to find bridges of length five (on the order of minutes), and searching for lengths greater than six is not practical for an interactive setting. This limitation could be addressed by modifying the breadth-first search algorithm to start from each end and meet in the middle, rather than starting from one end and going to the other.

Another characteristic difference of BUILDER compared to other algorithms is that the shape of the receptor is taken into account *before* the bridge generation stage, limiting the space searched for bridges to the allowable space that can fit inside the receptor. The space used to search for bridges is determined by the lattice, which is bump-checked against the receptor when it is created. In contrast, the other methods have to be bump-checked against the receptor after the generation is completed.

The final difference of BUILDER is that it is an interactive program, whereas the others are more automatic. Thus, BUILDER allows medicinal chemists to use their intuition at every key step in designing a compound, by choosing which templates and bridges are best at each stage, while the computer program performs the more intensive labor steps in finding these templates and generating bridges. The disadvantage of this approach is that it is more labor intensive for the chemist, whereas other approaches require the chemist to evaluate only the end products.

These differences make it hard to directly compare the test cases of the various algorithms to BUILDER. For example, for the HIV protease inhibitor application, we used the same starting fragments as in NEWLEAD. However, since BUILDER's strategy is to sequentially combine fragments, allowing the user control at each step to control the complexity composite it is creating, it is hard to say how many composite molecules were formed, only that there were 155, 3, 14 and 7 bridges, respectively, for each step illustrated in Fig. 9. In contrast, NEWLEAD found 11 total structures. The user was exposed to more fragment bridge possibilities using BUILDER, but ended with fewer total structures than with NEWLEAD. Similar difficulties would crop up with the other algorithms.

In the future we can automate more of the steps in BUILDER. Currently, when joining two fragments, the user must choose which atoms to use as attachment points for a bridge. If the user wishes to try a pair of attachment points, another run must be performed. Instead, the breadth-first search could be modified to allow multiple atoms both at the start and the end of the breadth-first search. Thus, it would be possible to simultaneously search for paths that join to any atoms in the target fragments.

The lattice construct could also be exploited to a larger extent. For example, each lattice point could be given an energetic score, based on its possible interactions within the site. This information could be used to influence which atom types are allowed in that position. It could also be used to prune the search. Instead of doing a full breadth-first search, the list of points adjacent to each lattice point could be presorted by energy, and only the energetically most favorable ones would be accepted. In this way, it may be possible to design bridges that improve binding energy as well as connect groups. This has not yet been developed, because the main role for the bridges is to connect fragments located in the key binding regions.

A final area for development is to work on methods for generating the initial fragment templates. Docking large molecules and deleting the unnecessary portions works well, but it may be more direct to dock smaller molecules into the 'hot spots'. Then a diverse library of small molecules and fragments could be directly used for the initial fragments.

In the coming years, with the number of receptor structures quickly increasing, it is important to develop tools to aid in designing drugs to these structures. BUILDER combines some of the automation of computer algorithms with an interactive system that allows chemists to use their intuition at every step in the design

process. Thus, while there are many directions in which BUILDER can be developed, it already provides a useful tool for structure-based drug design.

Acknowledgements

We wish to thank Richard Lewis, Simon Friedman and Elaine Meng for stimulating discussions. We gratefully acknowledge support from NIH Grants GM07175 and GM31497 (I.D.K.), and the facilities of the Computer Graphics Laboratory (RR-1081, R. Langridge).

References

1 Lam, P.Y.S., Jadhav, P.K., Eyermann, C.J., Hodge, C.N., Ru, Y., Bacheler, L.T., Meek, J.L., Otto, M.L., Rayner, M.M., Wong, Y.N., Chang, C.H., Weber, P.C., Jackson, D.A., Sharpe, T.R. and Erickson-Viitanen, S., Science, 263 (1994) 380.
2 Ring, C.S., Sun, E., McKerrow, J.H., Lee, G.K., Rosenthal, P.J., Kuntz, I.D. and Cohen, F.E., Proc. Natl. Acad. Sci. USA, 90 (1993) 3583.
3 DesJarlais, R.L., Seibel, G.L., Kuntz, I.D., Furth, P.S., Alvarez, J.C., Montellano, P.R., DeCamp, D.L., Babe, L.M. and Craik, C.S., Proc. Natl. Acad. Sci. USA, 87 (1990) 6644.
4 Appelt, K., Bacquet, R.J., Bartlett, C.A., Booth, C.L.J., Freer, S.T., Fuhry, M.A.M., Gehring, M.R., Herrmann, S.M., Howland, E.F., Janson, C.A., Jones, T.R., Kan, C.-C., Kathardekar, V., Lewis, K.K., Marzoni, G.P., Matthews, D.A., Mohr, C., Moomaw, E.W., Morse, C.A., Oatley, S.J., Ogden, R.C., Reddy, M.R., Reich, S.H., Schoettlin, W.S., Smith, W.W., Varney, M.D., Villafranca, J.E., Ward, R.W., Webber, S., Webber, S.E., Welsh, K.M. and White, J., J. Med. Chem., 34 (1991) 1925.
5 Shoichet, B.K., Stroud, R.M., Santi, D.V., Kuntz, I.D. and Perry, K.M., Science, 259 (1993) 1445.
6 Ripka, W.C., Sipio, W.J. and Galbraith, W.G., J. Cell Biochem., 40 (1989) 279.
7 Baldwin, J.J., Ponticello, G.S., Anderson, P.S., Christy, M.E., Murcko, M.A., Randall, W.C., Schwam, H., Sugrue, M.F., Springer, S.P., Gautheron, P., Grove, J., Mallorga, P., Viader, M., McKeever, B.M. and Navia, M.A., J. Med. Chem., 32 (1989) 2510.
8 Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R. and Ferrin, T.E., J. Mol. Biol., 161 (1982) 269.
9 Meng, E.C., Shoichet, B.K. and Kuntz, I.D., J. Comput. Chem., 13 (1992) 505.
10 Lawrence, M.C. and Davis, P.C., Protein Struct. Funct. Genet., 12 (1992) 31.
11 Miller, M.D., Kearsley, S.K., Underwood, D.J. and Sheridan, R.P., J. Comput.-Aided Mol. Design, 8 (1994) 153.
12 Bacon, D.J. and Moult, J., J. Mol. Biol., 225 (1992) 849.
13 Itai, A. and Nishibata, Y., Tetrahedron, 47 (1991) 8985.
14 Itai, A. and Nishibata, Y., J. Med. Chem., 36 (1993) 2921.
15 Rotstein, S.H. and Murcko, M.A., J. Comput.-Aided Mol. Design, 7 (1993) 23.
16 Bohacek, R.S. and McMartin, C., J. Am. Chem. Soc., 116 (1994) 5560.
17 Moon, J.B. and Howe, J.W., Protein Struct. Funct. Genet., 11 (1991) 314.
18 Rotstein, S.H. and Murcko, M.A., J. Med. Chem., 36 (1993) 1700.
19 Gillet, V., Johnson, P., Mata, P., Sike, S. and Williams, P., J. Comput.-Aided Mol. Design, 7 (1993) 127.
20 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.
21 Goodford, P.J., J. Med. Chem., 28 (1985) 819.
22 Boobbyer, D.N., Goodford, P.J., McWhinnie, P.M. and Wade, R.C., J. Med. Chem., 32 (1989) 1083.
23 Danziger, D.J. and Dean, P.M., Proc. R. Soc. London Ser. B, 236 (1989) 114.
24 Danziger, D.J. and Dean, P.M., Proc. R. Soc. London, Ser. B, 236 (1989) 101.
25 Böhm, H.-J., J. Comput.-Aided Mol. Design, 6 (1992) 593.
26 Böhm, H.-J., J. Comput.-Aided Mol. Design, 6 (1992) 61.
27 Miranker, M. and Karplus, M., Protein Struct. Funct. Genet., 11 (1991) 29.
28 Pearlman, D.A. and Murcko, M.A., J. Comput. Chem., 14 (1993) 1184.
29 Ho, C.W. and Marshall, G.R., J. Comput.-Aided Mol. Design, 7 (1993) 3.
30 Bartlett, P.A., Shea, G.T., Telfer, S.J. and Waterman, S., In Molecular Recognition in Chemical and Biological Problems, Special Publication Vol. 78, Royal Chemical Society, London, 1989, p. 182.
31 Tschinke, V. and Cohen, N.C., J. Med. Chem., 36 (1993) 3863.
32 Ho, C.W. and Marshall, G.R., J. Comput.-Aided Mol. Design, 7 (1993) 623.
33 Leach, A.R. and Kilvington, S.R., J. Comput.-Aided Mol. Design, 8 (1994) 283.
34 Fine, R.M., Wang, H., Shenkin, P.S., Yarmush, D.L. and Levinthal, C., Protein Struct. Funct. Genet., 1 (1986) 342.
35 Shenkin, P.S., Yarmush, D.L., Fine, R.M., Wang, H. and Levinthal, C., Biopolymers, 26 (1987) 2053.
36 Lewis, R.A. and Dean, P.M., Proc. R. Soc. London Ser. B, 236 (1989) 125.
37 Lewis, R.A. and Dean, P.M., Proc. R. Soc. London Ser. B, 236 (1989) 141.
38 Lewis, R.A., J. Comput.-Aided Mol. Design, 4 (1990) 205.
39 Lewis, R.A., Roe, D.C., Huang, C., Ferrin, T.E., Langridge, R. and Kuntz, I.D., J. Mol. Graph., 10 (1992) 66.
40 Ferrin, T.E., Huang, C.c., Jarvis, L.E. and Langridge, R., J. Mol. Graph., 6 (1988) 13.
41 Van Gunsteren, W.F. and Berendsen, H.J.C., Mol. Acc. Chem. Res. Phys., 34 (1977) 1311.
42 Ryckaert, J.P., Cicotti, G. and Berendsen, H.J.C., J. Comput. Phys., 23 (1977) 327.
43 Kabsch, W., Acta Crystallogr., A32 (1976) 922.
44 Kabsch, W., Acta Crystallogr., A33 (1978) 817.
45 Leach, A.R., Prout, K. and Dolata, D.P., J. Comput.-Aided Mol. Design, 2 (1988) 107.
46 Bolin, J.T., Filman, D.J., Matthews, D.A., Hamlin, R.C. and Kraut, J., J. Biol. Chem., 257 (1982) 13650.
47 Bernstein, F.C., Koetzle, T.F., Williams, G.J., Meyer, E.E., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T. and Tasumi, M., J. Mol. Biol., 112 (1977) 535.
48 Erickson, J., Neidhart, D.J., VanDrie, J., Kempf, D.J., Wang, X.C., Norbeck, D.W., Plattner, J.J., Rittenhouse, J.W., Turon, M., Wideburg, N., Kohlbrenner, W.E., Simmer, R., Helfrich, R., Paul, D.A. and Knigge, M., Science, 249 (1990) 527.
49 Comprehensive Medicinal Chemistry Database, version 89.2. Available from Molecular Design Ltd., San Leandro, CA.
50 Kellogg, G.E., Semus, S.F. and Abraham, D.J., J. Comput.-Aided Mol. Design, 5 (1991) 545.
51 SYBYL, Version 6.03, Tripos Associates, St. Louis, MO, 1993.