Acta Cryst. (1995). D**51**, 450–457

Mastering the *LORE* of Protein Structure

BY BARRY C. FINZEL

Physical and Analytical Chemistry, The Upjohn Company, Kalamazoo, MI 49001, *USA*

(*Received* 27 *June* 1994; *accepted* 22 *November* 1994)

Abstract

A detailed description of the design, operation and capabilities of *LORE*, a protein-database management tool to supplement more traditional protein map-fitting and model-building programs, is presented. The program includes elements for searching the library of known crystal structures for substructures of similar geometry. Substructures may be as simple as a single hairpin turn, or as complicated as an assembly of different elements of secondary structure. The programs also include elements for manipulating structural segments in complex ways to enable a sophisticated molecular editing capability of enormous utility in modeling and structure-refinement applications.

1. Introduction

A collection of software has been developed to make effective use of known protein structures in molecular modeling and crystallography. The program includes elements to identify substructures with particular geometric and sequence attributes from a library of known structures, and tools to superimpose homologous parts of protein stuctures. It also includes analytical tools and coordinate manipulation components designed to empower the user with the capability to directly include structural information from other structures in their own developing molecular model. This software has been an indispensible tool for *ab initio* map fitting, protein structure refinement, model analysis, and homology modeling problems in our laboratory.

The program originally grew out of frustration while practicing well established techniques for electron-density map interpretation. As protein crystallographers, we have often struggled to fit an atomic model to electron density, even when the density apparently arises from a common structural motif. Moreover, many motifs are not so common or easily recognized. There really were no convenient tools in map-fitting graphics programs such as *FRODO* (Jones, 1985) to make use of past model-building successes. Since Alwyn Jones first showed us the way to rapidly search a known structure for like substructures (Jones & Thirup, 1986), we have been working to develop program tools that will make model building less tedious and more reliable by making it easier to incorporate structural details already known from other structures.

In societies without written language, lore masters have often served as the collective memory of the culture, recalling the history of the entire society. The collection of stored atomic coordinates is, in some sense, the memory of the society of crystallographers. It contains all that we know about protein structure. The software described here has been designed to retrieve this coordinate lore and remind us of what we have already learned. It sets before our eyes the variety of macromolecular structure, so that we can learn from it again and again. That is why we call it *LORE*.

2. Concepts, definitions and program features

To understand and appreciate the functionalities of *LORE*, the reader must be familiar with some general concepts. We define several of these in the following paragraphs to simplify the later program description.

Fragments

A 'fragment' in the vocabulary of *LORE* is any collection of residues and atoms loaded into memory. Many fragments may be held at once and each has its own separate identity. Fragments are composed of some arbitrary number of residues and corresponding atoms, assembled in a linked list data structure of indeterminent size. When fragments are first created, they may simply consist of pointers to residues in library structures. As atomic coordinates are added to fragments, they are rotated and translated to a particular position, usually to overlay some target substructure. (The target is defined below). As a consequence of this open-ended fragment definition, fragments need not be connected amino-acid residues, and may even include prosthetic groups or solvent molecules.

When creating a fragment, *LORE* anticipates that it will be positioned to overlay the current target, and when possible a residue-to-residue correspondence is established between the fragment residues and target residues. This correspondence simplifies subsequent fragment manipulations such as superposition and coordinate substitution.

Target

One definition of 'target' is 'to set as a goal'. This is the definition most applicable to the target in *LORE*. The

ISSN 0907-4449

user defines the target as a guide or template to shape subsequent *LORE* activity. The target serves a fourfold purpose: (1) it provides the basis for deriving residue-to-residue correspondences with subsequently created fragment objects; (2) it provides the database search algorithm with a template of C_α geometry needed to identify fragments of like geometry from known structures; (3) it defines the position and orientation of created fragments; and (4) specifies the number of residues in the residue mask (defined below). The target is defined by specifying residue ranges from the current molecular model. It may be as simple as a single residue or residue range, or as complicated as an assembly of disconnected residue segments. Residue and atom data are loaded into memory and held until overwritten by another target. Fragments subsequently retrieved from the library are always superimposed on some part of the target and the r.m.s. difference in position of superimposed atoms is shown. Often, *LORE* relies on the target to provide sequence or atomic coordinate data for undefined or missing atoms.

Much of the complexity of *LORE* arises out of the author's attempts to free the user from the limitations imposed by the target. The residue mask (defined below) and tolerance parameters (see information about the search algorithm below) are two ways to introduce flexibility to the target. After all, the object is to learn something we did not already know. If we require the user to precisely specify (with atomic coordinates) the geometry of the objects we hope to find, the purpose of a search is defeated. Nevertheless, some target specification is currently unavoidable.

The database

The database of known protein structures for *LORE* is layered to enhance efficiency in searching (Fig. 1). The lowest layer contains atomic coordinates of Protein Data Bank (PDB) entries (Bernstein *et al.*, 1977). These are reformatted into residue-indexed direct-access files to simplify extraction of selected residue ranges. Layered over this is a database of 'chain information' containing amino-acid sequence and C_α geometry data for individual polypeptide chains in the PDB, and pointers to the complete atomic coordinate data in the lower layer. Chain information is the primary data used in searching. Atomic coordinate data in the lower layer is referenced only after homology has been confirmed through examination of sequence and geometry data contained in the chain information. The highest layer is a text-based index of chains. This index points to chain information and includes chain rankings based on properties such as uniqueness, resolution of the structure determination, *R* value, *etc.* The rank is helpful in selecting the specific subset of chains most appropriate to a particular application. For example, the user can restrict a search to only proteins from a given structural family simply by modifying the rankings in the index file and then selecting only the chains that meet a minimum rank requirement at run time. In our laboratory, structures are

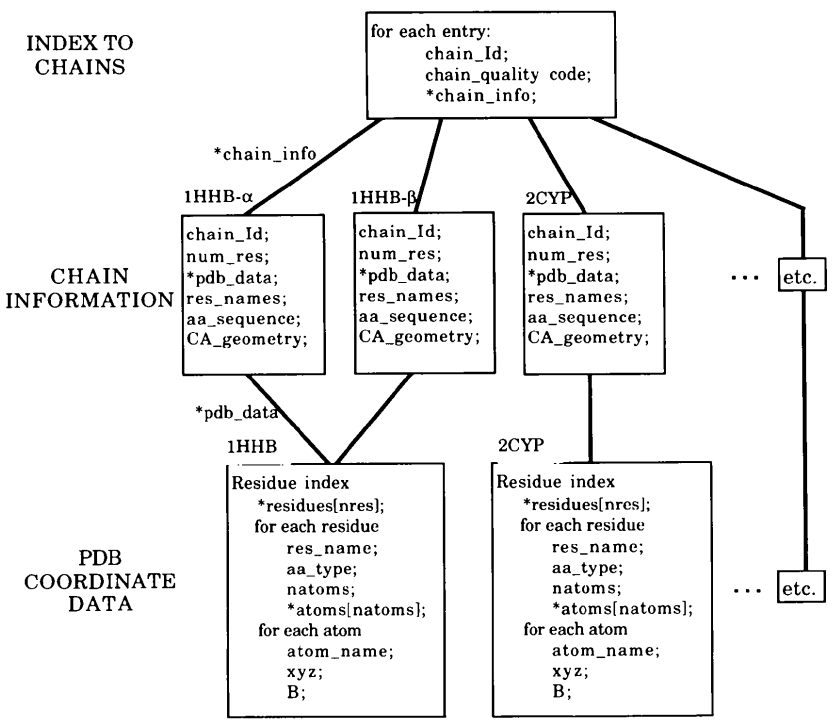


Fig. 1. *LORE* database design. Components of the layered database are connected through pointers as illustrated. Data elements that represent pointers to additional data begin with '*', like in C structure declarations. Data derived from two PDB entries (Hemoglobin, 1HHB; and cytochrome *c* peroxidase, 2CYP) are shown. The hemoglobin α- and β-chains are two different entries in the chain index.

most often ranked by quality (resolution, R value) and only the best chains are selected unless a larger database subset is necessary. All the database files can be derived directly from PDB files with software that is part of *LORE*.

The residue mask

The 'residue mask' is a component of the *LORE* user interface that provides a simple but very effective device for exercising residue-by-residue control over different *LORE* commands. The mask is the single most important construct to understand for effective use of *LORE*. It is through the mask that users specify particular sequence requirements in substructure searching (*e.g.* a glycine at the fourth residue position), or allow for conformational uncertainty when C_α positions are not known or trusted. A facsimile of the residue mask interface is shown in Fig. 2. Because it influences so many different *LORE* functions, the status of the mask is always on display in the *LORE* terminal window. Algorithmically, the mask consists of two logical arrays. The first array defines a logical state for each of the 20 amino acids, and specifies allowed sequences. This construct is used primarily in searching. The second array defines an on/off status for each residue in the target and, consequently, corresponding residues in a fragment. This on/off status has some impact on most *LORE* operations involving residues. It can be set to disable superposition at certain residues, for example, or identify amino acids and residue positions that will be the target for side-chain rotamer expansion (see below).

Target	Status	Allowed Amino Acids
Segment 1:		
5 L	On	ACDEFGHIKLMNPQRSTVWY
6 R	On	ACDEFGHIKLMNPQRSTVWY
7 D	On	ACDEFGHIKLMNPQRSTVWY
8 S	On	ACDEFGHIKLMNPQRSTVWY
9 N	Off	DGNPST
10 Q	Off	G
11 K	Off	DGNPST
12 S	On	ACDEFGHIKLMNPQRSTVWY
13 L	On	ACDEFGHIKLMNPQRSTVWY
14 V	On	ACDEFGHIKLMNPQRSTVWY
Segment 2:		
41 Y	Off	ACDEFGHIKLMNPQRSTVWY
42 E	On	ACDEFGHIKLMNPQRSTVWY
43 L	On	ACDEFGHIKLMNPQRSTVWY
44 K	On	ACDEFGHIKLMNPQRSTVWY
45 A	On	ACDEFGHIKLMNPQRSTVWY
46 L	On	ACDEFGHIKLMNPQRSTVWY

Fig. 2. A reproduction of the residue mask interface as it might appear while configuring *LORE* for a search of the database. The user moves the cursor through a vertical stack of residue names derived from the current target definition to set attributes of the residue mask. In this example, the target consists of two segments (residues 5–14 and 41–46). Residues 9, 10, 11 and 41 have been toggled off. The positions of C_α atoms in these residues will be ignored during the search. One-letter amino-acid codes at the right of each residue name show allowed amino acids. These are specified with single keystrokes or by Boolean combinations of select amino-acid types or predefined groups (such as amino acids common in helices). In the present case, a glycine must occur in homologous fragments at residue 10. Turn-promoting residues are required at residues 9 and 11.

The search algorithm

The search algorithm used in *LORE* for finding library substructures with conformational similarity to a target is derived from that of Jones & Thirup (1986). They showed that a rapid search can be conducted under the premise that similar substructures must have similar elements in a matrix containing inter-C_α distances. Distances between all C_α atoms are pre-computed for library structures and included in the chain information files of the database. If at least some C_α positions are defined in the target residue range, the entire list of active chains can be screened very rapidly to find similar structures. The individual elements of a triangular matrix of inter-C_α distances characteristic of the target residues $[d_T(i,j)]$ are compared with corresponding elements for each potential fragment $[d_F(i,j)]$ in the database. Any observed difference $|d_T(i,j)_\circ - d_F(i,j)|$ greater than a user selected tolerance (*e.g.* 1.0 Å) causes immediate rejection of the fragment. This search technique was easily extended for application in searches where the target consists of multiple disconnected residue ranges. In this case, the additional equivalence (again with a user-selected tolerance) of appropriate inter-C_α distances encoding the positional relationship of C_α atoms in one residue range to those in another is required. The coupling of this simple but rapid algorithm with the flexible target specification and residue mask gives *LORE* unique capabilities. By turning off the position of residue i in the mask, the user directs that all distances involving residue i should be ignored. This is an excellent way to overcome small deficiencies in the original C_α model.

The result of the search is not superimposed atomic coordinates, but pointers to structures. Fragment atom data is not automatically loaded. This separation of functions is set up deliberately to offer the user the opportunity to perform the superposition with different residues enabled than for the search, or to repeat the superposition if needed.

The superposition algorithm

In *LORE*, the best correspondence between two sets of atoms is found by the method of Kabsch (1978), aligning a specific subset of atoms common (by name) to both sets. The superposition is driven by the residue-to-residue alignment established at the time a fragment is created. The superposition algorithms are also interfaced to the residue mask. Only target atoms from residues marked on at the corresponding mask position are used.

The result of a superposition operation is a transformation matrix and vector that will overlay the fragment atoms onto the target. The r.m.s. fit of common atoms then becomes an attribute of the created fragment.

Typically, only the fragment residues that correspond to the target are transformed and loaded into memory, but the derived transformation can as easily be applied to overlay other atoms, too. In this way, *LORE* can be used to superimpose whole protein molecules based on the alignment of only a few substructures, or to display bound inhibitor positions when it is actually the proteins that have been overlaid.

Rotamers

The idea of 'rotamers' (Ponder & Richards, 1987; Janin, Wodak, Levitt & Maigret, 1978), common conformations of amino-acid side chains that represent geometries of particular stability, is now widely accepted. To speed modeling, many graphics programs include push-button or menu-driven interfaces that generate a set of possible amino-acid side-chain conformations from which the correct conformation may be chosen. The addition of a rotamer look-up capability to other *LORE* algorithms is very beneficial, because the user may then assess the frequency of occurrence of different side-chain conformations in given main-chain conformational contexts. This functionality permits the interactive extension of generalizations about the relationship of side-chain conformations and secondary structure (MacGregor, Islam & Sternberg, 1987) to very specific substructural motifs. Frequency analyses of this type can be quite predictive (Finzel, Kimatian, Ohlendorf, Wendoloski, Levitt & Salemme, 1990), or lead to the formulation of rules to guide protein modeling (Blundell, Sibanda, Sternberg & Thornton, 1987).

Options for atom manipulation

Additional flexibility is built into *LORE* to help the user manage the way sets of atoms are manipulated during various operations. A list of atom names (the superposition atom list) defines atoms from corresponding residues that will be overlaid during superposition. This can be set to C_α, all main-chain atoms (N, C_α, C, O), or any arbitrary list. Since the r.m.s. fit of a fragment to the target is often taken as an indicator of a good fit, it is very useful to be able to limit the superposition to only trusted atom positions. This list, when coupled with the residue alignment control available through the residue mask, offers good flexibility without significantly complicating the user interface. A similar list of atom names specifies which atoms are replaced when target residues are replaced by their counterparts from a given fragment. In some applications, such as structure refinement, it is as important to leave atom parameters undisturbed during rebuilding as it is to make necessary corrections. If only certain side-chain atom positions are wanted from a given fragment, this can be easily arranged. Finally, since *LORE* does almost everything in memory, it is desirable to be able to optimize memory utilization. One can specify a list of names of atoms that are to be loaded into memory from the disk when fragments are created. Side-chain or solvent atoms can be easily ignored when these are not needed.

3. Program implementation

LORE is implemented as a mixed Fortran-77 and C-language program. While the software is self-contained and includes all functions necessary for substructure manipulation, it is not a graphics program, and includes no stand-alone fragment display capability. We have, therefore, implemented the program to run as a separate submode of *PSFRODO* (Pflugrath, Saper & Quiocho, 1985). This is a convenient combination because *PSFRODO* already contains tools for electron-density display and model building that complement the database access functions of *LORE*. Fragments can be graphically displayed with a modified *PSFRODO* interface that allows independent viewing of up to 24 fragments simultaneously. The *LORE* software also has been written to retrieve (or send) new coordinates directly from (or to) *PSFRODO* memory to simplify information exchange between elements of the unified package. The software could be similarly implemented as a subprogram of any molecular modeling software, such as *CHAIN* (Sack, 1988), for which source code is available.

Program functions are accessed through a flexible *LORE* command language interpreter. Typed commands invoke routines that perform a specific action. Case insensitivity, unique command abbreviations and extensive type-ahead capabilities are features of this interface. A subset of *LORE* commands is listed in Table 1. A brief summary of each command and the resulting action is given. Commands can be roughly grouped into five functional classes: those that impact database administration, target specification, fragment creation or manipulation, atomic coordinate *I/O*, or options that might impact any of the above. Residue mask specifications are input through a separate, highly interactive command structure activated whenever certain commands are invoked.

To demonstrate how this software might be used, we include in this report two example applications of *LORE*. The first will illustrate how one can use *LORE* to expand upon a crude polyalanine C_α model to include all main-chain atoms and, eventually, side chains. This will show how the target is specified, how the search of known structures for similar substructures is done, how the backbone conformations of located fragments are incorporated in the growing model, and how to use the rotamer library to add side chains. A second example illustrates some general superposition capabilities of *LORE*. We shall superimpose two related molecules by targeting optimum correspondence between three disconnected but conserved elements of secondary structure, and then save the coordinates of the reoriented molecule for later reference.

Table 1. *LORE commands*

Database Administration Tools	
INDEX	Search chains listed in the named index file.
USE	Search only chains with a quality rating above the requested minimum.
LIST	List chains currently enabled for searching.
SAVE	Save the current target residue range as a new chain in the database.
Target Specification	
TARGET	Make the selected residue range(s) the current target.
REVERSE	Invert the direction of the advancing polypeptide chain in target segments.
Fragment Manipulation	
AVERAGE[1]	Create a new fragment by averaging selected fragments.
EDIT	Enter a subcommand mode to manage fragments in memory.
KEEP	Keep only selected fragments.
DELETE	Delete selected fragments from memory.
SORT	Sort fragments by goodness of fit or sequence homology (Dayhoff, *et al.* 1978.
MASK[1]	Select only fragments with certain sequence characteristics.
FIND[1]	Perform a search of the database for fragments with specific sequence or structural characteristics.
LOAD[1]	Retrieve atomic coordinates for fragments identified by FIND and superimpose them on the target.
ROTAMER[1]	Create fragments that represent different side chain rotamers.
SUPER[1]	Perform general superposition of atoms onto the target.
Atomic Coordinate I/O	
DISPLAY	Display selected fragments on the graphics device.
MAKEPDB	Make PDB files from selected fragments.
REPLACE[1]	Replace coordinates of target residues with those from the selected fragment.
Control Options	
MASK	Edit residue mask status.
SET	Enter subcommand mode to set control variables.
SATOM	Names of atoms in corresponding residues that will be overlaid.
RATOM	Names of atoms to be replaced with new positions.
LATOM	Names of atoms to be loaded into fragments.
TOLER	Define tolerances for Cα geometry search.
IGNORE	Ignore solvent molecules in library structures (Yes or No).
MUTATE	Mutate amino acid type during replacement (Yes or No).
DMODE	Set display mode to (all atoms, Cα backbone, ribbon).
Analysis Tools	
FREQUENCY[1]	Determine rotamer frequencies in a given population of fragments.
VDWCHK[1]	Delete fragments with outrageous non-bonded contacts.

[1]Commands that depend upon the state of the residue mask.

Example 1. C_α*-backbone model expansion*

The rough C_α model that serves as the starting point for this example was built as previously described (Finzel *et al.*, 1990) before entering *LORE*. We select residues A5 through A20 as the target (Fig. 3*a*). C_α positions from this residue range define the geometry of similar objects that will be located from the database. The structure is obviously that of a 16-residue section of α-helix. This is a trivial example, but one that will serve to show how *LORE* commands must be combined to complete the model.

The search is conducted with commands shown in Fig. 3(*b*). Parameters defined with the set command define attributes of the search; the quality (and therefore the number) of chains to be selected from the chain index, and the tolerance that defines the largest acceptable difference (Å) in inter-C_α distance in a homologous fragment. The search is completed in a few seconds, with 121 similar fragments found. We keep only the 50 most similar with the *EDIT* command, load and superimpose these 50 onto the target with the *LOAD* command, and then graphically examine some fragments. As fragments are selected for display, attributes of the created fragments are shown, including the r.m.s. fit of each fragment to the original target. If we find a backbone model we like, we can select it to replace our C_α model. In this case, they are so similar that it is difficult to choose, so we can compute an average main-chain geometry from the 20 best fragments (Fig. 3*c*) and then replace our original model with the average main-chain atom positions.

Now we continue by adding side-chain atomic coordinates with the help of the rotamer utility. All rotamers for a phenylalanine are created at residue A13 and displayed with commands in Fig. 3(*d*). The residue(s) and amino acid(s) for rotamer expansion are selected through the residue mask interface (not shown). Once rotamers are generated, we simply set parameters to allow model mutation, select the rotamer we want to

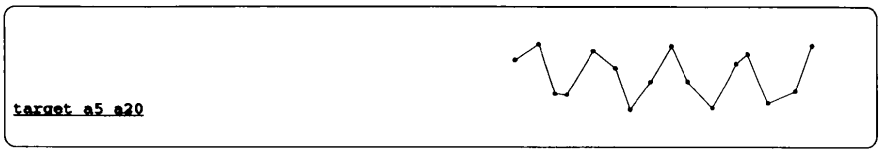
keep, and replace (Fig. 3*e*). All side chains along the chain can be added the same way to complete the model.

Example 2. General superposition

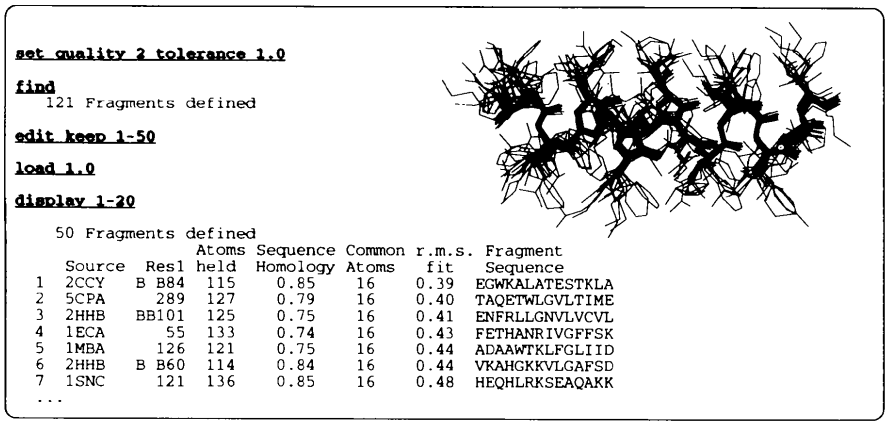
In the second example (Fig. 4), we will superimpose elements of secondary structure known to be conserved in two different protein structures so that the two structures may be compared, the *E. coli* ribonuclease H (PDB entry 1RNH; Yang, Hendrickson, Crouch & Satow, 1990) and the RNase H domain of the HIV-1 virus (PDB entry 1HRH; Davies, Hostomska, Hostomsky, Jordan & Matthews, 1991). We assume no knowledge of the identity of corresponding residues in the two structures, but we can use the search capabilities of *LORE* to find these correspondences for us.

If it is not already done, we first must add atomic coordinates for one of the structures (*e.g.* 1RNH) to the searchable database of known structures. This can be done directly from the PDB entry with a supplementary program (*MAKCHN*) that is part of *LORE*. The HIV-1 RNase H model is defined as our current model. We will superimpose the *E. coli* model onto this.

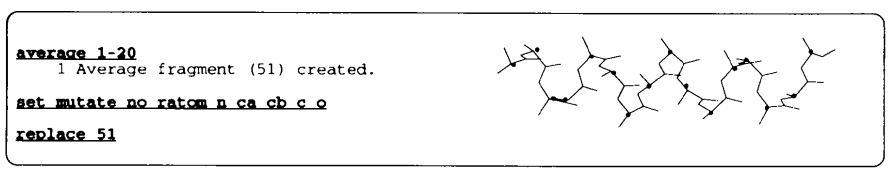
We must first select a target. The target should define structural attributes unique to this class of proteins. We choose three segments, one long helical section and two



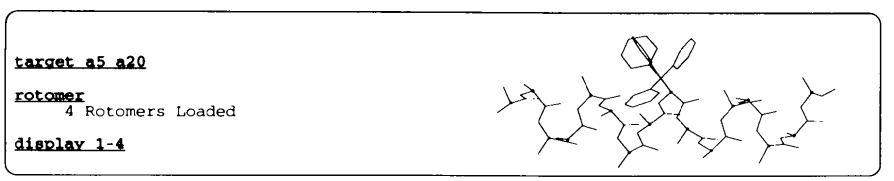
(*a*)



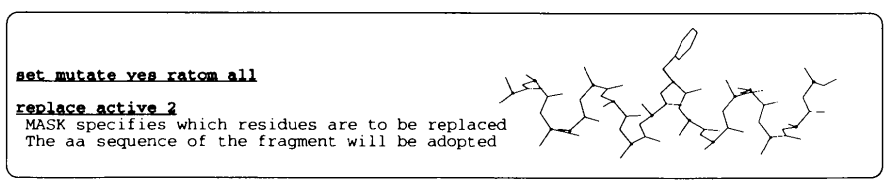
(*b*)



(*c*)



(*d*)



(*e*)

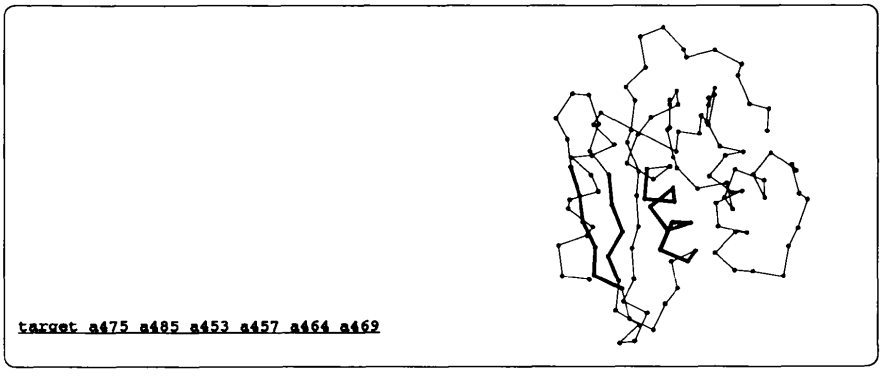
Fig. 3. Example of C_α-backbone model expansion performed with *LORE*.

paired β-strands (Fig. 4*a*). These form part of the structural core of the HIV-1 RNase H molecule that we expect to be conserved. We will look for structures in the database that share the geometry of each individual segment, and also the same structural relationship between the segments.

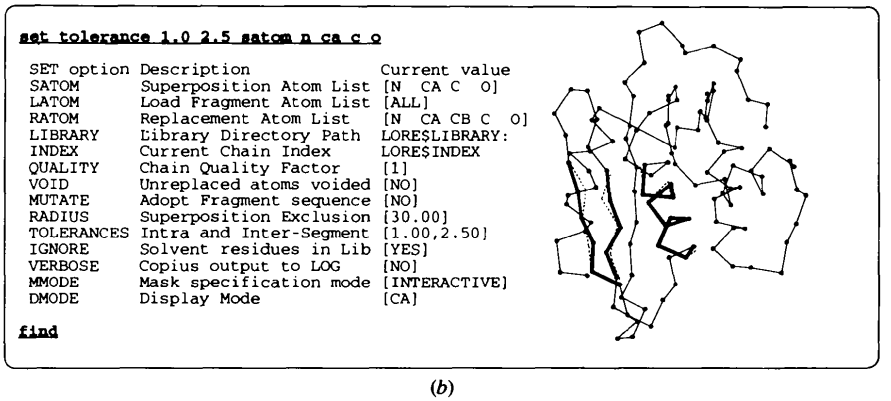
Before we conduct a search or do any superposition we must adjust some search parameters. We increase the intersegment tolerance to 2.5 Å, because it regulates the allowed tolerance in the relative orientation of one of our target segments (an element of secondary structure) to another. The intrasegment tolerance, which allows for variation within a segment, is 1.0 Å. We establish that the superposition will be performed with all main-chain atoms (N, C_α, C, O) (Fig. 4*b*).

Now we conduct the search just like in example 1. Multi-segmented searches take longer, but we still get the result in a few seconds. Only one other structure in our library contains this $\alpha\beta\beta$ motif, the *E. coli* RNase H we sought. The dashed C_α backbone in Fig. 4(*b*) shows how these segments conform to those in the target.

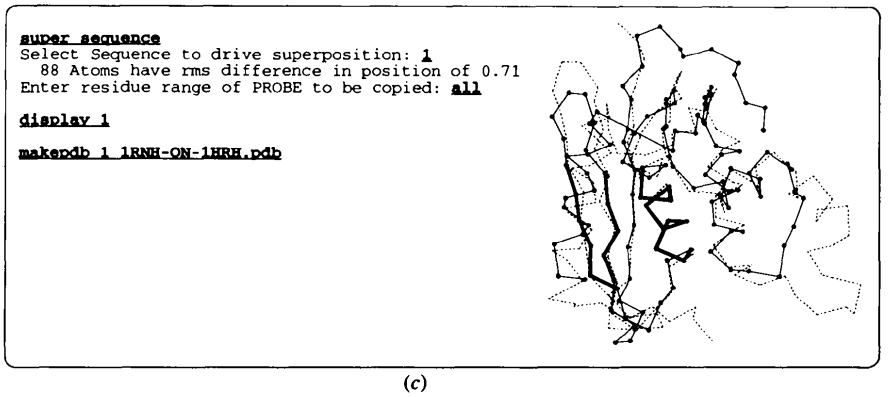
We could use *LOAD* to pull this three-segment fragment into memory as in example 1, but we are still interested in seeing the entire 1RNH molecule, so we will use *SUPER*. In invoking *SUPER*, we use the *SEQUENCE* option that defers to a selected fragment



(*a*)



(*b*)



(*c*)

Fig. 4. Example of intermolecular superposition performed with *LORE*.

or sequence entry in memory for identification of residues to be paired. We then instruct it to load the entire *E. coli* model. The reoriented atomic coordinates are now available as fragment 1 (Fig. 4*c*). They can be displayed or saved for examination at another time.

4. Summary

LORE is not the first program to be written to take advantage of protein structural data, nor will it be the last. It does have some unique capabilities that, to our knowledge, are not joined in any other software. While *O* (Jones, Zou, Cowan & Kjeldgaard, 1991) includes extremely powerful tools for *ab initio* map interpretation and model building using substructure libraries (Zou & Mowbray, 1994), it does not include the atom manipulation or target specification flexibility that readily extends such tools to applications in structure refinement or model analysis. *FRAGLE* (Finzel *et al.*, 1990), the direct predecessor of *LORE*, does not use the open-ended fragment data structures that makes *LORE* so useful as a general structural comparison tool.

Over the past six years, we have used *LORE* extensively in map fitting, structure analysis, and homology modeling, and have come to rely heavily on database structural information to guide our efforts. It has become an indispensible part of our software toolkit.

References

BERNSTEIN, F. C., KOETZLE, T. F., WILLIAMS, G. J. B., MEYER, E. F. JR, BRICE, M. D., RODGERS, J. R., KENNARD, O., SHIMANOUCHI, T. & TASUMI, M. (1977). *J. Mol. Biol.* **112**, 535–542.

BLUNDELL, T. L., SIBANDA, B. L., STERNBERG, M. J. E. & THORNTON, J. M. (1987). *Nature (London)*, **326**, 347–352.

DAVIES, J. F. II, HOSTOMSKA, Z., HOSTOMSKY, Z., JORDAN, S. R. & MATTHEWS, D. A. (1991). *Science*, **252**, 88–95.

DAYHOFF, M. O., SCHWARTZ, R. M. & ORCUTT, B. C. (1978). In *Atlas of Protein Sequence and Structure*, Vol. 5, Suppl. 3, edited by M. O. DAYHOFF, pp. 345–352. Washington DC: Nat. Biomed. Res. Found.

FINZEL, B. C., KIMATIAN, S., OHLENDORF, D. H., WENDOLOSKI, J. J., LEVITT, M. & SALEMME, F. R. (1990). In *Crystallographic and Modeling Methods in Molecular Design*, edited by C. E. BUGG & S. E. EALICK pp. 175–188. New York: Springer-Verlag.

JANIN, J., WODAK, S., LEVITT, M. & MAIGRET, B. (1978). *J. Mol. Biol.* **125**, 357–386.

JONES, T. A. (1985). *Methods Enzymol.* **115**, 157–171.

JONES, T. A. & THIRUP, S. (1986). *EMBO J.* **5**, 819–822.

JONES, T. A., ZOU, J.-Y., COWAN, S. W. & KJELDGAARD, M. (1991). *Acta Cryst.* A**47**, 110–119.

KABSCH, W. (1978). *Acta Cryst.* A**34**, 827–828.

MACGREGOR, M. J., ISLAM, S. A. & STERNBERG, M. J. E. (1987). *J. Mol. Biol.* **198**, 295–310.

PFLUGRATH, J. W., SAPER, M. A. & QUIOCHO, F. A. (1985). In *Methods and Applications in Crystallographic Computing*, edited by S. HALL & T. ASHIDA pp. 404–407. Oxford Univ. Press.

PONDER, J. W. & RICHARDS, F. M. (1987). *J. Mol. Biol.* **195**, 775–791.

SACK, J. S. (1988). *J. Mol. Graphics*, **6**, 224–225.

YANG, W., HENDRICKSON, W. A., CROUCH, R. J. & SATOW, Y. (1990). *Science*, **249**, 1398–1405.

ZOU, J.-Y. & MOWBRAY, S. L. (1994). *Acta Cryst.* D**50**, 237–249.