Acta Crystallographica Section D
Biological Crystallography
ISSN 0907-4449

Towards complete validated models in the next generation of *ARP/wARP*

Serge X. Cohen,[a] Richard J. Morris,[b] Francisco J. Fernandez,[a,c] Marouane Ben Jelloul,[a] Mattheos Kakaris,[a] Venkataraman Parthasarathy,[c] Victor S. Lamzin,[c] Gerard J. Kleywegt[d] and Anastassis Perrakis[a]*

[a]Netherlands Cancer Institute, Department of Molecular Carcinogenesis, Plesmanlaan 121, 1066 CX, Amsterdam, The Netherlands, [b]EMBL European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton CB10 1SD, England, [c]European Molecular Biology Laboratory, Gebaude 25a DESY, Notkestrasse 85, 22603 Hamburg, Germany, and [d]Department of Cell and Molecular Biology, Uppsala University, Biomedical Centre, Box 596, SE-751 24 Uppsala, Sweden

Correspondence e-mail: a.perrakis@nki.nl

Received 13 February 2004
Accepted 27 October 2004

The design of a new versatile control system that will underlie future releases of the automated model-building package *ARP/wARP* is presented. A sophisticated expert system is under development that will transform *ARP/wARP* from a very useful model-building aid to a truly automated package capable of delivering complete, well refined and validated models comparable in quality to the result of intensive manual checking, rebuilding, hypothesis testing, refinement and validation cycles of an experienced crystallographer. In addition to the presentation of this control system, recent advances, ideas and future plans for improving the current model-building algorithms, especially for completing partially built models, are presented. Furthermore, a concept for integrating validation routines into the iterative model-building process is also presented.

1. Introduction

The *ARP/wARP* software suite (Morris *et al.*, 2003) is a package of utilities aimed at delivering an essentially complete macromolecular atomic model from a phase set. Given data extending to at least 2.5 Å and reasonable initial phase estimates, *ARP/wARP* is capable of building a fairly complete and refined protein model within hours. The underlying algorithms are described in Lamzin &Wilson (1997), Perrakis *et al.* (1999) and Morris *et al.* (2002). An overview and comparison of current model-building packages can be found in Badger (2003) and the current status of *ARP/wARP* is described in Morris *et al.* (2003) and will not be repeated here. We instead focus here on the design of a new control system aimed at providing the user with a high degree of flexibility and additional building/completion/validation routines whenever necessary. More importantly, we aim to combine all these features into a highly automated model-building expert system that can build a model, identify problematic regions, choose the strategy for improvement and iterate through building, refinement and validation steps until the model is essentially of submission quality, at least for the protein parts of the model. Automated modelling of non-protein (nucleic acids, ligands, ions) regions presents a more complicated task that we are beginning to address (see Zwart *et al.*, 2004), but is far from complete automation. The same restriction applies for the modelling of 'unusual' parts of a protein model, *e.g. cis*-peptides and glycosylation, which are not currently being addressed. Automated modelling of the water structure is available but a number of issues have to be addressed, possibly also in the context of validation.

1.1. Automated model building

There has been significant progress in automated procedures for building protein models in crystallographic density maps (Badger, 2003). *ARP/wARP* (Perrakis *et al.*, 1999) introduced iterative automated model building (Perrakis *et al.*, 1999) in high- and medium-resolution maps. *RESOLVE* (Terwilliger, 2003*a*) and *MAID* (Levitt, 2001) utilize alternative approaches and offer automated model-building tools at lower resolution. More recently, approaches based on pattern recognition (Holton *et al.*, 2000; Ioerger *et al.*, 1999; Ioerger & Sacchettini, 2002) have started to become available. Automated tools are also included in popular interactive model-building packages such as *O* (Jones *et al.*, 1991), *QUANTA* (Oldfield, 2003), *MAIN* (Turk, 1992) and *XtalView* (McRee, 1999). Template-convolution methods as exemplified in Kleywegt & Jones (1997*b*) and implemented in an FFT formulation (Cowtan, 1998) also offer tools to automate model building.

In a strict sense, however, a true 'automated model-building' package should be capable of delivering a model that is both complete and validated in all cases; *i.e.* as error-free as possible as judged by the high standards of an experienced crystallographer. Although *ARP/wARP* is close to fulfilling these criteria for high-resolution data and good starting phases, it is not unfair to state that current software is not close to true automation. Ironically, the 'bottleneck of initial model building' has recently transformed into the 'bottleneck of model rebuilding, error checking and validation'. It is our intention to address these issues that follow the initial automatic generation of a protein model and extend our software package to deliver complete, error-free and validated models that are of a completeness and quality directly comparable to that of models that are produced after careful human inspection.

To achieve this goal, it is essential to combine the development of model-rebuilding tools with robust validation protocols that will 'criticize the model' and a sophisticated control system that will take the appropriate action for corrections. The 'final model' will be refined and validated and be delivered together with an extensive model-quality report.

1.2. Iterative validation and rebuilding

Currently, the steps of quality control and rebuilding of intermediate models are still predominantly manual (and therefore time-consuming) exercises. They require expertise on behalf of the crystallographer to recognize model regions that are unusual for some reason or other (poor fit to the density, uncommon main-chain conformation, non-rotamer side-chain conformation *etc.*), although validation software packages can make this task easier and more systematic (Kleywegt & Jones, 1996). However, not only does the crystallographer need to be able to recognize 'outlier' aspects of the model, he or she also needs to assess whether the outliers are 'errors in the model' (that need to be corrected prior to any model analysis and interpretation) or whether they are genuine, albeit unusual, features of the structure (in which case they may warrant description in the paper; Kleywegt, 2000; Kleywegt & Jones, 1997*a*). Full or partial automation of the quality-control and rebuilding task offers substantial benefits. Obviously, it would reduce the amount of time required to obtain the final model and let the crystallographer focus on the interpretation and analysis of the model rather than on its construction and debugging. Possibly even more important, though, would be the fact that the process would be made less dependent upon the individual skills of the crystallographer and the local validation practices (or lack of such) and thereby might improve the average quality of the models in the structural database. One possible approach to automating the quality-control and rebuilding process is to include a 'model-criticism module' as part of an iterative and automated model-building scheme. Such a module would take an intermediate model as input (together with an electron-density map that is minimally biased), apply heuristic rules to identify outlier residues and instruct the building program to remove and rebuild these residues. That is the intention of the software module *ElAl* (short for 'Electronic Alwyn') that we develop and intend to incorporate into *ARP/wARP*.

2. Results and discussion

2.1. The need to design a new generation of *ARP/wARP* control system

Our plans for producing a package that is able to produce complete validated models without user intervention imposed an imperative for the redesign of the *ARP/wARP* control flow scheme. The previous scheme was a C-shell script that supported a standard workflow and extremely little decision-

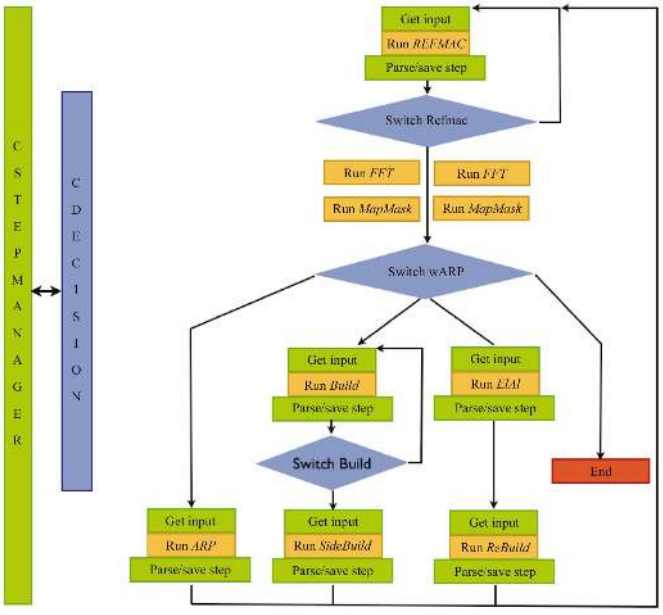

Figure 1
The new *ARP/wARP* workflow as implemented in Python (*pyWARP*). The 'Step Manager' class and classes that exchange information with the Step Manager are in green, 'Runnable' classes are in orange and the Decision Functions are in blue boxes. The blue 'Decision' box denotes the communication of the Step Manager with decision-making modules.

making. We implemented a new flexible control scheme in Python (*WARPy*) that will allow better functionality and future extensions. We have implemented the basic architecture and workflow and we expect to distribute a working version around the end of 2004.

The cornerstone of the *ARP*/*wARP* concept is the recycling of phase information between real space (model building) and reciprocal space (model refinement). In the original implementation, model building follows five or ten model-refinement cycles and this cycle is typically iterated ten times. For high-quality maps, 10×10 (100) cycles is excessive since an almost complete model can be obtained after three to five building cycles. For low-quality maps, however, the procedure may need to be run for as many as a few hundred cycles to obtain optimal results. Moreover, for high-resolution data ten refinement steps between autobuilding prove excessive, while for lower resolution data these might not be sufficient. Decisions as to when it is appropriate to attempt to assign the sequence to the model have to be taken at the start of the run, without real knowledge of the behaviour of the algorithms on that particular problem. Finally, the model of the last cycle is always considered as the final model, even if intermediate models are of higher quality.

To address these issues and to anticipate future developments, we decided to redesign the *ARP*/*wARP* control system. The main concepts underlying the philosophy of the design are outlined below.

2.1.1. A modular approach. A central feature in the new design and implementation of the *ARP*/*wARP* control system is the creation of flexible modules that take into account some principles of object-oriented programming. We chose the Python programming language for this task. We created a Runnable Class that is a wrapper around specific programs: it first runs the program and then extracts information about the run from the log files. This class is specialized for each software module, *i.e. REFMAC* (Murshudov *et al.*, 1997), *FFT*, *ARP* and the autobuilding modules. The central point of our design, however, is the Step Manager Class, which is the module that controls program flow. The Step Manager stores information about every step that has been taken by the software in the past, is aware of the current state and consults Decision Functions for making decisions for the future steps. The general design is depicted in Fig. 1.

2.1.2. An architecture that allows dynamic decision making. The 'classical' *ARP*/*wARP* flow scheme will be superseded by the more generalized scheme that is depicted in Fig. 1. The main new feature of this scheme is the implementation of decision points that control the program flow. There are three main decision points, which correspond to Decision Functions. These are separate procedures that examine crystallographic statistics and provide feedback to the Step Manager.

The first decision point (Switch Refmac) follows the execution of the reciprocal-space refinement step, typically performed by *REFMAC*. In the 'traditional' *ARP*/*wARP* flow, there was no decision to be made here: after refinement, maps were calculated and the model was updated by addition and deletion of atoms. In the new flow scheme the crystallographic R factors, likelihood gradients and geometrical targets are examined. If there is room for improvement of the current model (*i.e.* if the free R was still dropping sharply or if the geometry targets produced an 'over-restrained' model), then refinement of the current model is continued until convergence is reached. If refinement appears to have converged then the control system issues the command to calculate the $2mF_o - DF_c$ and $mF_o - DF_c$ maps and proceeds to the second decision point.

The second decision point (Switch wARP; Fig. 1) is the most elaborate one. There is a multitude of decisions that can be made at this step after consulting crystallographic refinement statistics. The most straightforward one is to just proceed with model update (addition and deletion of atoms, leftmost choice in Fig. 1) and re-enter refinement. This is equivalent to the 'old' protocol's 'small cycle'. The second option is to proceed with model reconstruction (middle choice in Fig. 1) and automated model building, which is equivalent to the 'old' protocol's 'big cycle'. A new option that will be provided is partial model reconstruction (right choice in Fig. 1), which will allow building of loops and disordered areas or problematic areas as identified by the validation module we describe later in this paper.

The third decision point (Switch Build; Fig. 1) is with regard to map reinterpretation while autobuilding. Apart from being a technicality of the current building algorithm (if iterated the algorithm extends or joins some fragments, since the regularization of terminal residues can improve the connectivity with adjacent free atoms), it offers attractive possibilities. For example, several autobuilding algorithms can be tried sequentially and the best one can be chosen at this decision point.

2.1.3. Interchangeable and addable modules for specific tasks. The Runnable Class essentially serves as a wrapper around each module and also enables a powerful plug-in facility. The 'Step Manager' does not 'care' whether the model optimization was performed by *REFMAC* or any other program. Thus, in principle, anybody could write a wrapper for another refinement program. The Step Manager executes the Refinement Class with a specified input (PDB file for coordinates and MTZ file for diffraction data) and expects refined coordinates and map coefficients in return (in PDB and MTZ formats as well). All the wrapper has to do is to arrange for the input and output to be communicated from and to the Step Manager to and from the program of choice, extract the statistics and pass the information to the Step Manager. The program flow can then be resumed in a transparent way with no further complications. The same principle can be applied for every module. Moreover, it is quite straightforward to add additional steps to the procedure as long as the main decision tree is not altered. These make the design not only a good user tool, but also a good development test-bed for different algorithms.

2.1.4. Minimal user input combined with dynamic adjustment of default values. The required input for *ARP*/*wARP* is rather minimal and consists of the diffraction data labels and

the sequence file, together with phases that are available from an experiment or are extracted from a partial model. However, there is multitude of parameters with default values that can be adjusted from the user interface. The only way to know how to adjust these parameters is through experience: not only 'general crystallographic wisdom', but rather in-depth knowledge of and experience with the particular case in hand. Typically, users execute the program once with default values and based on the examination of the log files they perform additional runs with adjusted parameters, *e.g.* tighter restraints or different cutoffs for atom addition. The new architecture allows the decision modules not only to provide feedback for the subsequent steps but also to adjust parameters based on the output and statistics of previous steps. In this sense the control program behaves as an expert system. The new architecture also allows users to take steps 'backwards' if certain decisions turn out to be suboptimal: a situation that is often faced by human users and will certainly arise at least in the first incarnations of the 'Decision Functions'.

2.2. Main-chain tracing

There is high activity in the implementation of new algorithms for main-chain tracing, which will not be discussed here since they are beyond the scope of this publication. It has to be noted, however, that no matter how good a tracing algorithm might become, it will almost invariably produce a number of main-chain fragments that have to be assigned to the protein sequence, unless of course the whole main chain can be fully built in one go. Once the positions of fragments in sequence are known, *i.e.* they are 'docked', we can not only build a more complete model including side chains, but also attempt to build less ordered loops of the main chain. These issues are discussed in some detail in the following paragraphs.

2.3. Sequence docking (assignment)

The first step in docking pieces of the main chain to a given sequence is to make some plausible guesses about the residue type associated with the built C^{α} atoms. The initial residue-type guesses made by *ARP*/*wARP* are based on the free atoms found within the vicinity of each C^{α} atom. For each C^{α} atom of every fragment a 'connectivity vector' is calculated showing how this C^{α} atom connects to free atoms, if at all. These 'observed' connectivity vectors are then compared with the 'theoretical' connectivity vectors of the 20 standard amino acids. Based on the similarity of the observed connectivity vector to each of the 20 theoretical vectors a (pseudo) probability is calculated for each residue in the traced structure to be each of the 20 standard amino acids. We refer to these values for one C^{α} atom as the 'probability vectors'. The probability vectors for the C^{α} atoms of each fragment constitute a 'probability matrix'. Details of the computation of connectivity vectors, normalization issues (which resemble those proposed by Terwilliger, 2003*b*) and a pattern-recognition-based approach for 'true' probability estimates will be discussed elsewhere (to be submitted).

The values of the probability vectors contain large errors. Depending on the resolution, in 70–90% of all cases the amino acid that is predicted to be the 'most probable' one at a given position is incorrect. However, as long as the correct residue type is among the top choices for most positions, the fact that stretches of a number of sequential residues need to match to the sequence means that in practice the correct solutions are nevertheless found. For example, consider a C^{α} atom of a Phe residue. Although the calculated probability of this residue being a Tyr or a His may well be greater than the calculated probability of it being a phenylalanine, the latter probability is nevertheless likely to exceed the probability of it being a smaller residue such as Ala or Ser. The information content here ends up as this residue is 'likely to be big and bulky' and 'unlikely to be small'. In other words, the probability assigned to that C^{α} as being in a Phe residue has a high value, even if it is not the highest.

The power of the 'probability matrices' is explored in the context of a 'sliding' algorithm that has been inspired by the 'slider' options in the *O* program (Zou & Jones, 1996). A fragment is 'placed' in the sequence and the values of the probability matrix for that sequence context are retrieved. For example, let us consider a fragment of five residues and a short sequence AGYAWGAAGF. The fragment is placed in the sequence AGYAW. For residue one, the probability of it being an Ala is retrieved from the probability matrix, for residue two the probability of Gly is retrieved and so on. These probabilities are then multiplied (in practice, their logarithms are added) and this gives the probability of that fragment being in that position. The fragment is then 'slid' to the second position of the sequence (GYAWG) and the probability of it being in this position is computed. This procedure is iterated for every position in the sequence and every fragment. Finally, we compute a 'contrast score' for each fragment that shows the contrast between the 'most probable' and the 'second best' choices. The 'contrast score' would be equivalent to the likelihood ratio in probabilistic terms. The rationale is simple: if the 'best' choice is considerably better than the 'second best' choice, it is very likely to be correct. This approach is better suited to this particular problem than the more commonly used *Z* score where the 'best' choice is compared with the average. The fragment with the highest contrast score is then assigned to the appropriate sequence position. It is very important that at this stage the contrast score for every remaining fragment is recomputed, taking into account the fact that sequence positions already occupied by the fragments that were previously placed are no longer available. This procedure, which is summarized in Fig. 2, is iterated until all fragments are docked or until the contrast score is too low to allow unambiguous placement.

2.3.1. Handling of non-crystallographic symmetry. The geometric redundancy generated by the presence of non-crystallographic symmetry (NCS) can be exploited in model building to aid main-chain tracing and side-chain docking, especially in cases where the quality of the electron density varies between otherwise similar NCS-related protein chains.

On first sight, the sliding algorithm that is described above has a limitation when NCS is available owing to the usage of the contrast score. For example, in the case of twofold NCS the sequence is twofold redundant. Thus, if the input sequence was simply repeated twice, all contrast scores should be zero, since the first and the second score would be identical. Simply computing the contrast between the first and the third scores causes trouble later along the line. To handle NCS, we devised a method in which the same sequence has an occupancy equal to the NCS number. Every time a fragment is docked the occupancy for the newly occupied sequence positions is reduced, until the occupancy for a position reaches zero and fragments can no longer be docked there.

Using fragments that align to the same part of the sequence but in different NCS-related molecules, it is straightforward to derive the NCS operators. Since it is common for the main tracing algorithm to trace NCS molecules with differences (reflecting both the varying quality of the electron density and the fact that NCS-related molecules often show genuine differences), the fragment alignment multiplicity and the NCS operators are combined to generate a master molecule with maximum sequence coverage. That master molecule is then used as a template to increase the completeness of each individual fragment by a copy-and-paste mechanism (which can be more accurately described as match-and-extend; Fig. 3).

The calculated NCS operators can also prove very powerful for crystallographic refinement within *REFMAC*. The NCS operators can also be used to intersperse electron-density averaging cycles during the model-building process. Such an approach would be most powerful for addressing low-resolution cases.

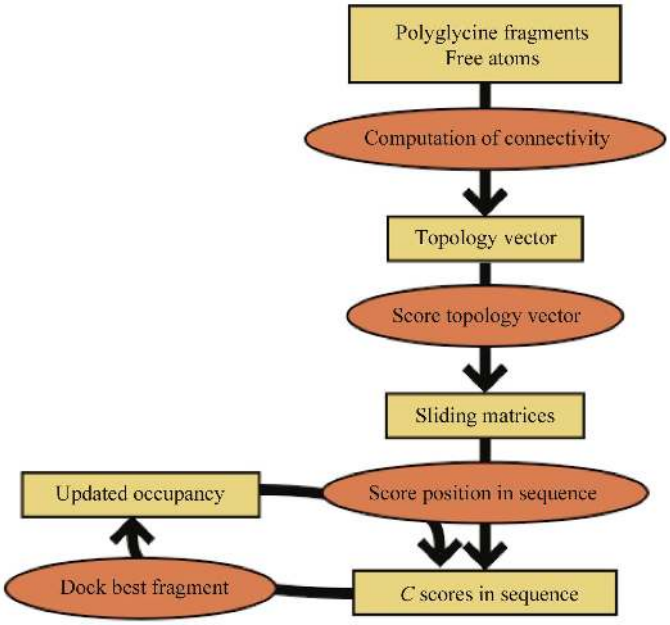

Figure 2
Scheme for side-chain docking (sequence-assignment procedure)

2.4. Building side chains

Side chains represent around 40% of the scattering mass of a protein. While 'sequence docking' solves the problem of side-chain-type identification, it is still needed to position the side-chain atoms and refine them so as to fit the electron-density maps. We use the well established method of fitting common rotamers (Jones *et al.*, 1991) followed by real-space refinement.

2.4.1. Handling of antibumping restraints. It is essential to implement some sort of antibumping restraints in order to prevent side chains from occupying the positions of existing side-chain or main-chain atoms. Gaussian geometrical restraints have to be calculated repeatedly in subroutines

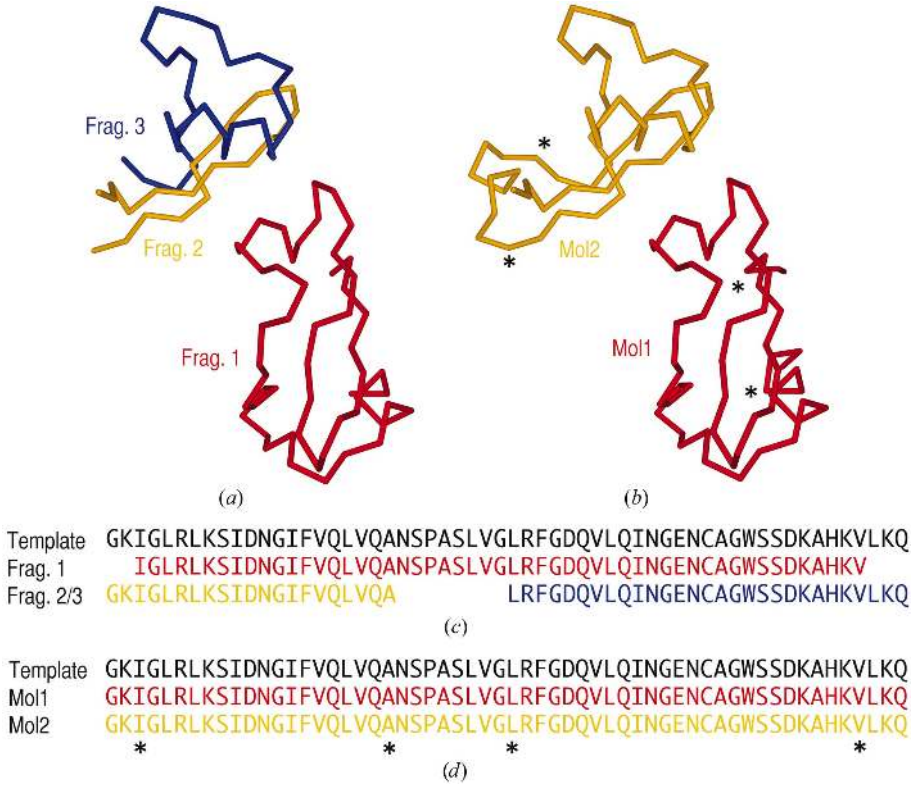

Figure 3
Exploitation of non-crystallography symmetry (NCS) coupled to side-chain docking in data collected from a syntenin crystal. Frags. 1, 2 and 3 are three fragments produced by the main-chain tracing algorithm (*a*), while Mol1 and Mol2 are produced by NCS-based extension (*b*). Successful side-chain docking allows the construction of a sequence alignment *versus* the protein monomer (template) sequence (*c*) from which NCS operators can be calculated. These NCS operators can be used to build a template molecule which, when mapped back onto the fragments, leads to a more complete model in three-dimensional space (*b*) as well as in sequence space (*d*). Asterisks delimit the start and/or end of the extensions, except for the N- and C-termini.

that are iterated millions of times and are suboptimal for the specific case. We chose to use what we refer to as a 'real-space residual map', a method that has been proposed and implemented for real-space refinement in *O* (Jones & Liljas, 1984). A copy of the electron-density map is made in computer memory and all existing atom density is then subtracted from this map (which is in effect equivalent to 'masking out' neighbouring atoms in real-space refinement in *O*), resulting in a 'real-space residual map' (Fig. 4). Every time that a new side chain is placed, its density values are also subtracted. Since the target for side-chain fitting is the real-space correlation with the electron-density map, using the correlation with the 'real-space residual map' eliminates the need for geometrical antibumping restraints. However, it is important not to perform the side-chain placement sequentially but to first place well ordered side chains with clear positions and proceed later with side chains that are less well ordered. It has to be noted that the 'real-space residual map' is only useful in the context of the refinement scheme we propose, where side chains are refined one at a time in real space.

2.4.2. Rotamer fitting. For rotamer placement we use the 'Penultimate Rotamer Library' (Lovell *et al.*, 2000). All conformations are pre-calculated once. The best fitting rotamers are chosen first for all small hydrophobic and polar residues, then for aromatics and finally from shorter towards longer polar charged residues and Met. This way, we ensure that density is occupied first by residues that are usually well ordered and we only position residues that are more likely to be disordered at the end. The order in which different side chains are placed was initially based on 'common crystallographic sense'. We are currently revising that order in the context of the results obtained from the EDS server (Kleywegt *et al.*, 2004) and we aim to derive a statistically valid 'order' for different resolutions. For example, an examination of real-space fit of residues reveals that while between 2.8 and 3.0 Å the order should be Cys-Met-Pro-Trp-Thr-Ser-Val-Phe-Tyr-His-Ile-Asp-Leu-Asn-Gln-Arg-Glu-Lys, between 1.2 and 1.4 Å it should be Cys-Tyr-Trp-Phe-Thr-Val-His-Ile-Ley-Ser-Met-Asn-Pro-Asp-Gln-Arg-Lys-Glu.

2.4.3. Real-space torsional refinement. For real-space refinement of selected rotamers we chose a torsional parameterization in which all χ angles of side chains are allowed to change with no restrictions. The φ angle of the residue is also allowed to change, but with a strong restraint that does not allow movement greater than 30°. The advantage of allowing the φ angle to change is that a much better positioning of the β C atom can be obtained. This operation results in distortion of the chiral angles of the C^{α} atom, but in practice this is so small that it is taken care of by the next *REFMAC* cycle and can in principle be easily corrected within our code. The function that we optimize is flexible and can be as simple as a summation of the densities at atomic centres at the beginning of refinement (to increase speed) and as complex a real-space correlation function using TLS information (Winn *et al.*, 2003) towards the end of refinement (to increase accuracy). The *SIMPLEX* optimizer (Nelder & Mead, 1965) is used as it presents significant advantages for this problem: it does not need derivative calculation and has a large radius of convergence. Its only disadvantage is the frequent evaluation of the objective function, but in our implementation it can refine 10–20 residues per second given the size of the side chains and the chosen parameterization.

2.4.4. Double conformations. The current implementation can check whether more than one rotamers fit the density with similar scores and propose double conformations that can be considered for refinement at high resolution.

2.5. Building loops

The *ARP/wARP* automated model-building routines re-interpret the complete map at every step, discarding completely all chemical information associated with previously built atoms. In every building step, the complete model is rebuilt. This procedure is an excellent protocol for initial model-building steps, where portions of the early model are likely to be incorrect. Also, discarding the complete model and rebuilding it on the basis of the map is a good way of avoiding and reducing bias. However, after a significant portion of the protein model has been built complete rebuilding is inefficient for two reasons. Firstly, it is much faster to only build the missing (or likely incorrect, see below) portions of the model. Secondly, especially after the main-chain fragments have been assigned to the sequence, the additional information from the sequence registration can be exploited to build less ordered regions of the model.

The second reason is the key for producing complete models. The current formulation of the main-chain building algorithm in *ARP/wARP* can be phrased as follows: 'Find in

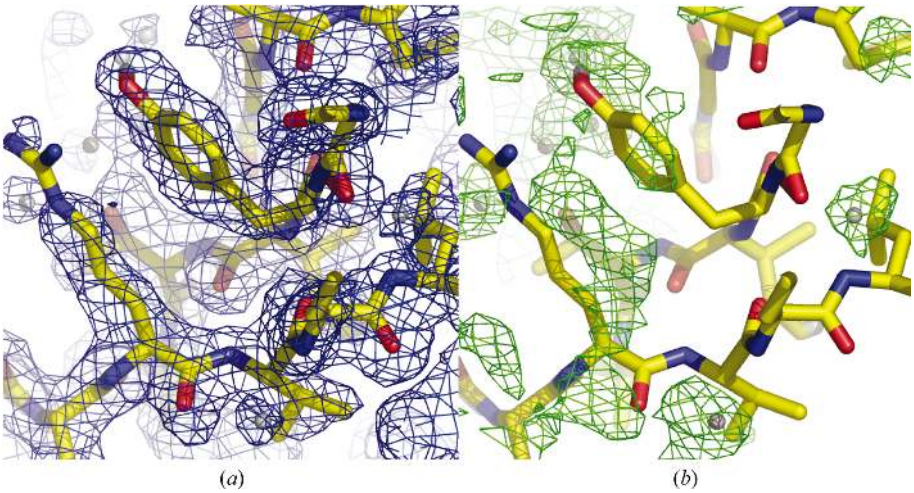

Figure 4
$2mF_o - DF_c$ (*a*) and real-space residual map (*b*) for a region of a map with side chains fitted and refined.

the map as many connected peptides as possible in as many fragments as necessary. The number of peptides is not exactly known owing to possible disordered regions. The start and the end of the fragments are not known. To avoid mistakes use very strict density and geometry criteria'. Once the main-chain fragments have been docked into the sequence, the same problem can be drastically reformulated. For example, if we have two fragments traced and it is known that one spans the sequence from residues 5–56 and the second residues 65–90, essentially the same algorithm as before can be employed but with additional information. For the example above: 'Find in the map nine connected peptides. The fragment of nine peptides starts at the Cα coordinates of residue 56 and ends at the C$^{\alpha}$ coordinates of residue 65. Find the fragment with correct geometry that has the most reasonable density'. These strong constraints (on starting and ending residue and hence the length of the loop) justify relaxation of the criteria used by the tracing program (*e.g.* with respect to the strength of the density) and thereby make it possible to trace loops with less well defined density.

2.5.1. Short loops. If the gap between fragments that are docked in tandem to each other is less than or equal to five peptides, then a semi-exhaustive search algorithm can be employed. The standard Ramachandran plot was sampled every 24°, which resulted in 92 φ–ψ angle combinations that are generously allowed. For fragments up to five peptides long it is computationally feasible to calculate all the above allowed conformations starting from the last preceding residue, filter those that connect correctly to the first succeeding residue and choose from these the one that fits the density best. This algorithm produces the best possible fit (since it is in practice an exhaustive search), which can be refined in a way similar to that described in §2.4.3. For six peptides or more the algorithm is computationally expensive (more than around 10 s on a standard PC in our current implementation) and its better to employ the ideas outlined in §2.5.2.

2.5.2. Long loops. The search algorithm explained in Morris *et al.* (2002) for main-chain tracing is being modified for this task. Candidate C$^{\alpha}$ atoms are computed in the area that the loops are likely to occupy using the 'residual density map' to avoid overlaps with the existing model. Peptides that fit between these C$^{\alpha}$ pairs are computed as in the main-chain tracing algorithm. The search algorithm is then employed with the additional constraint of a fixed start and end and a fixed number of peptides. The best C$^{\alpha}$ trace for that loop can then be used to build and refine the correct main-chain and side-chain atoms. The details of this approach are presently under development.

2.6. Iterative validation

ElAl is a new program designed to fit in the new *ARP*/*wARP* iterative execution scheme (Fig. 1) and to serve as a validation module that will communicate suggestions for rebuilding to the algorithms outlined in §§2.4 and 2.5. *ElAl* exploits information collected as part of the Uppsala Electron Density Server (EDS) project (Kleywegt *et al.*, 2004). In particular, EDS has enabled the collection of residue-type and resolution-specific statistics (average values and standard deviations) pertaining to the real-space fit (Jones *et al.*, 1991), which enables the use of very specific cutoffs to decide whether or not the real-space R value for a certain residue is unusual or not. For instance, a real-space R value of 0.4 in a 2.9 Å map would not be out of the ordinary for a lysine residue (Z score 1.7), whereas it would be very unusual for both a leucine residue (Z score 2.7) at the same resolution and a lysine residue at 1.9 Å resolution (Z score 3.6). *ElAl* calculates a number of properties (presently limited to real-space fit statistics and occupancy-weighted average temperature factors) for each residue. Subsequently, it checks for each of the properties whether or not the property for that residue exceeds a certain cutoff value. If it does, it is considered to be an outlier and a contribution is added to an empirical 'badness-of-fit' score (the higher this score, the more unusual the residue). When the entire model has been processed, the top N poor residues (where N has to be smaller than a certain absolute number and smaller than a certain percentage of all residues) are determined. These residues (and possibly their neighbours) can then be removed from the model prior to an automatic model-rebuilding step. Note that no sophisticated decision process is used to assess whether an outlier residue is an error or a genuine feature. The rationale for this is that if an outlier is a true feature that is supported by solid electron density then it will be rebuilt in essentially the same fashion (at the expense of only a marginal amount of computer resources, probably less than would be required for a more sophisticated decision-making routine).

In the future, the approach embodied in *ElAl* could be extended and made more sophisticated. For instance, one could calculate real-space fit values for side-chain atoms alone to find residues whose side chain needs to be rebuilt (but not necessarily their main chain). An obvious extension is also to include geometric criteria (*e.g.* bond-length and bond-angle violations, deviations from planarity, unusual chirality) as well as database-derived properties (*e.g.* Ramachandran outliers, unusual side-chain conformations). Finally, one could use a more sophisticated scheme to decide which residues to reject, *e.g.* favouring longer stretches of ill-fitting residues over isolated outliers. However, we expect the statistically derived criteria on the basis of the information gathered from the EDS to be the most valuable for our purpose.

2.7. A web-based facility for remote submission of *ARP*/*wARP* jobs

Traditional crystallographic practice has always presumed that software is installed locally and executed on on-site facilities. We decided to offer the crystallographic community access to a 16-processor Intel/Linux cluster for running *ARP*/*wARP* on an experimental basis. The reasons for this are threefold. Firstly, smaller laboratories often cannot afford to have the best hardware all the time, thus this service using a state-of-the-art machine may be very helpful to them. Secondly, laboratories often have problems with the local

installations or handling particular project cases. The common practice of many laboratories has been to contact the authors by e-mail and, after an electronic communication overload, exchange data files and log files to understand (and eventually solve) the problem. The remote-submission mechanism offers users a clear way to send their data to a well debugged installation and, if they wish, to ask the authors for intervention, without the hassle of file exchange *via* email, since all the data will be at the server. Finally, the server will be updated to have the latest debugged versions of the software available and serve as a test-bed for new algorithms.

The transfer of data is being performed *via* the HTTP protocol through the *CCP*4*i* interface (Potterton *et al.*, 2003). There are currently no security measures employed during data transfer. Users have to set up the job the usual way in the *CCP*4*i* interface and just choose the option for remote submission and type their e-mail address. Following remote submission, the user is immediately sent an e-mail containing a user name and a password, which she/he can use to login to the server through a web browser, submit the job and follow job progress interactively. All results can be downloaded at the end. There are different ways the data can be treated after the job has been run and the results downloaded by the user. The user can choose to delete all data without backup (allowing the *ARP*/*wARP* team to keep only the resolution limits and the final number of fragments and residues traced, for statistical reasons) if there are confidentiality issues involved. Most importantly, however, users can choose to share their data either just with the *ARP*/*wARP* developers or (advisable practice) with the wider crystallographic community. It is hoped that users will choose to allow dissemination of their data, which is crucial for the development of better software.

2.8. A web-based database for *ARP*/*wARP* optimization and development

As outlined above, it is very important for future developments to have access to a variety of true experimental data sets. Until now, we have used test data that have been made available in the authors' laboratory or by collaborators. However, we decided to appeal to the user community to share their experimental data with the *ARP*/*wARP* development team. We constructed a database where users can submit *via* a simple web-based interface their native diffraction data, phase information (in the form of experimental phases or as a molecular-replacement model) and the final model. This data will be archived and used for algorithm development, testing and 'training' of various algorithms to recognize patterns in a variety of map resolutions and quality. Provided that the user community cooperates, this resource will prove of enormous value for the development of *ARP*/*wARP*. It has to be noted that users can choose to share their submitted data not only with the *ARP*/*wARP* team, but also with the collaborators of the EU projects BIOXHIT and SPINE or the user community in general. We do hope that most users will actually choose to share their data with the wider possible audience, *i.e.* the crystallographic users and developers community.

We thank the NIH for funding algorithm development for the *ARP*/*wARP* package (SXC, FF, AP), grant R01 GM62612-01 and the EU for infrastructure support *via* the AUTOSTRUCT Framework V network, QLRT-1999-30398. RJM and VP acknowledge financial support from SPINE QLG2-CT-2002-00988. GJK is a Research Fellow of the Royal Swedish Academy of Sciences (KVA), supported through a grant from the Knut and Alice Wallenberg Foundation.

References

Badger, J. (2003). *Acta Cryst.* D**59**, 823–827.
Cowtan, K. (1998). *Acta Cryst.* D**54**, 750–756.
Holton, T., Ioerger, T. R., Christopher, J. A. & Sacchettini, J. C. (2000). *Acta Cryst.* D**56**, 722–734.
Ioerger, T. R., Holton, T., Christopher, J. A. & Sacchettini, J. C. (1999). *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, edited by T. Lengauer, R. Schneider, P. Bork, D. Brutlag, J. Glasgow, H.-W. Mewes & R. Zimmer, pp. 130–137. Menlo Park, CA: AAAI Press.
Ioerger, T. R. & Sacchettini, J. C. (2002). *Acta Cryst.* D**58**, 2043–2054.
Jones, T. A. & Liljas, L. (1984). *Acta Cryst.* A**40**, 50–57.
Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.
Kleywegt, G. J. (2000). *Acta Cryst.* D**56**, 249–265.
Kleywegt, G. J., Harris, M. R., Zou, J., Taylor, T. C., Wahlby, A. & Jones, T. A. (2004). *Acta Cryst.* D**60**, 2240–2249.
Kleywegt, G. J. & Jones, T. A. (1996). *Acta Cryst.* D**52**, 829–832.
Kleywegt, G. J. & Jones, T. A. (1997*a*). *Methods Enzymol.* **277**, 208–230.
Kleywegt, G. J. & Jones, T. A. (1997*b*). *Acta Cryst.* D**53**, 179–185.
Lamzin, V. S. & Wilson, K. S. (1997). *Methods Enzymol.* **277**, 269–305.
Levitt, D. G. (2001). *Acta Cryst.* D**57**, 1013–1019.
Lovell, S. C., Word, J. M., Richardson, J. S. & Richardson, D. C. (2000). *Proteins*, **40**, 389–408.
McRee, D. E. (1999). *J. Struct. Biol.* **125**, 156–165.
Morris, R. J., Perrakis, A. & Lamzin, V. S. (2002). *Acta Cryst.* D**58**, 968–975.
Morris, R. J., Perrakis, A. & Lamzin, V. S. (2003). *Methods Enzymol.* **374**, 229–244.
Murshudov, G. N., Vagin, A. A. & Dodson, E. J. (1997). *Acta Cryst.* D**53**, 240–255.
Nelder, J. A. & Mead, R. (1965). *Comput. J.* **7**, 308–313.
Oldfield, T. J. (2003). *Acta Cryst.* D**59**, 483–491.
Perrakis, A., Morris, R. & Lamzin, V. S. (1999). *Nature Struct. Biol.* **6**, 458–463.
Potterton, E., Briggs, P., Turkenburg, M. & Dodson, E. (2003). *Acta Cryst.* D**59**, 1131–1137.
Terwilliger, T. C. (2003*a*). *Acta Cryst.* D**59**, 38–44.
Terwilliger, T. C. (2003*b*). *Acta Cryst.* D**59**, 45–49.
Turk, D. (1992). PhD thesis. Technische Universität München, Germany.
Winn, M. D., Murshudov, G. N. & Papiz, M. Z. (2003). *Methods Enzymol.* **374**, 300–321.
Zou, J.-Y. & Jones, T. A. (1996). *Acta Cryst.* D**52**, 833–841.
Zwart, P. H., Langer, G. G. & Lamzin, V. S. (2004). *Acta Cryst.* D**60**, 2230–2239.