Enforcing secondary and tertiary structure for crystallographic phasing

Developing ARCIMBOLDO and BORGES

Massimo Domenico Sammito

ENFORCING SECONDARY AND TERTIARY STRUCTURE FOR CRYSTALLOGRAPHIC PHASING.

Developing ARCIMBOLDO and BORGES

Programa de Doctorat en Biotecnologia
Facultat de Farmàcia
Universitat de Barcelona

Massimo Domenico Sammito

Barcelona, April 2015

UNIVERSITAT DE BARCELONA
Facultat de Farmàcia

CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS
Instituto de Biología Molecular de Barcelona
Departamento de Biología Estructural
Unidad de Excelencia María de Maeztu

PARC CIENTIFIC DE BARCELONA
Crystallographic Methods Laboratory

ENFORCING SECONDARY AND TERTIARY STRUCTURE FOR CRYSTALLOGRAPHIC PHASING.

Developing ARCIMBOLDO and BORGES

Programa de Doctorat en Biotecnologia
University of Barcelona

Doctoral thesis presented by Massimo Domenico Sammito, B.S degree in Computer Science, Ms.C. in Bioinformatics, for the degree of Doctor (Ph.D) from the University of Barcelona.

Student:
Massimo Domenico Sammito

Thesis director:	**Tutor:**
Dr. Isabel Usón, ICREA Prof.	**Dr. Josefa Badia Palacín, UB Prof.**

April 2015

To my boyfriend Dario and my family

"Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza"

Dante Alighieri
Divina Commedia. Inferno, Canto XXVI, vv. 118-120
Firenze, Italia
1304-1321

SUMMARY

ARCIMBOLDO[1-4] is an *ab initio* phasing method for macromolecular crystallographic X-ray diffraction data, which combines location of model fragments such as polyalanine α-helices with the program PHASER[5] and density modification and main chain autotracing with the program SHELXE[6].
The method has been named after the Italian painter Giuseppe Arcimboldo (1526-1593), who used to compose portraits out of common objects such as fruits and vegetables (Figure 1-1). Following the analogy, ARCIMBOLDO composes an unknown structure by assembling small secondary structure elements, which are conserved across families of unrelated tertiary structure. Exploiting this method requires a multi-solution approach due to the difficulty to recognize correct solutions at early stages. Moreover, phasing a structure starting from partial information provided by such a small percentage of the total model (around 10% of the main chain atoms) is challenging and requires evaluation of alternative hypotheses under statistical constraints to avoid combinatorial explosion.
ARCIMBOLDO methods have proven successful in many cases of previously unknown structures[3] and also on a pool of test structures[4]. The program can accept any Sohnke space group and all the most frequent ones are represented in the pool of structures solved so far. In both studies data were collected in the most common protein space groups.

Figure 1-1 ***L'Ortolano***

Giuseppe Arcimboldo. Civic Museum "Ala Ponzone", Cremona, Italy

Data quality is crucial for phasing methods, and particularly sensitive for ARCIMBOLDO, where low resolution (worse than 2.1 Å) and lack of completeness (less than 98%) drastically decrease the chance of success.
Location of secondary structure elements is not indicated as phasing method for large structures or complexes (over 400 residues) unless very long helices are present and high resolution data are available. Such cases would require the placement of many fragments in order to assemble 10% of the main chain, which can lead to an unmanageable number of solutions. To approach correctly this different scenario we have implemented dedicated methods in ARCIMBOLDO_BORGES[7] and ARCIMBOLDO_SHREDDER[8]. These programs exploit libraries of folds or large search models and are described later in the text.
The current implementation[4], coded in Python, is deployed as a standalone binary, freely available under registration from http://chango.ibmb.csic.es/download. The binary is compatible with common Linux distributions and latest versions of the Mac OSX operating system. Users can find online manuals, tutorials and documentation in our website.
As of 30th April 2015, it has been downloaded 664 times and distributed to 121 research groups; furthermore, it has been installed in many European synchrotron facilities such as the Alba Synchrotron in Spain, the Diamond Light Source in United Kingdom and SOLEIL Synchrotron in France. The software is also available through SBGrid Consortium (https://sbgrid.org), a network of institutions across 19 countries, which provides a distributed grid network of computers to run structural biology software.

We have recently started a collaboration with the San Diego Supercomputer Center (http://www.sdsc.edu) in California (USA), to develop optimized and dedicated versions of the programs for their platform with the aim of addressing difficult phasing cases.
Due to this recent spread in the crystallographic community ARCIMBOLDO has been presented in many international conferences such as the International Union of Crystallography Meeting in Madrid (ES) 2011 and in Montreal (CA) 2014; the European Crystallographic Meeting in Bergen (NO) 2012, Warwick (UK) 2013; and many schools and workshops such as the International School of Crystallography in Erice (IT) 2012 and Macromolecular Crystallography School in Madrid (ES) 2014.

This thesis is organised in the standard scientific format comprising five main parts:

1. INTRODUCTION: introducing the theoretical topics directly or indirectly related to the contents of the thesis and also discussing the state of the art of current scientific production related to the objective proposed.
2. OBJECTIVES: listing all general goals and particular aims of the doctoral project conducted.
3. MATERIALS AND METHODS: detailing the hardware and software environment, including third party software and algorithms employed in the project.
4. RESULTS AND DISCUSSION: presenting all the produced algorithms, software, experiments and tests that correspond to the prefixed objectives.
5. CONCLUSION: summarising the whole project and listing its achievements by the end of the doctoral studies.

Chapter 1 in the Introduction, is dedicated to the subject of the X-ray macromolecular crystallography. After introducing the relevance of this science in the structural biology field and consequently in biomedicine, Section 1.2 focuses on the phase problem and on established methods employed to overcome it. Particular attention is paid to molecular replacement (Section 1.3) and *ab initio* methods (Section 1.5) presenting their strengths and limitations. Section 1.6 is dedicated to modelling, structure prediction and general use of external fragments in the determination of a protein structure. Chapter 2 introduces technical computational environments, such as grid networks (Section 2.2), supercomputers (Section 2.3), multiprocessing machines (Section 2.1) providing references for specialised reading. Each Chapter in the Results and Discussion section is dedicated to one of the designed programs implemented to solve and investigate set objectives. Chapter 3 is dedicated to ARCIMBOLDO and its underlying method to phase structures through unspecific secondary structure elements. The algorithm is described in detail, comprising underlying mathematics and geometry. Both the *lite* version (Section 3.1), as the simplest implementation for ARCIMBOLDO, and the extended distributed computing approach (Section 3.3), are explained. Section (3.2) deals with the testing of the deployed version considering both computing resources and generality of the method. The list of structures that, to our knowledge, have been solved with ARCIMBOLDO is presented in Section 3.4, discussing some of the cases that use particular features available in the program. Finally the solution with ARCIMBOLDO_LITE of a 13-fold superhelix[9] previously unknown structure is described in (Section 3.5). Chapter 4 deals with ARCIMBOLDO_SHREDDER, a method for phasing exploiting distant homologs. The method uses only crystallographic data to evaluate fragmented portions of the template model, the algorithm is described along with the statistics used, and designed formulas (Section 4.1). Section (4.2) is dedicated to the published case of MltE, which was solved with this method and from which the current program configuration has been derived. Chapter 5 deals with BORGES, a program to define, extract, superpose and cluster libraries of small local folds. First, the central mathematical constructions and operations

are described. It introduces the novel concept of Characteristic Vector[7] to describe secondary structure elements (Section 5.1), statistics to prove its generality (Section 5.2) and its use to describe local fragment distortion (Section 5.4). The method is presented in its first prototype version (Section 5.5) that allowed the generation of some basic libraries, described in Section 5.10. These libraries have been successfully exploited several times for phasing test and unknown structures. The present implementation (Section 5.6) reorganises procedures under the same general idea and introduces a new algorithm. Its new features allow the creation of more complex folds described in Section 5.11, namely knowledge-based DNA binding motifs of contiguous fragments with exposed loops. Structural comparison, superposition (Section 5.7) and geometrical clustering (Section 5.8) are crucial in the creation of such libraries and their algorithms and parameterization are then discussed.
Chapter 6 illustrates the ARCIMBOLDO_BORGES program. This software makes use of the BORGES libraries to enforce unspecific tertiary structure for phasing. Section 6.1 describes the single-machine implementation while Section 6.3 elaborates on the supercomputing and distributed version. As for ARCIMBOLDO, this Chapter also discusses test cases (Section 6.2) and previously unknown phased structures (Section 6.4). In particular two interesting cases are considered: the case of the coiled coil plectin fragment of the Rod domain (Section 6.5) and the case of a virus structure that is the first success, for an ARCIMBOLDO-based method, on an unknown all-β structure (Section 6.6). The last Chapter (7) is rather technical but can be interesting for developers and details a series of procedures designed and implemented in the programs presented. The management of the I/O (Section 7.3); grid support for many middleware systems and for remote access (Section 7.4), are arguments discussed and for which algorithms and protocols have been established. This Chapter also describes the development environment (Section 7.1) and the mechanism for deploying the program binaries (Section 7.2).
The thesis includes also the following parts:

- OUTLOOK: anticipating the on-going projects and elucidating about the possible developments starting from the achieved objective.
- REFERENCES: listing all the bibliography cited in the text. Hyperlinks are provided for the digital version.

APPENDICES: including personal scientific production, posters and communications presented, attendance at schools and congresses.

ACKNOWLEDGEMENTS

I would like to acknowledge Prof. Dr. Isabel Usón for supporting me during all my Ph.D studies and for the preparation of this thesis. She has stimulated my scientific curiosity and taught me all that I know. I would also like to thank Claudia Millán and all the members of the "Crystallographic Methods" group for the constant help and contributions to the work presented in this thesis.
Special acknowledgment goes to Dr. George Sheldrick for the invaluable discussions about my projects during the stages spent in his group and for providing computing resources and support. Thanks also to Dr. Lynn Ten Eyck for giving access to the San Diego Supercomuting Center, that opened new possibilities for my project, and to Dr. Kay Diederichs for the access to the remote grid of the university of Konstanz. Finally I would like to acknowledge Dr. Garib Murdushov, Dr. Randy Read and Dr. Arlie McCoy for the support in many steps of my Ph.D project. In particular for inviting me to visit their groups, where I had the opportunity to learn about detailed aspects of phasing methods.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

aa	Amino acids
CC	Correlation Coefficient
CPU	Central Processing Unit
cryo-EM	Cryo-electron microscopy
CV	Characteristic Vector
CVL	Characteristic Vector Length
DNA	Deoxyribonucleic acid
DSSP	Define Secondary Structure of Protein
FOM	Figure of Merit
FwMPE	F-weighted Mean Phase Error
GB	Gigabyte
GDT	Global Distance Test
GPU	Graphics Processing Unit
INITCC	Initial Correlation Coefficient
LLG	Log Likelihood Gain
MAD	Multi-wavelength Anomalous Dispersion
MB	Megabyte
MC	Matthews Coefficient
MD	Molecular Dynamics
MIR	Multiple Isomorphous Replacement
ML	Maximum Likelihood
MPI	Message Passing Interface
MR	Molecular Replacement
NCS	Non Crystallographic Symmetry
NMA	Normal Mode Analysis
NMR	Nuclear Magnetic Resonance
OS	Operating System
PCA	Principal Component Analysis
PDB	Protein Data Bank
RAM	Random Access Memory
RIP	Radiation-damage Induced Phasing
RMSD	Root Mean Square Deviation
RT	Running Time
SAD	Single wavelength Anomalous Dispersion
SAXS	Small Angle X-ray Scattering
SI	Sequence identity
SIR	Single Isomorphous Remplacement
SIRAS	Single Isomorphous Replacement with Anomalous Signal
SSM	Secondary Structure Matching

SVD	Singular Value Decomposition
SVM	Support Vector Machine
ZSCORE	Z-Score

INTRODUCTION

1 X-RAY CRYSTALLOGRAPHY

The origins of modern crystallography are connected to the discovery of the diffraction of X-rays by crystals[10, 11] for which Max von Laue received, in 1914, the Nobel Prize in Physics. X-ray diffraction was first used in the past century by William Henry Braggs and his son William Lawrence to determine the three-dimensional structure of crystals of inorganic compounds[12] and elements, as in the structure of diamond[13]. Since that discovery, crystallography has become an essential tool of investigation throughout the sciences, as it provides conclusive information on molecular structure down to the atomic level. Furthermore, crystallography uniquely complements structural information from other methods suiting different conditions, such as nuclear magnetic resonance (NMR) in solution and solid-state studies, small angle scattering (SAXS) in solution or electron microscopy (cryo-EM) from complex systems. For such a species as us, humans, for whom in most cases vision constitutes the predominant sense in our apprehension of the physical world, it is evident how the visualization of the main players in the chemical reaction processes inherent to nature and life, provides a powerful frame to relate all our functional knowledge and understand underlying mechanisms. Nevertheless, contrarily to what happens for instance in microscopy, the product of the crystallographic analysis is not a direct image of the molecules in the crystal. In the diffraction experiment, only the scattered intensities and not the phases from the X-rays, neutrons or electrons are directly measurable. However, the phases for each diffracted beam are essential to structure determination: without them the three-dimensional structure cannot be computed. This gives rise to **the phase problem**, central to crystallography. Obtaining the missing phases has ever been a quest in the crystallographic forefront. Even though in the last century a number of ways to solve the phase problem have been developed, the large number of parameters in today's complex problems and the frequent limitations in the data quality attainable in challenging studies still tend to hamper structure solution and phasing becomes a bottleneck in structure determination. The present work deals with the development of computing methods to solve the phase problem for macromolecular structures, relevant to structural studies underlying biotechnology and biomedicine.

1.1 Macromolecular crystallography in biomedicine

Although inorganic crystal diffraction patterns were obtained and solved at the beginning of the last century, the first macromolecules were only determined almost forty years later, using fibre diffraction. With that approach, the ground-breaking structure of DNA in B form was obtained by Watson & Crick[14] interpreting the invaluable diffraction patterns from Rosalind Franklin and Maurice Wilkins. Many others followed and contributed to the development in the field, and a number of polysaccharides, fibrous proteins and filamentous virus structures were revealed opening new perspectives to the study of their biological processes and functions and revolutionising medicine and its research. The first single crystal pattern of a soluble, globular, hydrated protein crystal was that of pepsin[15] recorded by John D. Bernal and Dorothy Crowfoot Hodgkin. The work was a tactical breakthrough because it showed that such crystals are better examined in the wet state. Advances in crystallization and improvements in single crystal techniques allowed the solution of the first globular proteins, like myoglobin[16], haemoglobin[17], lysozyme[18] or insulin[19]. These early achievements were already targeting molecules of medical relevance: haemoglobin is the oxygen transporter protein in our blood, lysozyme, is a bacterial-cell-wall lysing protein and insulin regulates the metabolism of carbohydrates and fats and it is directly related to one of the most widely spread diseases of our time in the western countries, the *diabetes mellitus*. At the end of the 1970s, Don Craig Wiley was one of the pioneers studying virus structures allowing to understand, for example, the main mechanism of attack of the influenza virus, the haemagglutinin [20, 21] action on the blood cells. Wiley continued the study of viruses throughout his career culminating in other important discoveries as the structure of the HIV-1[22] obtained in 1997. The determination of macromolecular structures has kept increasing the number of available three-dimensional models of the molecules that are relevant to a number of cellular processes and diseases. Nowadays, challenging structures such as ribosomes, huge viruses, membrane proteins, molecular machines or nucleic acids complexes are tackled by crystallographers. Protein structures are crystallized with a number of cofactors, inhibitors, activators, and reaction intermediates, interacting with them. Their mutations are also studied in order to discover relevant mechanisms of action that may shed light on the understanding of their role in diseases. The progress seen in this field has been possible thanks to the parallel advances in many sciences involved in crystallography and medicine, such as computer science and engineering, physics, materials science and instrumentation, molecular biology, genetics, biochemistry, biophysics, chemistry, and crystallography itself. And we are going towards a future in which new developments in all of these fields will also contribute to increase our knowledge on macromolecules that remain difficult to study with the present tools, such as membrane proteins, molecular machines or very large complexes.

1.2 The phase problem

Characterized by wavelength in the range 0.1 to 100 Å, X-rays are a form of electromagnetic radiation, and consequently they can be detected using image plates or other types of light detectors. This measurement is incomplete, as a wave is not only characterized by its amplitude, related to the measured intensity, but also by a phase, which is systematically lost in our measurement. This is what happens in the diffraction experiment, where only the scattered intensities and not the phases from the X-rays are directly measurable. However, the phases for each diffracted beam are essential to structure determination: without them the three-dimensional structure cannot be computed.

$$\boldsymbol{F(h)} = \int_{cell} \boldsymbol{\rho(x)} e^{(2\pi i \boldsymbol{h}\cdot \boldsymbol{x}) dv} \qquad (1)$$

$$\boldsymbol{\rho(x)} = \frac{\mathbf{1}}{\boldsymbol{V}}\sum_{\boldsymbol{h}} \boldsymbol{F(h)e^{(-2\pi i h\cdot x)}} \qquad \mathbf{(2)}$$

The fundamental relationship between experimental X-ray diffraction data and the electron density function in the crystal is given by the Fourier transform of the individual structure factors F. Each structure factor F is a complex number with amplitude and phase. The intensities of the scattered beams recorded are roughly proportional to the square of the structure factors. The objective of a crystallographic determination is to solve the inverse problem of determining the molecular structure within the crystal from the intensities of the scattered beams. To compute the corresponding inverse Fourier transform, with the structure factors as coefficients, their phases should be known as well as their moduli. As expressed in the Fourier transform, each atom contributes to all structure factors and a particular portion of the structure is not exclusively related to a subset of data or vice versa. This opens up the way for using methods that provide suitable approximations of the phases by exploiting different features of the molecules inside the crystal. If the molecules are small, as in chemical crystallography (often called "small molecule" crystallography), the problem tends to be heavily overdetermined and there is even the possibility of starting from random phases and then, by means of constraints[23], improve them until they are close to the true values. However, proteins crystals have intrinsic limitations (notably data resolution, solvent content, size of the structure) that decrease the data to parameter ratio and require methods addressing this lack of information.

1.3 Molecular replacement methods

One way to provide initial approximations to the phases is to derive them from a structure related to the unknown one contained in a crystal. Protein structure is known to be determined by the amino acid sequence as shown by Anfinsen[24]. Thus, a possible way to phase an unknown structure is to use a protein of similar sequence, such as a mutant, a homolog or even a component of a complex that has been previously characterized. This known protein is used as a search model that, if correctly placed in the unit cell, will account for the experimental data recorded. This method is known as Molecular Replacement (MR)[25-27]. In order to determine the correct position of the model in the unit cell, different target functions are in use [28]. Frequently, the search is divided conducting a rotation search to orient the model, followed by a translation search. A six-dimensional search can also be pursued, or even the simultaneous placement of several fragments. However, higher dimensional searches demand increased computation and challenging cases are best served by more sophisticated target functions, better accounting for errors derived of lack of completeness and differences in the model as well as for limitations in the experimental data. For instance, data resolution has a notable effect on the characterization of rotation and translation peaks in the Patterson that require local analysis[29]. Concerning the model, high scores in similarity, both in terms of primary and tertiary structure, increase the likelihood of a solution. The sequence identity can be used as an indicator of similarity[30] but structurally speaking the definition of similarity is more complicated: for example low sequence identity between two structures cannot exclude three-dimensional similarity as this tends to be more conserved than primary sequence[31], on the other hand two sequence identical structures can present different relative domain movements, which has to be taken into account during phasing. MR can use any source of coordinates or electron density as search models, opening up the way to

exploit structural information coming from experimental sources other than crystallographic studies. Nuclear magnetic resonance[32] (NMR), low angle scattering[33] (SAXS), cryo-electron microscopy[34] (cryo-EM) or even modelling. Cryo-EM is recently approaching resolution limits previously limited to X-ray crystallography[35]. As microscopy does provide a direct image at lower spatial resolution, tighter symbiosis between both fields should prove very advantageous. Another extremely interesting development, the use of modelled structures for MR has been shown to be feasible[36]. Modelling can be applied to increase the convergence when starting from a poor template from a distant homolog[37]. Furthermore if *de novo* structure prediction can reach enough accuracy, this approach provides an *ab initio* method[38].
Nowadays, MR has become the main phasing method for macromolecules, due largely to the number of already determined structures and their public availability, but also to the development of sophisticated mathematical treatment such as the one providing approximations to Maximum Likelihood (ML) functions[39, 40]. Fast and yet able to better model the differences and limitations of the known template, their use has decisively increased the MR radius of convergence.

1.4 Density modification and autotracing

Once MR has correctly located one or several models in the unit cell, this set of atomic coordinates is used to generate phases for the measured structured factors and to compute a three-dimensional electron density map. Initial phases originated from the model should be similar but will not be identical to those of the unknown structure. The starting model itself can be partial, incomplete, a single domain, for example, rather than the entire structure. All atoms contribute to scattering and the map generated from the partial phases in the model also carries information of the position of absent atoms, but the interpretation of these regions of the map is usually not straightforward. Often, the map needs to be improved before being interpreted and analysed for structure completion. Density modification techniques are general methods that lead to map improvement and aid its interpretability. The central idea behind density modification is that protein electron density maps have particular features that can be enhanced by applying prior information as a constraint to the phases or maps. There are two main approaches to density modification: ‘classical’ and ‘statistical’ methods.
Classical ones are based on performing alternative steps in real and reciprocal space that will introduce meaningful modifications to enhance features expected for a well-phased protein electron density map. For example, solvent flattening, first described by Wang[41], assumes that the noise arising from the solvent region of the crystal is heavily contributing to errors in the phases, because in such regions, electron density is essentially constant, and it is thus more likely that a correct map will have flat density in the solvent region. Application of solvent flattening either in real or reciprocal space generates a new set of phases that can be combined with the original ones, repeating this procedure until convergence. Another classical density modification technique is histogram matching[42], which basically consists in sharpening the density distribution so that its shape fits the corresponding skewed distribution calculated from sets of protein of similar expected structural characteristics, instead of a Gaussian histogram, characteristic from a noise map. One more illustration of the possible approaches in density modification is the “sphere of influence” algorithm[6], a simple strategy that is based on classifying different regions of the crystal as solvent or protein depending on the variance of the variance of the electron density on a sphere of radius equal to 2.42 Å. This distance relates atoms bonded to a common partner in a protein, and the algorithm provides a way to introduce general stereochemical constraints.

Statistical density modification methods are based on the introduction of the additional information about well-phased maps in the form of probability distributions, thus reducing model bias. Their application is usually more computationally demanding than classical approaches. Various software integrate one or more of the approaches indicated as the well-known programs DM[43], RESOLVE[44] and SHELXE[45]. This last software uses the sphere of influence and alternates density modification cycles with *autotracing*[46] steps leading to the partial interpretation of intermediate maps by tracing polypeptide chains. The algorithm aims to extend first helices, and frequent peptides within the conformational space constraints. Trace derived phases are combined with the previous ones[47] refining only one B-factor per residue. SHELXE also uses *no-go map* regions to avoid tracing in disallowed regions of the unit cell, as symmetry axes. Curved fragments can be generated by splitting and merging overlapping traced chains.

1.5 *Ab initio* methods

Direct solution of the phase problem is possible for small molecules (up to 200 independent non-hydrogen atoms), by enforcing probabilistic relations and evaluation of starting phase sets through figures of merit (FOMs). These methods are called *ab initio* or direct because no previous particular stereochemical knowledge or multiple wavelength measurements or derivative data are required. However, they are not suited for macromolecular structures for two reasons: their larger size negatively affects probabilistic phase relationships and, in general, their resolution is too limited.

The extension of direct methods[48] to larger structures (around 1000 independent atoms) was accomplished by the introduction of dual-space recycling methods[49], based on iteratively enforcing atomicity in real and reciprocal space, in order to constrain a few out of a large pool of trials starting from random atoms into correct, mathematically distinguishable solutions. These methods were successful in solving many antibiotic and disulphide-bridge rich peptides structures[50], which were too large for direct methods, even though atomic resolution data were available. They have also become extremely useful for experimental phasing (MIR, MAD, SIR, SAD, SIRAS, RIP), because they allow determining large substructures of heavy atoms or anomalous scatterers.

In the absence of atomic resolution data, constraints beyond atomicity are required in order to achieve *ab initio* phasing. Improvement of the experimental data may be obtained by data extrapolation to include non-measured reflections beyond the diffraction limit[51]. In general, density modification algorithms[52] are also highly efficient in aiding map interpretation in medium resolution data. At very low resolution the problem is formulated differently from the atomic resolution field and implementation of very different techniques[53] has been used to generate random sampling population from a density distribution to allow the determination of the molecule envelope and crystal packing of very large complexes.

1.6 Modelling and libraries

The fundamental conclusion that relates primary sequence with the tertiary conformation adopted by a protein is crucial in all fields intersecting structural biology. The “genetic code” itself explains the non-linear correspondence between sequence and structure even though the genetic information codified in synonymous codons still influences locally the three-dimensional arrangement[54]. Structural organization of proteins is codified in evolutionary hierarchies of families and superfamilies[55] and their domains are clustered[56] revealing conservation of functions across structures. This non random disposition justifies the search of conserved fold among known structures to impose stronger constraints during phasing, guide model building and improve refinement of X-ray structures, but also to study the physical properties of the folding process. Macromolecular structures solved worldwide through experimental methods are collected and made available via the Protein Data Bank (PDB)[57]. Many tools are provided to search and explore among over 100,000 deposited structures. PDBeMotif [58] is able to extract small continuous fragments, binding sites and ligands, while Dalí Server[59] allows browsing precomputed structural neighbourhoods. Both methods employ primary sequence and multiple alignments in their algorithms and aim to identify structural proximity with the template instead of structure variability. The search for conserved local fold and their organization in libraries has served improvements in model building, where the interpretation of electron density map can benefit from the statistical knowledge originated from conserved geometry. For example, Cowtan[60] explores Fast Fourier features through the definition of eneapeptides fragment libraries while Oldfield[61] searches for most frequent motifs by analysing clusters of Cα distance matrices. Structural constraints are also considered in refinement algorithms and tools are provided to generate restraints such as ProSMART[62]. Apart from distance matrices, spatial vectors have been explored to express geometrical relationships between fragments[63, 64] but they are not supplying biological or structural information. Machine learning algorithms such as Support Vector Machines[65] (SVM) are also used to extract frequency vectors relating conserved continuous peptide fragments in databases[66]. Geometrical characterization of fragment libraries is increasingly studied and their use is currently enforced in both experimental sciences and bioinformatics. The original coordinates from which these libraries are generated are usually coming from X-ray diffraction experiments, and their correct use and interpretation require some knowledge of the technique and its theory to obtain accurate results.

2 COMPUTING RESOURCES

X-ray macromolecular crystallography demands calculations requiring adequate computing resources. The growth of computer engineering and computer science made possible the implementation of more complex algorithms leading to the solution of big complexes, transmembrane proteins, ribosomes and other challenging structures. Still, the amazing software produced in the early days was highly efficient, very well written code and most of these programs are still used today in many research groups. The impulse that many crystallographers have put into the study of algorithmic, programming and numerical optimization has had recursion in other modern fields such as bioinformatics. Nowadays, new software tends to focus on parallelization, the simultaneous execution of CPU instructions at a given instant time, more than on optimization. The cost of memory and processors are not anymore unattainable for research groups and, more importantly, fast network connections allow the spread of the cheaper cloud computing respect to the most expensive and sophisticated use of supercomputers. Cloud computing[67] stands for general services, software, slot of computing time available through a network of connected computer managed by apposite middle layers between software and hardware called middleware. While supercomputer[68] typically refers to a tight, usually more expensive, infrastructure of computing units connected by fast intra-networks and physically curated by specialized personal. Despite the radical advances our technology has made, a brute force solution of protein structure cannot be calculated. It is necessary to constrain the problem and interpret, extrapolate and analyse outcome to drive further calculations, in other words a correct use of computing resources is not to reduce time but to extends the boundaries of the computable.

2.1 Parallelization and threading

Parallelizing processes[69] means to break the sequence of instructions allowing multiple executions of operations at the same time. This is only possible if they are addressed to physically different computing units, which are not interfering with each other. Code must be written in such a way the computer scheduler can effectively execute this procedure, avoiding cross dependences and common shared variables. Usually a code can be parallelizable only if the underlying algorithm, regardless of the programming language employed, is logically structured in that way. Even if interrelations between separate parallel operations are minimized some joint operations cannot be avoided. For example, it

could be necessary to wait for all pending parallel processes to end before continuing the sequential execution of the code, or subtler, Input/Output access must be regulated among parallel processes to avoid deadlock situations or undefined state of the machine. This necessarily leads to the programming of synchronization locks and semaphores to regulate such situations.
Threading[70], instead, is softer and in certain aspects more secure, which pays off in Running Time (RT). Threads are not necessarily directed to independent unit cores, can share common memory area and take advantage of the overclocking of computing units. Actually, single instructions of different threads are alternated at high speed in the unit core offering a discrete parallelization effect and allowing to easily manage contingent communication between them.
In recent years, new parallelization models have been introduced employing the computing unit of the graphical processor (GPU). Both, open sources, OpenGL[71], and commercial, CUDA[72], are widely used in companies and in scientific research but they require more expensive machines and specific programming training having a different hardware interface.

2.2 Distributed computing over a grid network

Grids are networks of connected workstations in which a program can distribute transparently its own calculations. Computers, which are the bottom layer of a grid network, do not necessarily require the same hardware/software configuration. Such a situation is common especially in scientific research where many computer grids are the result of collaborative efforts among small groups. It turns out that a middle layer is necessary as interface between the physical hardware implementation of the grid components and the user top layer in which all machines of the grid are simply seen as computing entities able to process a specific task. This middle layer, called *middleware*, is the central core of the model and has to provide job submission, queue managing and file transfers within the network. Different middleware is available such as the widely used HTCondor[73], SGE/Opengrid[74] or Torque/MOAB[75] grid. Each implementation differs in terms of services, commands and required configurations but presents four minimum components:

- A *submitter*: a machine, or frontend, from which jobs can be submitted and where output must be collected after the processing of task jobs in the grid. A submitter itself can be (or not) part of the grid, that is can also receive tasks to execute.
- A *queue manager*: a machine to collect, list and order pending jobs from all the users involved. It should allow the user (or the automatic program) to interrogate the queue for inspecting the current state of a job and modifying that state if needed. For instance, a user may request to remove a job, decrease its priority or hold its current state until release.
- A *scheduler*: a machine that extracts the highest priority job from the queue list and assigns it to a specific executor machine respecting the job requirements and distributing the load charge evenly throughout the grid. It also has to manage file transfer between the submitter and the executor machine and check the formal correction (the system exit of the process) of the task execution.
- An *executor*: a machine that receives a job task, usually the starting of a process or the execution of bash commands, and produces results. It can simultaneously be a submitter.

Modern middleware offers supplementary services to batch queue clusters of jobs, redirects input or output in special ways and even manages checkpoints from which to restart interrupted tasks.

2.3 Supercomputing facilities

From all the proposed alternatives, supercomputers are without any doubt the most powerful and fascinating computing resources. Supercomputers are characterized by sets of identical computing units, usually located in the same building, all directly connected with high rate optical fibre. They are constantly monitored and maintained by an expert team who ensures synchronization and correct functioning. The single computing units are generally high quality machines and optimal performance is obtained by cooling systems installed in the dedicated rooms. Supercomputers by themselves are not meant for parallelization; in fact they may act as single, super powerful computers in which complex processing can be performed. Numerical weather prediction[76] or real-time image and video processing[77] involve complex mathematical models containing nonlinear partial differential equations impossible to solve with analytical methods. Instead, they can be numerically estimated provided enough computing power is available. In structural biology this same complexity can be achieved by *de novo* protein folding or by molecular dynamics simulations[78]. Although conceptually all these operations illustrate well the relevance of supercomputers in research, their use is not jut limited to the solution of complex tasks but on the contrary is widely devoted to the execution of hundreds, thousands or billions of atomic operations in parallel. In this configuration the supercomputer works as grid computer networks with the evident difference of the hardware/software quality and the possibility to parallelize more than simply external processes, or bash commands. For a human user it is more intuitive queuing several tasks as separated processes and processing some input to produce output files but in fact, for a computer program accessing the local file system to produce input files and parsing and interpreting output files is a bottleneck in RT, constituting extremely slow operations compared to an operation in RAM. To overcome this problem, supercomputers widely support Message Passing Interface[79] (MPI), a standard protocol of communication between machines, that is supported by all modern programming language such as Python (mpi4py), Java (openmpi) and C (mpi). Through this interface, it is possible to write parallel code directly connecting a task with the program and functions skipping any unnecessary Input/Output interface with the file system. Many attempts[80] have been done to port sequential programs into parallel MPI, but best results are achieved with a valid parallelization design and a correct implementation supporting these facilities.

OBJECTIVES

The overall goal of this thesis is to develop crystallographic phasing methods enforcing secondary and tertiary structure as a constraint, with the aim of solving practical cases of structures where conventional methods fail. Hence, to contribute to the scientific community through the implementation of these new algorithms into distributed software.
In detail, we can distinguish methodological objectives from technical ones.
The first group involves the following specific objectives:

- To program ARCIMBOLDO, a software with the aim to extend *ab initio* phasing methods to larger structures diffracting below atomic resolution. Building on the experience gained in the group from a prototype programed in Perl, performing combination of sequential location in the unit cell of ubiquitous secondary structure elements, such as small α-helices, with density modification of the assembled substructures. Maintaining the underlying approach and the name, the goal was to recast the method in an efficient and flexible implementation in Python, introducing new algorithms to cluster, filter and prioritize intermediate solutions.
- To develop alternative strategies in ARCIMBOLDO exploiting the computational power of different hardware and software environments.
- To test efficiency and generality of the ARCIMBOLDO method on sample structures, tuning the internal parameterization of designed algorithms.
- To program ARCIMBOLDO_SHREDDER, an algorithm to generate, evaluate and exploit possibly conserved fragments from a distant homologous protein in the phasing of an unknown structure.
- To program BORGES, an algorithm to define and extract customized libraries of local folds to be used for phasing. In particular, to accomplish this objective requires:
    - To define a geometrical description for local folds, characterizing each component fragment with a mathematical representation of both its secondary structure features and their spatial relative location.
    - To statistically test this novel mathematical description to be general and precise.
    - To extract and analyse local folds of helical DNA binding motifs and the general motifs of parallel, antiparallel and parallel-antiparallel β-sheets, parallel and antiparallel contiguous helices, disulphide linked tetrapeptides.
- To program, ARCIMBOLDO_BORGES, a phasing method enforcing unspecific tertiary structure through the libraries of local folds previously computed.
- To solve unknown protein structures by the mean of each one of the proposed methods, ARCIMBOLDO_LITE, ARCIMBOLDO_SHREDDER and ARCIMBOLDO_BORGES.
- To analyse *a posteriori* the behaviour of the algorithms in each step of the runs.

Technical objectives comprise:

- To make software portable in both Linux and Mac OSX Operating Systems.
- To make software user-friendly improving input/output user-interface.
- To parallelize computation on a single workstation machine, on a distributed grid computer network and on supercomputer facilities.
- To test performance of the distributed version and produce on-line documentation.

MATERIALS AND METHODS

Crystallographic software

PHASER
http://www.phaser.cimr.cam.ac.uk/index.php/Phaser_Crystallographic_Software
Version 2.5.6 and 2.5.7 from both CCP4[81] and PHENIX[82] distributions.
Used for computing of MR rotation and translation function, packing symmetry, rigid body refinement, Gyre rotation refinement, normal mode analysis.
PHASER was always executed through keyword scripts, to ensure porting of our software also in grid computer networks.

SHELXE
http://shelx.uni-ac.gwdg.de/SHELX/
Version 2013 and 2014 from Shelx distribution server
Used for computing of density modification and autotracing, *pdb* optimization, FwMPE against the final structure and its relative origin shift.

COOT
http://www2.mrc-lmb.cam.ac.uk/personal/pemsley/coot/
Version 0.8.1 from CCP4[81] distribution.
Used for displaying macromolecules and the crystallographic unit cell and symmetry copies.

Programming resources

The workstation where the program has been developed and binary has been generated is an Intel i7 X980 @ 3.33 GHz with 8 physical cores sharing 12 GB of DDR3-RAM with the Linux distribution Ubuntu 14.04 installed.

Following programs are installed for developing:

PYTHON v. 2.6.5 and v. 2.7.6
https://www.python.org
Used for writing the code of all the programs and libraries.

PIP v. 1.5.4
https://pypi.python.org/pypi/pip/1.5.4
Used for installing python packages.

PYINSTALLER v. 2.1.1dev-67610f2-mod
https://github.com/pyinstaller/pyinstaller/wiki
Used for generating the binaries of the programs.

GCC v. 4.8.2
https://gcc.gnu.org/gcc-4.8/
Employed by PyInstaller to link system libraries.

Following python libraries are installed and used for developing:

BIOPYTHON v. 1.65
http://biopython.org/wiki/Main_Page
Used for reading *pdb* files and generating a structure object representing the protein with Bio.PDB.

MATPLOTLIB v. 1.3.1
http://matplotlib.org/1.3.1/index.html
Used to create dynamic graph plots during execution.

NTPLIB v. 0.3.1
https://pypi.python.org/pypi/ntplib/
Used to read the right time from the Internet or from the system.

NUMPY v. 1.9.2
http://www.numpy.org
Used for computing numerical calculations at high speed.

PARAMIKO v. 1.10.1
http://www.paramiko.org
Used for low level SSH machine connections.

PYCRYPTO v. 2.6.1
https://www.dlitz.net/software/pycrypto/
Used to ensure security in low level remote connections.

SCIPY v. 0.13.3
http://www.scipy.org
Used for machine learning algorithm, K-means clustering algorithm.

TERMCOLOR v. 1.1.0
https://pypi.python.org/pypi/termcolor
Used to display customized coloured output in the terminal output.

Computing resources

Local grid
http://www.ibmb.csic.es
At the Institute of Molecular Biology – Consejo Superior de Investigaciones Científicas we have configured a local HTCondor[73] grid comprising 120 nodes totalling 175 GFlops.

Remote grid
http://www.fcsc.es/index.php/es/
We had remote access on the supercomputer Calendula from the FCSCL in León, Spain integrating a grid with 248 nodes and 500 GFlops running both HTCondor[73] and SGE/Opengrid[74].

Software test
http://www8.hp.com/us/en/products/proliant-servers/product-detail.html?oid=5177949
Eight identical 8-core machines of an HP proliant BL460c blade system from the local grid, that were used for the ARCIMBOLDO_LITE and ARCIMBOLDO_BORGES tests as single workstations with dual quad core Xeon processors E5440, 2.83GHz and 16GB RAM. The Linux distribution installed was Ubuntu Server 10.04 LTS.

Gordon Supercomputer
https://www.sdsc.edu/News_Items/PR120711_gordon.html
We have access to the Gordon cluster at the San Diego Supercomputer Center, which is characterized by 1,024 dual-socket Intel Sandy Bridge nodes, each with 64 GB DDR3–1333 memory. Each node has 16 physical cores; nodes share data through the Data Oasis parallel file system, and are physically connected by Dual Rail QDR infiniband network. The Torque[75] distributed resource manager available at Gordon is used to queue jobs and request computing power resources.

Statistical analysis and plots

STATA[83] v. 12.1
http://www.stata.com
Used for function and scatter plots, and statistical analysis of data.

GNUPLOT v. 4.4
http://www.gnuplot.info
Used for 3d scatter plots.

GOOGLE CHART v. 2013
https://developers.google.com/chart/
Used to plot dynamic data and generation of histogram and function plots for the *html* output of the programs.

Molecular graphics

PYMOL v. 1.7.4
https://www.pymol.org
Used for figures of proteins and electron density maps.

RASMOL v. 2.7.5
http://rasmol.org
Used for visualization of macromolecules.

RESULTS AND DISCUSSION

3 PHASING THROUGH NONSPECIFIC SECONDARY STRUCTURE ELEMENTS: THE ARCIMBOLDO METHOD.

The advent of the dual-space recycling Shake and Bake algorithm[49, 84] made possible to solve small macromolecular structures by *ab initio* methods, from native, atomic resolution X-ray data alone. The underlying direct methods[85, 86] are based on probability theory, considering the structure composed of randomly distributed equal atoms to derive relationships among reflection phases from the measured intensities. The atomicity constraint is extremely powerful at atomic resolution and for structures with less than two hundred non-hydrogen atoms but it is seldom appropriate for macromolecular structures, which are usually diffracting to lower resolution and typically contain thousands of atoms in the asymmetric unit.
Our group proposed in 2009 a new method, ARCIMBOLDO[1], to extend *ab initio* methods to proteins twice as large as those solved by “shake and bake” methods and diffracting to barely 2.0 Å. A prototype implementation was distributed. The program was named after the Italian painter Giuseppe Arcimboldo (1526-1593), who used to compose portraits out of common objects such as fruits and vegetables. Following the analogy, ARCIMBOLDO composes an unknown structure by assembling small secondary structure elements, conserved across families of unrelated structures. The idea was to overcome the atomicity barrier imposing more stringent restraints that still would be protein-unspecific. Natural candidates were short, straight α-helices, as they are ubiquitous fragments in proteins. In fact, at present, the Structural Classification Of Protein server in Berkeley (SCOPe 2.05) classifies only 931 families out of 4756 as all-β structures. However, using helices to provide initial phases for density modification and autotracing algorithms requires first locating them in the unit cell. This is done resorting to MR functions to find a correct rotation and translation.

Exploiting this method requires a multi-solution approach due to the difficulty to recognize correct solutions in the early stages. Moreover, phasing a structure starting from partial information provided by such a small percentage of the total model (around 10% of the main chain atoms) is challenging and requires evaluation of alternative hypotheses under statistical constraints to avoid combinatorial explosion.
From that initial prototype providing proof of principle and first successes, a fundamental reorganization of the algorithm has been done and a completely new implementation improving usability, portability, and efficacy has been written; new algorithms were also introduced to cluster, filter and prioritize solutions.
The rest of the chapter focuses on the latest distributed version of the program and its results on test and unknown structures.

3.1 ARCIMBOLDO_LITE: single workstation implementation

The most popular version of ARCIMBOLDO is probably the *lite*, for phasing with secondary structure fragments. Deployed as a single binary, it was the first one characterized by an easy input/output interface, now shared by the other versions. Furthermore, it does not require a MYSQL database and parameterization is reduced to a minimum, providing automated, data dependent defaults.
Target structures for this version are proteins in the range of 90-300 aa, diffracting to a resolution of 2.0 Å or better and complete data (98% in the highest resolution shell).

ARCIMBOLDO provides suitable default values for supported parameters wherever possible. This minimizes the mandatory input required to run a job, but also maintains full user control by overriding specified parameters.
The user must input processed and scaled diffraction data in CCP4 binary *mtz* format, as well as SHELX *hkl* format. In both files, data may be given as either amplitudes or intensities.
A configuration file in the proprietary *bor* format should be also given as input. A detailed description is given in the Section 7.3. Through this file the user must name the input data files, indicating also the correct labels to address data signal and its sigma, and address third-party software, PHASER and SHELXE. The file also serves to override default parameters. The user should specify the expected crystal content by providing the molecular weight of the protein and the number of components in the asymmetric unit or, alternatively, the sequence in *fasta* format from which these two values will be inferred. The default search model is a straight polyalanine helix of given length, specified by the user, however other standard *pdb* models may be input. Even though the most frequent search fragment is a polypeptide, ARCIMBOLDO can be instructed to locate other species such as nucleic acids, cofactors or ligands.
The output of the program is summarized in an *html* file containing graphs and tables showing results for all steps. Detailed output description is also found in the Section 7.3.

Figure 3-1 shows the flow diagram of the program.

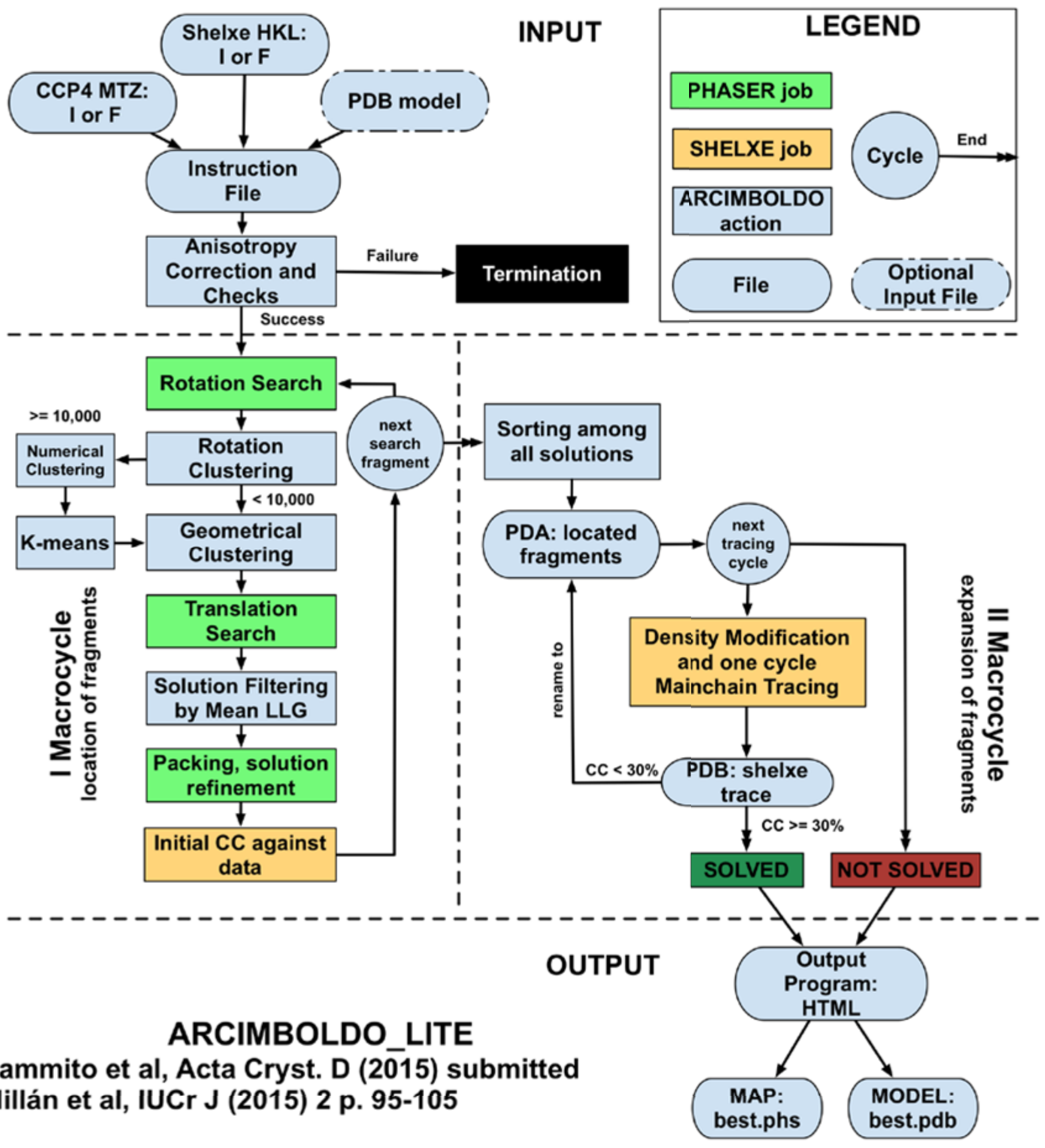


Figure 3-1 ARCIMBOLDO_LITE workflow

The program will automatically check user-defined parameters and terminate in case of fatal issues. The flow of ARCIMBOLDO_LITE can be divided in two macro-cycles. In the first one all fragments are sequentially located by PHASER in the unit cell. Default is to search two helices of fourteen residues. Solutions are grouped in rotation clusters and prioritized by considering both LLG and INITCC. In the second macro-cycle selected solutions will be treated with SHELXE to apply density modification and autotrace the remaining main chain. Traced solutions are evaluated against CC and best model and map are linked to the output *html*.

3.1.1 Internal failure checks

A series of checks are automatically performed to prevent common mistakes and to test all third-party software required:

- The instruction file is checked for formal correctness.
- In Linux distributions the file */proc/cpuinfo* is read to count the number of physical cores available in the workstation, while the command *sysctl -n hw.ncpu* is issued

in Mac OSX. If an external *pdb* model search has been specified, this *pdb* is checked to ensure it is written in a standard format and contains at least one atom.

- Existence and user accessibility of all given paths is checked. Python, PHASER and SHELXE versions are checked to be compatible with the deployed binary.
- If the user sets a different space group from the one in the *mtz*, it is checked to be one of the Sohnke groups in standard setting.
- Basic crystallographic constraints for the data are tested directly in PHASER and read from its output. For instance, compatibility of the space group with the cell dimensions, or improbable values of molecular weight. Existence and correctness of the *mtz* labels is tested performing an anisotropic correction.
- ARCIMBOLDO_LITE does not support resolution under 2.5 Å, so if this is the case a message is displayed in the *html* file and the program stops. For resolutions between 2.5 Å and 2.0 Å a warning message is displayed.
- A SHELXE job is launched to test both the input SHELXE line and the formal correctness of the *hkl* file. SHELXE also performs a statistical check on the normalized intensities, calculating the E^2-1 value[87, 88], and distinguishes between intensities and structure factors. To pass the test it must be possible to read the Initial Correlation Coefficient (INITCC) from the output file.

Failure in any of these tests causes termination of the program. The program will print to the standard output details about the error suggesting possible solutions. PHASER or SHELXE error messages will also be printed.

ARCIMBOLDO runs in two macro-cycles: the first one locates, with PHASER, all requested copies of the search models specified in the configuration file, while the second one reorders all partial solutions produced and selects a number of them equal to the available cores minus one. A core, in fact, is always reserved for the main program running cycle and its threads. Selected solutions are evaluated with SHELXE, which applies density modification and traces main chain atoms.

3.1.2 Rotation search of the first fragment

In the first macro-cycle all fragments are sequentially searched for.
A rotation search is computed with PHASER, and the resulting top 75% rotation peaks are written into an *rlist* file. Their FOMs (LLG and ZSCORE) are read from the PHASER output. As default, rotations rendering negative LLG are excluded.

Rotations are expressed in Euler angles but our software converts them to quaternions and 3x3 rotation matrices, to simplify geometrical operations.

$$\boldsymbol{q} = [q_0, q_1, q_2, q_3]^T = \begin{bmatrix} q_0 \\ q_{1:3} \end{bmatrix} \qquad (3)$$

A quaternion $q \,\varepsilon\, H$ is represented as a vector with an associated angle.
Basic operations such the *adjoint* norm and the inverse of the quaternion are computed by:

$$\bar{q} = \begin{bmatrix} q_0 \\ -q_{1:3} \end{bmatrix} \qquad (4)$$

$$\|\boldsymbol{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{5}$$

$$\boldsymbol{q}^{-1} = \frac{\overline{\boldsymbol{q}}}{\|\boldsymbol{q}\|} \tag{6}$$

Quaternion multiplication is not a commutative operation, and can be obtained by:

$$\boldsymbol{p} \cdot \boldsymbol{q} = \boldsymbol{q}_m(\boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{Q}(\boldsymbol{p})\boldsymbol{q} \tag{7}$$

where:

$$\boldsymbol{Q}(\boldsymbol{q}) = \begin{bmatrix} q_0 & -q_{1:3}^T \\ q_{1:3} & q_0 I_3 + C(q_{1:3}) \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{8}$$

To convert Euler angles to quaternions the following transformation[89, 90] is applied:

$$q_{323}(\phi, \theta, \psi) = \begin{bmatrix} c\phi/_2 c\theta/_2 c\psi/_2 - s\phi/_2 c\theta/_2 s\psi/_2 \\ -c\phi/_2 s\theta/_2 s\psi/_2 + s\phi/_2 s\theta/_2 c\psi/_2 \\ c\phi/_2 s\theta/_2 c\psi/_2 + s\phi/_2 s\theta/_2 s\psi/_2 \\ c\phi/_2 c\theta/_2 s\psi/_2 + s\phi/_2 c\theta/_2 c\psi/_2 \end{bmatrix} \tag{9}$$

where **323** stands for the Euler angles *z-y-z* convention[91] used by PHASER[39], ϕ, θ, ψ are *spin*, *nutation* and *precession* angles, expressed in radians and the following notation is used for compactness:

$$c_{\phi/2} \leftrightarrow cos\left(\frac{\phi}{2}\right), c_{\theta/2} \leftrightarrow cos\left(\frac{\theta}{2}\right), c_{\psi/2} \leftrightarrow cos\left(\frac{\psi}{2}\right) \tag{10}$$

$$s_{\phi/2} \leftrightarrow sin\left(\frac{\phi}{2}\right), \ s_{\theta/2} \leftrightarrow sin\left(\frac{\theta}{2}\right), s_{\psi/2} \leftrightarrow sin\left(\frac{\psi}{2}\right) \tag{11}$$

whereas the conversion to rotation matrix is:

$$\begin{aligned} R_{323}(\phi, \theta, \psi) &= R_3(\phi)R_2(\theta)R_3(\psi) \\ &= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & c_\phi c_\theta s_\psi + s_\phi c_\psi & -c_\phi s_\theta \\ -s_\phi c_\theta c_\psi - c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \end{aligned} \tag{12}$$

where the same notation as above is used.

3.1.3 Clustering rotation peaks

All rotations and their symmetry equivalent orientations are clustered in groups gathering rotations whose angular difference is smaller than 15°.
If the number of rotation peaks exceeds 10,000 a K-mean clustering[92] is performed, which compares numerical values of the Euler angles before geometrically comparing rotation references for each group so created. This prior numerical clustering reduces the number of geometrical comparisons under the assumption that similar values for Euler angles represent the same rotation. Nevertheless, introducing the K-mean clustering step involves a randomization procedure in the generation of initial seeds, which subsequently converge in a sub-optimal clustering. This compromises reproducibility of the results and therefore this step is performed only if direct geometrical comparison of all rotations is not affordable.

Algorithm 1: Numerical clustering of rotations

```
Input:
    list of Euler angles representing:    $rotations = [[\phi^i, \theta^i, \psi^i]]\ i \in [0, length(rotations)[$
Output:
        list of lists. Each sublist is a rotation cluster.

if length(rotations) > 10,000
        for i = 0 to length(rotations)-1
                quaternion$_i$ = get quaternion from Euler angles # Eq.(9)
                cp$_i$ = get cantor number of $[\phi^i \cdot 1,\ \theta^i \cdot 2, \psi^i \cdot 3]$ #Additional function
                rot_descriptor$_i$ = $[quaternion_i^1, quaternion_i^2, quaternion_i^3, quaternion_i^w, cp_i]$
        start_k = $\sqrt{length(rotations)/2}$
        return list of rotation clusters after applying K-means
                clustering with initial K=start_k
else:
        return list of lists. Each sublist contains one rotation
```

Additional functions:

```
Cantor_tuple
Note:
     Recursive procedure extended to real numbers
Input:
        $K = (k_1, k_2, \dots, k_n)\ \forall k_{1\dots n} \in R$
Output:
        $cp \in R$

if length(K) == 0
        return null
if length(K) == 1
        return $k_1$
$k_s = k_1$
return $\frac{1}{2} \times [Cantor\_tuple(k_{s+1}, \dots, k_n) + k_s] \times [Cantor\_tuple(k_{s+1}, \dots, k_n) + k_s + 1] + k_s$
```

Rotations are represented as arrays of five real numbers: four numbers to describe the rotation in quaternion format and a number that is the result of the Cantor tuple applied to the Euler angles and their relative positions.

K-mean clustering is performed for the list of all rotation peaks, and k parameter is initialized at the empirical value[93]: $k \approx \sqrt{n/2}$.
This initial k value is improved by computing the overall standard deviation inside each cluster and it is increased until no improvement is found upon new cluster addition. Independently from this intermediate clustering step, geometrical comparison of rotations is always required.

The rotation with the highest LLG is chosen as reference to find similar rotations defining a cluster. The non-clustered rotation with the highest LLG is taken as reference to test against all the remaining ones, until all rotations are associated to a cluster. To reduce the effect of the arbitrary choice of reference rotations, this geometrical clustering is iterated. A first run is performed using an angle threshold of 3° and this angle is incremented in each run until the final threshold is reached.
Rotations are compared to each other taking into account their symmetry equivalent orientations, determined by the crystal Laue point group. Thus, each time a comparison between two rotations is required, the procedure generates all symmetry equivalent rotations for one of the two, and tests all of them against the other to compute the angle between them.

The angle between two rotations is calculated by one of the following algorithms:

Algorithm 2: Numerical clustering of rotations

```
Input:
        rot1,rot2,structure,threshold
Output:
        True or False

atom1 = read the first CA of the structure
atom2 = read the last CA of the structure
atom3 = atom1
atom4 = atom2
rot_matrix1 = convert_euler_in_rotmatrix(rot1) # applying equation (12)
rot_matrix2 = convert_euler_in_rotmatrix(rot2) # applying equation (12)
atomR1 = rotate(atom1,rot_matrix1)
atomR2 = rotate(atom2,rot_matrix1)
atomR3 = rotate(atom3,rot_matrix2)
atomR4 = rotate(atom4,rot_matrix2)
        $\boldsymbol{v}_1 = atomR_2^x - atomR_1^x, atomR_2^y - atomR_1^y, atomR_2^z - atomR_1^z$
        $\boldsymbol{v}_2 = atomR_4^x - atomR_3^x, atomR_4^y - atomR_3^y, atomR_4^z - atomR_3^z$
        $\theta_{12} = arctan2(|v_1 \times v_2|, v_1 \cdot v_2) * \frac{180}{\pi}$
if $\theta_{12}$< threshold
        return True
else
        return False
```

Additional functions:

```
*rotate*
**Input:**
        atom,rot_matrix
**Output:**
        atomR

        $atomR^x = rot\_matrix_{11} \cdot atomR^x + rot\_matrix_{12} \cdot atomR^y + rot\_matrix_{13} \cdot atomR^z$
        $atomR^y = rot\_matrix_{21} \cdot atomR^x + rot\_matrix_{22} \cdot atomR^y + rot\_matrix_{23} \cdot atomR^z$
        $atomR^z = rot\_matrix_{31} \cdot atomR^x + rot\_matrix_{32} \cdot atomR^y + rot\_matrix_{33} \cdot atomR^z$
return atomR
```

Rotation Matrices: The Cα coordinates of the first and the last residue from the search model are extracted and a vector is created connecting them. A copy of this vector is made and each vector is then rotated by one of the two rotations to be compared. The rotation is performed by applying to the crystal coordinates of the vector the rotation matrix obtained from the Euler angles. Once both vectors are rotated, and their coordinates expressed again in orthogonal, the angle between them is computed as the two *arguments-arctan* of the vector product (or cross product) and scalar product (or dot product) of the two vectors. ARCIMBOLDO_LITE employs this algorithm as default.

Algorithm 3: Comparison of two rotations by angle between two quaternions

```
**Input:**
    quat1,quat2,threshold
**Output:**
    True or False

inv1= inverse_quaternion(quat1)           # applying equation (6)
pro = quaternion_product(quat2,inv1)       # applying equation (7)
$\theta_{12} = arccos(pro_{q0}) * \frac{180}{\pi}$

if $\theta_{12}$< threshold
        return True
else
        return False
```

Quaternions: the quaternion product of the conjugate quaternion of the first rotation and the quaternion of the second rotation is computed. The arccosine of the angular component of the product is the angle between the two rotations, which is then expressed in degrees.

Algorithm 4: Angle between paired characteristic vectors

```
Input:
    rot1,rot2,structure1,threshold,max_shift
Output:
        True or False, n_residues_of_elongation, direction_elongation

cvs = generate CVs for structure1        # applying algorithm 6
cvsR1 = rotate copy of CVs by rot1
cvsR2 = rotate copy of CVs by rot2

if length(cvsR1) != length(cvsR2)
        $cvsR_1 = [cvsR_1^0, cvsR_1^{floor(length(cvsR_1)/2)}, cvsR_1^{length(cvsR_1)-1}]$
        $cvsR_2 = [cvsR_2^0, cvsR_2^{floor(length(cvsR_2)/2)}, cvsR_2^{length(cvsR_2)-1}]$

smallest_angle = null
for shift = 0 to max_shift
        for direction = 0 to 1
                all_angles_list = empty list
                for t = 0 to length(cvsR1)-max_shift
                        if shift != 0 and direction == 0
                                $ind_1, cv_1 = cvs_1^{t+shift}$
                                $ind_2, cv_2 = cvs_2^{t}$
                        elif shift != 0 and direction == 1
                                $ind_1, cv_1 = cvs_1^{t}$
                                $ind_2, cv_2 = cvs_2^{t+shift}$
                        else
                                $ind_1, cv_1 = cvs_1^{t}$
                                $ind_2, cv_2 = cvs_2^{t}$
                                $\theta_{12} = arctan2\,(cv_1 \times cv_2\,, cv_1 \cdot cv_2) * \frac{180}{\pi}$
                        all_angles_list append $\theta_{12}$
                        smallest_angle = min(mean(all_angles_list), smallest_angle)
                if smallest_angle has been updated
                        save n_residues_of_elongation = shift
                        save direction_elongation = direction

if smallest_angle < threshold
        return True, n_residues_of_elongation, direction_elongation
```

Vector Distribution: Two copies of the search model are rotated. Each copy is rotated by one of the two rotations under comparison. Then for each one a distribution of characteristic vectors is computed, defined as vectors connecting centroids of three Cα and of three O for each overlapping tripeptide of the main chain search model. More details about these vectors is given in the BORGES chapter, but for the purpose of this algorithm we can consider them as standard three-dimensional vectors. These two distributions of sequential vectors can be compared computing angles between corresponding paired vectors and taking the mean of all these angles as the difference in degrees between the two rotations. The application of the rotation to the search models, as well as the computation of the angle between vectors is performed as described for the Rotation

Matrices algorithm. This algorithm is the slowest of all, and it is not manageable for a pool with more than 30 rotations. However it is the only algorithm at present in which it is possible to check, directly in real space, for rotations representing elongated helices. This is feasible because after having applied the rotation to both models it is possible to shift to the left or to the right the comparison between paired vectors, and that is equivalent to comparing a pattern of a helix rotating around its own axis and therefore representing an elongation.

A user specified Non Crystallographic Symmetry (NCS) rotation matrix in the configuration file will be used to cluster rotations in the same way as the crystallographic symmetry rotations.

These operations result in a sequence of grouped rotations (or combinations of rotations) ordered by LLG inside each cluster. From this point, solutions will be always associated to the original rotation cluster ID. Results are written into a formatted plain text file so that they can be used by the program to continue an interrupted run skipping all operations previously performed including clustering step.
If no rotations are available, for example in a case in which all the rotations are associated to negative LLG, the program will stop and print a warning message to the standard output.

3.1.4 Hardware dependent solution filtering

To prevent calculation of an unmanageable number of jobs for a single workstation, the number of clusters is limited to twice the number of machine cores and the number of allowed rotations within each cluster is limited to sixty-four times the number of cores.

3.1.5 Translation search

The PHASER Maximum Likelihood based Translation Function[40] is calculated for all selected rotations, grouped in clusters. Translations are internally sorted by their LLG, the top peak being defined as 100% while solutions with an LLG on the mean value are defined as 0%. All translations above the 75% cut off are saved in *sol* files, which are parsed and their fractional coordinates are associated to its LLG and ZSCORE. Each rotation can lead to different translation solutions, thus an internal coding is used by the program to discriminate combinations. This association is required to backtrack solutions and to analyse the evolution of their relative FOMs in all steps.
The program will skip the translation search for the first fragment in the case of the triclinic space group P1, where translation coordinates in the unit cell are arbitrary due to the absence of symmetry restrictions on the allowed origins.

3.1.6 Solution filtering by mean LLG

The number of solutions to evaluate is then reduced by filtering out all solutions with an associated LLG lower than the mean LLG value among all solutions in all clusters. This simple filter removes solutions, which are unlikely to be correct. Again, the number of allowed solutions in each cluster is proportional to the hardware of the actual workstation.

3.1.7 Packing symmetry check

Solutions are tested to be consistent to the crystallographic symmetry restrictions imposed by the space group. In particular, the equivalent positions are generated and clashing

atoms, defined as atoms at distances below 3.0 Å, are identified and counted. ARCIMBOLDO default settings do not allow any clash between atoms. This assumption is reasonable if we consider that search fragment are short secondary structure elements that are unlikely to overlap in correct solutions.
If no solution survived the packing filter, and not a single fragment was located, the program will terminate printing a message to the standard output. In case previous fragment searches have been successful, the program leaves the first macro-cycle and proceeds to the second macro-cycle in order to extend the available partial solutions with SHELXE.

3.1.8 Solution filtering

For the second time, solutions are analysed to filter out likely incorrect solutions. Solutions associated to a ZSCORE equal or higher than 7.5, are likely to be correct[40]. For this reason, all translations in the same cluster with a lower ZSCORE are automatically discarded. In addition, if the user has specified a minimum LLG for the translation step, this value is used as threshold to filter out unwanted translations. By default ARCIMBOLDO will accept all translations associated to a positive LLG.

3.1.9 Solution refinement

Rotation angles and translation coordinates are locally refined to reach the local maxima of the log-likelihood function. Solutions are grouped in rotation clusters and evaluated together to let PHASER merge equivalent solutions and reduce their number.

3.1.10 Solution prioritization

Solutions are sorted by their full resolution INITCC as calculated with SHELXE. The Correlation Coefficient (CC) is influenced by the size of model fragment with respect to the total mass of the protein. This implies that while it cannot be interpreted as an absolute value, the relative differences between sorted solutions can indicate promising ones. By default ARCIMBOLDO uses all atoms of a partial solution to evaluate CC, but it is often convenient to select a *pdb* optimization option[94] in SHELXE to find the largest subset of atoms, which maximizes the initial CC.

3.1.11 Successive fragment searches

Rotation and translation searches of the second and successive fragments are performed by fixing previous solutions, sorted by their refinement LLG. The program can limit the number of previous solutions to avoid producing an unmanageable number of PHASER jobs. The default is to not exceed 1,000 PHASER jobs in total.

The presence of fixed fragments slightly modifies the clustering algorithm. The cluster numbering is not defined anymore by a single integer id but by a tuple of integer numbers each one referring to previously identified or new rotations. ARCIMBOLDO defines combinations of rotations for the searched fragments. (m,n) and (n,m) combinations are treated separately, even if geometrically equivalent. The reason for this choice is that their FOMs can be different and, at this step, it could be difficult to identify the ones representing correct solutions. In this way the decision of selecting solutions between these two pools is delayed until a complete figure of merit describing both rotation and translation is available. The merging operation is performed after the crystal packing check and before the solution refinement is started, to filter out equivalent solutions.

PHASER can determine the presence of translational NCS and account for it in the target function[95, 96] while simultaneously placing the related copies. ARCIMBOLDO will recognize the tied solutions and skip from the first to the second fragment after the translation search. The option can be inactivated and in the case of coiled coils, where accounting for the translation through the model may be more successful than searching for individual helices.

3.1.12 Solution expansion

After all fragment searches are performed, ARCIMBOLDO moves on to the second macro-cycle. All solutions from all fragment searches are read and sorted in a single table by their INITCC. Solutions with more fragments located, and thus with more atoms, are likely to have higher CC, while partial solutions obtained at initial stages tend to be at the bottom of the table. Nevertheless, exceptions are possible. Sorting the table as described allows ARCIMBOLDO to solve cases where an inappropriate fragment search has been set up. Prioritized solutions will include the top CC and at least the two top LLG solutions even if their INITCC value was poor.

The *lite* version of ARCIMBOLDO in multiprocessing will limit the number of SHELXE expansion jobs to the available physical machine cores. From these, ARCIMBOLDO reserves one core for its main process and all the others are monitored and used for external SHELXE processes. Users can modify both the number of cores to employ and the number of solutions to evaluate. Different combined parameterizations allow, for example, testing twice the number of solutions in two cycles of parallelization cores. In addition, users can choose to launch more jobs than available cores, forcing the machine to activate hyper-threading.

SHELXE requires the starting model and the crystallographic data to perform density modification of the initial phases derived from the model and autotracing of the main chain atoms in the resulting electron density map. The program also takes as input several detailed parameters, which can be crucial for the success of the method.

ARCIMBOLDO generates a list of parameters tailored to the resolution of the data and the type of search model employed:

For data at ultra-high resolution, equal or better than 1.0 Å, map sharpening and low density elimination[97] is most effective in gradually improving the phases. A large number of density modification cycles is needed to reach convergence. Furthermore, indicating a low solvent content percentage is consistent with high resolution of the data. As resolution gets lower decreasing the number of density modification cycles, lowering the density sharpening parameter and increasing the solvent content is appropriate. Details are summarised in the Table 3-1.

The number of autotracing cycles, wherein SHELXE builds polypeptide main chain from the electron density map, is by default set to 8. Each cycle is processed as an independent SHELXE job, which takes as input the trace of the previous cycle. This allows monitoring each cycle independently and stopping the program once a solution is found. It also avoids that one single process occupies the CPU for a long period of time. If the user selects *pdb* model optimization in SHELXE, it will be performed after each autotracing cycle instead of just once, as it would happen in a standard SHELXE run.

Resolution range (Å)	**SHELXE line**
<= 1.0	-m200 -a8 -s0.25 -v0.5 -u2999 -t10 -q -y[*res*]
]1.0,1.3]	-m100 -a8 -s0.35 -v0.25 -u2999 -t10 -q -y[*res*]
]1.3,1.5]	-m50 -a8 -s0.45 -v0.1 -u2999 -t10 -q -y[*res*]
]1.5,2.0]	-m15 -a8 -s0.5 -v0 -u2999 -t10 -q -y[*res*]
> 2.0	-m10 -a8 -s0.6 -v0 -u2999 -t10 -q -y[*res*]
Legend	
-m	Density modification cycles
-a	Main chain autotracing
-s	Solvent content fraction
-v	Low density elimination factor
-u	Allocated memory in MB for fragment optimization
-t	Time factor for helices and peptide searches
-q	Include helical fragments as seeds for tracing
-y	Highest resolution for calculated phases from the input coordinate model

Table 3-1 SHELXE line parameterization in ARCIMBOLDO_LITE

Default parameters for SHELXE as they vary according the resolution of the data.

In order to avoid model bias, the model input into SHELXE will be used to calculate initial phases and discarded in favour of the new trace after each autotracing cycle. Side chains or non-polypeptide models will be lost after the first cycle. Nevertheless, if the start model is a cofactor, nucleic acid or heavy atom substructure, accounting for it throughout the expansion process may be needed. This can be done specifying -K in the SHELXE command line, which will instruct the program to retain the model input and to avoid tracing in the volume it occupies. When this option is activated, autotracing cycles cannot be distributed in sequential SHELXE jobs and all of them are performed in a single process.

3.1.13 Post-solution aftermath analysis

For developing purposes, ARCIMBOLDO is able to analyse the full run of a test case by comparing its results against the known structure. Specifying a path to the solved structure in *ent* format activates this option just after performing the rotation function search and clustering at the first fragment search. From this, reference phases for the measured structure factors and geometrical parameters are calculated.

ARCIMBOLDO can fail to solve at different steps, for example the correct rotation may be missed, or even if the rotation is correct it may not lead to the right translation. Furthermore, a correct partial solution may remain unnoticed and not be sent to extension in SHELXE, or be tested and not succeed to solve. A detailed analysis of every step in the ARCIMBOLDO procedure was carried out in order to recognize potentially correct solutions that failed to solve the structure and improve the software efficacy in future runs. Therefore the following analysis was performed:

Rotation function analysis

The procedure starts by extracting all the helices (or any other fold given as input search) in the *ent* file. For each one of them a Gyre[98] LLG is computed in PHASER. Gyre function allows calculating the rotation LLG value for a model in its current position. As the models were cut from the real structure their orientations must be optimal, thus computing this LLG value provides range values for the correct solution. For each of the perfect models the RMSD against the input search model is also computed to quantify the overall distance between them. High RMSD indicates that the search model employed could be too different from the real structure and this difference should be taken into account in the ARCIMBOLDO run by either increasing the RMSD parameter or changing the search model. The procedure tests the correctness of the clustered orientations by computing the RMSD between search model and real structure after applying to the first each orientation combined with the optimal translation. Only orientations that show a small RMSD value (less than 1.0 Å) are likely to produce a correct translation in the following steps of the ARCIMBOLDO run.

Translation function analysis

SHELXE is employed to evaluate translation solutions. The electron density map is generated from the located fragments and five cycle of density modification are applied. Then weighted mean phase error[99] (FwMPE) between resulting map and deposited structure is used as figure of merit to discriminate hits. Values of FwMPE smaller than 80º indicate not random solutions, and lower values increase the chances of success of the autotracing algorithm.

3.2 Testing ARCIMBOLDO_LITE

The recent spread of ARCIMBOLDO_LITE among many crystallographic groups prompted us to plan an extensive examination of the single machine implementation over a pool of test structures to refine internal parameterization and to optimize involved algorithms. The aim was to explore RT performance, subject to hardware and software specifications, as well as the effectiveness of the algorithm, this one conditioned by a correct parameterization.

To test ARCIMBOLDO_LITE RT performance two structures were chosen: the C-terminus of the human translation initiation factor eIF5 deposited in the PDB with the code 2IU1[100], and the structure of a surfactant protein from the foam nest of the frog *Leptodactylus vastus* deposited under the *pdb* code 4K82[101]. While both proteins have a similar molecular weight, 24443.10 and 23492.90 kDa respectively, 4K82 diffracts to a better resolution of 1.60 Å than 2IU1, which reaches 1.70 Å. Both structures could be solved with an older version of ARCIMBOLDO and required the location of at least two helices before producing interpretable electron density maps, thus their choice was appropriate as a general target case for the application. ARCIMBOLDO_LITE was run with default options, but searching for 4 helices of 14 residues.

Results of the ARCIMBOLDO_LITE runs on different machines are listed in Table 3-2. The values reported can only convey a general indication for a real job run in different working environment, but are not predictive of precise performance in other structural contexts. Running time (RT) is extremely correlated to the number of solutions produced at intermediate stages, which is unpredictable *a priori*.

HARDWARE	CORES	RAM (GB)	RAM per CORES (GB)	2IU1	4K82
				Running Time (hh:mm)	
Debian 6.0 Xeon X5560 @ 2.80 GHz	8	24	3.0	04:26	01:18
Debian 6.0 i7 2600 @ 3.40 GHz	4	12	3.0	05:34	01:36
Debian 6.0 i7 3930K @ 3.20 GHz	6	16	2.67	04:10	01:21
Debian 6.0 Xeon E5410 @ 2.33 GHz	8	32	4.0	07:46	01:34
Debian 6.0 2 Duo E4500 @ 2.20 GHz	2	0.51	0.26	32:19	08:45
Ubuntu 10.04.2 i7 950 @ 3.07 GHz	4	6	1.5	06:41	02:03
Ubuntu 10.04.3 Xeon E5440 @ 2.83 GHz	8	16	2.0	06:58	01:56
CentOS 7.1.1503 Xeon CPU E52650v2 @ 2.60GHz	16	125	7.8	01:55	00:32
Ubuntu 10.04.4 i7 X980 @ 3.33 GHz	6	12	2.0	05:01	01:21
openSUSE 13.2 Xeon E5-2680 v3 @ 2.50 GHz	24	16	0.67	04:34	01:00
Mac OSX Yosemite 10.10 I7-3720QM @ 2.60 GHz	4	8	2.0	03:16	01:10
Mac OSX Mavericks 10.9 I7-3720QM @ 2.80 GHz	4	4	1.0	08:54	01:40
				Mean	
				07:37	02:01

Table 3-2 ARCIMBOLDO_LITE performance

RT performance of the structures 2IU1 and 4K82 over a pool of workstations with different hardware and operative systems.

All workstations managed to solve the structures. Table 3-2 clearly shows that RT depends on the hardware specification of each workstation, whereas no particular correlation with the installed Operating System (OS) is observed. In particular, the number of cores influences the number of parallel external PHASER and SHELXE processes started, while the RAM can increase performance of each process. Nonetheless, RT is not linearly proportional to either of the two parameters but it is limited by the weakest component in the workstation including also the quality and speed of CPU and RAM. As example of an extreme case the 5th entry in Table 3-2 produces a RT of 32h:19m for the 2IU1 structure with a machine of 2 cores sharing 512 MB of memory: in this case calculations required to swap RAM memory contents to the hard disk, slowing down execution. Still, a large amount of RAM does not necessary lead to high performance as evidenced in the case of the 4th entry. The best performance derived from a workstation with 16 over-clocking capable cores sharing 125 GB RAM. This machine is dedicated to image processing during crystallographic data collection; the available memory is sufficient to ensure that all parallel processes will run at maximum possible CPU speed. Therefore, this test is intended to provide an orientation about appropriate hardware for typical ARCIMBOLDO_LITE targets.

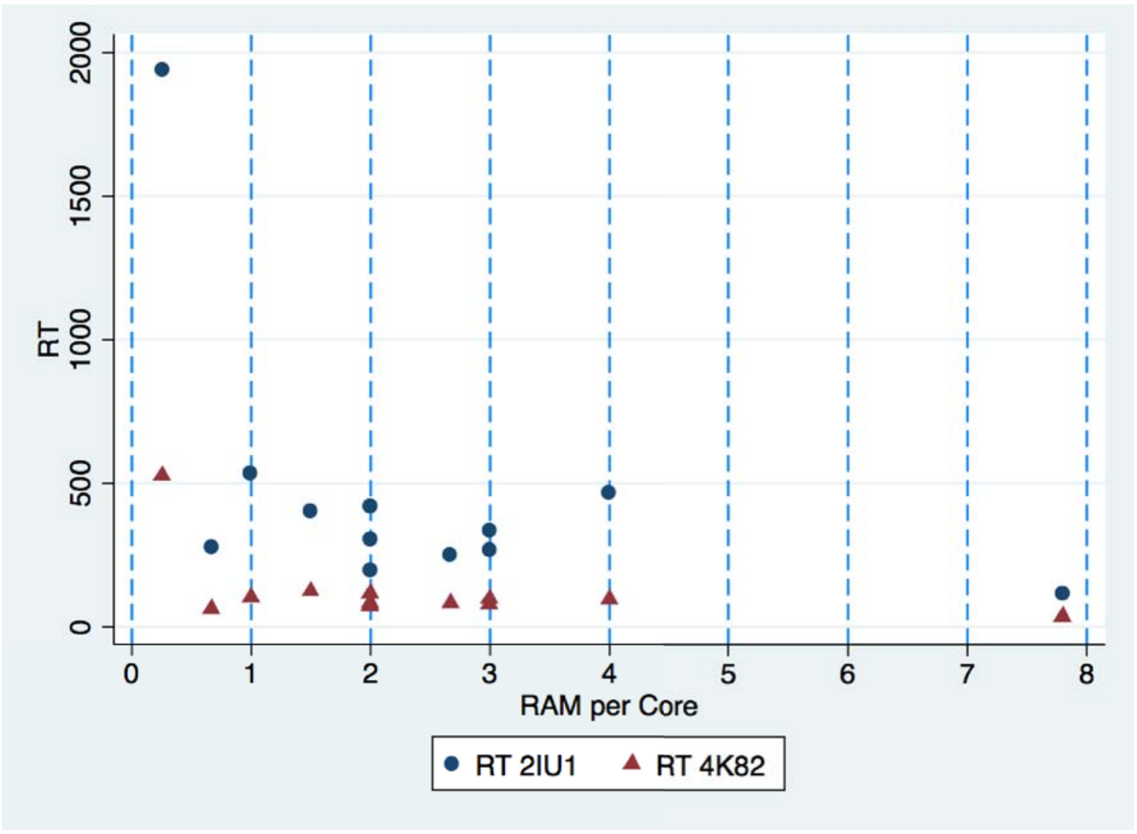


Figure 3-2 ARCIMBOLDO_LITE performance sampling on several workstations

RT for both 2IU1 and 4K82 is plotted against the ratio of memory per core. RT is expressed in minutes.

As suggested by Figure 3-2, a broader study with more samples would be required before venturing a detailed correlation between hardware and performance. Hence, with the present study it is only possible to recommend a minimum requirement of 2.5 GB per core for satisfactory ARCIMBOLDO_LITE performance on a 25KDa structure with several fragment search stages. However, independently of the RT performance, it must be considered that a machine with few cores will reduce the chances of success for ARCIMBOLDO_LITE, as solutions pursued through expansion will be accordingly reduced. In such cases, the parameter overriding default number of expansions should be increased. In conclusion, it is advisable to run ARCIMBOLDO_LITE on a machine with a minimum of 8 cores. The user may force the program to deal with more solutions despite the hardware limitation, or can limit the number of cores employed without reducing afforded solutions, although these configurations are not encouraged on the contrary the use of a suitable machine or a distributed computed approach, Section 3.3, is more indicated.

The second aspect of this study aimed to test the effectiveness of ARCIMBOLDO_LITE by evaluating its performance against a pool of 294 structures[38, 102]. These tests were run on the eight identical 8-core machines of an HP proliant BL460c blade system, as single workstations with dual quad core Xeon processors E5440, 2.83GHz sharing 16GB RAM. The Linux distribution was Ubuntu Server 10.04 LT.

The test pool held cases with resolutions comprised between 2.2 and 0.54 Å, sizes between 44 to 120 residues in the asymmetric unit and 41 different space groups, approximately following the frequency distribution seen in the PDB that is, with a predominance of $P2_12_12_1$ (Figure 3-3)

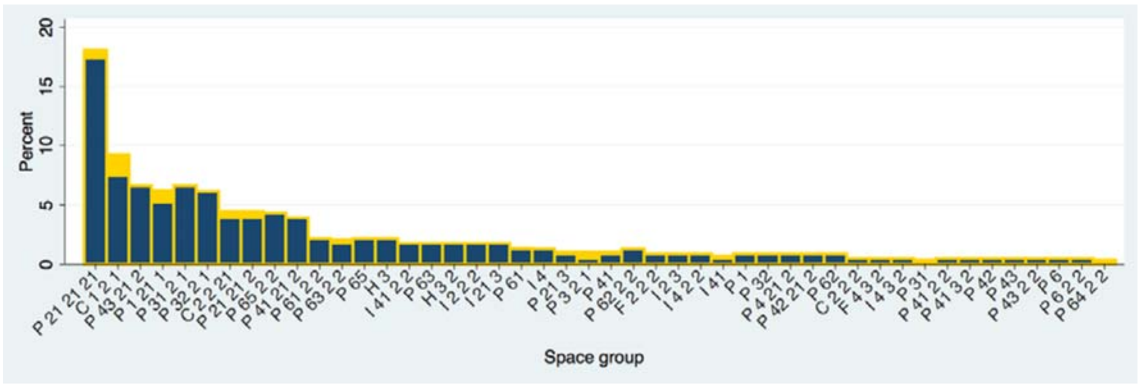


Figure 3-3 Space group representation in the helical pool test

Histogram graph displaying frequencies of space groups in the pool of 294 structures. Frequency is annotated as percentage respect to the total pool. Gold bars show values for the total pool of 294 structures while the blue coloured portion refers to the subset of 230 helical structures.

Table 3-3 summarizes the description of the test pool and its results.

Resolution (Å) Ranges	Structures	Helical Structures	Solved with ARCIMBOLDO	Success rate (%)
Total	294	230	139	60
1 - 0.54	24	11	11	100
1.3 - 1.0	43	33	24	73
1.6 - 1.3	78	62	38	61
2.2 - 1.6	149	124	66	53

Table 3-3 ARCIMBOLDO_LITE helical test solutions by resolution range

Structures containing helical fragments constitute 78% of the total test cases examined. The test pool is split into resolution ranges. For each range the number of helical structures is reported along with the number of cases solved with ARCIMBOLDO_LITE. The percentage of solved structure in each resolution shell is also displayed, indicating the dependency between data quality and success of the method.

230 out of 294 structures were suitable cases for ARCIMBOLDO_LITE being characterized by a minimum helical content of 8%. Running ARCIMBOLDO_LITE blindly on the pool, with a fragment search setup to find two polyalanine helices of 14 residues and using a common SHELXE parameterization resulted in 100 of the 294 structures being solved. SHELXE was set to perform 5 iterations involving 30 cycles density modification and autotracing, twentyfold increasing the time for random search and taking into account the presence of helices, solvent content of 55% and fragment optimization against the CC. Grouping structures according to data resolution and performing individual tests on the subgroups allowed to determine specific SHELXE parameterization for each resolution shell resulting in the values quoted in Section 3.1.12 and summarised in Table 3-1.
As an example the test case of 1L9L[103], Figure 3-4, is discussed.

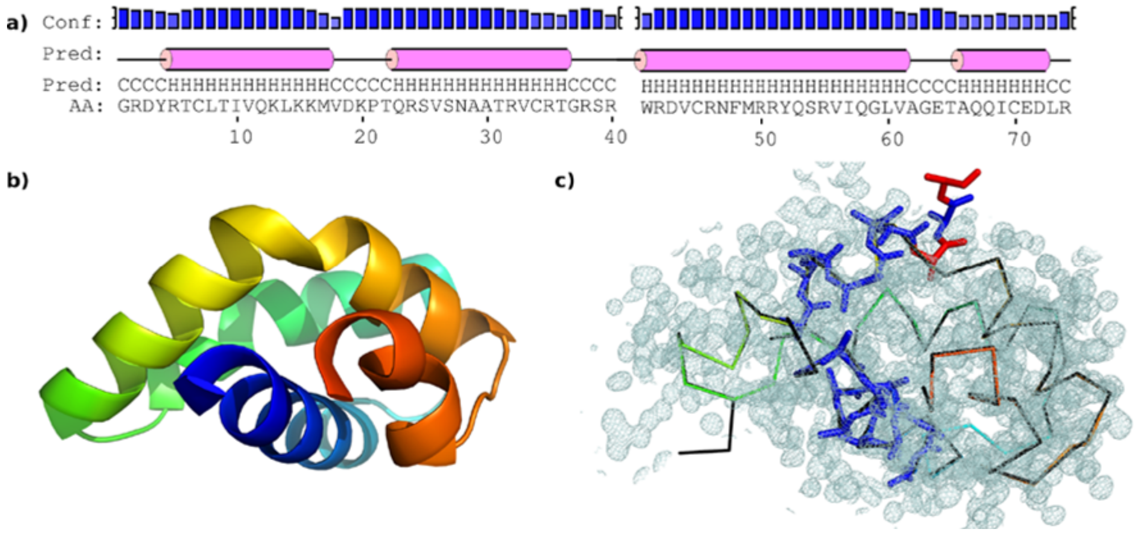

a)
Conf:
Pred:
Pred: CCCCHHHHHHHHHHHHHCCCCCHHHHHHHHHHHHHHHCCCC HHHHHHHHHHHHHHHHHHHHHCCCCHHHHHHHCC
AA: GRDYRTCLTIVQKLKKMVDKPTQRSVSNAATRVCRTGRSR WRDVCRNFMRRYQSRVIQGLVAGETAQQICEDLR
10
20
30
40
50
60
70
b)
c)

Figure 3-4 ARCIMBOLDO_LITE test case: 1L9L

The test case 1L9L, crystal structure of Granulysin at 0.92 Å: a) the secondary structure prediction indicates an all-helical structure; b) the cartoon representation of the protein with gradient colours displaying the direction from N-terminal to C-terminal (*rainbow display*); c) the structure (in *rainbow*) is represented along with the SHELXE electron density map (in cyan) and located helices represented as sticks (in dark blue). The main chain traced by SHELXE (black) fits accurately the deposited coordinates. Two residues were eliminated from the initial fragments and are displayed in red.

The Granulysin structure is a monomer of 74 residues and crystals have a solvent content of 20.6%, it was deposited in 2002 along with native data to 0.92 Å resolution in $P2_1$. ARCIMBOLDO_LITE was run searching for two helices of 10 residues, that led to a correct solution in SHELXE with 41.77% CC computed from 75 residues traced. The map produced is interpretable: from the solution obtained it is easy to build and refine correctly the remaining part of the structure. In this test case it was essential to increase the number of density modification cycles performed for each autotracing cycle in SHELXE. In fact, this algorithm can benefit from the high data quality, which significantly aids the autotracing procedure. After performing 200 cycles of density modification FwMPE decreases from 63.6º to 36.2º. At this resolution it is reasonable to assume a low solvent content, and in fact the solution was obtained approximating it to 35%, this value worked well although the real solvent content was even lower, but further increasing the solvent content will prevent to achieve interpretable map after density modification. It turns out that up weighting high-resolution density by the density sharpening factor *–v* in SHELXE helped in converging to an interpretable map. This algorithm is generally useful on experimental phases from anomalous or MAD data and is switched off in SHELXE when

phases are extracted from coordinates, but its use is recommended for such high quality data where details of the atomic resolution can be exploited. On the contrary, it becomes deleterious in other resolution shells, and should be used with caution with data below 1.5 Å. In these particular test cases, the fragment optimization procedure[94] internal in SHELXE was also used, resulting in the removal of 2 out of 3 incorrect placed residues of one helix. This last step it was not fundamental in this case, in fact a correct solution was also found without it, but it helped in converging faster.
Considering all results from the blind run and from the individual curated cases some considerations for the parameters have been extracted. In particular the correlation of some parameters with the resolution of the data allowed differentiating 5 groups:

- Ultra-high resolution (below 1Å): 200 cycles of density modification are requested for each autotracing cycle, solvent content is approximated at 25% and density sharpening is activated and a strongly weight is given to the highest resolution shell by a factor of 0.5.
- High-resolution (1.0 to 1.3 Å): density modification cycles are halved, solved content increased to 35% and density sharpening down-weighted at 0.25.
- Medium-high resolution (1.3 to 1.5 Å): density modification cycles are halved again, solved content increased to 45%, and density sharpening is almost deactivated at 0.1.
- Medium resolution (1.5 to 2.0 Å): density modification cycles are reduced to 15 iterations, to avoid bias in the map, solvent content is set to 50% and density sharpening is switched off.
- Low resolution (above 2.0 Å): It should be considered that structures diffracting at such resolution are not intended to be solved with ARCIMBOLDO_LITE. For this resolution shell an ideal parameterization becomes uncertain, instead more extensive SHELXE trials are advised. The default line in ARCIMBOLDO_LITE reduces the number of density modification cycles to 10, and increases the solvent content to an overestimated value of 60%, which enhances solvent flattening and may lead to a better interpretable map. Also for this range density sharpening is switched off.

The new parameterization implemented increased the number of successfully solved helical structures to 139 out of the pool of 230.
ARCIMBOLDO_LITE using helices as search fragments failed in 91 cases of structures containing at least one helix. This result may arise from a number of different reasons: inappropriate data quality despite the resolution limit, insufficient helical content or presence of more prominent features in the structure. Whereas data quality may be difficult to improve and in the case of our test set was predetermined, the latter suggested that helices should not be the only search fragments exploited in ARCIMBOLDO. It turns out that 5 of the test structures present predominant scattering features, in particular iron-sulphur clusters and a Heme group. 20 structures contain at least 4 disulphide bridges. ARCIMBOLDO_LITE was successful in solving 4 structures out of 5 tested by employing a chemical compound search fragment input as an external *pdb* model. Table 3-4 summarizes the tests.

	# Structures	# of Structures containing Fe, clusters or Heme Group	# Solved with ARCIMBOLDO	Success rate (%)
Total	294	5	4	80

Table 3-4 ARCIMBOLDO_LITE test performances with external fragments

ARCIMBOLDO_LITE managed to solve 4 structures out of 5 by searching for an inorganic chemical moiety, rather than for an helix.

In these particular cases the use of parameter *-K* in the SHELXE argument line was fundamental to retain the initial fragment throughout several autotracing cycles. SHELXE is programmed to trace only the polypeptide main chain and thus would lose the chemical inorganic compound after the first cycle if not otherwise instructed. Moreover, the CC might not reflect a successful solution if the atoms of the cofactor compound were not taken into account. In the case of other unsolved structures, helices may be too small, characterized by higher B-factors or simply too curved to be correctly matched with the internally generated straight helices. Some of them were solved through the use of libraries, introduced in Section 5.10, and their tests are summarized in Section 6.2.

3.3 A distributed computing approach to solve challenging structures.

As described above ARCIMBOLDO_LITE for a single machine is intended to simplify and make affordable the solution of a protein structure in simple cases where high resolution is available and the size of the structure is relatively small, under 200 residues. For more complex problems, it is more indicated to make use of distributed computing through Grid middleware, such as Condor or Sun Grid Engine or a supercomputer facility.
This section focuses on the differences between the single machine implementation and the distributed computing one. Technical details on distributing PHASER and SHELXE jobs over a Grid or a Supercomputer are given in Sections 7.4 and 6.3.

Main differences lay in the way the program filters and prioritizes solutions at different stages:
First, the number of rotation clusters and the number of solution in each cluster are not limited to the hardware of the machine where the binary is started, on the contrary the power of the grid is completely exploited by exploring all rotation clusters and their combinations in multiple fragment searches.
Second, the number of solutions selected for the expansion step with SHELXE is not anymore limited by the number of available cores in the workstation, and becomes a parameter. In the case of distributed grid computing, the default is to expand sixty solutions, but the user is allowed to modify this number via the instruction file. In the case of a supercomputer, default is the exact number of cores assigned among all requested nodes.
Third, each fragment search stage is exhaustively evaluated, including the expansion step of the selected solutions before proceeding on to the location of a new fragment. This implies that once a correct solution, able to produce an interpretable map is found, ARCIMBOLDO will end immediately avoiding the search of successive fragments.

3.4 Successful cases in ARCIMBOLDO

Since the original release in 2009, our group started to collaborate with other crystallographic groups to phase unknown structures with ARCIMBOLDO[1].
Recently, the binary distribution of the *lite* version simplified program use allowing external groups to solve autonomously their structures; such cases are marked with the name of researcher(s) in bold in the first column of the Table 3-5.

Data from	Space Group	# Residues	Search Fragment(s)	d(Å)	PDBID
J. Bravo	$P2_12_12_1$	111	2H12	1.3	
P. Czabotar	$P3_121$	120	1H14	1.3	
M. Graille	$P2_1$	310	1H16	1.45	4LUN
A. Mechaly, D. Guerin	P1	212	1H12, 1H9	1.46	
C. Tisné	$P2_12_12_1$	130	2H14	1.5	
V. Sautner, P. Neumann, K. Tittman	$P2_12_12_1$	73	2H14	1.6	
K. v Hecke	P432	165	2H14	1.6	
J. Hermoso	$P6_1$	50	1H10	1.7	
E. Valkov	$P2_12_12_1$	107	2H14	1.7	
D.C. Hissa, K. Gruber	$P2_1$	204	2H14	1.7	4K82
S. Trakhanov	$P2_12_12_1$	144	1H14	1.75	
V. Arcus	$P2_1$	112	2H12	1.8	4E1P
M. Rudolph	C2	182	6H10	1.82	4WG0
K. Zeth	$C222_1$	90	1H12	1.9	4AEQ
T. Pavkov	$P4_12_12$	111	1H13	1.9	
R. Bunker	P1	240	Model Helices	1.94	4BJS
S. Becker	$P2_1$	222	3H14	1.95	3GWH
A. Thorn, G. Sheldrick	I422	327	2 copy of an NMR model of 31 aa	1.95	3SZS
P. Czabotar	$P2_12_12_1$	99	Helices, modelling, refinement	2.09	
N. Verdaguer	$P6_322$	50	3H14	2.1	
O. Mayans	$P2_1$	240	Helices with side chains	2.1	4M3L
E. Pohl	$P2_1$	360	6H20	2.12	

Table 3-5 Previously unknown structures solved with ARCIMBOLDO

The table lists previously unknown structures successfully phased with ARCIMBOLDO. Size and resolution limit, space group and search fragment(s) employed are quoted for each structure. ***nHn*** codes for number of copies of given fragment type of quoted size. In this way, 2H14 stands for the sequential location of two helices of fourteen residues each.

The table encompasses mainly structures that have been solved by our group in collaboration with the owner of the data along with a few structures independently solved by external groups that have kindly informed us of their results (in bold). For each structure, the search models originally employed to phase it are reported. The latest release

of ARCIMBOLDO together with the last versions of PHASER and SHELXE may favour a different parameterization, usually simpler than the original. Structures are sorted by data resolution, which constitutes the main limitation for ARCIMBOLDO. The table starts at 1.3 Å, below the atomic resolution required for classical macromolecular *ab initio* methods based on direct methods. In the higher resolution range spanning 1.3 to 1.7 Å many structures were automatically solved with ideal helical fragments and ARCIMBOLDO default settings. In particular, the case of Dr. M. Graille at 1.45 Å is remarkable from its large size, 310 aa. Although bigger than other structures with comparable resolution, it has been solved starting from just one helix of 16 residues. Many factors may favour the phasing process: for example the presence of long helices enhances the probability of a correct location during MR; also variation on B-factors along main chain atoms can help discriminate preferable stable regions from flexible areas. Data completeness at high-resolution and a strong signal to noise ratio may help on the generation of accurate maps easier to trace. Medium and low resolution data are usually more difficult to phase, especially with ARCIMBOLDO_LITE as it auto-filters solutions in order to adapt to the hardware. In such cases, correct solutions are not always identified by top FOMs and a multiframe parallel approach is more indicated to increase the chances of success. It is noticeable how with decreasing resolution, the complexity in the search fragments increases: longer helices are required, or more fragments need to be located before expansion. In some cases search fragments are no more ideal helices but particular models: the structure from Dr. A. Thorn and Dr. G. Sheldrick at 1.95 Å was solved using as a model two helices from an NMR determination of the same structure, totalling 31 aa. Classical molecular replacement had not succeeded in placing the 7 or 8 requested copies in the asymmetric unit. Indeed, upon placement of the third one, partially correct solutions started to be eliminated by packing filters. ARCIMBOLDO succeeded placing the model sequentially and pursuing in parallel each partial solution, through density modification and autotracing expansion. The correct solution was only found after locating two copies of the model while placement of additional copies, did not lead to the phased structure. The structure from Dr. O. Mayans at 2.1 Å is another interesting case as its solution required to model side chains on the ideal helical main chain before locating and expanding it. The side chains were placed in energetically optimized conformers with the program SCWRL4[104] and even though not all side chains were correct, the results allowed SHELXE to bootstrap and the structure to be solved. This result suggests that at resolution below 2 Å, it is essential to introduce additional structural information and in this case incorporating side chains with errors was better than starting from the more perfect yet more incomplete model. This solution inspired a suitable approach for low resolution data that would use partial solutions found in ARCIMBOLDO for which main chain trace is correctly resolved but for which extension to the complete structure stalls. This method was first implemented by Dr. Kathrin Meindl and by Rafael Junqueira Borges, both in our group and their prototypes have been applied to complete phasing of the structures from Dr. P. Czabotar and Dr. E. Pohl at 2.09 and 2.12 Å, respectively.
A particular case is the structure from R. Bunker at 1.94 Å that is annotated as solved with *model helices*: this structure was solved by Dr. Bunker himself when ARCIMBOLDO was a prototype written in Perl. The precise model helix used in phasing is unknown but our ideal helices have been confirmed to work.
The presence in the table of 7 out of the 10 most frequent space group in the PDB database (4 out of the top 5) encourages to think that there is no a particular symmetry derived limitation for ARCIMBOLDO success.

The use of ARCIMBOLDO is recommended up to a resolution limit of 2.0 Å, although expertise, proper parameterization and user intervention can succeed in favourable cases at lower resolution. Sometimes, partial solutions produced from ARCIMBOLDO are not wrong but too poor for phasing to bootstrap and difficult to identify among the whole incorrect solutions found. Nonetheless, we have encountered instances where such challenging cases could be solved combining ARCIMBOLDO partial solutions with other methods, implemented in our group. The idea is to provide more correct phases to the expansion steps in SHELXE by either clustering in reciprocal space partial solutions (project implemented by Claudia Millán), or adding more features in real space like side chains to the located fragments (implemented by Rafael Borges). Other cases would require the incorporation of derivative data in combination with the fragment search stage. In any case, these approaches require to develop a fine parameterization and a sophisticated design to work with generality. Table 3-6 summarizes these cases, which will not be further discussed as they fall out of the scope of this thesis.

Data from	**Space Group**	**# Residues**	**Search Fragment(s)**	**d(Å)**	**PDBID**
X. Gomis - Rüth	$P2_12_12_1$	794	Frags + Se-MAD	2.0	4IJA
C. Artola, J. Hermoso	$P2_1$	682	Fragments, modelling and BUSTER refinement	2.7	4C5F
N. Valadares, R. Garrat	Pseudo-merohedrial	60-240	Coiled coils, twins	1.6-2.4	

Table 3-6 Previously unknown structures in which ARCIMBOLDO contributes on determine a partial solution

The table lists the structures, which have been solved with ARCIMBOLDO in combination with other methods, always programmed in our group, but that are not discussed in this text.

3.5 Structure of a 13-fold superhelix solved *ab initio*.

ARCIMBOLDO_LITE, as an *ab initio* phasing method, provides the possibility to phase a structure for which a suitable model is not available, but also in cases in which the crystallized structure is unexpected or unpredictable. This was the case of a study conducted by the group of Dr. Markus Rudolph at the Molecular Design and Chemical Biology F. Hoffmann-La Roche Ltd, in Switzerland and published in 2015[9]. The objective was to crystallise the glucorticoid receptor (GR), an oxosteroid hormone receptor, together with a small part of its co-regulator peptide[105] as shown in Figure 3-5.

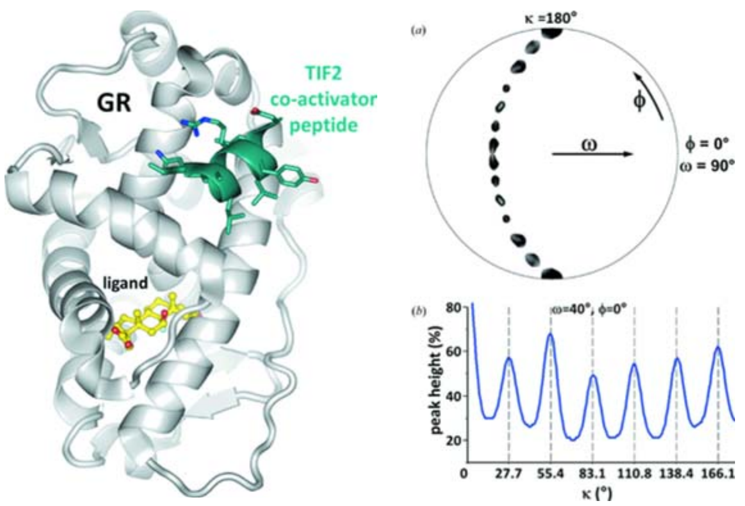


Figure 3-5 TIF2 following 13-fold NCS symmetry

On the left: the structure of GR (PDBID 3MNP) in complex with the TIF2 co-activator peptide in green. On the right: (a) The stereographic projection of the κ = 180° section of the self-rotation function. The crystallographic twofold axis is at $\omega = 90º, \phi = 90º$. (b) Signal of a recurrent peak at $\omega = 40º, \phi = 0º$ as a function of κ that is not visible in (a). The peaks follow the term (n × 360°)/13 with the integer n ≤ 6, indicating 13-fold NCS. Combination of this axis with the crystallographic twofold axis leads to the twofold NCS axes visible in (a).

GR is a nuclear hormone receptors ligand-activated transcription factor that to carry out its cellular function needs to translocate from its cytosolic complex into the nucleus where it interacts with co-regulators. Depending on whether transcription is activated or repressed the core-regulator can be a coactivator, e.g. transcriptional intermediary factor 2 (TIF2) or a corepressor, e.g. nuclear receptor co-repressor (NCoR). Genes regulated by GR are involved in sugar metabolism, cell inflammation and immunosuppression[106]. A pool of crystal diffracting to 1.82 Å in C2 space group was obtained. MR replacement was a natural choice for phasing the complex being its structure[107] previously determined since 2002, but all the available models produced negative LLG and completely random maps, suggesting that the crystallized content was not indeed a GR molecule. HPLC analysis confirmed the absence of GR and detected only TIF2. Moreover, the calculated Matthews Coefficient (MC) of 2.5 Å^3 Da^{-1} over the C2 asymmetric unit, indicated the presence of up to 22 peptide molecules of 1.6 kDa describing a difficult scenario in which phasing from *ab initio* would also be challenging.

Although biologically irrelevant, obtained results were fascinating to the expert crystallographers due to the intriguing presence of a 13-fold NCS (Figure 3-5). Nevertheless, the NCS alone would not explain the expected MC that would be larger, around 4.3 Å^3 Da^{-1}. Due to the high resolution of the diffraction data collected the crystal was expected to be packed tightly. Several MR jobs were launched testing 13-mer parallel β-barrels that when aligned with the NCS axis roughly reproduced the self rotation pattern. None of these tests were conclusive, and the analysis of Wilson plots for secondary-structure prediction, through the expertise of Dr. Alexander Popov, suggested a high predominance, over 55%, of α-helical content in the crystal and a maximal predicted β-stranded content of no more than 20%. Alpha-helical barrels were consequentially tested without success, while not random solutions were obtained with PHASER searching for 13 copies of ideal helices of varying size, even though resulting maps were not automatically interpretable by modelling software. At this point ARCIMBOLDO_LITE was launched in a workstation with 16 Intel Xeon CPUs (E5-2670 with 8 cores @ 2.60 GHz) and with 132 GB RAM running CENTOS 6.5. It was instructed to search one ideal helix of 10 residues. The run finished in approximately one hour with 98 traced residues of the correct solution comprising all the 13 copies. It was then systematically tested the effect of the fragment size in the solution of the structure resulting in a minimum helix length requirement of 8 residues but, as in the case of a nonamer helix, it was required to locate at least two helices before expanding. Helices of 12 aa did not reproduce a correct solution even deactivating crystal packing filtering (Section 3.1.7). All calculations were done at the maximum resolution of 1.82 Å. The solvent content was set to 65%, slightly higher than expected in order to enhance solvent flattening, and in the auto-tracing α-helices were searched for in all cycles (-q option). The NCS option in SHELXE is not useful in the present context given the absence of tertiary structure and hence was not applicable here. The full set of tests are displayed in the Figure 3-6, fragment sizes spanning 5 to 14 aa were combined with 1 to 8 copies search, results on CC values pointed out that suitable fragment lengths were found between 8 and 12 residues. This is not surprising considering that strategy underlying ARCIMBOLDO is to locate a partial, yet very accurate, substructure; previously experiences in SHELXE shown that a large penalty is paid for incorrect parts in this substructure[46]. Superior results are obtained with a smaller, error-free model than for a more complete but inaccurate model.

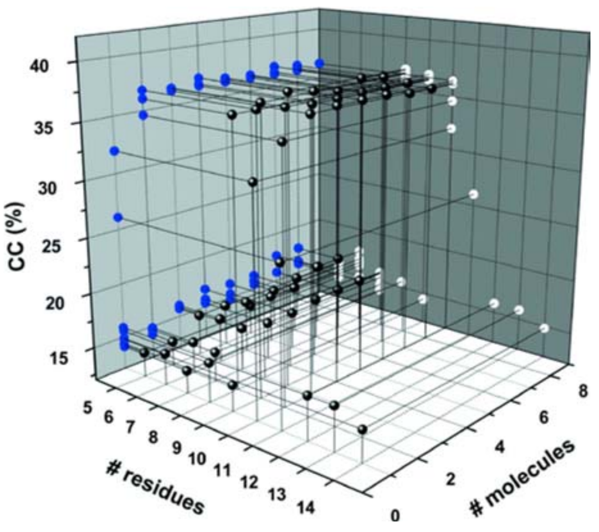


Figure 3-6 ARCIMBOLDO_LITE fragment size tests for TIF2

Length requirements for the ideal polyalanine α-helix search model. Helices between 5 and 14 residues were used as models, and 1 to 8 copies were searched for in PHASER, followed by three cycles of density modification and automated polyalanine chain tracing in SHELXE. For improved legibility the data points (black spheres) are projected onto the grey walls of the plot (blue and white dots).

The combination of both crystallographic and non-crystallographic symmetry arranges the 13 copies of the small helical peptide in a super-helix, Figure 3-7, with a left-handed twist. The height of a single turn is 66 Å containing the 13 copies. The projection of the asymmetric unit along the NCS axis is a barrel but the helices are not arranged in a parallel conformation, as initially supposed, explaining the differences in the self-rotation function. Superhelices are frequently observed in proteins with sequence repeats, they usually form with a non-parallel arrangement among the repeats, otherwise a ring-shape solenoid is most common[108]. However left-handed superhelices are rare thus further studies should be conducted to completely define their structural properties. The result obtained raises the possibility for α-helical peptides to aggregate *in vivo* in a similar way as the toxic β-amyloid[109] does. It could be conceivable that endogenous ligands could stabilize the quaternary structure of α -helical peptide aggregates. In turn, such a stabilizing small-molecule drug could help inhibit the fibrillation cascade that leads to β-type deposits.

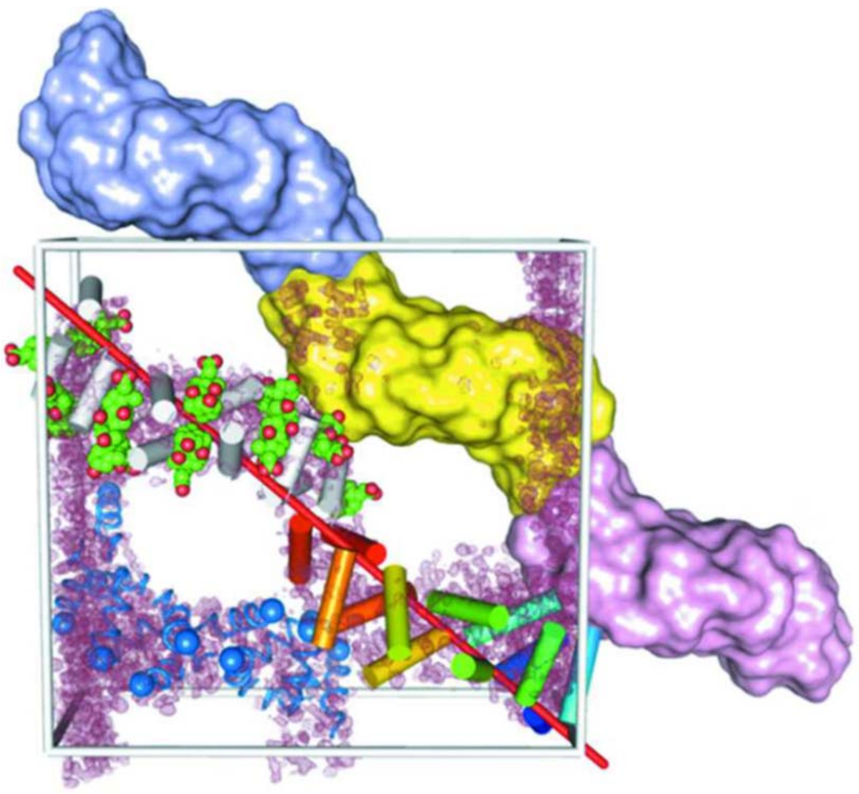


Figure 3-7 Superhelical TIF2 structure and packing in the C2 unit cell

The C2 unit cell is shown as a grey box with the origin at the bottom right and its four asymmetric units. One asymmetric unit consists of 13 short helices of the TIF2 peptide (coloured cylinders) that are arranged around the NCS axis, which is shown as a red line. Another asymmetric unit centred on the NCS axis is shown in grey with space-filling models (green) of the cholic acid molecules that wedge between the helices. The asymmetric units combine to form a continuous left-handed superhelix that traverses the crystal, which is well visible through a surface representation of three individually coloured asymmetric units. The arrangement of TIF2 helices is not all parallel.

4 PHASING FRAGMENTS FROM DISTANT HOMOLOGY MODELS: THE SHREDDER METHOD

Molecular Replacement requires the location of a model of known structure, close enough to the target structure, which can subsequently be used to derive starting phases that upon improvement will lead to the final solution. The degree of similarity between the target and model is usually quantified in terms of percentage of sequence identity (SI), whereby homologous structures are the ones sharing at least 30% of SI[110]. A better criterion for phasing potential would be the RMSD between model and target as it considers tertiary structure, usually more conserved than primary sequence. RMSD must be inferred by SI, as the target structure is yet undetermined, but a specific correlation[30] between both properties is manifest and is also considered in MR software to weight probability functions. Although an all purpose criterion does not exist, MR success can be expected to require at least 35% SI or an RMSD below 1.5 Å [111]. Whenever a close model is available and data quality is appropriate for model bias not to be an issue, MR is the best strategy to phase the target structure, but there are frequent cases in which only distant homology models are available. Such models are bound to contain accurate building blocks, but it may not be evident which sub-structure is the most suitable search fragment purely from the degree of conservation. If that was the case, highly sophisticated approaches for model weighting, enhancement and combination can already be found in SCULPTOR[112], MRTAILOR[113], SCEDS[114] or ENSEMBLER[115]. Recently, hybrid approaches on the frontier between MR and *ab initio* methods have succeeded in solving unknown structures. Suitable search models can be produced through *ab initio* modelling or starting from structures of distant homologs or NMR models with RosettaMR[37] or AMPLE[38, 116].
Our approach[8], implemented in the program ARCIMBOLDO_SHREDDER (http://chango.ibmb.csic.es/SHREDDER) is the subject of this chapter. Our method

explores the selection against the experimental data of search fragments generated from low SI models, to be used within ARCIMBOLDO, at resolutions under 2 Å. Evaluation of the rotation function around its maxima, for sets of fragments systematically shredded from the original model, is exploited as a suitable way to identify and eliminate less accurate portions. Results are combined into a score per residue and the template is trimmed accordingly. A second trimming step[94] is undertaken prior to density modification and autotracing expansion.
The antecedents to this method lie in a prototype program known as *GRINDER* from Iñaki Martínez de Ilarduya, in our group, which cuts pieces out of MR search templates to generate ARCIMBOLDO search fragments. Examination of FOMs in these exhaustive tests allowed identifying the rotations function as a stage providing information on model correctness. The proposed method was effective in solving the previously unknown structure of MltE[117, 118], and its solution is described later in this Chapter.

4.1 ARCIMBOLDO_SHREDDER workflow

The aim of the designed algorithm is to identify all local regions of highest structural deviation of a template structure in order to remove them and decrease the overall RMSD down to the point in which MR using ARCIMBOLDO can succeed.
Instead of relying on SI, the program tests systematically shredded models against rotation function and makes use of the target value LLG to discriminate between poor models and more probable ones. Analysis of FOMs derived from clearly different rotation solutions are kept apart, as some of them may be incorrect and several correct, referring to location of different monomers in the asymmetric unit. Thus, partial solutions coming from the same rotation cluster are considered globally rather than isolated to produce a discrete number of alternative models to test within ARCIMBOLDO. As a last consideration, optimizing models at the rotation stage can be determinant for finding correct translation solutions. In our experience with small fragments, the correct translation is usually harder to find than a correct rotation and model optimization increases its chances of success.

The workflow of the program is described in Figure 4-1.

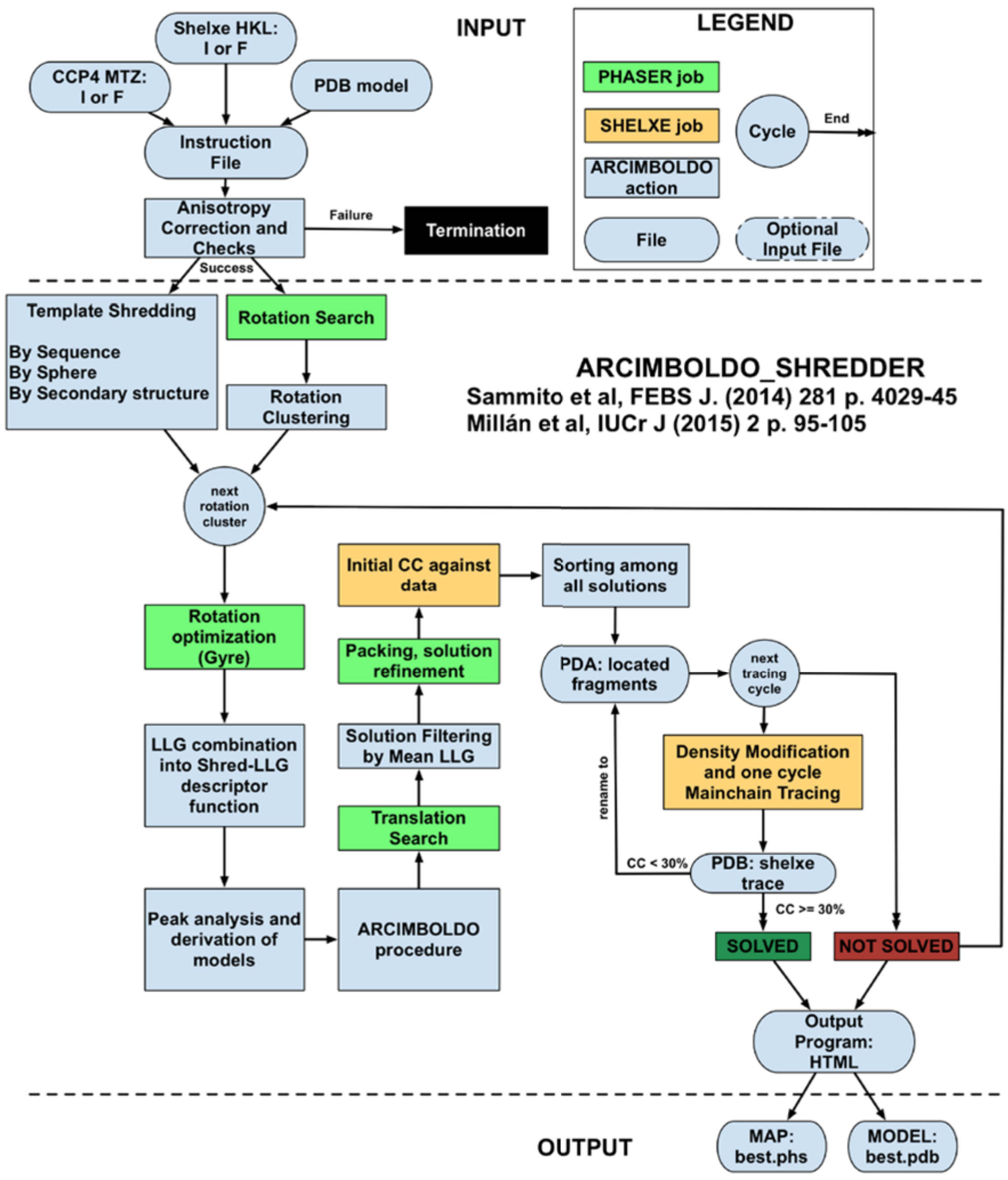


Figure 4-1 ARCIMBOLDO_SHREDDER workflow

ARCIMBOLDO_SHREDDER shares the same development environment of ARCIMBOLDO and the other programs presented in this thesis, thus initial checks are always executed to ensure both correct function of the software involved and appropriate hardware requirements. Details are provided in Section 3.1.1. The program requires data in both SHELX *hkl* and CCP4 *mtz* format while the configuration *bor* text file gives internal parameters. After initial checks are performed, the core of the program starts by generating a series of models cut from the input *pdb* model derived from the distant homolog. This procedure can be parameterized to instruct the program to perform either a sequential or spatial spherical shredding or systematically discarding secondary structure elements from the original model.

4.1.1 Sequential shredding

This is the default mode in ARCIMBOLDO_SHREDDER and the first one to have solved automatically an unknown structure. By employing this option the program will create all possible models obtained cutting linearly in the primary sequence, a number of residues spanning from **n** to **m**. Default is to generate all models shredding from 4 to 20 residues. This range can be also evaluated in a sparse way introducing a *step* value: default is a *step* of one residue, leading to the analysis of the whole range from 4 to 20, but the user can modify all the parameters and adapt them to generate an amount of models scalable to the available hardware resources. The number of evaluations generated can be easily calculated with the following:

$$\sum_{\substack{i=n \\ mcm(i,s)=s}}^{m} r - i + 1 \tag{13}$$

Where **n** and **m** are the minimum and maximum number of residues to be shredded, **s** is the step integer of the span and **r** is the number of total residues of the input model.
Instead of using sequential spans of residues to *omit* them from the original model, the program can be inversely instructed to generate models corresponding to such *fragments* extracted from the rest. Additional available parameterization, not activated by default, is the choice to trim the input model into polyalanine and/or remove residues not belonging to any defined secondary structure element, before generating shreds, or on the contrary to just maintain the side chain for Cysteine, conserving disulphide bridges. It is also possible locking parts of input model if there are reasons to trust them. For instance, requiring to never remove residues belonging to a given α-helix and/or a β-structure element.

4.1.2 Spherical shredding

Alternative shredding methods are provided within the program. One is to shred the starting model by spatial proximity. Hence, ARCIMBOLDO_SHREDDER can be instructed to cut out models by centring a sphere of radius **r** in each selected Cα atom of the input model and extract as many models as model residues that include all the neighbour atoms around the radius of the sphere. Again, the user chooses whether to generate polyalanine models, or with side chains or just keep the disulphide bridges. As in sequential shredding, it is possible to choose between omitting the residues in the volume of the spheres, creating holes in the input model, and extracting the sphere generating globular three-dimensional portion of the structure.

4.1.3 Shredding by secondary structure

This last option aims to generate shred models by cutting out secondary structure elements. In particular the input model is firstly parsed and all the secondary structure features are mapped, secondly all the possible combinations obtained by eliminating from **n** to **m** secondary structure elements are calculated and models are generated accordingly. Although it is not mandatory, the user can specify the number of helices and/or β-strand each model must always contain to enforce tertiary structure properties such as parallel or antiparallel conformation for β-strand or α-helices.

4.1.4 Evaluating models against the rotation function: the Shred-LLG

After shredded models are generated, a rotation function search at 2.0 Å is performed in PHASER using as search model the distant homologous structure. Rotation clustering is performed as described in Section 3.1.2 and Section 3.1.3, with the only difference that the RMSD input to PHASER is higher than the standard 0.2 Å used in an ARCIMBOLDO run. By default RMSD is set to 1.2 Å but the user should fine-tune this parameter to reflect the sequence identity of the input model against the target structure. Not properly modifying this value can lead to retrieving only negative LLG values, which by default are excluded from further calculations.
At this point, for sphere shredding or secondary structure shredding, all models produced are joined in a library to launch an ARCIMBOLDO_BORGES, as described in Chapter 6. Further studies on the FOMs for models of heterogeneous length are required before adapting the Shred-LLG analysis to these options and normalization steps must be included to make them comparable.

In case the sequential shredding is activated, as it is by default, each rotation cluster is examined in turn and Gyre tests are performed in PHASER[98], to refine Euler angles and get an optimized rotation LLG for each shredded model.
The program stores the LLG distribution for each sequentially ordered group of models of equal size, obtained by shredding out the same number of residues generating a series of curves as the ones shown in Figure 4-3. Each distribution is also normalized by averaging the LLG over the number of remaining residues in the corresponding shredded models. This normalization is a crude approximation under the assumption of a linear and even contribution per residue to the total LLG, but is essential in order to compare FOMs from models of different sizes.
When a correct span is removed, the LLG should decrease whereas if the part omitted is incorrect, reducing the model should lead to higher LLG values. Thus, the local maxima in the plotted curves indicate the most incorrect regions, and their removal improves the model. Persistence of peaks and consistency of results among all curves is a good indication to separate correct from spurious peaks. Thus the program needs to identify peaks in each curve but being discrete distributions, peak identification is accomplished by iteratively searching for local maxima under a delta threshold:

Algorithm 5: Peak search in a curve

```
Input:
        list of all points $[x_1,x_2,\dots,x_n]$
Output:
        list of peaks

$\delta = 1$
maxp = empty list
while $\delta > 0.5$
    $\mathrm{maxp}_\delta = \forall x_i \mid \begin{cases} x_i - x_{i-1} \geq \delta \\ x_i - x_{i+1} \geq \delta \end{cases} \quad i \in 1 \dots n$
    $\delta = \delta - 0.001$
    append $\mathrm{maxp}_\delta$ to maxp

return maxp
```

A single discrete descriptor (the Shred-LLG function) is then computed to correlate the omit residue regions with a variation in the LLG. Each point of this function is related to a single residue of the template model. The Shred-LLG is computed for a given residue as the LLG average over all N_{res} residues omitted and $N_{model\,s_{res\emptyset}}$ models in which this residue is omitted as shown in Eq. 14. This last normalization step compensates for under-representation of terminal residues. The alternative of shredding the template in a circular way, omitting residues from the N-terminus when the C-terminal end is reached, may result in models that are too disconnected, depending on the structure of the template.

$$\boldsymbol{ShredLLG_{res}} = \frac{\sum \boldsymbol{LLG_{model-res\emptyset}}}{\boldsymbol{N_{res}}} \cdot \frac{\mathbf{1}}{\boldsymbol{N_{model_{res\emptyset}}}} \tag{14}$$

Example of the function described is shown in Figure 4-4. The Shred-LLG descriptor function is analysed to find peaks corresponding to local maxima discriminated above the variance of their local environment. 75% of the top omit LLG, as well as the lowest LLG among the selected peaks are set as thresholds for residue selection. Residues characterized by higher function values are more likely to be incorrect since omitting them entails an increase in the LLG. Conversely, residues whose omission corresponds to a decrease in the LLG are expected to be correct. Plateau regions above the graph thresholds are identified as they may gather spans of incorrect residues. Once a peak is identified in the descriptor function, the shape of the corresponding peaks in all LLG distributions for the various shreds is analysed, selecting the sharpest peak to identify the precise residue range to be omitted. The final proposed models combine all selected omit ranges. As a conservative approach, the program will automatically generate multiple models to be used in a multi-solution frame whenever the descriptor function does not show a clear numerical discrimination or if peak retrieval is not obvious. Thus, the default is to produce four models eliminating peaks, plateaus and residues characterized by values above 75% of top and by values above minimum peak height, respectively.

4.1.5 Pursuing optimized models for phasing target structure

All trimmed models are then sequentially used as ARCIMBOLDO models for which to compute translations, evaluate packing, refine solutions and finally apply density modification and autotracing with SHELXE. A default RMSD of 0.8 Å is input to PHASER, above what is usually used for ARCIMBOLDO helix search cases. This parameter may need adjustment according to the specific case. More details regarding the ARCIMBOLDO run are found in Section 3.1. Results are presented in a combined way as html output and success identified through a final CC > 30% causes the program to stop remaining calculations because the structure has been solved.

4.2 Solving MltE: an outer membrane protein involved in cell wall recycling in bacteria

ARCIMBOLDO_SHREDDER successfully solved the structure of MltE, a bacterial outer membrane-anchored endolytic peptidoglycan lytic transglycosylase[119], where others methods, including the standard ARCIMBOLDO approach had failed. Rod-shaped single crystals of MltE from Escherichia coli were obtained[120] by Dr. J. Hermoso and Dr. C. Artola-Recolons at the Institute of Physical chemistry “Rocasolano” in Madrid. Two X-ray diffraction datasets, both extending to 2.0 Å resolution, were collected from different crystals at beamline ID23-2 of the European Synchrotron Radiation Facility (Grenoble, France). Crystals belong to space group C222$_1$ and were expected to contain two copies of the 194 amino acids MltE monomer in the asymmetric unit, corresponding to a solvent content of 45%. A MR approach was tried but no structure with clear homology over a broad span of the sequence was identified, but similarity of predicted fold and 35% identity over a range of 66 amino acids (34% coverage, E-value 0.034) was determined for Slt70 from E. coli. This structure is a 70 kDa soluble lytic transglycosylase, determined at a resolution of 1.9 Å in complex with a 1,6-anhydromurotripeptide and deposited in the PDB under the code 1QTE[121]. As expected, MR was unsuccessful due to the low overall similarity between the model and the target structure. In fact, once the MltE structure was determined, an RMSD of 3.1 Å could be calculated for the superposition of 160 Cα atoms of Slt70. Performing a pairwise alignment with CLUSTALW[122] between the model and the target resulted in a poor local alignment in which the worst aligned region was identified in the first 60 N-terminal residues up. Taking residues 457–596 from 1QTE, truncated to polyalanine or maintaining conserved side chains, as an MR search fragment did not lead to a solution. Originally, this structure was solved when the current ARCIMBOLDO_SHREDDER procedure was not yet established. Indeed, this structure was the motor for building up its strategy, suggesting that local similarity between distant homologs can be exploited if the region of high difference can be identified and omitted. One among the many possible blind ways of doing this was to generate 113 polyalanine models by cutting out all possible spans of 26 contiguous residues from the original, non-solving template of 140 aa. Iñaki Martinez de Ilarduya wrote the first prototype of the sequential shredding script in C to actually perform this operation and each one of these models was used within ARCIMBOLDO to search for two copies with PHASER; finally all resulting substructures comprising one or two placed models were subject to further trimming by iterative peak list optimization[94] on a residue basis against the E-based CC[123]. Each pair of equivalent amino acid from both monomers was omitted in turn, and a CC was calculated. Whenever this led to an increase in the CC, which naturally tends to be higher the larger the number of atoms, the pair of residues was eliminated from the model. The INITCC increased from 8.0 to 10.9%. Whereas the untrimmed solution could not be successfully expanded with SHELXE, the remaining model, containing 85 amino acids per monomer, rendered a main chain trace of 247 amino acids, characterized by a CC of 36.1% after ten cycles of main chain tracing interspersed with density modification were performed. The resulting electron density map could be easily traced and the side chains assigned. This stepwise residue optimization against the CC is now a standard option within SHELXE[46].

Solving the structure established the potential of this approach, but the computational power required to evaluate not just these 113 models but also those generated with parameterizations that did not lead to a solution within ARCIMBOLDO was problematic. Moreover the chosen span of 26 aa could be too restrictive or too general in other cases; we sought a general approach to incorporate shredding models and evaluate them in one step. Once the structure was solved, it was possible to compute the FwMPE for all PHASER solutions after the translation step to characterize them depending on whether this value was near 90º (random phases) or lower than 80º (non random solutions). We found that no interpretable map was reached after placement of the first fragment, the lowest value being 82°. It was only at the stage of the second fragment, where high ZSCOREs for the translation function, and correspondingly low numbers of translation solutions produced, gave indications as to the potential of a particular model to yield a solution. This preliminary analysis revealed that the first and second fragments displayed the same informative pattern for the distribution of the rotation function LLG against the sliding window of residues omitted (Figure 4-2 (a)), suggesting that the rotation function would be a discriminating stage in which model evaluation, prioritization and optimization could be exploited. Incidentally, rotation being the first step in the process makes it

especially convenient, as the number of solutions is still restricted even if many models are evaluated.
The question at this point was to test if the rotation function LLG entails information on the correctness of the model and then to compare the results with the real graph of local deviations between the model and the target.

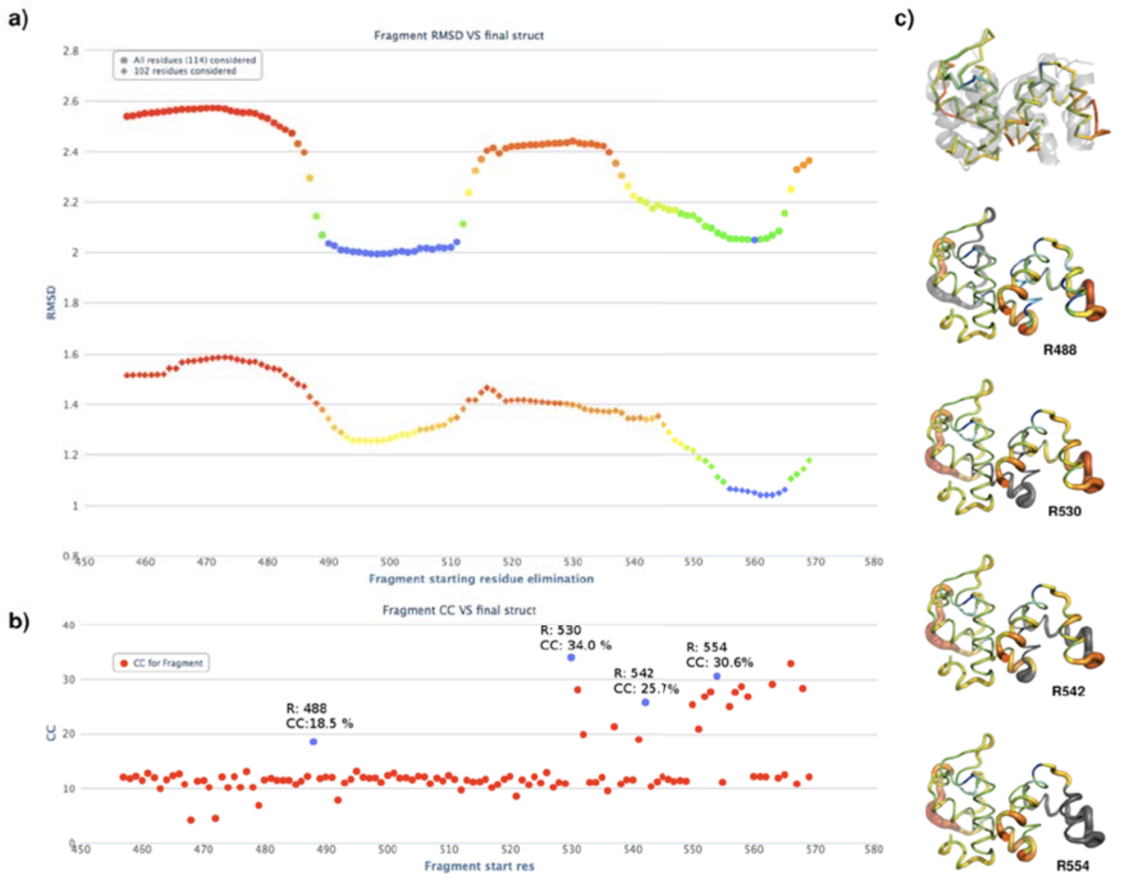


Figure 4-2 Relationship between model correctness and phasing success

(a) Plot showing the global RMSD of two sets of Cα alignments between the MltE structure and each 1QTE model with a missing gap of 26 residues starting at the residue indicated on the horizontal scale (top, all 114 residues; bottom, the best 102 residues). (b) Final SHELXE CC of the trace, the points discussed in the text are labelled and shown in blue. (c) Plots of the 1QTE template (coloured ribbon) superposed on the MltE final structure (PDB ID 2Y8P; grey cartoon) and for the four models highlighted in (b), colour coded (using red for the largest RMSD and blue for the smallest RMSD) after the 140 Cα superposition, RMSD as in (a) and with the omitted region in black. The width of the backbone also reflects the RMSD.

Hence, we first studied the structural conservation at a local scale for 1QTE and MltE. Figure 4-2 (a) shows the RMSD characterizing a Cα superposition for each of the models, taking into account all 114 residues or only the geometrically closest 102, corresponding to a core that produces a structurally meaningful superposition. For the complete models being used, values vary between 2.0 and 2.6 Å, the curve showing two regions of high differences alternating with two regions corresponding to more similar models. However, neither low RMSD nor high LLG in the rotation function show perfect correlation to the final success in phasing, as judged from the SHELXE CC (and confirmed by FwMPE) scoring the best trace from each solution, presented in Figure 4-2 (b). The 19 models leading to a solution are characterized by a CC of the trace above 18%, well discriminated from the average 12% value that corresponds to unsolved structures. Four points in these curves are discussed below, and the corresponding models are displayed in Figure 4-2 (c), together with a plot showing the superposition of the 1QTE template to the final MltE

structure (PDB code 2Y8P). The first case corresponds to a model where the gap starts at residue 488, and represents an isolated, less marked, CC maximum that is located in the first descent of the full RMSD curve but before it effectively drops. The corresponding cartoon representation shows that the omitted region (in grey) enters an area of high deviation that is thus removed, although this also applies for many other non-solving models with lower global RMSD. This draws attention to the fact that a spurious success may depend on very small differences in the model and random correct sampling. The other three selected points are located towards the right part of the curve, where the second area of lower RMSD occurs. Again, there is a discrepancy in the point representing model 530, which, although corresponding to the top CC obtained, still displays a rather high RMSD. Models 542 and 554 are characterized by lower RMSDs and are located in the region of the graph where solutions are more frequent. The cartoon shows the large deviations in the areas removed (grey, broad coil). The apparent discrepancies in the eventual phasing success and the accuracy of the model and its rotation location are better explained by looking at the core RMSD curve. This curve represents the RMSD calculated for the Cα of the 102 best agreeing amino acids, thus considering a core of the structure that produces a meaningful structural superimposition. In this curve, the second minimum is more pronounced than the first, and better reflects the phasing success. As the most incorrect residues are likely to be eliminated by the necessary peaklist optimization against the CC, this core better represents the nature of the structure entering the density modification stage. In summary, rotation appears to map the overall RMSD of the model while, for phasing success, a smaller, more accurate structure is preferable to a more complete one impaired by higher deviations. Given the same overall RMSD, it is better to have a more accurate core and some areas of higher deviation that may be readily trimmed than to have differences evenly spread with no means to improve the model.
From this preliminary analysis it was clear that it was possible to prioritize shredded models by their FOMs at early stages without pursuing all the models through parallel ARCIMBOLDO runs. Moreover, considering them grouped according to rotation clusters allowed to compare FOMs and to normalize results. The ARCIMBOLDO_SHREDDER approach, as described in the previous Section 4.1, was implemented and sequential spans of 4,5,…,19 residues were omitted from the 1QTE model generating 16 curves for each rotation cluster representing a different orientation spotted. The top LLG rotation cluster represented a correct orientation and the plots for each group are shown in Figure 4-3. As can be seen, the graph for the shortest omit is noisier but already contains all peaks representing the four areas with most significant deviations. As the omitted spans grow, the graphs become smoother. Peak analysis, as described in the previous Section, was conducted and conserved peaks among different curves were identified. In the picture vertical dashed blue lines mark these peaks. The procedure follows by computing as described above (Eq. 14) the Shred-LLG per residue estimator. Figure 4-4 (a) shows the Shred-LLG function combining rotation LLG of all omit shreds for the two correct rotations. Both present maxima corresponding to the spans whose omission is most favourable for the four most incorrect areas in the template. In the case of the green line derived from the first rotation cluster, the areas identified by our algorithm and eliminated from the model are those of residues 502–513, 536–548, 557–575 and 594–596 plus the residue 457. The most correct assignment, established from the true RMSD between template and target structure would be 508–513, 538– 543, 563–588 (in this region two highly differing spans with a few correct residues in between are found) and 592–596, even clearer in the purple line as the shoulder for 522–529 disappears. The peak identifying the last C-terminal residues behaves similarly and, accordingly, the spans eliminated for this second cluster are 500–515, 538–549, 560–573 and 587–596, plus 457–

460. Cartoon representations of the 1QTE template, colour-coded to match the areas of high RMSD identified in each of the graphs, are displayed in Figure 4-4 (b).

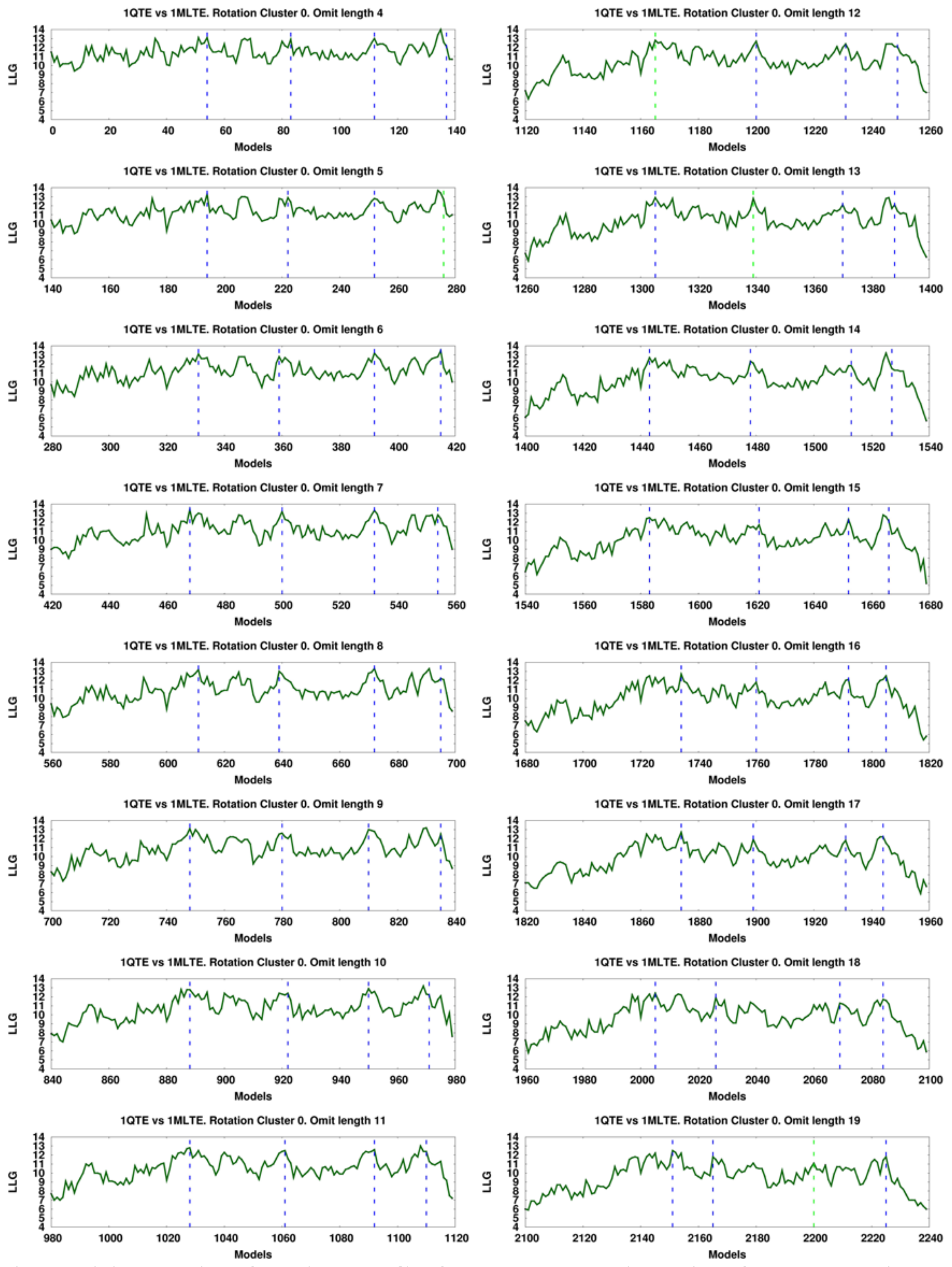


Figure 4-3 Rotation function LLG of the correct orientation for each omit group generated in the solution of MltE

Rotation function LLG graphs obtained around the peak of a correct rotation by shredding out all possible spans of 4, 5,…, 19 amino acids from the original template. The peaks contributing to the areas selected for trimming (Figure 4-4) are marked by blue vertical lines. The green vertical lines highlight the peaks in the four curves that determine the length of the areas to be omitted.

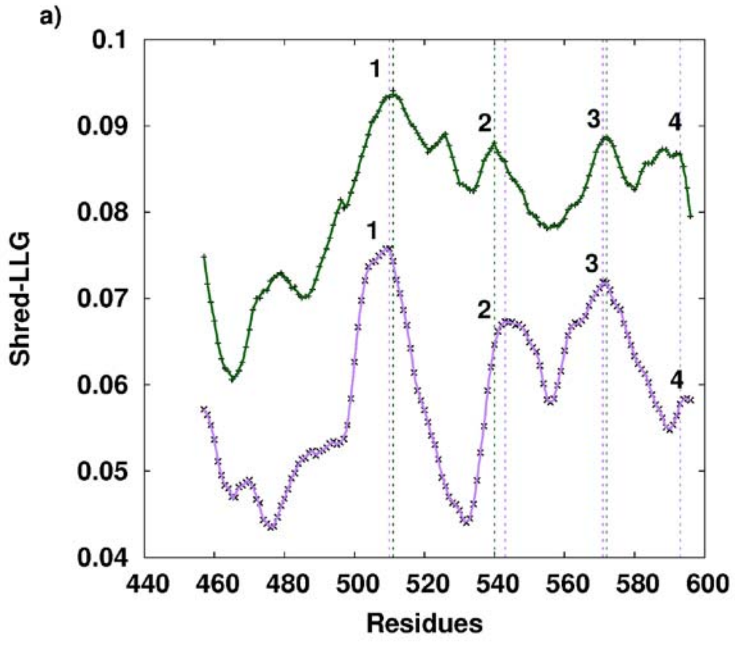


Figure 4-4 Evaluation of model RMSD from the rotation function of omit shreds.

(a) Shred-LLG graph estimating the average LLG derived from the elimination of each residue in the 1QTE poly alanine template. The green and purple lines correspond to the two correct rotation clusters. The pattern revealing the four most incorrect areas [508–513, 538–543, 563–588 (comprising two highly differing spans with a few correct residues in between) and 592–596] appears as 502–513; 536–548; 557–575 and 594–596–457 in the green line, and as 500–515, 538–549, 560–573, 587–596 plus 457–460 in the purple line, where the shoulder for 522–529 disappears. Vertical dashed lines mark local maxima. (b) Plots of the 1QTE template (coloured ribbon), with colour and ribbon width indicating the RMSD from MltE (2Y8P) and omitted residues shown in grey, matching the four areas of high RMSD identified in the graphs in (a). Red colour or wide ribbon indicate large deviations, whereas blue colour or thin ribbon stand for low RMSD.

In practice, it makes little difference, as either model, proceeding within ARCIMBOLDO to translation search, packing check, rigid body refinement and density modification autotracing, succeeds in solving the MltE structure.

The effect of eliminating individual residues on the rotation function LLG would not be correlated with their RMSD to the true structure. The RMSD of the model derived from the first rotation cluster Figure 4-4 (b), top, fitted to the target structure on the optimal rotation for the original template is reduced from 2.7 to 1.93 Å. On the contrary, RMSDs for 15000 randomly trimmed models of the same number of residues render RMSDs spread from 2.04 to 3.06 Å as shown in Figure 4-5. This difference is significant, and visible in Figure 4-6 as the shredded model is set apart from the distribution by a ZSCORE of 4.68, well separated from the best randomly trimmed model, characterized by a ZSCORE of 3.95. Thus we can conclude that our optimization is not following a random behaviour but it can lead to a correct solution and possibly phase the structure. It is not possible to select only one orientation cluster by its top LLG but all found rotations must be explored sequentially. LLGs characterizing different rotations are not necessarily comparable and the only strategy available is to locally optimize a model in a multi-frame approach that generates alternative possibilities to test. There are many steps in the ARCIMBOLDO_SHREDDER procedure where further improvement remains to be investigated, comprising:

- Use of finer statistics to evaluate, merge and select peaks, beyond relying only on the Kurtosis of the omit curves but taking into account the local contour of the peak.
- Better selection of the spanning residues to be omitted for the final produced models that would take into account the structural context in which these residues are found: secondary structure elements, exposed or buried region etc.
- Better normalised statistics for the Shred-LLG to enhance diagnosis of poorly aligned regions and reduce noise.

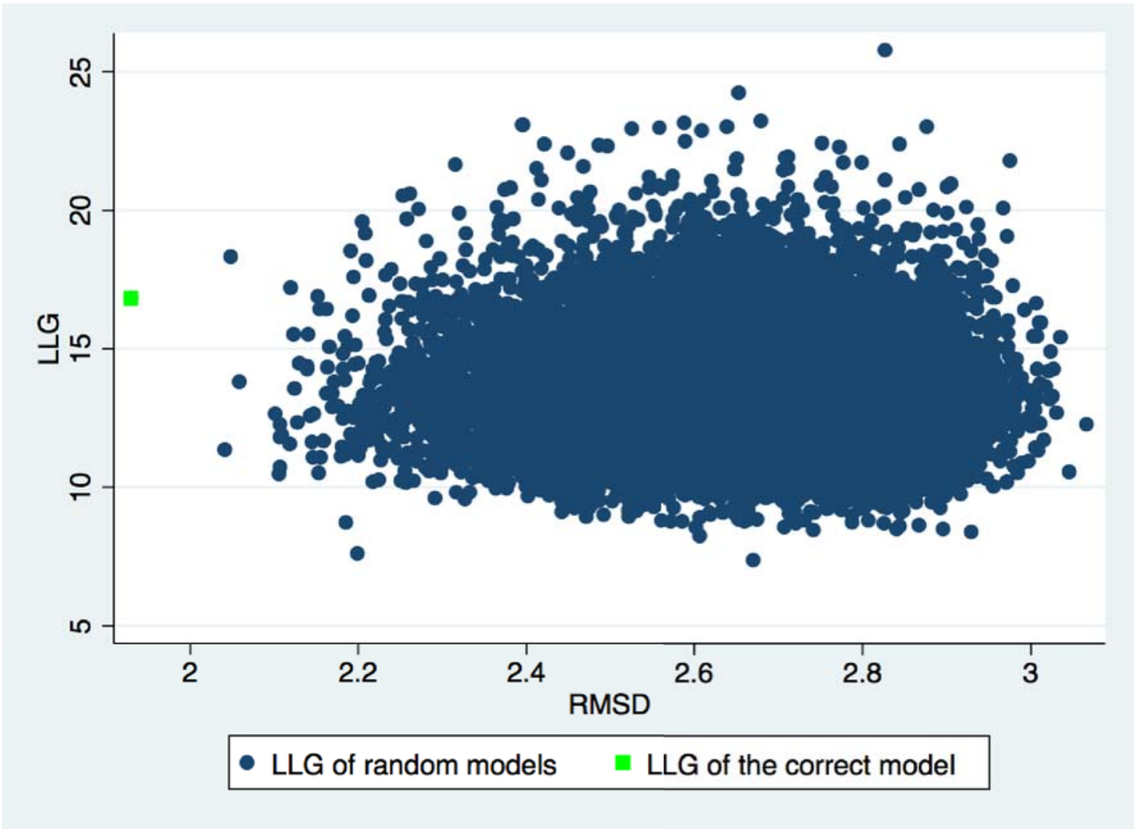


Figure 4-5 LLG vs RMSD of 15,000 random models

15,000 models were obtained cutting out randomly the same number of residues omitted in the correct model. RMSD against the real structure and rotation function LLG for the correct orientation are correlated in the graph. Random models are displayed in navy blue while the green square is the correct solution. Although, the RMSD of 1.93 Å for the correct model is clearly better than for any random model, its LLG does not correspond to a top value.

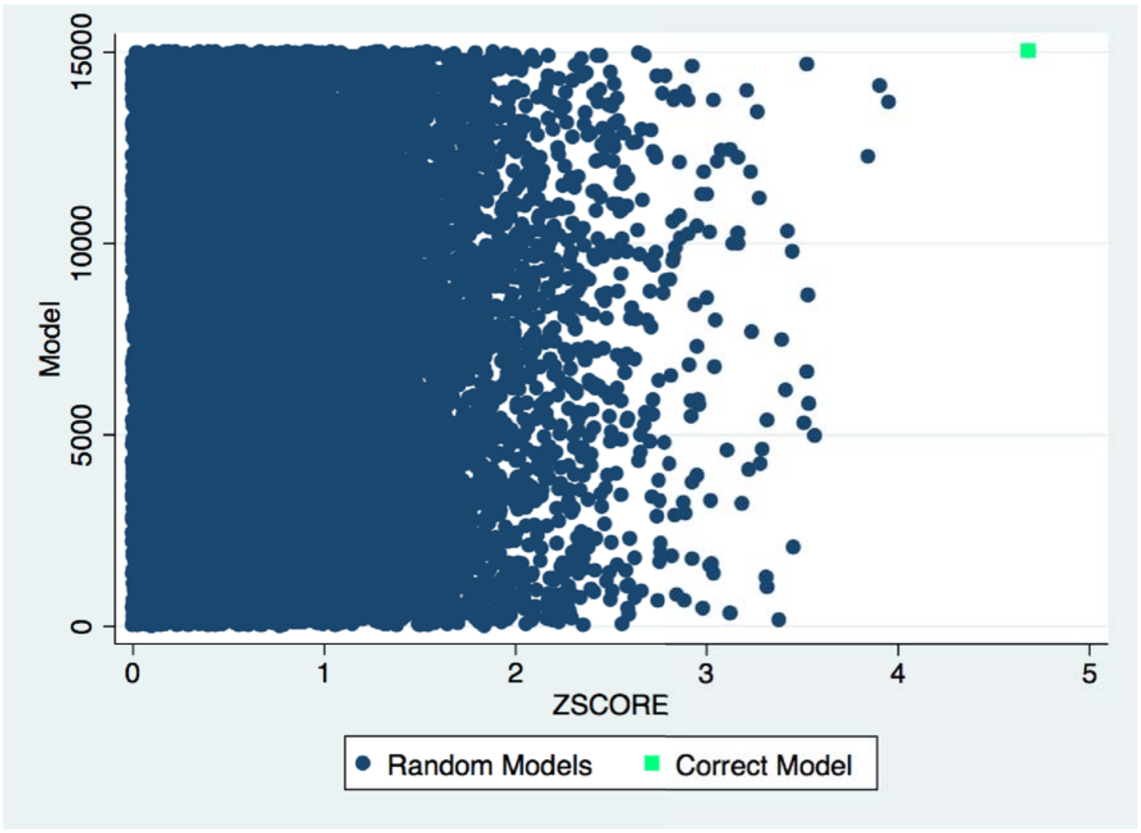


Figure 4-6 Singularity of the correct model respect to 15000 random ones.

ZSCORE values are computed from the data of the Figure 4-5. The correct model displayed in green is separated from any of the random models. Its ZSCORE corresponding to 4.68 places the solution in the tail of the distribution.

4.3 Novel structures solved with ARCIMBOLDO_SHREDDER

ARCIMBOLDO_SHREDDER has only recently been published and released, but the method has already solved some previously unknown structures, summarized in Table 4-1. Not all features implemented in the program have been exhaustively tested yet, and further testing is required to optimize parameterization. In particular, only sequential shredding combined with the evaluation of Shred-LLG has been used in a fully automated procedure for phasing previously unknown structures, although volume shredding has succeeded in a more manual way. With the exception of the MltE structure, described in the previous section, the entries in the following table are not yet published and are therefore not further discussed.

Data from	Space Group	# Residues	Search Fragment(s)	d(Å)	PDBID
X. Gomis - Rüth	$P2_12_12_1$	560	Phase clustering from Sphere Shredded models	1.5	
J. Hermoso	$C222_1$	378	2 copy of shredded model of 85 aa.	2.0	2Y8P
R. Hurtado	$P2_12_12_1$	1450	Shredder with side chains	2.35	
R. Hurtado	$P2_12_12_1$	532	Shredder with side chains	2. 5	

Table 4-1 Previously unknown structures solved with ARCIMBOLDO_SHREDDER

As seen from the table, the quoted structures are considerably bigger than the general ARCIMBOLDO target structure and the resolution of two of them is lower, well under 2Å. In all cases a distant homolog was identified by an HHPred[124] search and CLUSTALW2[122] was used for pairwise alignment in order to determine a possible core of conserved residues. The case from Dr. J. Hermoso was discussed in the previous Section, and its solution was fundamental to develop the current implementation in ARCIMBOLDO_SHREDDER. The same algorithm has solved two structures from Dr. R. Hurtado, challenging cases being both large structures yielding only 2.35 and 2.5 Å resolution data, respectively. Even though close homologs were known for these structures, MR was not straightforward, whereas ARCIMBOLDO_SHREDDER was able to isolate differences brought upon by hinges in the models and phase the structures without any particular configuration. The input model was provided with side chain atoms and the shredding, as default, maintained these atoms in the extracted models. It is worth mentioning that forcing the program to polyalanine trimming did not lead to a solution. This outcome is in line with the principle that at lower resolution a larger fraction of the total structure is required as starting information, even at the cost of accuracy, and it is worth maintaining partly or totally the side chain atoms, in order to increase the chances of expansion bootstrapping into interpretable maps. The fourth case reported, from Dr. Gomis-Rüth, is a particular case in which the shredding method was not sequential but considering comparable volumes. It was not performed automatically but with intervention, aiming to create partially overlapping models conserving a specific fold composed of three antiparallel strands and one α-helix. These manual shredded models were used as input search model in ARCIMBOLDO_LITE and partial solutions were selected to be clustered in reciprocal space combining them in such way as to generate a merged solution from which correct phases bootstrap. This approach is implemented within the ALIXE[50] program, developed in our group by Claudia Millán, and will be shortly released as a standalone binary.

5 LOCAL FOLD LIBRARIES AND THEIR PROPERTIES: THE BORGES ALGORITHM.

All ARCIMBOLDO-derived phasing methods aim to build a bridge between the pure *ab initio* and the MR approach by enforcing unspecific structural features rather than atomicity. In the case of ARCIMBOLDO these features are provided by secondary structure elements, such as ideal α-helices, whereas ARCIMBOLDO_SHREDDER makes use of distant homologues to extract conserved structural regions. Although both methods provide good alternatives for solving unknown structures they present intrinsic limitations. For example, no suitable homolog might exist for a given structure or it might not be retrievable because of its primary sequence being only partially known or even undetermined. Likewise, a structure may also present a low or absent helical content, or may be dominated by other features, such as β-strand repeats in β-barrels or β-sandwiches, which can dominate diffraction and consequently interfere with helix location. Finally, the limiting implicit condition required from ARCIMBOLDO to correctly locate the first fragment in order to find subsequent ones, may become increasingly challenging for larger structures as the signal for a small fragment becomes weaker. One possible solution to all these obstacles could be to extend the methods by enforcing tertiary structure, as in ARCIMBOLDO_SHREDDER, instead of relying on isolated secondary structure elements but maintaining non-specificity of the search models, as in ARCIMBOLDO, to achieve generality in the approach. The task of generating such composite local folds can be seen as the inverse problem of extracting them from an existing collection. This situation can be described by a literary analogy borrowed from the novel the “Library of Babel” written by the Argentinian writer Jorge Luis Borges, from whom the new method was named after. In this piece Borges describes an imaginary infinite library containing all possible books of four hundred and ten pages. The pages are finite but not the number of books, thus in such library a reader could find anything and also its opposite, all truths and all lies, nothing or everything. In the same way, by following this analogy, our finite and non-random PDB structural database contains, indeed, a vast amount of structural information. Hence, it is reasonable to suppose that for any unknown structure, given small enough fragments (for example, two helices or three strands in a particular configuration) close geometrical models are bound to occur in some of the deposited entries. Conversely, if a given local fold is impossible, it should certainly not be found in the database.

A new *quasi ab-initio* method was introduced in 2013 in our group comprising the setup of two programs:

- BORGES[7], for the definition, extraction, clustering and superposition of library of local folds,
- ARCIMBOLDO_BORGES[3, 7] for the ensuing use of these libraries for phasing (as described in Chapter 6).

BORGES (http://chango.ibmb.csic.es/BORGES/) runs on a workstation that automatically accesses and distributes calculations to a cluster or supercomputer. Existing tools to analyse and retrieve structural information[64, 125] are meant to identify overall, rather than local, geometry or to focus on libraries to be exploited in conventional molecular replacement[61], model building and map interpretation[126] and refinement[62]. In contrast, our approach is tailored to analyse detailed secondary structure geometry and spatial relations among different fragments through a distribution of vectors defined by the centroids of Cα and carbonyl oxygens from overlapping tripeptides.

5.1 Characteristic Vector Definition

Secondary structure properties are usually derived from the hydrogen bond pattern. Analysing this network implies checking the environment of the amino acid, made up of non-consecutive residues, even of symmetry equivalents not explicitly contained in the PDB set of coordinates. These hydrogen bonds are made possible by the protein backbone adopting torsion angle values in characteristic ranges usually displayed in the Ramachandran plot. Thus, analysis of relevant torsion angles may suffice to characterize the secondary structure. **D**efine **S**econdary **S**tructure of **P**rotein (DSSP)[127] is the standard algorithm employed to predict H positions and consequentially hydrogen bonds and thus determine the secondary structure environment for each residue.

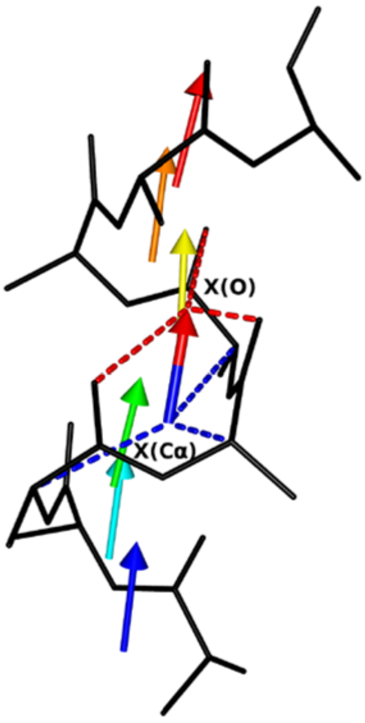


Figure 5-1 α-helix CVs

In our approach, a further simplification can be achieved by examining the atomic distribution within archetypal secondary structure elements. In an α-helix (Figure 5-1) all carbonyl moieties are aligned, pointing in the same direction, towards the C-terminal end, whereas the corresponding Cα atoms are arranged towards the N-terminal side. All individual Cα-O vectors of each amino acid are approximately collinear. As a consequence if the centroid of all Cα atoms and the centroid of all O atoms in a helical segment are computed, it can be noticed that the resulting XCα-XO vector relating the centroids is also collinear with the helix axis and its modulus, defined as the Euclidean distance between both points, comes close to the maximum possible value of 2.4 Å, from the individual Cα-O distance given by the covalent angle.

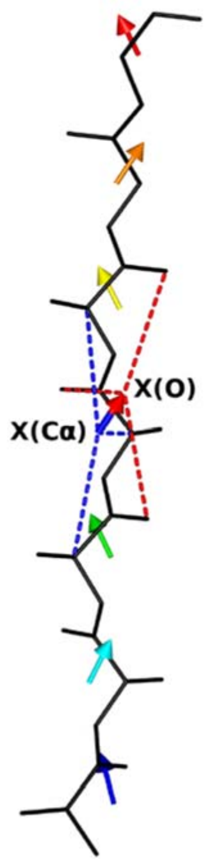


Figure 5-2 β-Strand CVs

In the case of β-strands (Figure 5-2), carbonyl bonds alternate their orientations on both sides of the backbone chain. Their average also runs in the direction of the strand but the resulting modulus is shorter as consecutive vectors run in tilted directions and their orthogonal components tend to cancel out. In the case of turns and coils, the less systematic alignment tends to even out, leading to shorter distances between Cα and O centroids.
We denominate such vectors as **Characteristic Vectors** (CVs) because they not only describe the secondary structure of the involved main chain atoms but also locate them in a spatial context in which geometrical comparisons are possible.

5.2 Statistical Analysis over the module

Systematic calculation of CVs on different randomly chosen fragments has led to the hypothesis that CVs could represent the secondary structure type of a fragment. However a rigorous description needed a statistical analysis in which CVs would be tested against the well-known DSSP[128] algorithm, to correlate results and determine the robustness of our method
The statistical computation was performed on a selected pool of structures from the RCSB PDB database. The query options used are listed in Table 5-1. The intention was to collect protein structures, of good quality from high resolution X-Ray structures, but correcting for redundancy of well known structures. Moreover, all structures not yet released were obviously excluded together with structures that did not contain enough main chain residues to extract at least two contiguous CVs. A pool of 18,646 structures was obtained.

Query	**Structures Found**	**Conjunction**
Experimental Method: X-Ray	95,768	and
High. Res. between 0.5 Å and 2.1 Å	52,710	and
There is a Protein Chain	51,800	and
Representative Structures at 90% of sequence identity	18,997	and
Structures released	18,993	and
Structures in which at least two CVs are extractable	18,646	

Table 5-1 Extraction of the test pool to analyse CV properties

The pool of 18,646 structures was extracted by filtering in the PDB all X-Ray structures resolved up to a maximum resolution of 2.1 Å that contains main chain atoms and filtering out for identical sequences up to 90%.

A Python script was written in order to parse the group and compute both the DSSP prediction for each residue and the CV from non-overlapping fragments of a fixed length. In the BORGES implementation, characteristic vectors are computed from contiguous overlapping fragments with a sliding window of one residue. However, in this test overlap is avoided to reduce data dependency. The script also verifies that according to DSSP prediction all residues in the fragment belong to the same secondary structure assignment. To ensure that strong outliers or errors in the local geometry did not affect the vector statistics we selected only the fragments for which DSSP predicts a continuous structural pattern. Any fragment containing residues annotated as turns, bends or loops was discarded. Instead, we imposed that for α-helices the only accepted annotation would be **H**, which according to the standard DSSP indicates an α-helix, and for β-strands **E** and **B**. This restrictive choice reduces the number of fragments obtained, especially with increasing fragment length, but should give more reliable results. Finally, the script ended by deriving a simple statistical analysis, computing the mean and standard deviation of all predicted helical and β-strand fragments. The script was executed several times for each secondary structure pattern group changing the fixed length (3, 9, 15 and 21) of the fragments to be extracted, in order to relate algorithm precision with fragment size and once more, to reduce correlation among data, each test for a different fragment size is performed on an independent subset of proteins extracted from the main pool of structures.

Although choices were designed to reduce the dependency of the outcome a subtle correlation will always be present due to the origin of the atomic coordinates themselves, which are the result of standard X-ray diffraction protocols. This property of the data cannot be modified but is generally considered as unimportant.

Figure 5-3 shows statistics characterizing the distribution obtained with helical fragment of various lengths. The corresponding results for β-strand CVs are shown in Figure 5-4.

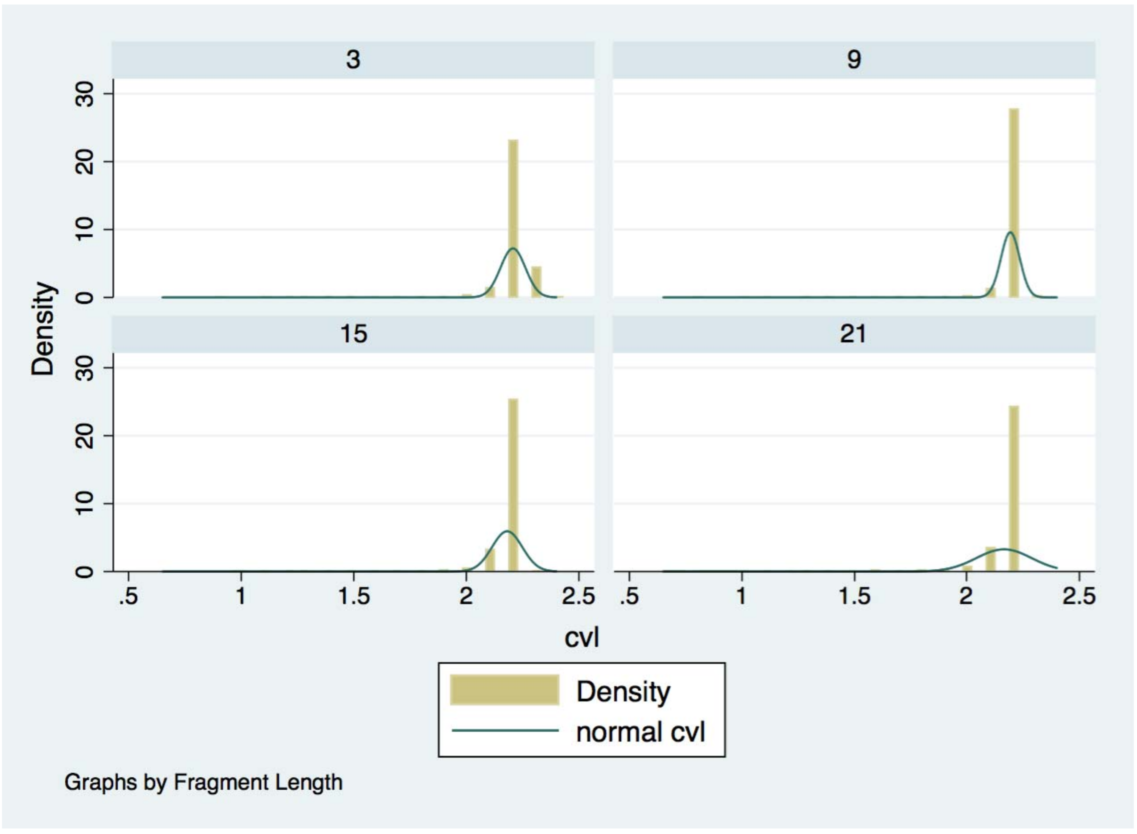


Figure 5-3 Graph of helical distribution of CVs of different sizes

Distributions of CVs from fragments of 3, 9, 15 and 21 residues, identified as helical by DSSP.

Fragment Length	Obs.	Median	Mean	St. Dev.	Variance	Skewness	Kurtosis
3	89,206	2.2	2.21	0.06	0.003	-2.63	33.85
9	22,028	2.2	2.19	0.04	0.002	-12.14	238.97
15	9,351	2.2	2.18	0.07	0.005	-8.02	98.28
21	18,939	2.2	2.17	0.12	0.015	-6.47	54.15

Table 5-2 Descriptive statistics for each subgroup of distribution of helical CVs

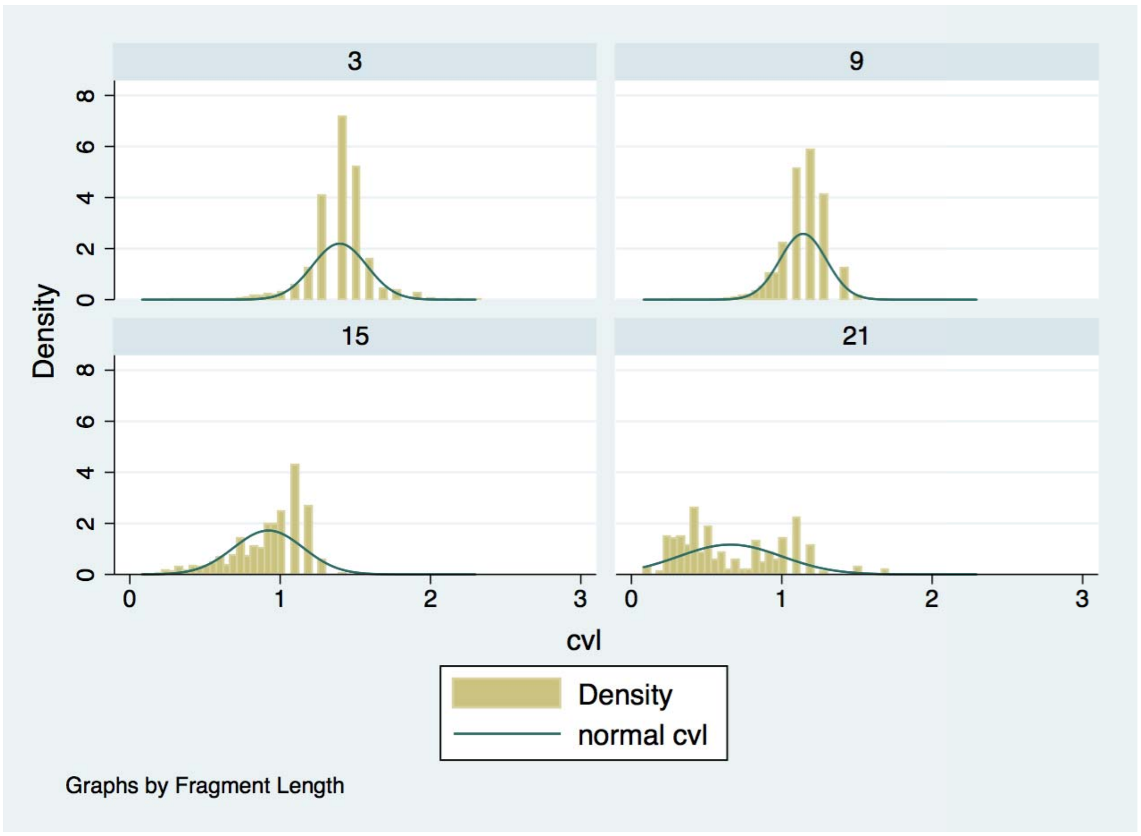


Figure 5-4 Graph distribution of CVs for β-strands of different sizes

Distributions of CVs from fragments of 3, 9, 15 and 21 residues that were identified as β-strand by DSSP.

Fragment Length	Obs.	Median	Mean	St. Dev.	Variance	Skewness	Kurtosis
3	77,072	1.4	1.40	0.18	0.033	-0.42	5.67
9	13,459	1.2	1.14	0.15	0.239	-0.51	3.72
15	1,965	0.96	0.92	0.23	0.054	-0.80	3.25
21	238	0.53	0.66	0.34	0.117	0.57	2.42

Table 5-3 Descriptive statistics for each subgroup of distribution of β-stranded CVs

Each subgroup is described with classical descriptive statistics. Skewness and kurtosis indicate asymmetry[129].

This preliminary analysis revealed two main properties of the vector:

1. The vector dependency on the secondary structure type: it is clear that α-helix and β-strand ranges are distinctly separated.

2. The vector dependency on the length of the fragment: shorter fragments entail a larger vector size and this effect is stronger for β-strands than for the α-helical group.

While the first property is essential to classify fragments according to their secondary structures, the second property could pose a problem as large fragments might be difficult to classify. As expected, the population of the fragments decreases with their lengths and this effect is further enforced by the imposed restriction to consider only very regular fragments. Analysing results from this perspective, it is not surprising to find only 238 β-strands of twenty-one residues, not out of scarcity of long β-strands but simply due to the stringent regular pattern adopted from DSSP (sequences of E and G) and the reduced number of *pdb* entries used to compute them. It is also clear that the characteristic vector is more regular for α-helices than for β-strands. This is evident from two different aspects: the standard deviations for the α-helical group are lower than for the β-strands and the differences of the means for the α-helical groups are markedly lower compared to the β-strands. For α-helices we observe a left asymmetric leptokurtic distribution as indicated by the negative skewness and high positive kurtosis, but the median is always comparable to the mean indicating that asymmetry derives from the peakedness of the distribution. Whereas, for β-strands we still observe the same asymmetric distributions but the median tends to decrease as increasing the fragment length.

To establish whether differences in the medians are significant or not or, in other terms, if all medians are generated from the same distribution of data, we performed the non parametric Kruskal-Wallis[130] test, testing the null hypothesis that there was no difference. The test was computed separately for the α-helix and the β-strand group. Our results showed the null hypothesis could be rejected, implying that there exists a significant difference among distributions of CVs obtained from different fragment size, leading to the conclusion that the criteria used for assigning secondary structure type to a CV must be dependent of the fragment size from which it was computed.

Table 5-4 Kruskal –Wallis equality-of-populations rank test for helical CV distributions

	Fragment length	Obs	Rank Sum	
	3	89,206	6.64e+09	
	9	22,028	1.44e+09	
	15	9,351	5.62e+08	
	21	18,939	1.09e+09	
Chi-squared		3,702.055		with 3 d. f.
probability		0.0001		
Chi-squared with ties		8,496.791		with 3 d. f.
probability		0.0001		

Table 5-5 Kruskal –Wallis equality-of-populations rank test for β-stranded CV distributions

	Fragment length	Obs	Rank Sum	
	3	77,072	4.04e+09	
	9	13,459	2.44e+08	
	15	1,965	1.45e+07	
	21	238	1.18e+06	
Chi-squared		23,623.909		with 3 d. f.
probability		0.0001		
Chi-squared with ties		24,511.019		with 3 d. f.
probability		0.0001		

It is reasonable to think that tripeptides should present a more constant backbone conformation for a defined secondary structure pattern, than longer fragments[131]. Extracting all continuous overlapping tripeptide CVs computed along main chain atoms from all deposited structures we observe the ranges, summarised in Table 5-6, actually adopted inside BORGES to classify tripeptide adscription to secondary structure elements:

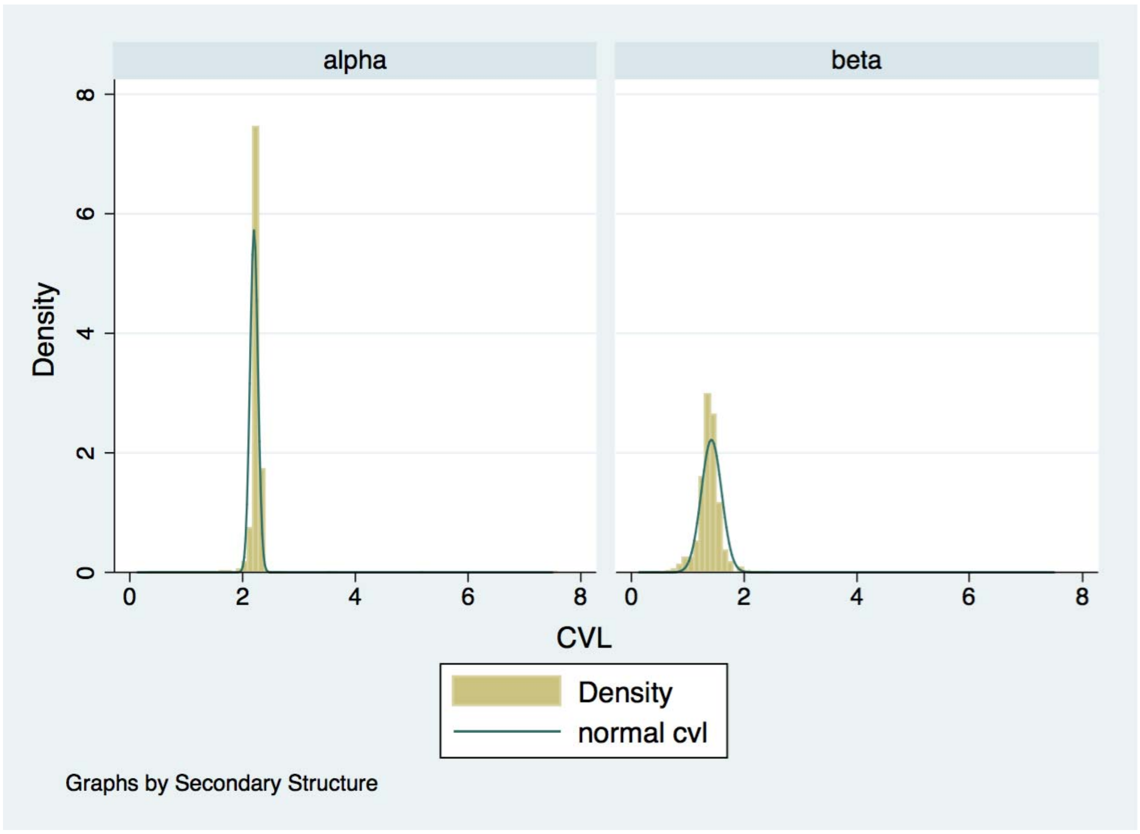


Figure 5-5 Global distribution of CVs in the whole PDB

Distributions of α-helical CVs and β-stranded CVs from tripeptides, are shown as histogram bars. The interpolated Gaussian function is represented in green. α-helical CVs are more abundant and precise than β-stranded ones.

	Mean	Standard Deviation
α-Helix	2.2	0.18
β-Strand	1.4	0.24

Table 5-6 CV Mean and standard deviation for secondary structure elements

These ranges are the result of both the previous statistical analysis and empirical experiments. Opportunely, the lower limit of the α-helix range is distinctly higher than the upper limit of the β-strand one, so we do not risk mistaking a β-strand tripeptide as an extremely distorted α-helical one.
Longer fragment classification requires the analysis of the discrete distribution of CVs along the main chain: a helix is assigned when the CV lengths of the interested residues maintain the same helical pattern, whereas matching consistency is required to assign a β-stranded conformation. Coil regions are characterized by an irregular alternating pattern in the distribution, which is the result of lack of regularity in the main chain typical of such regions.

Thus, the conclusions to this preliminary statistical assessment on the suitability of the properties of the characteristic vector and their distributions can be summarized in:

- The vector length is linked to the secondary structure type of a fragment.
- The vector length is very sensitive to any perturbation of the geometry of the structure that is also related to the size of the fragment itself. Optimal classification is achieved with tripeptides, where sharp peaks are observable for α and β-structures.
- Thus, secondary structure assignment on a real fragment requires the analysis of a discrete distribution of CVs for continuous overlapping tripeptides.
- The direction of the vector reflects the direction of the polypeptide chain, therefore it is possible to distinguish the orientation of two strands in a β-sheet and classify them as parallel or anti-parallel.
- Residues in a fragment, showing distortions from the regular secondary structure geometry cause an alteration in vector length.

5.3 Vector directions and fragments

As stated above, the characteristic vector is important not only for its module properties, but also for the direction it assumes. In fact, it reflects the global direction of the component vectors allowing, for instance, parallel and antiparallel strands to be distinguished from each other or helix direction to be characterized.

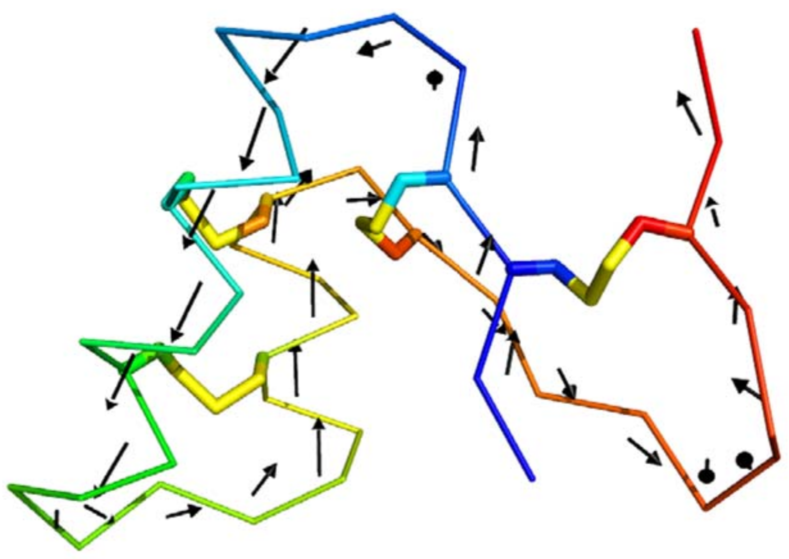


Figure 5-6 Hellethionin-D CV distribution

The molecule is represented in rainbow coloured ribbon to indicate the direction of the main chain. Disulphide bridges are enhanced and coloured in yellow. Arrows represent the CVs and follow the direction of the chain. For clarity, only every second CV is displayed for the α-helices.

Hellethionin[132] from *Helleborus purpurascens* is a peptide comprising 44 aa including 8 Cysteines forming 4 disulphide bridges. Thionins are probably involved in plant defence against pathogens and applications have been proposed for engineered plant resistance in agriculture and as immunotoxins in tumour therapy. Their 3D structure is related to that of viscotoxins, purothionins, or crambin. Hellethionin D (PDB entry 3SZS) is displayed in Figure 5-6 showing how the CVs reflect not only secondary structure properties but also their direction. This structure was solved in collaboration with Dr. A. Thorn and Dr. G.M. Sheldrick, by means of NMR derived fragments used as search models in ARCIMBOLDO. The complete solution procedure is still unpublished. The fold contains four fragments, two antiparallel helices of 14 aa and 9 aa, and two small antiparallel β-strands of 5 aa and 6 aa. The figure highlights how the vectors follow the geometry of the secondary structure according to a specific pattern: the same orientation for the α-helices enclosing them like an ideal cylinder and alternating directions for the β-strand. As seen in Figure 5-7 contributing vectors have practically constant length within regular α-helices.

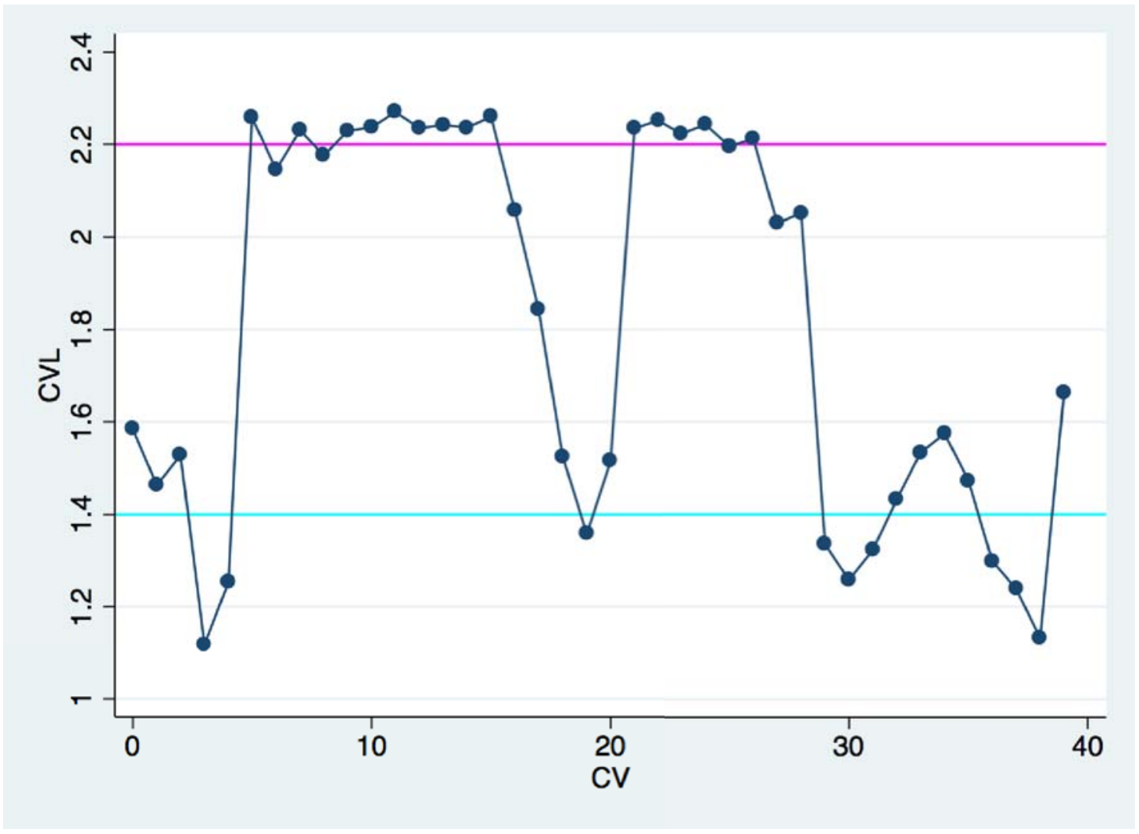


Figure 5-7 Analytical representation of the CV distribution for Hellethionin-D

Each CV is distributed by its modulus (CVL). The magenta line at 2.2 Å references for α-helical CVL whereas a cyan line at 1.4 Å indicates β-strands.

This is due to the covalent bonding in the backbone, imposing strong constraints on the geometry and thus allowing minimal variation in this regular pattern. β-strands present more flexibility although their pattern is clearly separable from α-helices. Extreme peaks in the sequence of vectors indicate the interruption of a regular secondary structure pattern and are associated with coils or turns.

In conclusion, the component of the contributing vectors that allows us to compare and relate secondary structure elements in space is neither the length nor the number of contributing vectors but their direction.

Geometrical relations among characteristic vectors including angles and distances can be computed and are used in the BORGES algorithm to extract libraries of specified fold.

5.4 A local geometrical description

Another advantage of using the CV distribution as fragment descriptor instead of a single global CV is the possibility to encode local fragment distortions. If for regular α-helices and β-strands we found, as foreseen, a quite constant distribution, we have also found a recurring pattern in the distribution of α-helices that showed a certain degree of curvature in their conformation. We call this subclass of fragment type **curved helix.** Furthermore, we can distinguish a kink that breaks the helix in two pieces from a smooth curvature throughout the helix. As can be seen in Figure 5-8, the CV distribution provides appropriate level of detail to map local distortion per residue basis.

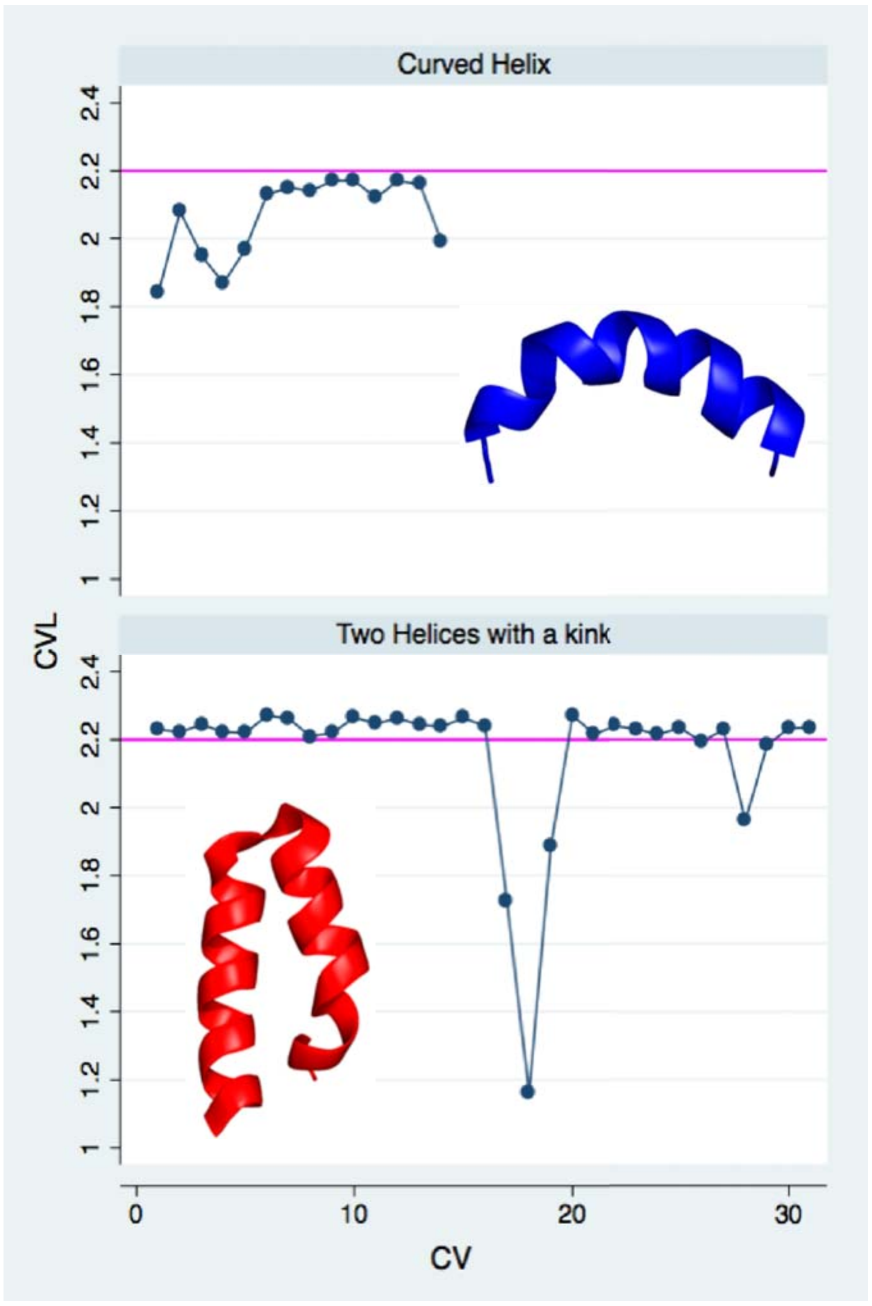


Figure 5-8 CV distribution of helical fragments.

Top: CV distribution of a curved helix. The small distortion in the pattern matches the geometrical curvature of the helix. Bottom: Two helices separated by a kink are displayed along with their CV distribution. The kink is clearly visible in the pattern as a strong peak.

Curved helices cannot be recognised by a singular CV but require the analysis of the whole distribution revealing that helices having a kink have a constant pattern preceding and following the kink and a strong peak in that residue. In contrast, the distribution of a gradually curving helix is smoother and the peak at the bulge is less marked.

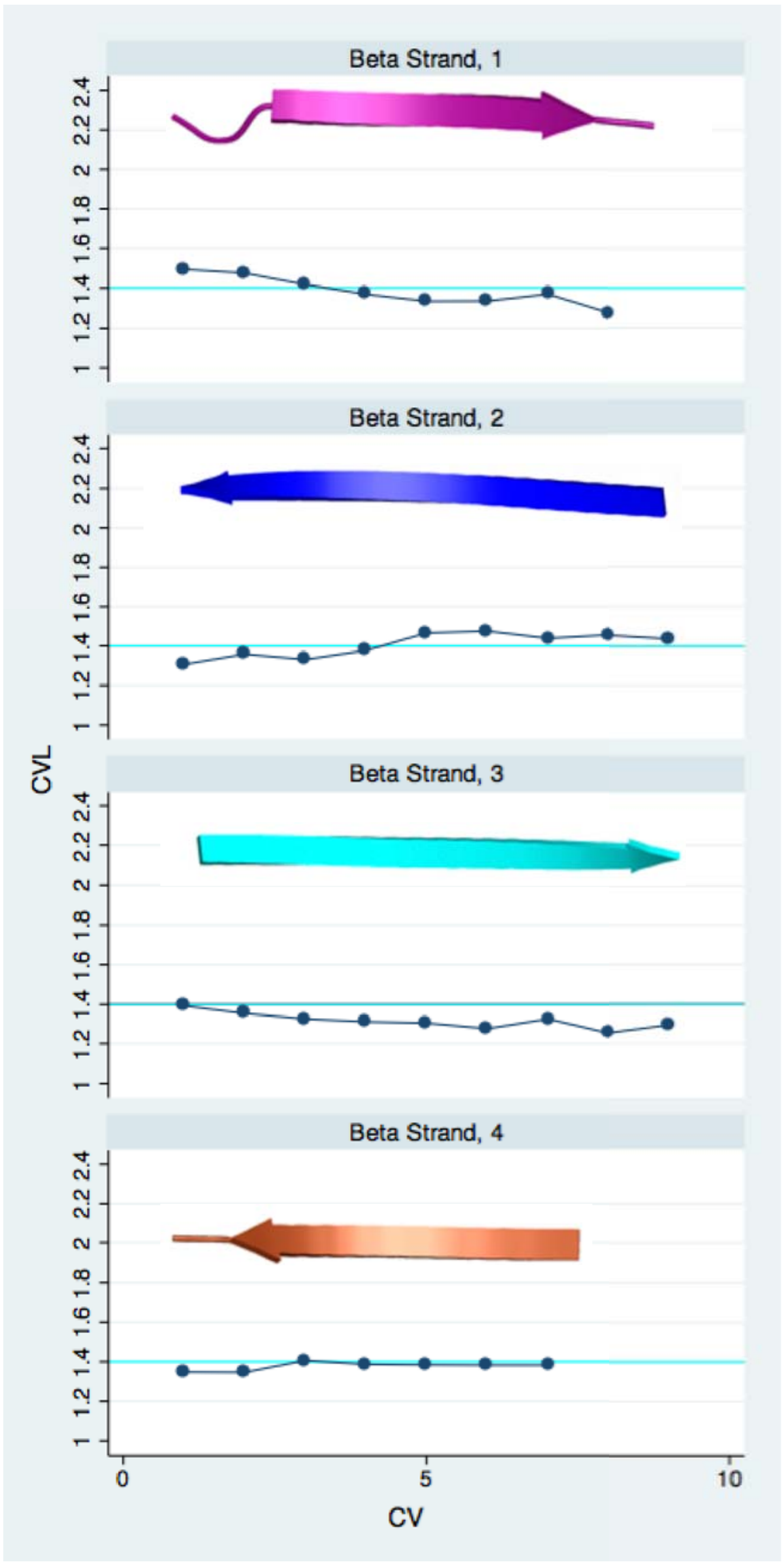


Figure 5-9 CV Distribution of β-strands

B-strands composing an antiparallel β-sheet are displayed together with their respective CV distribution. The pattern of the distributions reflects the local geometry conformation per each residue.

Similar conclusions can be derived for β-strand fragments. In Figure 5-9 a β-sheet of 4 antiparallel β-strands is decomposed to analyse the CV distribution of each fragment. Although all of them are in line with the reference β-strand mean of 1.4 Å, details of the local environment for each residue are reflected as well.

5.5 BORGES prototype implementation

A prototype to extract libraries of a given fold from the whole PDB database was designed, based on distributions of CVs describing secondary and tertiary structure. As introduced above, distributions are defined by the centroids of α-carbons and carbonyl oxygens of consecutive, overlapping tripeptides. The backbone conformation of a tripeptide captures an amino acid in the context given by its preceding and its following residue. For every tripeptide along the protein backbone, a vector is defined with its origin at the geometric centroid of the three α-carbon atoms and ending at the centroid described by the three carbonyl oxygens in the tripeptide. In the case of an α-helix, the CVs are parallel to its axis, and their direction is that of the polypeptide chain; for a β-strand, the CVs deviate around 45° from the direction of the polypeptide chain, with consecutive vectors being approximately orthogonal. As shown by statistical tests, the moduli of such vectors are determined by the type of secondary structure, their distributions falling into clearly distinct ranges: the resulting mean (standard deviation) values are α, 2.2 (0.18) Å; β, 1.4 (0.24) Å; or coil (where individual tripeptides may show any CV value, but the distribution along consecutive segments varies erratically). These values reflect the hydrogen bonding undergone by the carbonyl oxygens, and thus the main-chain environment and interactions. The CV modulus is maximized when all carbonyl moieties are aligned, as in the more regular α-helices; distortions from helix bending or kinks are concomitant to a change in the carbonyl orientation, leading to a sensible decrease in the resulting CV modulus. The alternating geometry adopted by the directions of the carbonyl groups in a strand leads to a substantial shortening of the resulting CV. Loop and coil regions tend to contain backbone torsions in the preferred Ramachandran regions, and thus it is not surprising that CVs for their tripeptides may adopt any value but consecutive CVs lack the constant distribution identifying the secondary structure elements. Beyond secondary structure, CVs are useful to ascertain tertiary structure relationships. To this end, a global CV is defined for the complete n-peptide in each secondary-structure fragment, that is, a vector defined from the centroid of all α-carbons to the centroid of all carbonyl oxygens in the fragment. Distances between different secondary structure elements can be calculated from these global CVs, through the geometrical difference of their origin points; and through their scalar product their relative orientation can be quantified. As detailed in previous sections, this formulation is conveniently accurate, matches DSSP assignment and discriminates well among local characteristics. It also provides the flexibility to define different thresholds for the geometry of different areas (for example, a more rigid definition of strands packed within a β-sheet flanked by a more mobile α-helix).

The geometrical definition of a library is conveyed to BORGES through a model template in *pdb* format. With this information, all main chain composite fragments in the *pdb* with a tertiary structure resembling the template within specified thresholds will be extracted and superimposed. Threshold values are defined for the distance between fragments, calculated from the distances relating CV origins. Limits are also defined for the deviation of the angles between fragments: the angles between corresponding CVs in the template and fragment cannot differ by more than a given value for the fragment to be accepted.

BORGES needs to compute every CV for each tripeptide in the *pdb* before screening for a given geometrical definition. But as both operations are independent, the whole PDB database can be annotated once in terms of characteristic vectors and stored in a dedicated database from which different fragment libraries may be derived.

BORGES can extract the searched fold from all deposited structures, or from a specified subset; it can include all NMR model solutions for a particular structure or select just the one with the lowest energy. However, artificial B-values are adopted for the extracted models, being meaningless for our purpose outside of their native structural context.

During the test of this first prototype version an annotated database was generated. This operation took 17 h on a four-core workstation, while a standard search against the resulting database to extract and cluster a given geometry (for example, two helices or three β-strands in a particular disposition) took under half a day in a grid of 100 cores, with more complex motifs or non-exhaustive samplings being considerably faster.

To extract a library from the annotated database, BORGES starts by analysing the template provided. It decomposes the template into secondary structure fragments, described by their CV distribution, and computes relative geometrical relationships between them. Let us define X_s and W_s as generic elements of secondary structure, α-helix or β-strand that belong to the search model. If X_s has t residues, BORGES associates to this fragment a distribution X of $t - 2$ CVs. The same is done for all other secondary structure elements W_s. BORGES also describes each fragment with a global CV defined as the vector between the centroid of all Cα and the centroid of all O atoms.

$$\boldsymbol{X_s = R_1^x, R_2^x, \dots, R_t^x \quad -- \quad X = x_1, x_2, \dots, x_{n=t-2}} \tag{15}$$

$$\boldsymbol{W_s = R_1^w, R_2^w, \dots, R_m^w \quad -- \quad W = w_1, w_2, \dots, w_{q=m-2}} \tag{16}$$

$$\boldsymbol{W_{CV} = \frac{1}{m}\sum_{i=1}^{m} C\alpha(R_i^w) - \frac{1}{m}\sum_{i=1}^{m} O(R_i^w)} \tag{17}$$

Internal relationships between secondary structure elements are computed: the relative orientation of the two fragments $\boldsymbol{X_{CV}}$ and $\boldsymbol{W_{CV}}$ is expressed as the angle γ; the distance between them is expressed as a vector connecting the Cα centroids of the two fragments X and W. The resulting distance vector is named $\boldsymbol{D_{xw}}$ of length r, and the direction of this distance is determined by the angle φ between $\boldsymbol{X_{CV}}$ and $\boldsymbol{D_{xw}}$.

$$\boldsymbol{\gamma = \measuredangle X_{CV}, W_{CV}} \tag{18}$$

$$\boldsymbol{D_{xw} = \frac{1}{t}\sum_{i=1}^{t} C\alpha(R_i^x) - \frac{1}{m}\sum_{i=1}^{m} C\alpha(R_i^w)} \tag{19}$$

$$\boldsymbol{\phi = \measuredangle X_{CV}, D_{xw}} \tag{20}$$

BORGES sequentially searches for each secondary structure element defined in the fold and extract only the models that satisfy all geometrical relationships among fragments. The order in which BORGES identifies fragments within the fold is not random. A sorting procedure, based on topological sorting, allows BORGES to identify a core of the fold, represented by the elements that are most closely packed together in space. It is convenient to start searching for these elements to discard early on incompatible combinations, thereby freeing memory and reducing computation time. The geometric conditions are evaluated by checking the less probable conditions first in order to filter out fast geometries that will not fit the template definition.

Filtering criteria.
Given Y_s and Z_S, two secondary structure elements in a structure of the PDB that match the secondary structure elements X_S and W_S defined from the template, the basic descriptions are computed in the same way, defining *Y*, *Z* as the distribution of CVs by (15) and (16), and $\boldsymbol{Y_{CV}}$ and $\boldsymbol{Z_{CV}}$ as the global CVs, by (17).
Secondary structure fragment *Y* will be compared to *X* and *Z* to *W* and the geometrical relationships within the pair *Y*, *Z* will be compared with those in the search model *X*, *W*.

Fragment length.
X and *Y* must have the same number of residues. So as a consequence they also will have the same number of CVs in their distributions.

$$|\boldsymbol{X_s}| = |\boldsymbol{Y_s}| \Leftrightarrow |\boldsymbol{X}| = |\boldsymbol{Y}| \tag{21}$$

Secondary structure.
X and *Y* should have the same secondary structure annotation, verified through the CV distribution. A pair of two successive CVs belonging to the same fragment may not differ in their length by more than 1.0 Å, or else a breakpoint is defined in the secondary structure element. Next, each CV of the distribution is compared with statistical values to check consistency and continuity for **ah**, α-helices, and **bs**, β-strands. Whenever outliers are found, before the entire fragment is rejected, a Ramachandran validation is performed for the torsions in the tripeptide, and it is also checked that the predecessor and successor of the outlier CV still belong to the predicted secondary structure annotation. In that case, a distortion of the fragment is registered, flagging as **ch**, curved helices, and **cbs**, strongly distorted strands.

$$\boldsymbol{shape}(\boldsymbol{X},\mu,\sigma) = \begin{cases} \boldsymbol{if}\ \forall \boldsymbol{i} \in \{\boldsymbol{1},\dots,\boldsymbol{n}\} & \big|\|\boldsymbol{x_i}\| - \mu\big| \le \sigma \\ & \Rightarrow \boldsymbol{regular} \\ \boldsymbol{if}\ \exists\ \boldsymbol{C} \subset \{\boldsymbol{2},\dots,\boldsymbol{n}-\boldsymbol{1}\} : \forall \boldsymbol{j} \in \boldsymbol{C} & \big|\|\boldsymbol{x_{j-1}}\| - \mu\big| \le \sigma \\ & \wedge \\ & \big|\|\boldsymbol{x_{j+1}}\| - \mu\big| \le \sigma \\ & \Rightarrow \boldsymbol{distorted} \end{cases} \tag{22}$$

$$\boldsymbol{type}(\boldsymbol{X},\mu,\sigma) = \begin{cases} \boldsymbol{if\ shape}(\boldsymbol{X},\mu = \boldsymbol{2.2},\sigma = \boldsymbol{0.18})\ \boldsymbol{is\ regular} & \Rightarrow \boldsymbol{ah} \\ \boldsymbol{if\ shape}(\boldsymbol{X},\mu = \boldsymbol{1.4},\sigma = \boldsymbol{0.24})\ \boldsymbol{is\ regular} & \Rightarrow \boldsymbol{bs} \\ \boldsymbol{if\ shape}(\boldsymbol{X},\mu = \boldsymbol{2.2},\sigma = \boldsymbol{0.18})\ \boldsymbol{is\ distorted} & \Rightarrow \boldsymbol{ch} \\ \boldsymbol{if\ shape}(\boldsymbol{X},\mu = \boldsymbol{1.4},\sigma = \boldsymbol{0.24})\ \boldsymbol{is\ distorted} & \Rightarrow \boldsymbol{cbs} \end{cases} \tag{23}$$

$$\boldsymbol{type}(\boldsymbol{X},\mu,\sigma) = \boldsymbol{type}(\boldsymbol{Y},\mu,\sigma) \tag{24}$$

Scalar product for relative orientation.
To check relative orientations between fragments, BORGES computes the scalar product of their global CVs given *γ*, the angle between *X* and *W*, and *α*, the threshold on the difference in the corresponding template and fragment angles, specified by the user.

$$\theta = \boldsymbol{cos}^{-1}\left(\frac{\boldsymbol{Y_{CV}} \cdot \boldsymbol{Z_{CV}}}{\|\boldsymbol{Y_{CV}}\|\|\boldsymbol{Z_{CV}}\|}\right) \tag{25}$$

$$\gamma - \alpha \le \theta \le \gamma + \alpha \tag{26}$$

Fragment distance.
For checking distance compatibility for Y and Z, a distance vector $\boldsymbol{D}_{yz}$ is defined. In the template, r is the length of the distance vector $\boldsymbol{D}_{xw}$, and φ is the angle between $\boldsymbol{X}_{CV}$ and $\boldsymbol{D}_{xw}$. Distance and angle have to agree with those in the template within the user-specified thresholds input in the configuration file, named d and β, respectively:

$$\boldsymbol{r - d \leq \|D_{yz}\| = \sqrt{\frac{1}{t}\sum_{i=1}^{t} C\alpha(R_i^y) - \frac{1}{m}\sum_{i=1}^{m} C\alpha(R_i^z)} \leq r + d} \tag{27}$$

$$\boldsymbol{\theta - \beta \leq \measuredangle Y_{CV}\,; D_{yz} \leq \phi + \beta} \tag{28}$$

Distribution difference.
This filter limits the maximum difference between each CV of the search fragment element X and the corresponding CV of the extracted secondary structure element Y, thus enabling the user to define the tolerance threshold, δ, within which local geometrical distortion of the fragments may differ. Physically meaningful values range between 0.15 and 0.40 Å; larger values would be comparing different types of secondary structure elements.

$$\boldsymbol{\forall i \in \{1, \dots, n\} : |\|x_i\| - \|y_i\|| \leq \delta} \tag{29}$$

Once all models are finally extracted from the PDB, they are grouped in geometrically similar clusters. Implemented clustering algorithms will be explored in Section 5.8. All fragments representing clusters are superimposed on the template and B values are all set to the same value practically flattening their contribution in MR jobs.

5.6 BORGES_MATRIX implementation

The first prototype established it was possible to define local folds, extract them by the mean of CVs and employ these libraries for successful phasing of new structures with ARCIMBOLDO_BORGES. However, the implementation required a MySQL database that was not efficient for the size of the data managed and the overall design was complicated by the dependence of the CV from the size of the fragments. In fact, even though CV distributions were computed to perform a local geometrical filter such as the *distribution difference* (Eq. 29), all operations concerning spatial conformation between fragments were computed from the global CVs (Eq. 18-20). This first approximation worked well to define the general properties of a fold but as seen in Section 5.2, CVs computed from different length size do not follow the same distribution, thus the criteria to assign secondary structure were dependent from the size of the fragments. Moreover, curved fragments are difficult to delineate by a single CV, their lengths would decrease respect to the mean of the statistical distribution but their absolute value could be difficult to separate from coiled regions. This was one of the reasons why the prototype required angle and distance thresholds that were markedly high in order to capture variations in the fold to extract. Moreover, it was not possible to extract coil regions and generally regions for which it was not possible to clearly associate a secondary structure feature, because using a single CV would be extremely affected by strong deviations. Lastly, this first algorithm required to re-compute the CVs for the deposited structures each time a run was executed, to obtain CVs derived from the same number of residues as the ones in template fragments.

A better description of a fragment may be provided by the construction of a CV distribution, as introduced in (Eq. 15-16), resulting in the following benefits:

- Each CV of the distribution is computed from a tripeptide, thus all CVs are comparable and follow the statistical distribution shown in Figure 5-5
- Description of the secondary structure of a fragment is given by the sequential annotation of helical or stranded CVs. Breaks in the sequence reveal the interruption of a secondary structure element (Figure 5-8).
- Local distortions for a fragment are reflected in the variation of CV length in the annotated sequence.
- Geometrical comparisons between fragments, orientations and distances, are computed comparing the overall distribution of CVs, considering fragment curvature.
- All the PDB database can be parsed and annotated once with a CV distribution, which is unique, all the geometrical relationships between these CVs can be calculated once and remain immutable.
- Extracting a fold from an annotated structure requires finding the subset of CVs in the CV distribution of the protein that matches the template CV distribution.
- Coiled regions, conserved loops, and non-folded regions can also be extracted, as they would map a CV distribution that can be searched against the PDB.

The workflow of the new BORGES_MATRIX implementation is described in Figure 5-10:

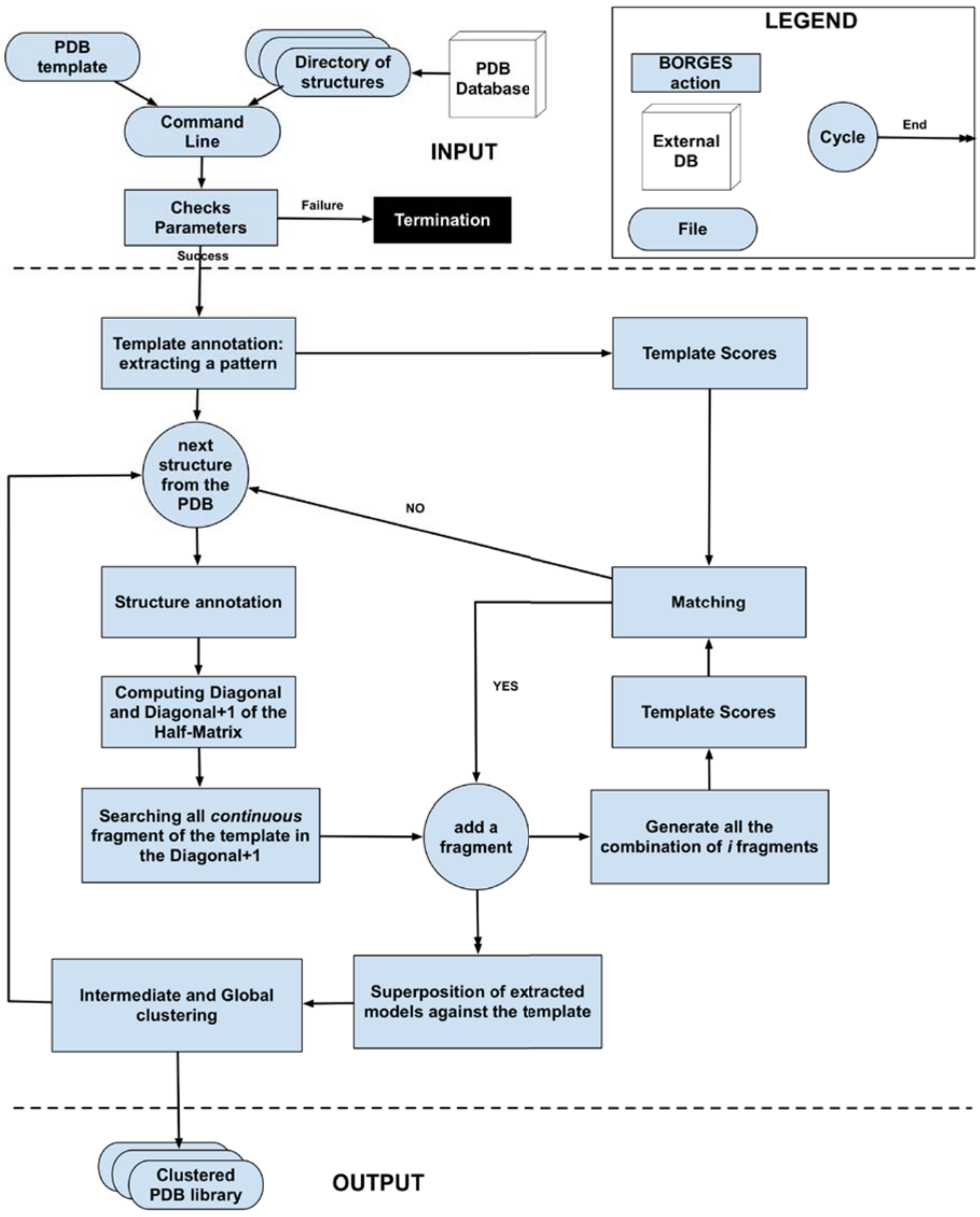

PDB template
Directory of structures
PDB Database
Command Line
INPUT
Checks Parameters
Failure
Termination
Success
LEGEND
BORGES action
External DB
Cycle
End
File
Template annotation: extracting a pattern
Template Scores
next structure from the PDB
NO
Structure annotation
Matching
YES
Computing Diagonal and Diagonal+1 of the Half-Matrix
Template Scores
Searching all *continuous* fragment of the template in the Diagonal+1
add a fragment
Generate all the combination of *i* fragments
Intermediate and Global clustering
Superposition of extracted models against the template
Clustered PDB library
OUTPUT

Figure 5-10 BORGES_MATRIX workflow

Five steps compose the workflow of a standard job in BORGES_MATRIX:

1. Read the parameterization and evaluate the template to define the local fold to extract.
2. Parse and annotate every protein stored in *pdb* file of a directory or alternatively access a precomputed database and pick sequentially annotated proteins.
3. Extract every occurrence of the local fold for each protein of step 2, compare them to the template and filter applying user defined thresholds.
4. Cluster geometrically all extracted models.
5. Superpose these models to the original template, and save to file setting a common B-value.

Step 2 implies it is possible and reasonable to generate the database once and for all, since this annotation will never change. Nevertheless, implementation of this database is an undergoing project (see Outlook section) and has not yet been performed within the new implementation. Currently the program annotates proteins in running time while executing a specific local fold search.

5.6.1 Structure annotation through CVs

The annotation of a *pdb* structure is a procedure that will convert the coordinate file to a mathematical description based on CVs used to extract local folds comparable to a template. Given a *pdb*, no matter if it is the template or a deposited structure in the PDB, we need to compute the corresponding CV distribution through Algorithm 6, in which a list of atoms is first filtered to remove possible disorder, by keeping the alternative configuration with higher occupancy, and later on is used to created a list of CVs computed from overlapping tripeptides with a sliding window of one residue.

Algorithm 6: Generation of Characteristic Vectors for a protein

```
Input:
    list of the atoms:
    $[atom\_id,atom\_name,chain\_id,res\_id,res\_name,x,y,z,occ,B]^i$
Output:
    list of CVs:
    $[id\_cv,cv\_module,[x^1,y^1,z^1],[x^2,y^2,z^2],[[chain\_id,res\_id]^3]]^n$
    chain_boundaries:
    $[[chain\_id,start\_cv\_id,end\_cv\_id]]^k$
Notes:
    i = number of atoms
    r = number of residues
    n = number of CVs
    k = number of chains

atoms   = filter_atoms_by_occupancy_and_remove_disorder(atoms)
                                                    # Additional function
coordCA  = $[x^i,y^i,z^i]$ of atoms$^i$ where atom_name is CA
coordO   = $[x^i,y^i,z^i]$ of atoms$^i$ where atom_name is O
s       = 3   # of residues per CV
n       = r - s +1

if n <= 0
    return empty list of CVs, empty list of chain_boundaries
```

```
vectorsCA = [x=0,y=0,z=0]$^n$
vectorsO  = [x=0,y=0,z=0]$^n$

for t=0 to s
        $vectorsCA^0 = \frac{vectorsCA_x^0+coordCA_x^t}{s}, \frac{vectorsCA_y^0+coordCA_y^t}{s}, \frac{vectorsCA_z^0+coordCA_z^t}{s}$
        $vectorsO^0 = \frac{vectorsO_x^0+coordO_x^t}{s}, \frac{vectorsO_y^0+coordO_y^t}{s}, \frac{vectorsO_z^0+coordO_z^t}{s}$

for t = 1 to n
            $vectorsCA_x^t = vectorsCA_x^{t-1} + \frac{coordCA_x^{t+s-1}-coordCA_x^{t-1}}{s}$
            $vectorsCA_y^t = vectorsCA_y^{t-1} + \frac{coordCA_y^{t+s-1}-coordCA_y^{t-1}}{s}$
            $vectorsCA_z^t = vectorsCA_z^{t-1} + \frac{coordCA_z^{t+s-1}-coordCA_z^{t-1}}{s}$
            $vectorsO_x^t = vectorsO_x^{t-1} + \frac{coordO_x^{t+s-1}-coordO_x^{t-1}}{s}$
            $vectorsO_y^t = vectorsO_y^{t-1} + \frac{coordO_y^{t+s-1}-coordO_y^{t-1}}{s}$
            $vectorsO_z^t = vectorsO_z^{t-1} + \frac{coordO_z^{t+s-1}-coordO_z^{t-1}}{s}$

for t = 0 to n
      vectorsH$^t$ = get vector module by euclidean distance
                                                              # Additional equation

      resids$^t$ = get residues used in vectorsH$^t$

      if not all residues are continuous in resids$^t$
            vectorsH$^t$ = 100 #break value for continuous CVs
      if not all residues belongs to the same chain
            save in chain_boundaries:
                              [[chain_id,start_cv_id,end_cv_id]]
   save in CVs list:
            [id_cv,cv_module,[x$^1$,y$^1$,z$^1$],[x$^2$,y$^2$,z$^2$],[[chain_id,res_id]$^3$]]$^t$

return CVs list, chain_boundaries
```

Additional equations:

$$vectorsH^t = \sqrt{(vectorsCA_x^t - vectorsO_x^t)^2 + (vectorsCA_y^t - vectorsO_y^t)^2 + (vectorsCA_z^t - vectorsO_z^t)^2}$$

Additional functions:

Name: filter_atoms_by_occupancy_and_remove_disorder
Input: List of the atoms:
[atom_id,atom_name,chain_id,res_id,res_name,x,y,z,occ,B]^i
Output: List of filtered atoms

```
for i=0 to length(atoms)
    if atoms$^i$ has all main chain atoms with occupancy >= 0.5
        if atoms$^i$ has disordered atoms
            save the one with higher occupancy
            if disordered atom has same occupancy
                save the first one
        else
            save atoms$^i$
return List of filtered atoms
```

For each tripeptide in a chain of the structure, the centroid of the three Cα and Oxygens are connected to define the corresponding CV. Only CVs computed from tripeptides with continuous residues are saved. Continuity is checked structurally evaluating the C-N bond length between adjacent residues. The module of the vector is calculated as the Euclidean distance and a unique id is associated to each vector. As stated in the previous Section the maximum number of CVs computed is equal to the number of residues in the chain minus two, as chain interruption or multiple disorder residues reduces the number of resulting CVs.
If we compare geometrically all possible pairs of CVs by applying Eq.18-20, we would end up with a complete spatial description of the structure relating every CV against the others. As seen in the simplified Figure 5-11, this operation corresponds to mathematically generating a complete undirected graph of relationships. Geometrical measures, angles and distances, do not change according to the order of the pairs. We can notice how among all the possible pairs some of them are formed by CVs belonging to different secondary structure elements while others are computed within the same fragment. In particular, each CV is connected to the following one sharing two residues and partially overlapping. We refer to these CVs as *continuous* as they describe the continuity of the chain itself, while all others pairs present a *jump* in the continuity and will provide information about further relations, including relations among fragments.

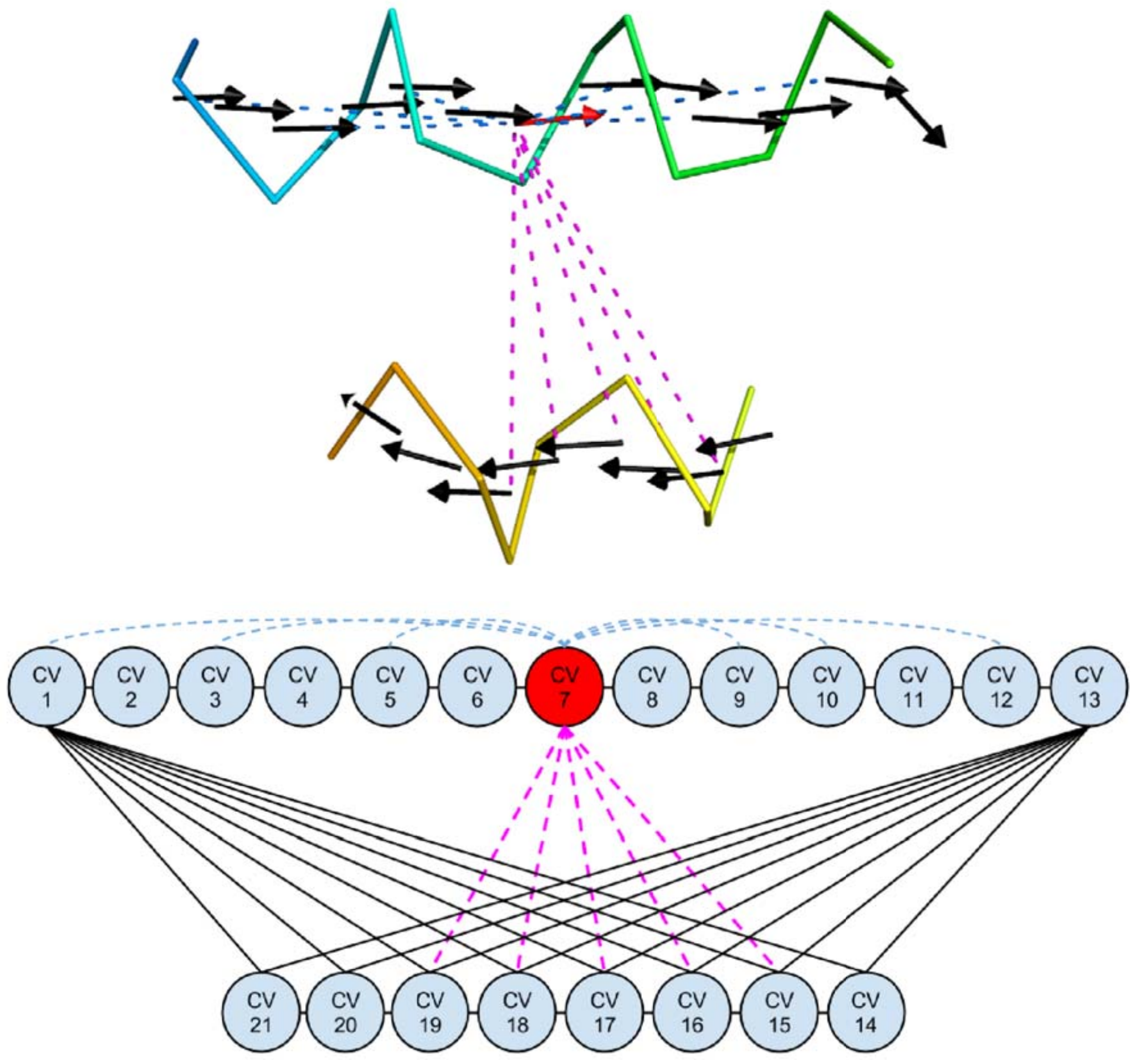


Figure 5-11 Graph representation of CV relationships

Top: Two helices of the Helletionin-D protein introduced in Figure 5-6, displayed as ribbon. All the CVs for both helices are also depicted, illustrating the direction of the helices. In BORGES_MATRIX during the annotation process all CVs are geometrically compared in pairs, a subset of these comparisons across the two helices is emphasized in magenta, while some connections between CVs belonging the same fragment are highlighted in blue. The same representation is adopted in graph 5-11 below, where the corresponding comparisons are represented along with new ones in solid grey lines. BORGES_MATRIX compares all possible pairs of CVs but for clarity only some of them are displayed in the picture.

In computer science, graphs can be easily mapped to matrices, and an undirected complete graph correspond to half of a matrix that contains $n^2/2$ elements, where **n** is the number of CVs (Figure 5-12).

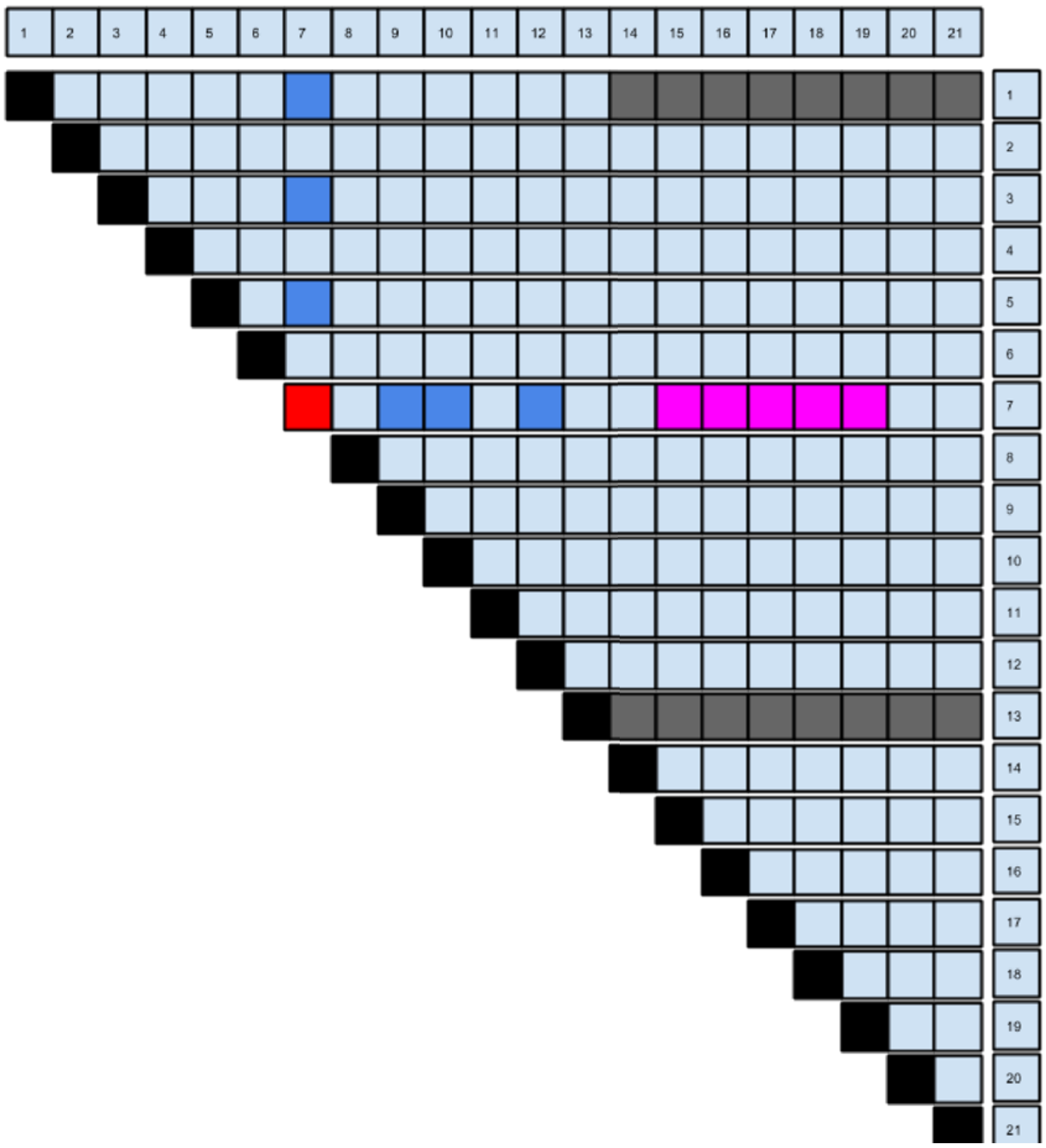


Figure 5-12 Half-Matrix representation for the CV Graph

The Graph in Figure 5-11 can be represented in a program as a half matrix in which CVs occupy the diagonal cells, in black; magenta cells are relations between CV 7 of the longer helix and some CVs belonging to the smaller helix; some comparisons between CVs belonging to the same helix are shown in blue and remaining relations between fragments, in grey.

The matrix presents some important properties:

- It is symmetric:

$$\boldsymbol{matrix_{ij} = matrix_{ji}} \tag{30}$$

- The Diagonal:

$$\boldsymbol{matrix_{ij} \forall i,j \in [1,n[\wedge i = j} \tag{31}$$

Instead of containing the geometric relations of each CV with itself, which are trivial, is used to store the description of CVs as obtained through Algorithm 7.

- The Diagonal+1:

$$\boldsymbol{matrix_{ij}}\forall \boldsymbol{i,j} \in [\mathbf{1},\boldsymbol{n}[\wedge \boldsymbol{j} = \boldsymbol{i}+\mathbf{1} \quad \mathbf{(32)}$$

  contains all geometric relations between possible *continuous* CVs. Secondary structure elements remain defined by contiguous subsets of this diagonal, which implies that searching for helices or strands, regardless of their relative orientations and distances, will only require to explore this particular set.
- All others cells in the matrix contain information about spatial relations between *jump* CVs, of particular interest being relations among fragments.

BORGES_MATRIX requires as a minimum set, the content of the Diagonal and the Diagonal+1. The rest of the matrix is not precomputed and specific cells will be generated in running time when required by the algorithm. It turns out that most of the information in the matrix is redundant when considering a specific fold search, thus it is preferable to compute this part of the cells only when needed. For the same reason, in the real implementation a Hash-Table substitutes the matrix, to occupy only the portion of memory required without loosing the benefit of the element direct access.
Algorithm 7 illustrates the generation of the Diagonal and Diagonal+1 by means of the CVs computed through Algorithm 6.

Algorithm 7: Generation of Diagonal and Diagonal+1

```
Input:
list of CVs
[id_cv,cv_module,[x1,y1,z1],[x2,y2,z2],[[chain_id,res_id]^3]]^n
Output:
[NxN] Matrix where
$matrix_{ij} \forall i,j \in [0,n[\wedge i = j$ is the Diagonal
$matrix_{ij} \forall i,j \in [0,n[\wedge j = i+1$ is the Diagonal+1.
The rest of entries are empty.
Notes:
Matrix can be implemented as hash-table to reduce memory
occupation

n = length(CVs)
for i = 0 to length(n)
    for j = 0 to length(n)
        if j == i
            $matrix_{ij} = [0,[CVs_i^1,0],0,0,0,[0]]$
        elif j == i+1
            $matrix_{ij} = [id\_cv_i - id\_cv_j,[CVs_i^1,CVs_j^1],\theta_{ij},d_{ij},\varphi_{ij},[resids_i,resids_j]]$
return matrix
```

Additional equations:

$$CV_i = [x_i^1 - x_i^2, y_i^1 - y_i^2, z_i^1 - z_i^2]$$
$$CV_j = [x_j^1 - x_j^2, y_j^1 - y_j^2, z_j^1 - z_j^2]$$
$$\theta_{ij} = arctan2(|CV_i \times CV_j|, CV_i \cdot CV_j)$$
$$DH_{ij} = [x_i^1 - x_j^1, y_i^1 - y_j^1, z_i^1 - z_j^1]$$
$$d_{ij} = \sqrt{(DH_{ij}^x)^2 + (DH_{ij}^y)^2 + (DH_{ij}^z)^2}$$
$$\varphi_{ij} = arctan2(CV_i \times DH_{ij}, CV_i \cdot DH_{ij})$$

A geometrical relationship between two CVs is determined by:

- Their *continuity*: two vectors are defined as *continuous* if they overlap by two residues, otherwise their relation is defined as *jump*
- the *angle* between the two vectors θ_{ij},
- their *distance* d_{ij}: measured as the length of a new vector connecting the Cα of the first CV to the Cα of the second,
- the *angle distance* φ_{ij}: measured as the angle between the first CV and the distance vector.

This description of a pair is used for computing the Diagonal+1 as well as any other cell of the half-matrix in running time, when required.

5.6.2 Fold extraction from an annotated structure

Once the template model is annotated with CVs the represented fold is searched for against the whole PDB, or any given set of structures. Given that the fold search from different structures constitutes an independent operation, it is easy to parallelize computation over a grid network, a supercomputer facility or just by multiprocessing in a single workstation, details of the parallelization means of our software are provided in Chapter 7. If no precompiled database is available, each structure is also annotated before performing the fold search.

Extraction of a fold from the annotated structure relies on the property of the complete graph created. Complete graphs are highly connected[133], that is, it always exists a path connecting any pair of nodes. This property is graphically represented in Figure 5-13.
Considering just the Diagonal+1 subset from the complete graph of the search template if we define the pattern of geometrical relationships that we wish to find in the annotated structure, we will have imposed a path from the first CV to the last CV of the Diagonal+1. If the annotated structure in the PDB contains this path (representing the desired fold), then it must be possible to extract it from its complete graph. The problem of the extraction of a fold is reduced to the search problem of a given path in a complete graph.
Note that in theory the choice of the path is arbitrary as long as all CVs of the template are considered at least once, but the use of the Diagonal+1 template as the path to search has the advantage of being easily manageable, more precise when weighted with the user thresholds and human-understandable. However, Diagonal+1 is just a minimal search pattern whereas the full half-matrix represents the maximum degree of geometrical relationships defined for any fold. The level of detail entailed in the full half-matrix is so high that it would require fine-tuned parameterization and cost high computational power. On the other hand, the geometrical relation between fragments contained in the Diagonal+1 may be insufficient. BORGES_MATRIX adopts a compromise, making use of the Diagonal+1 with additional CV pairs (corresponding to cells in the matrix) to improve outcome without being too RT demanding.
When BORGES_MATRIX annotates the search template it reads the Diagonal+1 (Eq. 32) and looks for *jump* pairs in it, in other words, it searches for breaks in the main chain that would identify separate fragments composing the fold. As example the template in Figure 5-11, contains two helices that are clearly not connected, thus there will be a *jump* pair in the Diagonal+1 relating a CV of the first helix with a CV of the second, in this example the pair is the following: (CV_{13}, CV_{14}). The program annotates the limits of each fragment, again in the example the fist helix is delimited at (CV_{1}, CV_{13}) and the second at (CV_{14}, CV_{21}). Note that if the input template describes a single, uninterrupted main chain, then BORGES_MATRIX will annotate the entire fold as a single fragment. Hence, it is not

required that the template contains only defined secondary structure elements. On the contrary, loops and coiled regions can also be included and extracted with BORGES_MATRIX.

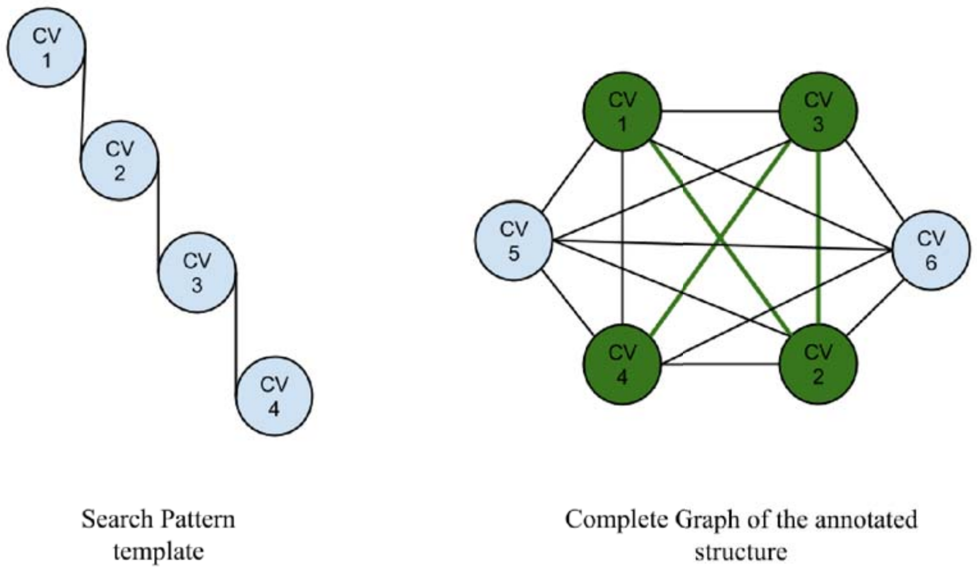


Figure 5-13 Complete Graph connectivity property

A complete graph is highly connected. The example graphically shows this property: a path of 4 CVs can be found in the complete graph if it exists.

In general, if z fragments are found, for each one of them f^k the limits (f_s,f_e) are defined. BORGES_MATRIX generates the search pattern by adding to the Diagonal+1 other pairs relating the fragments, obtaining:

$$\begin{cases} \boldsymbol{matrix_{ij}} \forall \boldsymbol{i,j} \in [\boldsymbol{1},\boldsymbol{n}[\wedge \boldsymbol{j} = \boldsymbol{i}+\boldsymbol{1} \\ \boldsymbol{matrix_{ij}} \ \forall \boldsymbol{f^k},\boldsymbol{f^l} \in [\boldsymbol{1},\boldsymbol{z}[\,;\ \forall \boldsymbol{i},\boldsymbol{j} \in [\boldsymbol{1},\boldsymbol{n}[\wedge \begin{array}{r} \boldsymbol{i} = \boldsymbol{f^k_s}\,; \boldsymbol{j} = \boldsymbol{f^l_s} \\ \boldsymbol{i} = \boldsymbol{f^k_s}\,; \boldsymbol{j} = \boldsymbol{f^l_e} \\ \boldsymbol{i} = \boldsymbol{f^k_e}\,; \boldsymbol{j} = \boldsymbol{f^l_s} \\ \boldsymbol{i} = \boldsymbol{f^k_e}\,; \boldsymbol{j} = \boldsymbol{f^l_e} \\ \boldsymbol{i} = \boldsymbol{f^k_{\lfloor\frac{s+e}{2}\rfloor}}\,; \boldsymbol{j} = \boldsymbol{f^l_{\lfloor\frac{s+e}{2}\rfloor}} \end{array} \end{cases} \qquad \textbf{(33)}$$

In this way, the search pattern is enriched in geometrical details describing spatial relations between fragments. Note that both index k and l in Eq. 33 range from 1 to z, implying that if a single fragment was found in the fold, Eq. 33 would work associating the fragment to itself, ensuring that even for single fragments geometrical relationships between remote CVs are included in the pattern procuring a more accurate and realistic description of the fold.

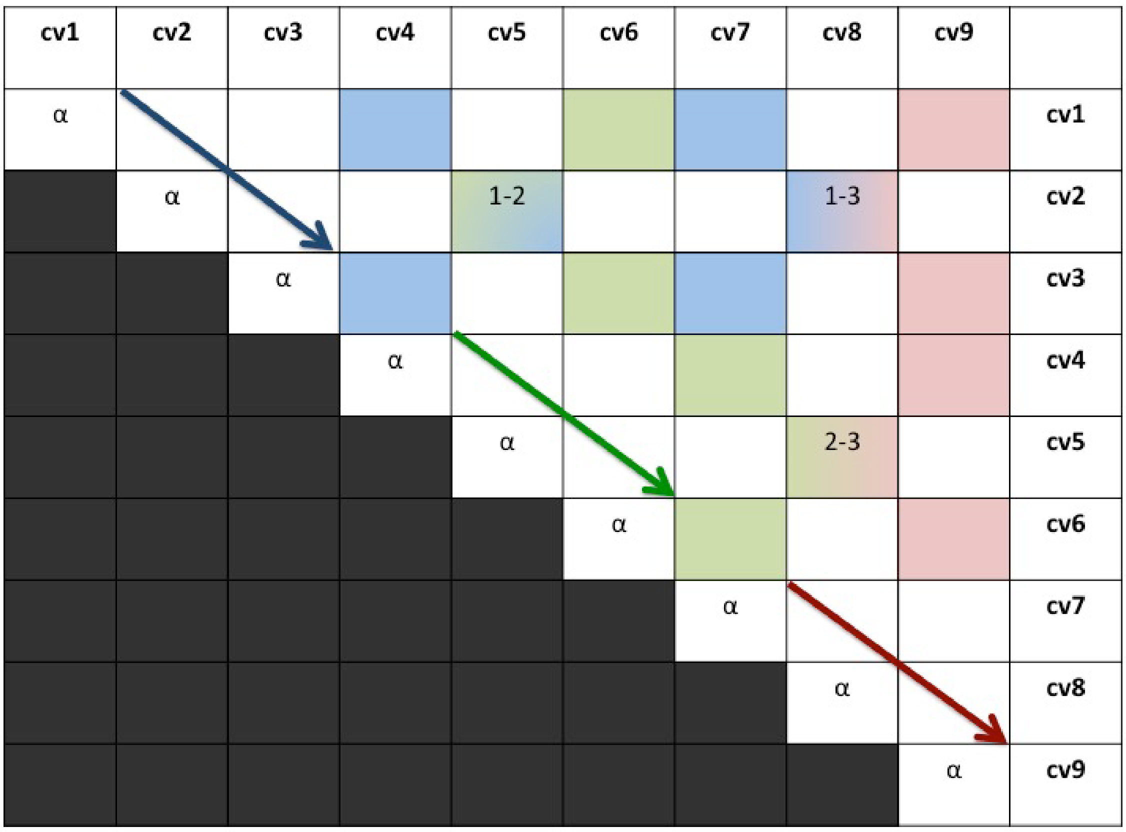


Figure 5-14 An example of search pattern

Illustration of the definition of a search pattern for 3 disconnected helices of 5 residues. Black cells are omitted from the matrix. The Diagonal stores the information of each singular CV, the Diagonal+1 shows the continuity of fragments. Coloured cells are computed and included in the search pattern and describe relations among fragments.

Figure 5-14 displays an example of a search pattern for a fold composed of 3 helices each one of 5 residues (a CV is computed for each 3 residues by a sliding window of 1). The Diagonal stores the secondary structure feature of each CV, the Diagonal+1 reports on continuity of the secondary structure elements highlighted with coloured arrows. From the figure, it can be inferred that the pairs (CV3, CV4), (CV6, CV7) are *jump* pairs, and they correspond to the interruption in the chain delimiting the boundaries for the three helices. Notice that knowing the start and the end of each fragment BORGES_MATRIX computes through Eq. 32, the content of the coloured cells and adds them to the search pattern. These new matrix entries introduce the geometrical description within a pair of fragments, giving a detailed description of the fold in space.

Once all cells composing the search pattern are computed, the search template is scored. This operation aims to summarize numerically the geometrical properties of the pattern. In particular two scores are generated:

- one refers to the local geometry of each fragment in the fold (entailing curvatures, distortions, kinks, etc.) that is named *continuous* score and it is computed only from *continuous* pairs
- the other refers to the spatial arrangement of these fragments in the space (entailing relative orientations such as parallel, antiparallel, etc.) that is named *jump* score and is computed only from *jump* pairs.

Enhanced tuning of the scoring function should be implemented in the future; at present it simply consists on the numerical sum of all equal geometrical features (CV lengths, angles, distance lengths, angle distances) for all pair of CVs in the pattern divided as described above. A better description would take some statistical metric to weight the contribution of each CV.

As example, the following might be the scoring output for two small helices of 6 residues:

```
=====================Frag. n.: 1 ==================================
SS    cv1  cv2        c/j    (cvl1, cvl2)    angle         dist.         angle dist.
ah    0        1      1      (2.20, 2.23)    7.00          1.72          21.23
ah    1        2      1      (2.23, 2.19)    6.94          1.75          20.63
ah    2        3      1      (2.19, 2.22)    10.5          1.63          29.32
ah    3        4      1      (2.22, 2.28)    9.94          1.70          25.12
=====================Scores Frag. n.: 1 ==============================
Score CV:                11.12         #Sum of all CV in the fragment
Score angle vectors:     34.38         #Sum of all angle vectors
Score distance:           6.80         #Sum of all distances
Score angle distance:    96.30         #Sum of all angle distances
=====================Frag. n.: 2 ==================================
SS    cv1  cv2        c/j    (cvl1, cvl2)    angle         dist.         angle dist.
ah    5        6      1      (2.28, 2.17)    10.1          1.80          14.06
ah    6        7      1      (2.17, 2.26)    10.5          1.54          29.01
ah    7        8      1      (2.26, 2.25)    4.72          1.83          23.46
ah    8        9      1      (2.25, 2.24)    10.0          1.69          20.80
=====================Scores Frag. n.: 1 ==============================
Score CV:                11.20         #Sum of all CV in the fragment
Score angle vectors:     35.32         #Sum of all angle vectors
Score distance:           6.86         #Sum of all distances
Score angle distance:    87.33         #Sum of all angle distances
=====================Check:  0 1 ==================================
SS    cv1  cv2        c/j    (cvl1, cvl2)    angle         dist.         angle dist.
ah     0       5      7      (2.20, 2.24)    143.10        36.27         21.13
ah     0       9      11     (2.20, 2.00)    164.04        10.30         70.71
ah     4       5      3      (2.15, 2.24)    166.41        12.47         41.14
ah     4       9      7      (2.15, 2.00)    153.43        46.78         10.74
ah     2       7      7      (2.22, 2.23)    165.65        10.48         61.23
=====================Scores Check: 0 1 ============================
Score CV:                      13.04
Score angle vectors:    792.63
Score distance:         116.31
Score angle distance:   204.95
=================================================================
```

In this Output the column **c/j** refers to the pair being *continuous*, if the number is equal to 1, or *jump* otherwise.

Scores are compared to the ones calculated after extracting a fold from a deposited protein, and the fold will be accepted if they match to a degree established by the user. These parameters are expressed as two percentages: one to match *continuous* pairs (parameter –C) and another to match *jump* pairs (parameter –J). Thus, if the user selects –C100 –J100, the scores obtained from the template and from the extracted fold should 100% match. On the contrary, -C0 –J0 will accept anything with the same length of the template.

BORGES_MATRIX proceeds extracting the pattern from an annotated structure executing the following steps:

1. Sort the fragments in the search pattern by their size. Searching first for longer fragments accelerates convergence since smaller fragments are ubiquitous.
2. Take one annotated structure from the database or annotate it in running time.
3. Create a matrix per Chain in the annotated structure, and search the fold separately in each matrix, unless a special parameter (-N) is input to merge all chains into a single one, corresponding to a single matrix. This option can be useful for searching a fold, which can be formed across chains or NCS copies. Note that symmetry copies are not contemplated.
4. If there is only one fragment in the search pattern:
    a. Probe all possible starting points in the Diagonal+1 and extract all the fragments of the size searched. A starting point is a cell in the matrix, whose CVs match the secondary structure type of the fragment searched.
    b. Compute scores for extracted models and compare scores against the template filtering by user threshold.
    c. Store the ones that survived the filtering step
5. If there are multiple fragments in the search pattern:
    a. Extract from the Diagonal+1 the first two fragments sorted in the step 1
    b. Filter fragments that produce *continuous* scores incompatible with the threshold specified by the user.
    c. Create all binary combinations of the remaining fragments.
    d. Compute *jump* scores for the combinations and compare them with the template filtering out incompatible ones.
    e. If there are more fragments to search, extract the next one in the list of step 1 and perform step b. Create ternary combinations and test them as in d. Repeat this step, increasing the combinatorial unit, until no more fragments remain to be searched.
    f. Store the combinations that survived all through the procedure.
6. If required (parameter –S), check for the sequence filtering out solutions that do not match the input sequence. The sequence inserted should match the template size. The symbol X indicates any valid residue. This mode is used to look for repeats or for conserved Cysteines.
7. If required (parameter –B), check for disulphide bridges filtering out solutions that do not form at least a disulphide bridge, which is located evaluating the distances between SG Cysteine atoms.

5.7 Model superposition

After having extracted all matching models from an annotated structure, BORGES_MATRIX proceeds to superpose them against the template and compute an RMSD. The problem of superposing two structures can be reduced to the problem of finding the optimal roto-translation that applied to the target structure minimizes/maximizes a defined FOM. The general approach is to minimize the RMSD, which usually requires the definition of a *core* of residues of equal size for both model and template. However, other approaches have been widely used, especially in the field of structure prediction, such as the global distance test (GDT), used for assessing current modelling techniques during the biannual critical assessment of protein structure prediction

experiment (CASP)[134], and TM-SCORE[135] that extends GDT to a protein-size dependent scale. BORGES_MATRIX relies on RMSD minimization, as most of the crystallographic community, but future implementation would explore other FOMs as complementary distance measures. Regarding the procedures to solve the orthogonal procrustes problem of finding the rotation matrix mapping the two structures, BORGES_MATRIX follows a Kabsch[127] based algorithm. The algorithm computes also a covariance matrix, which gives the statistical dispersion of atoms between the two sets. A newer implementation, Golub[136], uses Singular Value Decomposition (SVD)[137] to find the optimal rotation matrix. It is implemented in scientific programming libraries such as Biopython[138], currently employed by BORGES_MATRIX. An extensions to the Kabsch algorithm with Maximum Likelihood (ML), the Theseus[139] algorithm, is also supported in BORGES_MATRIX in particular it is activated to simultaneously compute multiple structural alignments in the intermediate clustering step between models extracted from the same *pdb* that is discussed in the next section.
In BORGES_MATRIX, superposition is driven by the minimization of RMSD between the extracted model and the search template, but the definition of the *core* is not based on pairwise local sequence alignments, such as Smith-Waterman[140] and its extensions. BORGES_MATRIX libraries do not necessarily share sequence identity among the models, and are extracted employing only geometrical criteria. On the other hand, Secondary Structure Matching (SSM)[141] algorithms based on graph matching and implemented in well-known graphical software as COOT[142], or in PYMOL[143] with its own implementation *Super*. SSM algorithms are more stable and give better results than sequence alignment based ones, but have difficulties to build a minimal graph from small, segmented folds. Models extracted with BORGES_MATRIX tend to share a central core in common with deviations concentrated in the boundaries of the fragments. Thus, the program generates alternative *cores* for all combinations obtained eliminating a given number of N/C-terminal residues per fragment. The user can select this number with the parameter –F, its default being set to 1. Obviously increasing it will introduce more computation to test each individual *core* but will also increase the chances of improving superposition based on aligned residues. The number of resulting models corresponds to the binomial coefficient:

$$\boldsymbol{if - Fn \Rightarrow T_n = \sum_{K=1}^{n} k = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = \binom{n+1}{2}} \qquad \textbf{(34)}$$

The so created *cores* are input in a shuffled random order to the referred superposition methods to find an initial roto-translation, which is then refined with Nilges[144] algorithm. Nilges iteratively searches for highest dispersions between atoms and identifies ordered regions of aligned atoms determining the local optimal roto-translation. Figure 5-15 reports an example of the effects of the different superposition methods and choice of the cores. A discontinuous β-sheet template, in cyan, of 4 parallel strands is superposed to an equivalent model extracted from the PDBID 1YLO. Both template and model have the same size of 24 residues, and even if the model presents the same overall fold, the single strands display locally strong deviations from the template. Applying the original Kabsch algorithm (a) over all 24 residues renders an RMSD of 3.49 Å. The alignment is clearly not optimal, the superposition distributes evenly atom deviations resulting in no pair being perfectly aligned. Golub improvement (b) is superior, producing 1.45 Å over the same number of residues, and it is not possible to improve without reducing the core. PYMOL with *Super* (c) obtains a good alignment of 0.83 Å over a core of 20 residues. This method is more sophisticated than a classical SSM approach but is based on the same theory of superposing graphs built from secondary structure connected elements. The SSM algorithm, via COOT, did not work with this example, being impossible to build the corresponding graph, possibly because of the lack of connectivity between fragments. The method implemented in BORGES_MATRIX (d) which is the combination of ML structural alignment algorithm based on Theseus over a core of residues built by removing as a maximum 3 residues per fragment from their edges and refining by Nilges the resulting roto-translation, yielded 0.59 Å RMSD over 18 aligned residues.

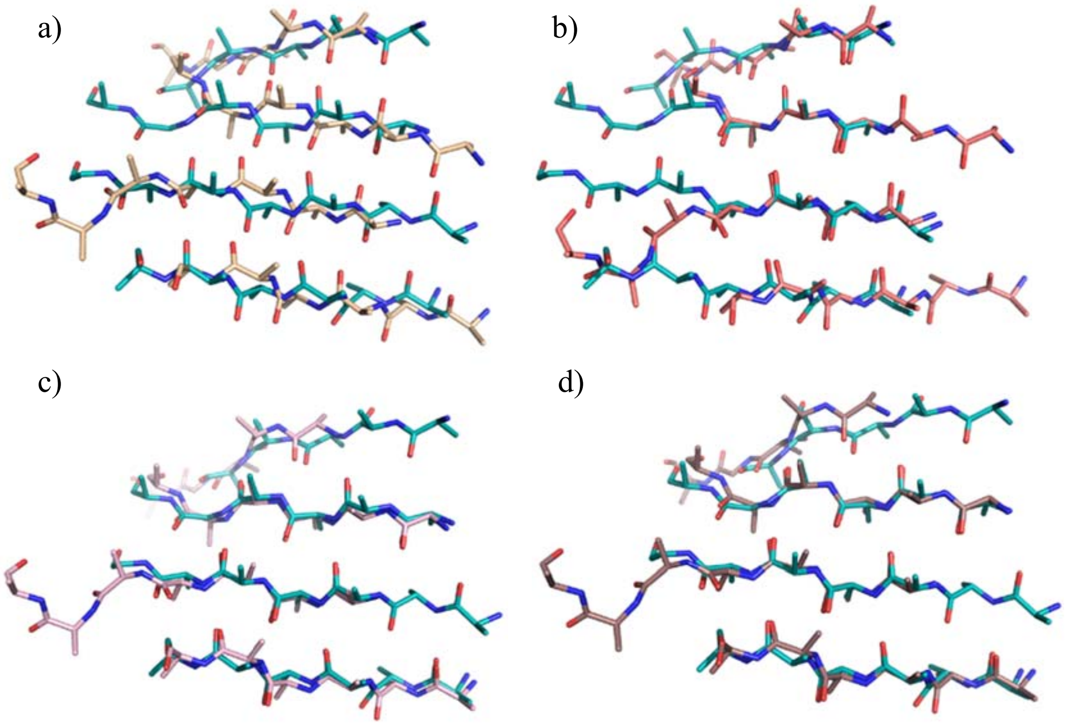


Figure 5-15 Example of the effects of different superposition methods

A template of four parallel strands (24 aa) in cyan is superposed to a model of similar fold (24 aa) with different algorithms. In a) the classical *Kabsch* algorithm using all 24 residues produces an RMSD of 3.49 Å. In b) *Golub* algorithm computed over the 24 residues produces 1.46 Å RMSD. In c) *Super* algorithm from PYMOL yields an RMSD of 0.82 Å over a core of 20 residues. In d) ML algorithm based on *Theseus* over a core of 18 residues, computed from BORGES_MATRIX allowing removal of a maximum of 3 residues per fragment (-F3) and followed by *Nilges* produced an RMSD of 0.59 Å.

No unique method is ideal for all applications, and BORGES_MATRIX allows to switch between different algorithms because each can be useful in a specific context. For example, superposing models to create libraries useful for phasing can require to optimally align a central core accepting deviations in the ends but if superposition is computed to evaluate structural homology an overall conservation of the tertiary structure would be a stronger indicator for such property.

5.8 Clustering models

BORGES_MATRIX is a versatile program that can be parameterized for different research tasks. Its original purpose was to generate libraries of models for phasing unknown structures but our perspective is to broaden its usage for the structural understanding of local folds conserved across proteins regardless of their function. The general strategy of using BORGES_MATRIX is completely different in the two cases.

Libraries employed for phasing would require the program to extract models sharing the same overall fold, in terms of secondary structure elements, but in which the difference between models must be significant enough to justify the use of the library instead of a single representative model. In this approach BORGES_MATRIX needs to extract models with a discrete tolerance for dissimilarities and then group them by a stringent clustering reducing the library to only models with differences above a certain RMSD threshold. Establishing this threshold is not trivial, as phasing depends on other considerations, such as the resolution of the crystallographic data. It is reasonable to think that a crystal diffracting to 3.0 Å cannot really benefit from a library whose cluster references differ by 0.5 Å. While at higher resolution, local differences between models can be more critical. An operative approach is to produce a number of clusters as large as could be realistic managed by the available computing resources and time constraints. These clusters can be later on divided in subclusters to consider smaller differences.
On the other hand, for bioinformatics research aiming to study the properties of small discontinuous local folds, BORGES_MATRIX might be used to search a detailed fold with a stringent parameterization and clustering can be used to directly spot subtle differences that should be later on investigated to infer some kind of structural feature.
Accordingly, the number of models extracted in the two cases would reasonably vary and would depend from many unpredictable factors: the search template itself, for example, can be more or less represented in the PDB database. It is not surprising that a three stranded antiparallel β-sheet can be extracted hundred thousand times from the whole PDB. BORGES_MATRIX in fact would extract any compatible fold with a minimum difference of 1 residue. It turns out that clustering is the most demanding step in the generation of the library. BORGES_MATRIX can extract complete libraries of models in one or two days depending on the complexity of the template and its parameterization. But their clustering can require weeks, especially when hundred thousand models are produced.
The current clustering approach is not yet optimal, it is extremely simple and focused on the creation of libraries for phasing. Projects are underway, to extend, improve and assess this step for a broader usage, employing new algorithms based on statistical analysis and data mining. Macromolecular structural clustering is a hot topic in fields as Molecular Dynamics (MD)[145] where Principal Component Analysis (PCA) is widely used for the study of atom trajectories in a MD simulation. Other software solutions, such as MACXLUSTER[146], combine complex distance measures, TM-score and GDT (Section 5.7), with the classical RMSD. They give access to hierarchical clustering[147], Nearest Neighbour[148] that being a fuzzy clustering allows one model to overlap more clusters, and 3D-Jury[149] that provides a method of identifying the most frequent structure found within a library but not a means of identifying different, frequently occurring structures, for example two different folds within a population. Many Bioinformatics groups create their own clustering approach although their research is principally directed to other topics as in the case of ROSETTA[150] a software for *ab initio* protein prediction, that also includes many algorithms and procedures to perform structural clustering through hierarchical an K-means clustering. From a further but related field, Computer Science, new approaches[151] are rising for computing optimization to make affordable large-scale data clustering.
BORGES_MATRIX performs two different clustering steps: one among models extracted from the same structure and another among models from different ones.

5.8.1 Intermediate clustering

This step immediately follows extracting all models from a structure. These models are more likely to be very similar because they could be extracted from equal monomers in different chains, or simply they differ only by a bounce of residues in the edges. For example from an helix of 30 residues BORGES_MATRIX can extract a maximum of 16 helices of 14 residues, and if two parallel helices are extracted from the same coiled coil, some of them will be obviously very similar differing by just one residue at each end, so that filtering redundant models at this step can reduce RT. BORGES_MATRIX outputs statistics regarding this clustering step. By default this procedure is always activated but can be switched off if extracting every single occurrence of a given fold is wished. The clustering is a K-mean clustering[92] in which the data descriptor vector for each extracted model contains geometrical parameters describing the structural conformation of the model[152, 153]:

- The RMSD of the model against the template is computed as described in the previous Section.
- The centre of mass $\mathbf{c_m}$

$$\boldsymbol{structure_{mass} = s_m = \sum_{atom}^{structure} atom_{mass}} \tag{35}$$

$$\boldsymbol{center^x_{mass} = c^x_m = \frac{\sum_{atom}^{structure} atom^x \cdot atom_{mass}}{structure_{mass}}} \tag{36}$$

$$\boldsymbol{center^y_{mass} = c^y_m = \frac{\sum_{atom}^{structure} atom^y \cdot atom_{mass}}{structure_{mass}}} \tag{37}$$

$$\boldsymbol{center^z_{mass} = c^z_m = \frac{\sum_{atom}^{structure} atom^z \cdot atom_{mass}}{structure_{mass}}} \tag{38}$$

- The moment of inertia tensor **T** of the structure that is related to the inertia of a rigid body to accelerate in rotation around an axis. It is calculated through:

$$\boldsymbol{d_x = (a^x - c^x_m)\ ;\ d_y = (a^y - c^y_m)\ ;\ d^z = (a^z - c^z_m)} \tag{39}$$

$$\boldsymbol{Q = \sum_{atom=a}^{structure} \begin{pmatrix} [d_y^2 + d_z^2]\cdot a_m & [-1d_x \cdot d_y]\cdot a_m & [-1d_x \cdot d_z]\cdot a_m \\ [-1d_x \cdot d_y]\cdot a_m & [d_x^2 + d_z^2]\cdot a_m & [-1d_y \cdot d_z]\cdot a_m \\ [-1d_x \cdot d_z]\cdot a_m & [-1d_y \cdot d_z]\cdot a_m & [d_x^2 + d_y^2]\cdot a_m \end{pmatrix}} \tag{40}$$

$$\boldsymbol{T = \begin{pmatrix} \sqrt{\frac{5}{2s_m} * (Q^1_{eigenvalues} - Q^2_{eigenvalues} + Q^3_{eigenvalues})} \\ \sqrt{\frac{5}{2s_m} * (Q^3_{eigenvalues} - Q^1_{eigenvalues} + Q^2_{eigenvalues})} \\ \sqrt{\frac{5}{2s_m} * (Q^2_{eigenvalues} - Q^3_{eigenvalues} + Q^1_{eigenvalues})} \end{pmatrix}} \tag{41}$$

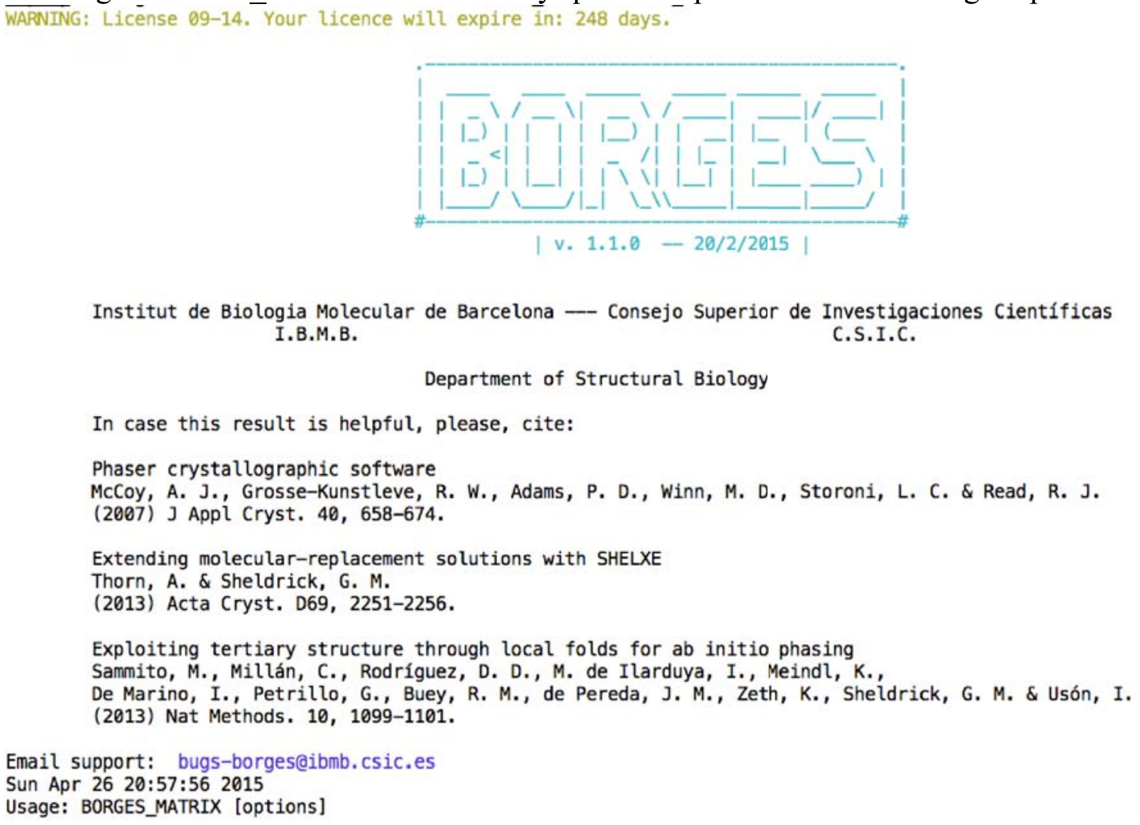
The number of clusters, K-value, is found dynamically starting from the empirical value of the root square of half of the models[93] and increasing to minimize the variance within each cluster. The model of lowest RMSD is chosen as representative for each cluster and output as *pdb* file.

5.8.2 Global clustering

Once structures in the database have been analysed, a second clustering among all extracted models takes place. This step can require days and even weeks if the number of models is in the order of hundred thousand. It is an iterative procedure in which all models superimposing against a reference with an RMSD below a specified threshold (default is 1.5 Å), are ascribed to the same cluster. The first reference is the search template used to extract the fold, next the template would be randomly picked from all the remaining unclustered models. Finally, the reference of each cluster is used to write the clustered sampled library. However the full hierarchy of classification is conserved in a way that would always allow to expand a full cluster if required.

5.9 User-interface and settings

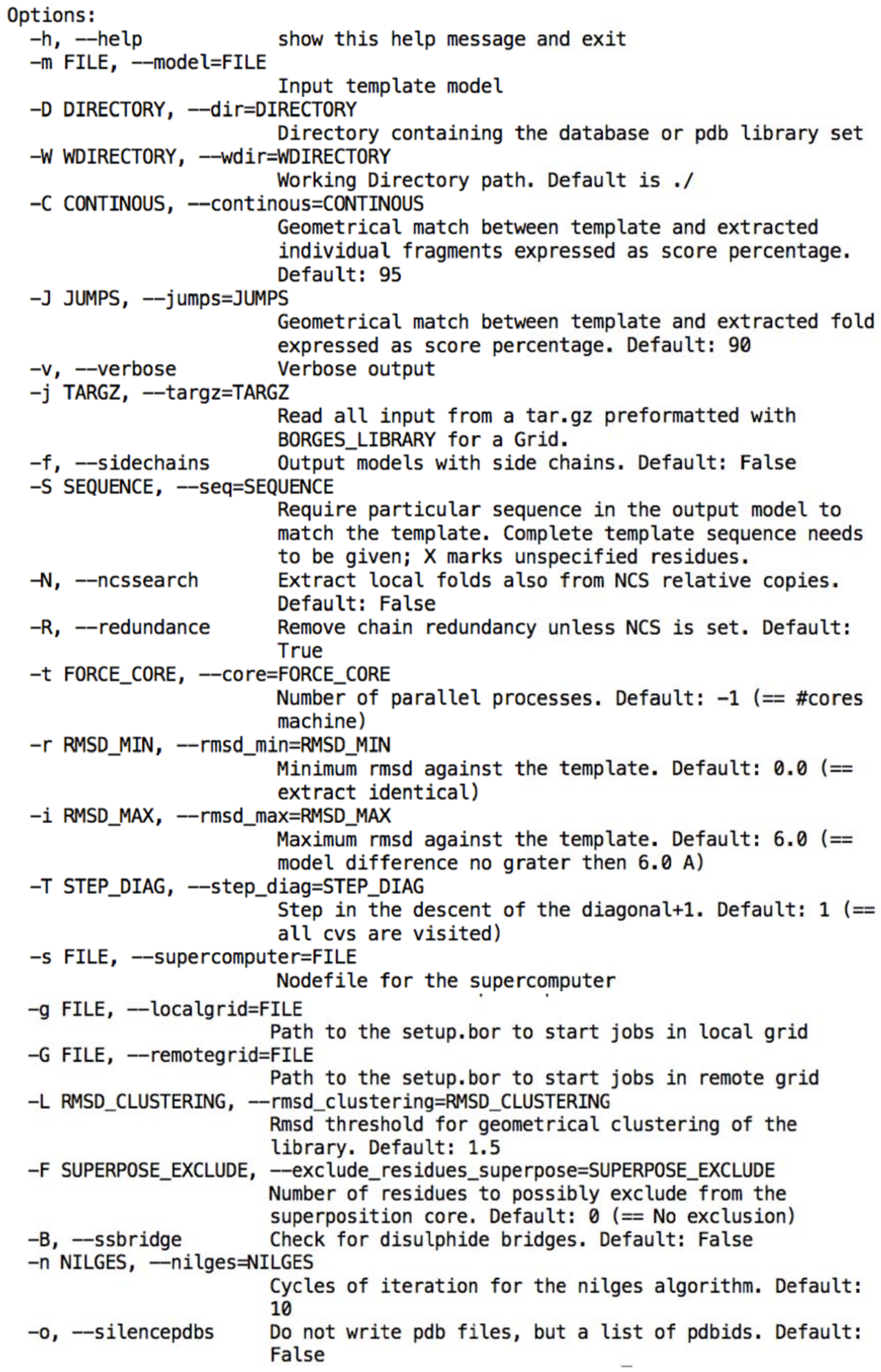
Invoking BORGES_MATRIX without any qualifiers produces the following help:



```
WARNING: License 09-14. Your licence will expire in: 248 days.

                              | v. 1.1.0  — 20/2/2015 |

        Institut de Biologia Molecular de Barcelona --- Consejo Superior de Investigaciones Científicas
                     I.B.M.B.                                          C.S.I.C.

                                  Department of Structural Biology

        In case this result is helpful, please, cite:

        Phaser crystallographic software
        McCoy, A. J., Grosse-Kunstleve, R. W., Adams, P. D., Winn, M. D., Storoni, L. C. & Read, R. J.
        (2007) J Appl Cryst. 40, 658-674.

        Extending molecular-replacement solutions with SHELXE
        Thorn, A. & Sheldrick, G. M.
        (2013) Acta Cryst. D69, 2251-2256.

        Exploiting tertiary structure through local folds for ab initio phasing
        Sammito, M., Millán, C., Rodríguez, D. D., M. de Ilarduya, I., Meindl, K.,
        De Marino, I., Petrillo, G., Buey, R. M., de Pereda, J. M., Zeth, K., Sheldrick, G. M. & Usón, I.
        (2013) Nat Methods. 10, 1099-1101.

Email support:  bugs-borges@ibmb.csic.es
Sun Apr 26 20:57:56 2015
Usage: BORGES_MATRIX [options]
```

Figure 5-16 Header of the current version of BORGES_MATRIX

BORGES_MATRIX started with no parameters and shows a header with the version number, release date, expiry date, and scientific references.



```
Options:
  -h, --help            show this help message and exit
  -m FILE, --model=FILE
                        Input template model
  -D DIRECTORY, --dir=DIRECTORY
                        Directory containing the database or pdb library set
  -W WDIRECTORY, --wdir=WDIRECTORY
                        Working Directory path. Default is ./
  -C CONTINOUS, --continous=CONTINOUS
                        Geometrical match between template and extracted
                        individual fragments expressed as score percentage.
                        Default: 95
  -J JUMPS, --jumps=JUMPS
                        Geometrical match between template and extracted fold
                        expressed as score percentage. Default: 90
  -v, --verbose         Verbose output
  -j TARGZ, --targz=TARGZ
                        Read all input from a tar.gz preformatted with
                        BORGES_LIBRARY for a Grid.
  -f, --sidechains      Output models with side chains. Default: False
  -S SEQUENCE, --seq=SEQUENCE
                        Require particular sequence in the output model to
                        match the template. Complete template sequence needs
                        to be given; X marks unspecified residues.
  -N, --ncssearch       Extract local folds also from NCS relative copies.
                        Default: False
  -R, --redundance      Remove chain redundancy unless NCS is set. Default:
                        True
  -t FORCE_CORE, --core=FORCE_CORE
                        Number of parallel processes. Default: -1 (== #cores
                        machine)
  -r RMSD_MIN, --rmsd_min=RMSD_MIN
                        Minimum rmsd against the template. Default: 0.0 (==
                        extract identical)
  -i RMSD_MAX, --rmsd_max=RMSD_MAX
                        Maximum rmsd against the template. Default: 6.0 (==
                        model difference no grater then 6.0 A)
  -T STEP_DIAG, --step_diag=STEP_DIAG
                        Step in the descent of the diagonal+1. Default: 1 (==
                        all cvs are visited)
  -s FILE, --supercomputer=FILE
                        Nodefile for the supercomputer
  -g FILE, --localgrid=FILE
                        Path to the setup.bor to start jobs in local grid
  -G FILE, --remotegrid=FILE
                        Path to the setup.bor to start jobs in remote grid
  -L RMSD_CLUSTERING, --rmsd_clustering=RMSD_CLUSTERING
                        Rmsd threshold for geometrical clustering of the
                        library. Default: 1.5
  -F SUPERPOSE_EXCLUDE, --exclude_residues_superpose=SUPERPOSE_EXCLUDE
                        Number of residues to possibly exclude from the
                        superposition core. Default: 0 (== No exclusion)
  -B, --ssbridge        Check for disulphide bridges. Default: False
  -n NILGES, --nilges=NILGES
                        Cycles of iteration for the nilges algorithm. Default:
                        10
  -o, --silencepdbs     Do not write pdb files, but a list of pdbids. Default:
                        False
```

Figure 5-17 BORGES_MATRIX command help

All parameters are input by command line. The parameters –C and –J explained in Section 5.6.2 define the threshold to extract the fold, input as *pdb* file path with –m, from the set of *pdb* structures in the directory indicated with –D. To reduce the number of extracted fragments from the Diagonal+1 the user can set up a step with –T, reducing the available starting position to generate fragments. Output directory can be modified with –W, side chains can be conserved from the extracted models with the option –f or the complete creation of the library can be muted with –o (useful for exploratory tests or for producing only statistics). Search of the fold can be performed from a non-redundant set of structures (-R) or considering also interactions between NCS copies (-N). A specific sequence, or part of it employing X for undefined residues, can be checked to be present in all the models extracted. BORGES_MATRIX can also filter out all the models that do not contain any disulphide bridge by activating the parameter –B. Models are filtered by global parameters based on RMSD against the template, it is possible to specify the minimum (-r) or the maximum (-i) RMSD required. Parameters relative to the superposition are –F, the number of removable residues per fragment, and –n indicating the numbers of cycle in which perform the Nilges algorithm. The only parameter for clustering is –L specifying the threshold RMSD for the superposition of the models against their reference cluster.
The program is launched in multiprocessing by default using all available cores minus one, unless the parameter –t is otherwise configured. A local (-g) or remote (-G) grid can used to speed computation by specifying the path of its configuration (see the Section 7.4) or alternatively a supercomputer can be addressed (-s) by specifying the list of assigned node machines for the program (Section 6.3).

5.10 Libraries of typical folds: β-sheets and helical arrangements.

The first prototype of BORGES[7] was used to generate a collection of libraries of standard folds ubiquitous across families and structures to employ them for *ab initio* phasing with the method ARCIMBOLDO_BORGES described in Chapter 6. Nomenclature to indicate a particular fold makes use of the abbreviation **u** for *up* and **d** for *down* referring to the direction of the CVs within each secondary structure element.
In particular, libraries describing the conformation of contiguous parallel (uu) and antiparallel (ud) helices and libraries of parallel (uuu), parallel-antiparallel (uud) and antiparallel (udu) β-sheets were extracted from the entire PDB database.

	Fold	**aa**	**PDB template**	**aa ranges**	**Parameterization**	**Models**	**Clusters**
Helical	uu	32	3RK2	E40-55, H157-172	(15º 3 Å 0.10 Å)	460,000	11,421
	ud	34	3KFW	X163-179, X182-198	(15º 3 Å 0.10 Å)	2,500,000	6,343
Sheets	uuu	20	1C7E	A4-9, A52-58, A86-92	(50º 3 Å 0.40 Å)	69,314	5,844
	uud	20	1AUK	A22-27, A274-281, A313-318	(50º 3 Å 0.40 Å)	79,938	7,734
	udu	20	4AEQ	B66-71, B86-92, B96-102	(50º 3 Å 0.40 Å)	925,300	7,650

Table 5-7 Summary of helical and sheet libraries computed

Table 5-7 summarizes the set of libraries created. Templates were not chosen with a particular criterion but were arbitrarily picked to represent a given local fold. Our aim was to create general libraries to phase structures unrelated to the template. α-helical templates are of different size for historical reasons: they were created for different purposes with no intention to compare their results.

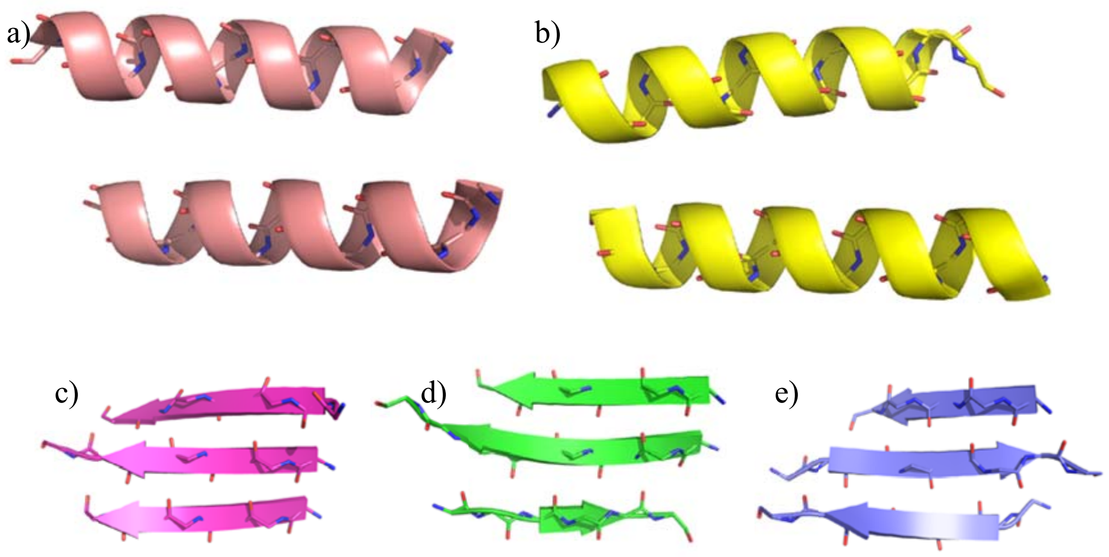


Figure 5-18 Library templates

a)Parallel helices, b) antiparallel helices, c)parallel β-strands, d) parallel-antiparallel β-strands, e) antiparallel β-strands.

However, the antiparallel helical library has just one residue more per fragment respect to the parallel one and can be reparsed to trim out the difference if required for a particular experiment. Here it is reported as it was generated since this library has been used by ARCIMBOLDO_BORGES for solving unknown and test structures. On the contrary, β-sheet libraries were organized from the beginning to share the same size as they were part of a study aiming to compare their population and performance. As evidenced by the reported parameterization, helical folds did not require a wide angular tolerance to extract a variety of models. In fact, 15º of angle difference between the global CV of each helix and its corresponding one in the template were enough to allow differences in the fold, but for β-strands this measure needed to be increased up to 50º before obtaining models differing from the template. This threshold is too ample to physically represent the real spatial deviation between β-strands, it turns out that the global CV direction and length for these fragments may significantly wander from the mean computed over tripeptides and more importantly, away from the direction of the fragment itself. On the other hand, distances were not a particular problem as a difference threshold of 3 Å was large enough to encompass any possible deviation of the fold without losing its overall description. The distribution difference parameter changed from helices, 0.10 Å, to strands, 0.40 Å, to reflect the more frequent backbone variation in β-strands relative to the observed regularity in α-helices. However, libraries were created in different moments while developing a stable version of the program, thus even if the values for the clusters and the number of models cannot be read as absolute indicators (the different prototypes would extract and/or cluster differently), the tendency shown is interesting: antiparallel helices are more frequent than parallel ones as it is for antiparallel vs parallel β-strands. Something not surprising considering that antiparallel conformations for both helices and strands may simply arise through a turn or a kink in the backbone while parallel conformations require structural constraints over a longer portion of the chain. Libraries contain all possible models representing a fold within a given tolerance, but in addition, the extraction procedure occasionally leads to the inclusion of outliers, which should be excluded.

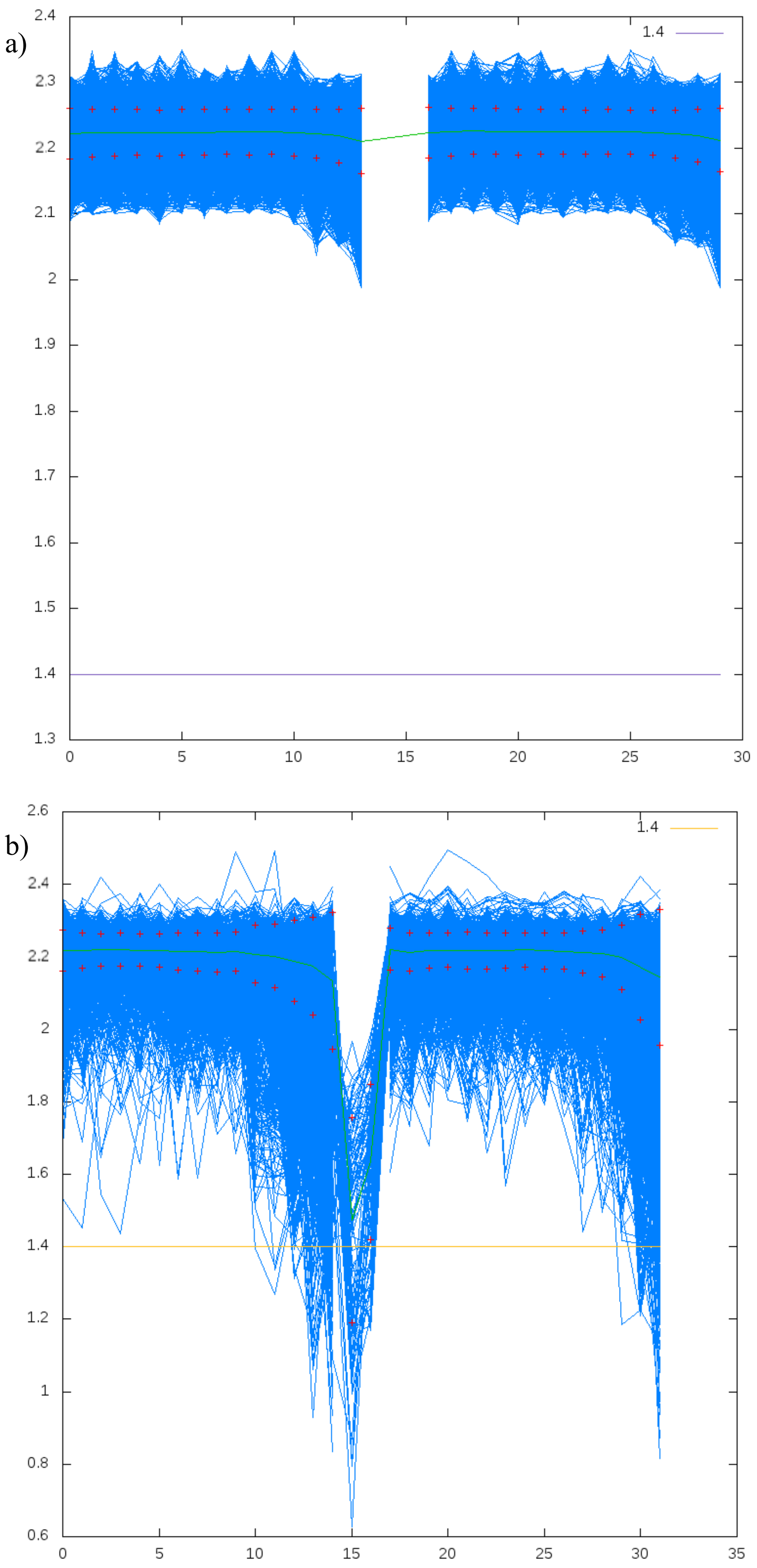


Figure 5-19 CV distributions for helical libraries

a) CV distributions for all models in the parallel library are displayed together with a green line indicating the mean at each point, and its relative standard deviation by red crosses. b) corresponding graph for the antiparallel library.

As an example Figure 5-19 shows the CV distributions of all models for (a) the helical parallel library and (b) for the antiparallel helical library. The parallel library contains only models of two disconnected helices, as expected from the template in Figure 5-18 (a). The green line is the set of means for all positional CVs and the red crosses represent the standard deviation from the mean per each point. The library accumulates more distortion at the edges of helices. Thus some models with terminal residues can slop the distribution, even though no visible outliers are detectable. The second graph (b) refers to the library of antiparallel helices. Visually the graph is clear-cut as the previous one, presenting continuous distributions, indicating there are models with connected helices even though the template is disconnected Figure 5-18 (b).

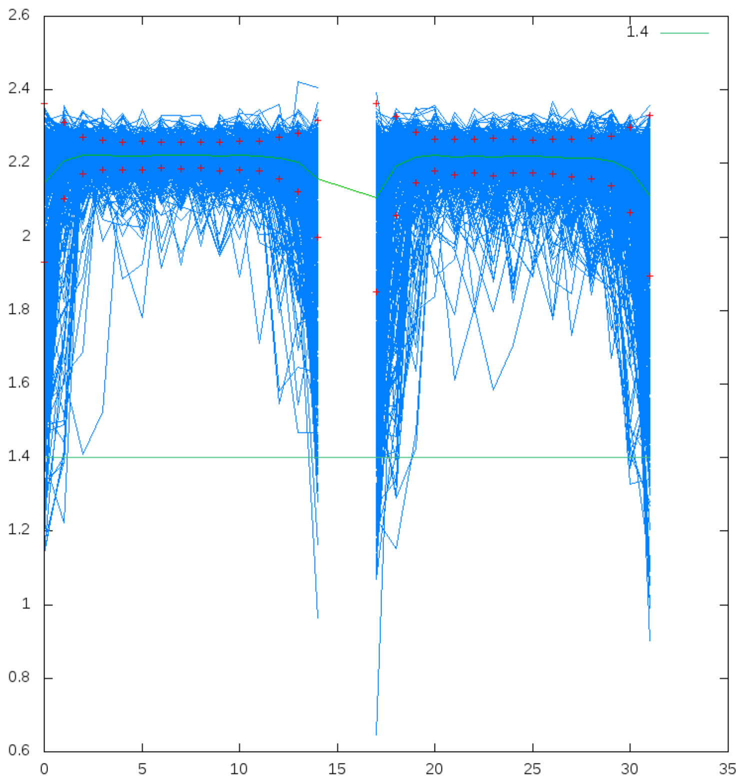


Figure 5-20 Antiparallel helical library processed with BORGES_MATRIX

The distribution of a sample library extracted with the new algorithm BORGES_MATRIX.

In the case displayed in Figure 5-21 (a), a model extracted from the *pdb* 3PZK: the two helices are connected by a small turn that was recognized as helical. The new implementation BORGES_MATRIX would have cut out this turn by analysing the CV distribution instead of relying on a single CV and would only return models matching the template. However, models like this do not harm a phasing procedure, and can be tolerated in a library introducing variability in the fold to derive initial phases. This model, as the others with connected helices, shows a drop in the CVL values around the CV_{15} to increase again as the second helix starts. Other peaks of decreasing CVL are visible at the end of the second helix with the typical pattern of a transition from a defined secondary structure region to a coiled one. Spurious peaks are also present around the CV_3 near the beginning of the first helix, with a pattern that usually indicates main chain curvature. For example, a model extracted from the *pdb* 2W5J, has curved helices in particular one in which a secondary structure prediction algorithm such as DSSP does not reveal helical content as

seen in Figure 5-21 (b). This is recognizable by its torsion angles pattern in the Ramachandran plot. The decision to regard such models as outliers or not again depends on the purpose of the library. For phasing, including slightly curved models may be the difference between solving a structure or not, as the unknown structure may well present this geometry, but for more speculative bioinformatics experiments it is essential to control the inclusion of such models and explicitly require the program to extract them or not. The new implementation in BORGES_MATRIX has taken this aspect into account and the parameterization is now able to filter out or include such models in the library. To confirm our supposition, a sample of the antiparallel library was recomputed with the new algorithm, Figure 5-20 imposing 80% of similarity for *continuous* pairs and 75% for *jump* pairs, the library has been extracted from just one third of the whole PDB database and produced more than 435,000 models clustered in 1,312 groups. This sample library, as the template, does not contain connected helices, and accumulates deviations in the edges of the helix when the threshold of 75% similarity is considered. Stringent, higher, thresholds would narrow the selection of the models while lower threshold would increase the variability of the sample. The last interesting visually outlier in the graph is the presence of some spurious peaks in which a CVL reaches a value exceeding 2.4 Å. Such values of the CV should be forbidden in the peptide stereochemistry. Indeed, the models from which this rare CVL is obtained can be exemplified by *pdb* entry 3A5D (Figure 5-21 (c). This is a huge, 4.8 Å structure presenting 330 Ramachandran outliers, and the ones belonging to the extracted model are coloured in red. Models containing improbable torsion angle conformation should be discarded in the generation of the library, for this reason BORGES_MATRIX also analyses the absolute values of the CVLs to be physical possible, spotting outliers. The Graph in Figure 5-20 reflects also this situation reducing visual outliers in the CVL. There is only one value that is very near to 2.4 Å (residue 198, chain A of the *pdb* 3I0K) and it is found in the extremity of the first helix corresponding to an Arginine that is also an outlier in the Ramachandran, telling us that the critical threshold for the highest CVL can be refined to be even more stringent. CVs provide an overall indicator for wrong torsions although, in order to validate main chain torsion angles originated by non-standard distortions, complementary information should be brought in.
Therefore, even if libraries computed with the prototype, and in general BORGES libraries, may contain a few spurious errors, outliers or unwanted models they are appropriate for phasing. Indeed the described library were used to phase successfully unknown and test structures and their results are discussed in Chapter 6.

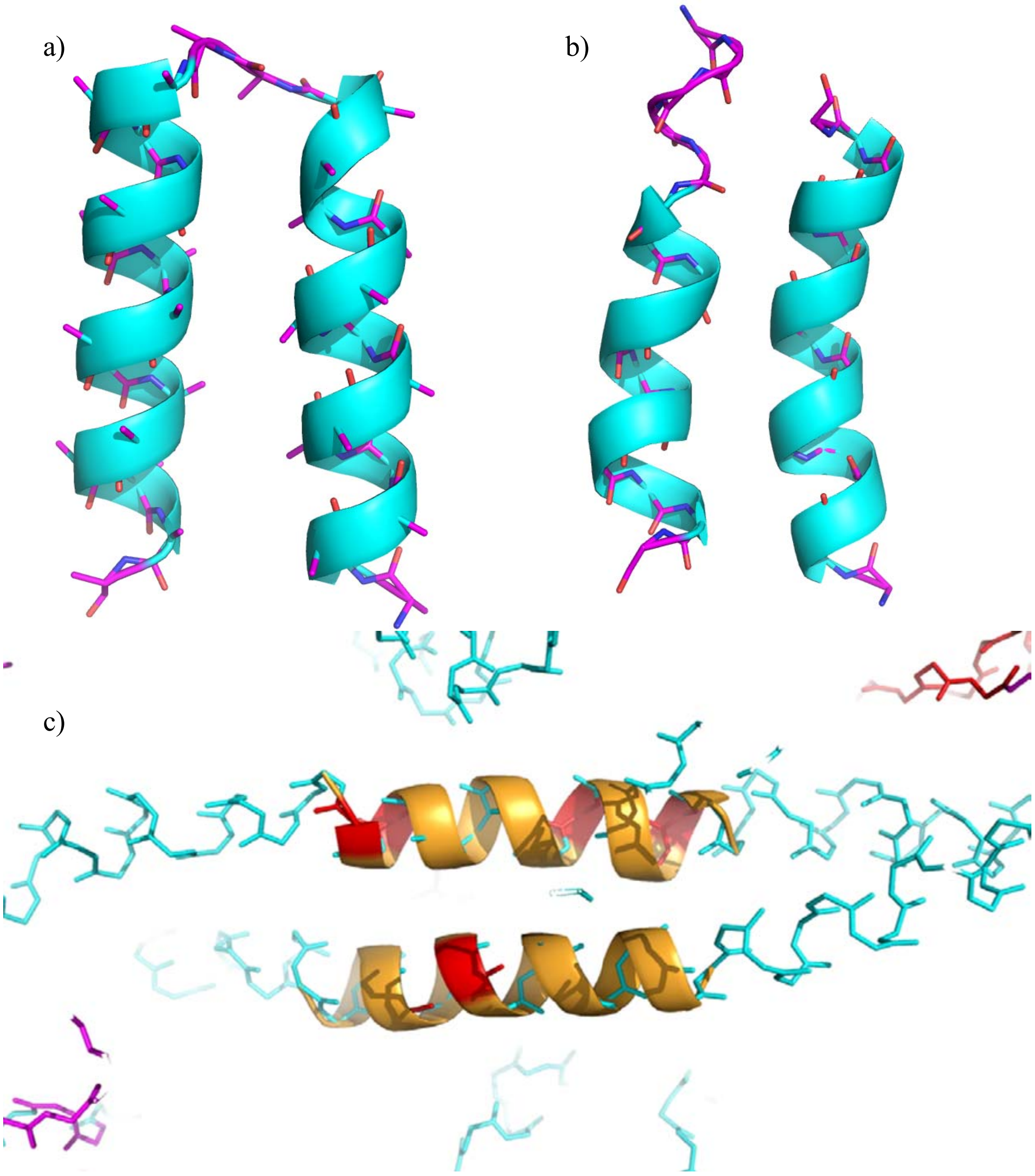


Figure 5-21 Outliers in the antiparallel helical library

Three examples of outliers identified in the helical antiparallel library. In a) non discontinuous model extracted from 3PZK in spite of the original template being composed of two fragments. In b) model from 2W5J presenting in regions a curvature that breaks the secondary structure element. In c) model from 3A5D, containing improbable torsion angles

5.11 Libraries of knowledge-based folds: DNA-binding motifs and their BORGES library version.

DNA-binding proteins play essential roles in all aspects of transcription, DNA repair and gene regulation, and therefore it is no surprise that 6–7% of all proteins expressed in eukaryotic genomes have been estimated to interact with DNA[154]. A number of co-crystal structures showed early on that nature has evolved to use a limited set of structural domains for DNA recognition, and accordingly DNA-binding proteins have been classified into eight major groups based on their structure and function. Although the number and diversity of DNA-binding structures solved in the last decade has greatly increased, most proteins still fall into one of these groups, which include the helix–turn–helix (HTH), zinc-coordinating, zipper-type, other α-helical and β-type proteins. Crystallization of DNA binding structure is complicated by the fact that frequently many synthetic DNA oligonucleotides differing in length and/or sequence are tested. Crystals tend to be more fragile and radiation-sensitive owing to the increased absorption of heavier atoms. Diffraction patterns are often anisotropic owing to base stacking and the formation of semi- continuous DNA helices throughout the crystal, and the resolution is generally limited. As an example, there are only five protein–DNA complexes determined at resolutions of 1.2 Å or better. Specific methods accounting for the particularities of DNA and its complexes may be more appropriate than the ones devised for proteins. For phasing, we explored taking advantage of the specific patterns of DNA-binding proteins to generate databases[155] of conserved structural motifs and domains to be used within our ARCIMBOLDO frame. One of the most structurally conserved domains is the zinc-coordinating group (also designated zinc-fingers) that is typically found in eukaryotic transcription factors. Other candidates are the helix–turn–helix group, which is found in many bacterial regulators (including the winged-helix motif;[156]), and zipper-type proteins. On the contrary, the family of β-type DNA-binding proteins shows too much structural variability to provide useful search fragments in ARCIMBOLDO. TATA-box binding proteins, on the other hand, are similar enough to be used in MR approaches[157]. Libraries of the above cited motifs were manually curated to conduct experiments establishing an appropriate strategy for phasing DNA-binding proteins with ARCIMBOLDO[155]. Zinc-coordinating and zipper-type target structures were solved successfully using protein–DNA specific fragment subsets within ARCIMBOLDO. In the case of the zipper-type complex the long helices already constitute efficient search fragments, an ideal regular helix being close enough to the more tightly wound zipper helix. In this case, a fragment library is clearly unnecessary. On the contrary, in the case of the zinc-finger motif the isolated secondary-structure motifs were not effective while the binding-motifs library was. The method is dependent on sufficiently high-resolution diffraction data, with the limit appearing to be around 2.0 Å. PHASER is generally successful in positioning fragments. Ways to enhance the efficiency of the procedure in the future are suggested by the more accurate models being distinguished by higher FOMs in PHASER, which opens the door to model refinement or library extension.

The family of zinc-coordination binding motifs constitutes the largest single group of transcription factors in eukaryotic genomes. They typically present a structurally conserved characteristic zinc environment (Figure 5-22) in which one or two Zn cations are coordinated by Cysteine and Histidine residues in a tetrahedral geometry. We can benefit from this common geometry of a small part of our target structure, as it can be predicted from the sequence. BORGES_MATRIX was employed to test the extraction of this motif, which unlike previous templates this one represents continuous fragments in which not only the secondary structure stretches but also the connecting loops are included as they have a functional role. This library was used to solve a test structure Klf4, a zinc-finger transcription factor indispensable for terminal maturation of epithelial tissues, and it is also involved in the generation of pluripotent embryonic stem cells from differentiated tissues. This structure is deposited in the Protein Data Bank under the *pdb* code 2WBS[158]

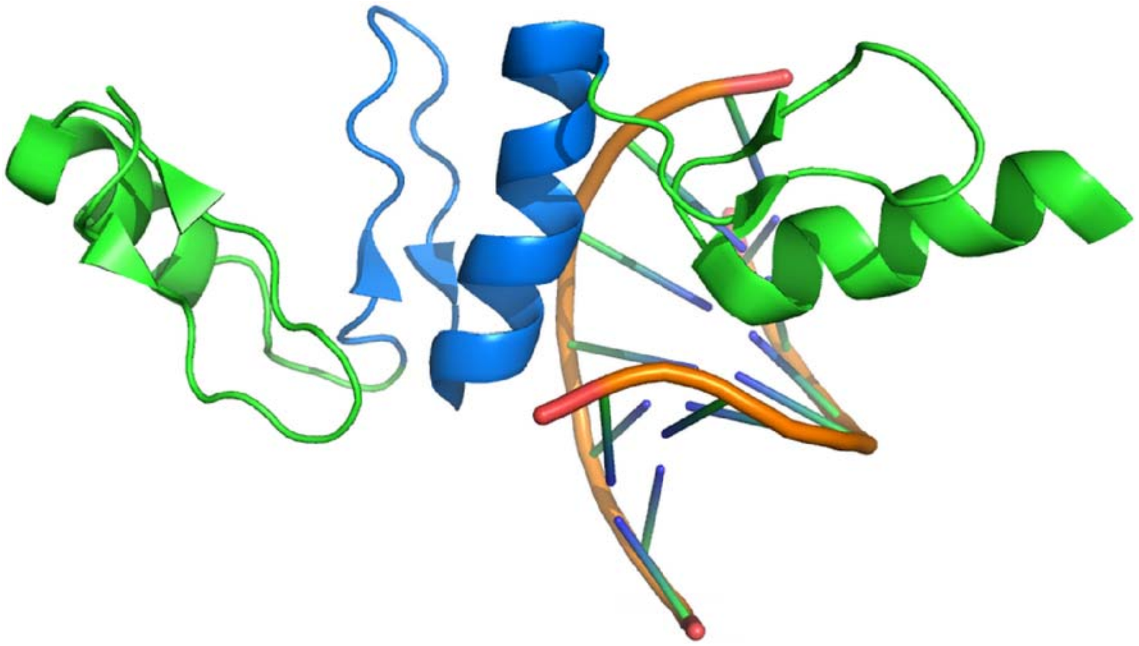


Figure 5-22 Cartoon representation of the Klf4 structure with bound DNA

The Zinc-finger motif (blue) appears three times in the structure. The motif is characterized by a short helix and two short β-stranded segments connected by a loop.

The structure, in space group $P2_12_12_1$, contains a seven base-pair double-stranded DNA helix surrounded by three connected zinc-finger fragments totalling 87 amino acids.
The selected template was the zinc-finger structure from *pdb* 1UBD[159]. This entry corresponds to the human YY1 Zinc-finger domain bound to the Adeno-associated virus P5 initiator element. A classical ARCIMBOLDO approach using a single helix as search model failed to produce the correct solution, presumably because of the short helix. Instead, exploiting the described library of superposed models to take into account the local variability, succeeded in phasing the structure.

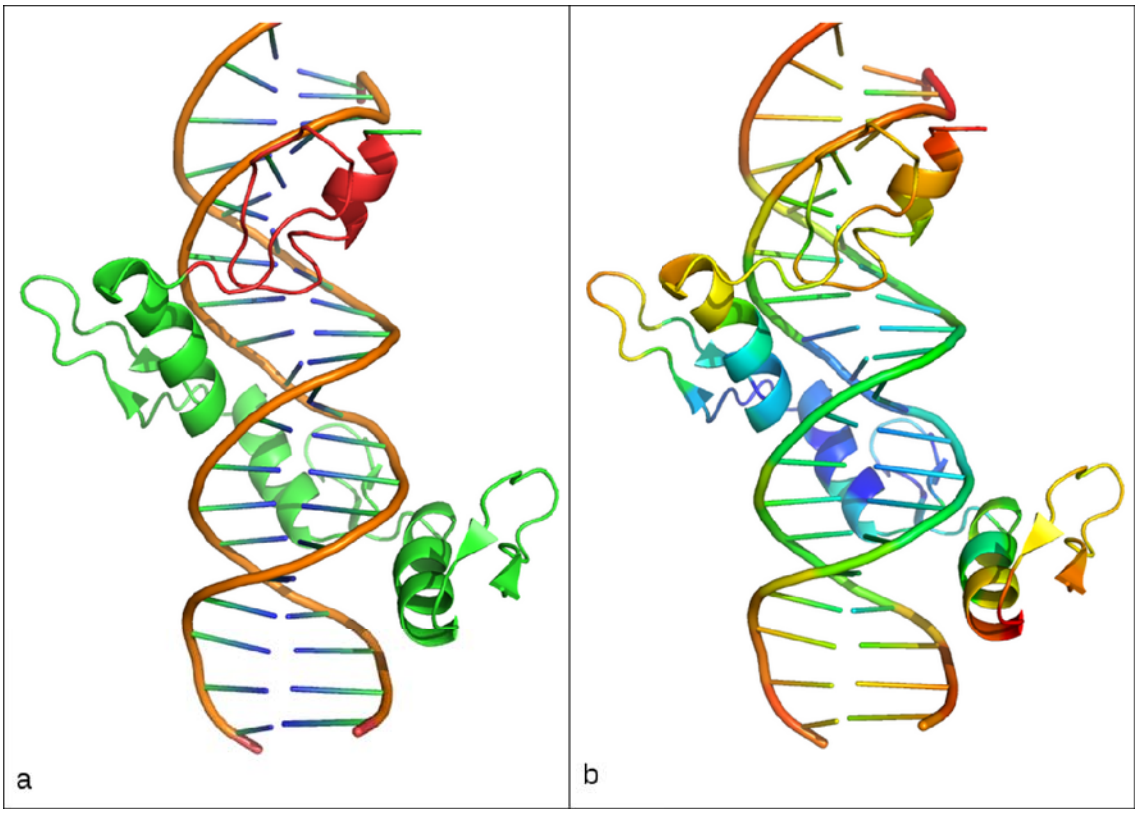


Figure 5-23 Cartoon representation of the 1UBD deposited structure that contains the human YY1 Zinc-finger domain bound to the DNA

a) The portion used as search model to create the library with BORGES_MATRIX is highlighted in red; b) rainbow colour representation of the deposited structure, colours ranging from red to blue with red corresponding to high B-factor values and blue for low values.

As shown in Figure 5-23, the selected template (a) (C381-408) is not necessarily the most rigid copy (b), judging from the relatively high B-factors. Moreover, the RMSD of the search model vs. the final structure exceeds 0.5 Å, which is what is usually required to phase from such a small model, justifying the creation of a library of models. The template model is 27 aa long, the fold observed in the manually curated database is characterized by a short helix of 14 aa, two very short antiparallel strands of variable length and a specific loop region that is also conserved in the fold but can somewhat vary in size.
BORGES_MATRIX recognizes the template as a single fragment since there is no break in the chain. The less conserved region in the fold is the mobile loop varying in size from 3 to 4 residues. BORGES_MATRIX is instructed about the loop range by a new parameter (-y) giving the initial and last residue in the range (C386-390). In these particular regions of the fold, thresholds are increased to reflect the variability of the coil, while all other residues would need to match the given parameters for *continuous* and *jump* pairs (-C75 ; -J75). The fold search was set against a subset of 31 zinc-finger structures, extracting at least one fold per each structure. The so obtained library of 115 models was not clustered but models were superposed against the template and B-factors set to a common value. The library was used for solving the test case of 2WBS producing a translation solution with acceptable packing, characterized by 7.08 ZSCORE and LLG of 46.10 that lead to a traced solution of CC 25.24%.
In conclusion, the experiment proved BORGES_MATRIX could extract continuous fragments and create libraries of knowledge-based fold for phasing.

6 ENFORCING UNSPECIFIC TERTIARY STRUCTURE FOR PHASING: THE ARCIMBOLDO_BORGES METHOD

The experience with sequential search of single helices in ARCIMBOLDO proved that *ab initio* methods can be extended to macromolecular structures when the atomicity constraint is substituted with ubiquitous secondary structure elements. The approach has been successfully used as described in previous Chapters and has a large applicability as 80% structures in the PDB contain helices. Even though in principle any secondary structure element could be used as a search fragment in ARCIMBOLDO, helices are most favourable as they are constant, rigid and periodic. The presence of long helices in the structure may require a better approximation than model helices. Increasing the length of the helix, the geometrical distortion is accumulated and might become predominant, needing a closer model from which to generate initial phases. The problem could be solved by searching multiple copies of an ideal smaller helix to map the long one, but a more appropriate approach could be the use of alternative, rather than single model helices. Other issues with helices are coiled coil structures where bundles of long helices are coiled tightly in extended chains. Our experience with ARCIMBOLDO and such folds shows a tendency to overlap the second helix on top of the first, producing deceptively high FOMs for solutions that are incompatible with packing. For example, we have observed a tendency to high ZSCOREs and LLGs in space groups such as C2 when small helices are placed on the twofold axis. Finally, although helices are abundantly represented in the PDB database, there are families, domains and folds, which are exclusively all-β structures. To address all these different scenarios a new program has been developed: ARCIMBOLDO_BORGES that enforces unspecific tertiary structure for phasing by the use of libraries of superposed models. These libraries represent geometrical differences, hence degrees of freedom internal to the local fold, increasing the number of searches in a particular space of structural conformation. Moreover, local folds fix spatial relationships among secondary structure elements imposing stronger constraints favouring allowed packing within the unit cell. For example, the use in MR of two antiparallel or parallel contiguous helices instead of single helices will avoid to place them superimposed and thus could generate a correct solution from which to expand the final structure.

We have seen in the previous Chapter 5 how to define, extract and cluster libraries of local folds, we are going to explore in this Chapter how employ them for phasing with the program ARCIMBOLDO_BORGES.

6.1 ARCIMBOLDO_BORGES on a single workstation

Although a general target case for ARCIMBOLDO_BORGES requires distributing computation over a grid network of computers or parallel processing in a supercomputer, a single-workstation implementation is also available. The same deployed binary can work in different environments and by default it is set to work on a single machine through parallel multiprocessing. The main differences lay in the automatic filters and selection stages that are activated to render the computation feasible in each context.

The following diagram describes the program workflow:

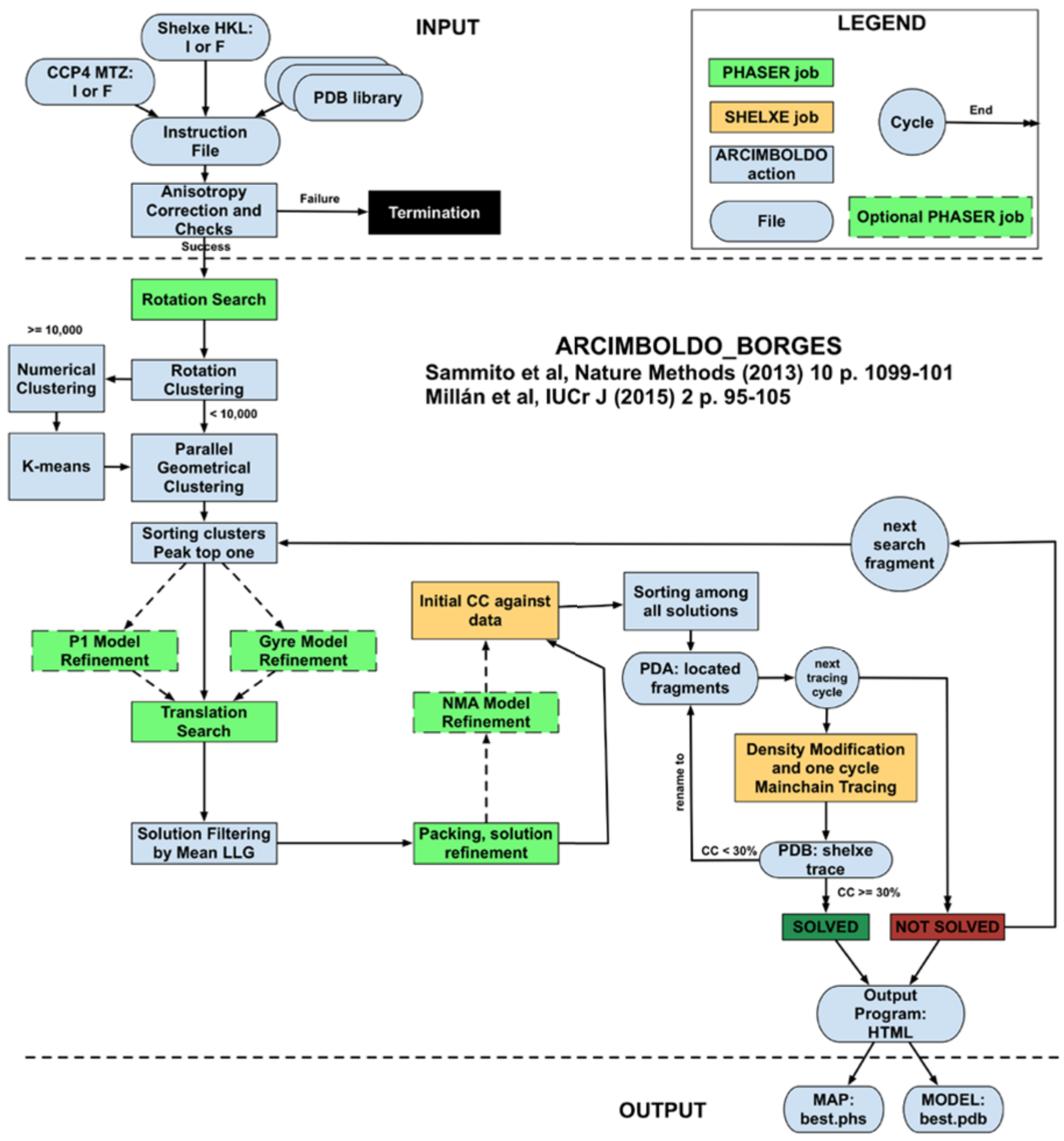


Figure 6-1 ARCIMBOLDO_BORGES workflow

ARCIMBOLDO_BORGES also checks input parameterization and runs diagnostics at the initial stage to spot hardware or software incompatibilities. Details for this step can be found in Section 3.1.1. The program uses all models in a pre-calculated library to evaluate the outcome of a PHASER rotation function. All resulting orientations are clustered. Each cluster represents a rotation peak and contains the set of models producing it. Clusters are treated independently, evaluating translations for all peaks and models and expanding with SHELXE as many selected solutions as the number of allocated cores. As the method is computationally demanding, the supercomputing settings are described as the norm.

6.1.1 Rotation search of the library models

A rotation search is performed for *n* randomly sampled models in the library, where *n* is the number of allocated cores at the supercomputer. Each rotation search is computed with PHASER, and the resulting top 75% rotation peaks are written into the *rlist* file. Their

FOMs (LLG and ZSCORE) are read from the PHASER output. By default, rotations rendering negative LLG are excluded.

6.1.2 Rotation clustering of a sampled group

Rotation equivalence is tested with the same algorithms described in Section 3.1 for ARCIMBOLDO_LITE, including symmetry equivalent orientations and rotational NCS if specified by the user. Clustering is performed in a slightly different way. Rotation clusters are first sampled on a reduced number of models that is proportional to the number of workstation cores; then completion of rotation search and clustering for all remaining models is performed.

6.1.3 Rotation search of the remaining library models

Rotation searches for the remaining models run in parallel and every time 100 of them are completed, a new clustering procedure is launched as an external ARCIMBOLDO_BORGES process to compare results against the originally sampled group. Geometrical clustering of rotations is performed with the same algorithms used for the initially sampled group, except that new solutions can be associated to an existing cluster or remain unclustered. After all rotation searches are completed a thread of the main process will expand the originally sampled clusters with all rotations assigned, whereas a new clustering step will be performed to group all unassigned rotations into new clusters. Finally, a last step is performed in which all rotation clusters are compared through their references, and if required, merged to remove redundancy.

These operations render a sequence of rotations grouped and ordered by LLG. Clusters are filtered to contain for each model only the top LLG rotation. As solutions are developed, cluster ID is maintained. Results are written to a formatted plain text file so that the program may them to resume an interrupted run, skipping previously computed steps. If no rotations are available, for example if all the rotations are associated to negative LLG, the program will stop and print a warning to the standard output.

From this point on, ARCIMBOLDO_BORGES pursues each rotation cluster sequentially and independently from the others. The order in which clusters are evaluated depends on their population[160]. Clusters gathering more rotations are performed first. When the program is run to parallelize jobs in a single workstation the list of clusters to evaluate is limited to the top four. The following steps always refer to a single rotation cluster.

6.1.4 Optional: Patterson correlation refinement

In case the library fold is composed only by helices and data extended to triclinic hemisphere are provided in *mtz* format, ARCIMBOLDO_BORGES computes at this point a P1 refinement with PHASER giving the model internal degrees of freedom. It does so by splitting each helix in the fold into a separate file to be treated by PHASER as an ensemble and locally refining the Euler angles of the helices against the data in P1[161]. The new coordinates of the merged ensembles composed by the separated helices, are saved and the RMSD between the starting model and the refined model is also computed and registered in the output files. The refined model is then used as input to perform a new rotation search. FOMs are saved, and, by default, this refinement operation is performed up to three times.

6.1.5 Optional: Gyre refinement of each helix in the fold

Recently PHASER incorporated a new mode named GYRE, to refine individually all the ensembles of a PHASER job around a pre-computed rotation and against the maximum likelihood rotation function. This mode can be activated instead of P1 Refinement.

6.1.6 Translation search

The PHASER Maximum Likelihood Translation Function is calculated for all rotations and models in the cluster. Translations are internally sorted by their LLG, the top peak being defined as 100% while solutions with an LLG on the mean value are defined as 0%. All translations above the 75% cut off are saved in *sol* files, which are parsed to associate fractional coordinates to their LLG and ZSCORE. Each rotation can lead to different translation solutions, but only the top LLG solution is saved.
The program will skip the translation search for the first fragment if the space group is triclinic P1.

At this stage, PHASER can determine the presence of translational NCS and simultaneously place the related copies. ARCIMBOLDO_BORGES will recognize the tied solutions and use them together in all following steps. The option can be inactivated and in the case of coiled coils, accounting for the translation through the model may be more successful than searching for individual helices.

6.1.7 Packing filter

Solutions should be physically possible and consistent with the crystallographic symmetry restrictions imposed by the space group. To discard pointless substructures, the equivalent positions are generated and clashing atoms, defined as atoms at distances below 3.0 Å, are identified and counted by PHASER. ARCIMBOLDO_BORGES default settings do not allow any clashes between atoms. This assumption is reasonable if we consider that search folds are composed by short secondary structure elements that are unlikely to overlap in correct solutions. If no solution survived packing, the program will give up on the cluster and proceed to evaluating the next one.

6.1.8 Solution prioritization

Solutions are sorted by their full resolution initial CC as calculated with SHELXE. The CC is influenced by the fragment size with respect to the total asymmetric unit contents. All models in a library have practically the same number of atoms. This implies that while it cannot be interpreted as an absolute value, the relative differences among sorted solutions are indicative. By default ARCIMBOLDO_BORGES uses *pdb* optimization[94] in SHELXE (-o) to find the largest subset of amino acids that maximizes the INITCC, but the user may inactive this option and retain all atoms in the model. The initial CC is computed for each solution before and after applying solution refinement with PHASER. Optionally it is also possible to execute normal mode analysis for each solution in PHASER[114]. The resulting models are then tested with initial CC against data in SHELXE and only the top model for each solution is selected. The highest FOM model among the two cases, or three if the optional normal mode analysis is included, is further considered. Prioritized solutions will include the top CC and at least the five top LLG solutions even if their INITCC value was poor.

6.1.9 Solution extension

The single workstation version for ARCIMBOLDO_BORGES limits the number of SHELXE extension jobs to the available physical machine cores. The user can force an increase in the number of threads but the program will not check for the stability and performance of the machine. An increase in the number of solutions to be sequentially expanded may also be selected.

SHELXE uses the starting model and the crystallographic data to calculate phases and perform density modification and main chain autotracing in the resulting electron density map. The program also takes an input line with parameters, whose choice is essential for the success of the method. Default parameters in ARCIMBOLDO are not necessarily SHELXE defaults but tied to the resolution of the data and the type of search model employed:

As was seen in the ARCIMBOLDO_LITE context, for data at ultra-high resolution, equal or better than 1.0 Å, map sharpening and low density elimination[97] is most effective in gradually improving the phases. A large number of density modification cycles is needed to reach convergence. Furthermore, indicating a low solvent content percentage is consistent with high resolution of the data. As resolution gets lower, decreasing the number of density modification cycles, lowering the density sharpening parameter and increasing the solvent content is appropriate. The number of autrotracing cycles, wherein SHELXE builds polypeptide main chain from the electron density map, is by default set to 8. Each cycle is processed as an independent SHELXE job, which takes as input the trace of the previous cycle. This allows monitoring each cycle independently and stopping the program once a solution is found. If the user selects model optimization in SHELXE, it will be performed after each autotracing cycle instead of just once, as would happen in a standard SHELXE run.

Resolution range (Å)	**SHELXE line**
<= 1.0	-m100 –a10 -s0.20 -v0.5 -u2999 -t10 -o -y[*res*]
]1.0,1.3]	-m50 –a10 -s0.30 -v0.25 -u2999 -t10 -o -y[*res*]
]1.3,1.5]	-m25 –a10 -s0.40 -v0.1 -u2999 -t10 -o -y[*res*]
]1.5,2.0]	-m10 –a10 -s0.45 -v0 -u2999 -t10 -o -y[*res*]
> 2.0	-m5 –a10 -s0.55 -v0 -u2999 -t10 -o -y[*res*]
Legend	
-m	Density modification cycles
-a	Main chain autotracing
-s	Solvent content fraction
-v	Low density elimination factor
-u	Allocated memory in MB for fragment optimization
-t	Time factor for helices and peptide searches
-q	Include helical fragments as seeds for tracing
-y	Highest resolution for calculated phases from the input coordinate model

Table 6-1 SHELXE line parameterization in ARCIMBOLDO_BORGES

Default parameters for SHELXE according to the resolution of the data.

In order to avoid model bias, SHELXE will use the input model to calculate initial phases and discard it in favour of the new trace after each autotracing cycle. Therefore, side chains or non-polypeptide models will be lost after the first cycle.

6.1.10 Next cycle iteration and completion

ARCIMBOLDO_BORGES evaluates by default the four most populated clusters. The user can override this number through the *bor* file. Cluster prioritization can alternatively be set to filter rotation clusters by LLG above the mean. All selected clusters, ordered by their top LLG, will be pursued from translation search to solution refinement. Once all clusters are evaluated, a second cluster sorting is performed, this time using the top LLG at the solution refinement step. This new order is used to score again each cluster, prioritize internally its solutions and proceed on to the computationally demanding SHELXE expansion.

ARCIMBOLDO_BORGES inspects traced solutions and if a CC above 30% is reached, the structure should be solved and the program stops. Messages are written to the standard output and links to the solution files will be displayed in the html and xml output files.

6.2 Testing ARCIMBOLDO_BORGES

ARCIMBOLDO_BORGES was tested on the same pool of structures employed for ARCIMBOLDO_LITE[4] introduced in Section 3.2. From the pool of 294 structures 151 remained to be solved with an alternative method, as 143 were already solved with ARCIMBOLDO_LITE. 147 out of 294 structures contain β-strands as the only, or main, secondary structure fragments. In addition, 20 structures featured four or more disulphide bridges anchoring a peculiar fold, as is typical in small toxin inhibitors. Such cases are outside the scope of a secondary structure fragment ARCIMBOLDO_LITE search as single strands do not constitute appropriate search fragments, but can be solved with libraries of small folds with ARCIMBOLDO_BORGES. Table 6-2 summarizes the results grouping structures by resolution limit and library fold type.

Resolution (Å) Ranges	**Structures**	**Candidate structures**	**Solved with ARCIMBOLDO_BORGES**	**Success rate (%)**
Total	294	151	38	25
1 - 0.54	24	10	5	50
1.3 - 1.0	43	18	8	44
1.6 - 1.3	78	40	12	30
2.2 - 1.6	149	83	13	16
Fragments		Structures	Solved with ARCIMBOLDO_BORGES	
Helices ud		7	2	
Curved helices		1	1	
Disulphide bridges		20	1	
β-uuu		13	1	
β-uud		24	2	
β-udu		131	31	

Table 6-2 ARCIMBOLDO_BORGES test solutions by resolution range

In particular, 6 precomputed libraries containing geometrically clustered variations of a particular fold were used. Two of them contain pairs of contiguous helices, parallel and antiparallel respectively. They are especially indicated for coiled-coils, as single helices

tend to be placed in the same position in single fragment searches. Three libraries correspond to 20 amino acids arranged in three stranded β-sheets of antiparallel, parallel and parallel-antiparallel disposition. These libraries of models, introduced in Section 5.10, superimposed to match the generating template have been computed with the first prototype of BORGES[7], described in Section 5.5, and are available for download from our website (http://chango.ibmb.csic.es/download), along with libraries of 24 amino acids arranged in four strands with different relative directions.
Beside these libraries a new one was generated with BORGES_MATRIX, containing a sample of disulphide bridges linking two tetra peptides in all possible conformations.
Although ARCIMBOLDO_BORGES is better suited for large computing resources, the new implementation can run either connecting to a local or remote grid or on a single multicore machine. As for ARCIMBOLDO_LITE, automatic parameterization is related to the available hardware, and only four rotation clusters will be sequentially evaluated by default when the program is run on a single machine. The test structures in this pool are small enough to be computed on the same 8-core workstations used for the ARCIMBOLDO_LITE tests. Typical run time was 2-5 days. The most successful library was the three antiparallel β-strands (udu). This arrangement is more frequent than the alternative ones. 31 out of 131 of the cases containing this fold were solved, not all were tested for lack of computing time. In particular, they were skipped if already solved by ARCIMBOLDO_LITE with helices. The antiparallel library was used first and only structures where this fold would not work were further considered for phasing with parallel-antiparallel, solving 2 structures and parallel libraries, which solved 1 case. Moreover, 2 test cases were solved with the libraries of helices and 2 structures with the disulphide linked peptides. Finally the 90 amino acid, 1.5 Å structure 3L32 in $I4_122$, was solved with a library of 162 models of single helices of 18 aa, displaying different degrees of curvature. This result confirms that the straight model helix in ARCIMBOLDO may not be the optimal search fragment for helical structures containing markedly curved or otherwise distorted helices.
The overall success rate of the program is of 25% of the identified cases. As seen for the ARCIMBOLDO_LITE test, high resolution cases yield a higher success rate: 50% for atomic resolution structures, decreasing to 34% for data ranging from 1.6 to 1.3Å and 16% below 1.6Å. Optimal parameterization has been found to be different for structures mainly composed of β-strands and is accordingly set as default for ARCIMBOLDO_BORGES. The Table 6-1 summarises the current SHELXE parameterization. In particular, if the program is launched with a non-helical library, the default expansion stage will not search for helices (eliminating flag -q), use lower solvent content than in helical cases (0.5 lower for each resolution range), and halving the number of density modification cycles. It will also keep the original fragment for half of the autotracing iterations. Another difference is the use of the *pdb* optimization within SHELXE (-o) that in case of libraries is switched on by default. We have observed that fragment search functions are frequently able to correctly place a model in spite of incorrect features, whereas density modification is more sensitive to model accuracy and may stall when starting from partially incorrect models. This situation is illustrated in Figure 6-2 (c) where the solution of 1V70 is shown together with the model from the library, in blue, from which starting phases were generated. Residues trimmed out by the optimization process, which are clearly absent in the deposited structure are shown in red.

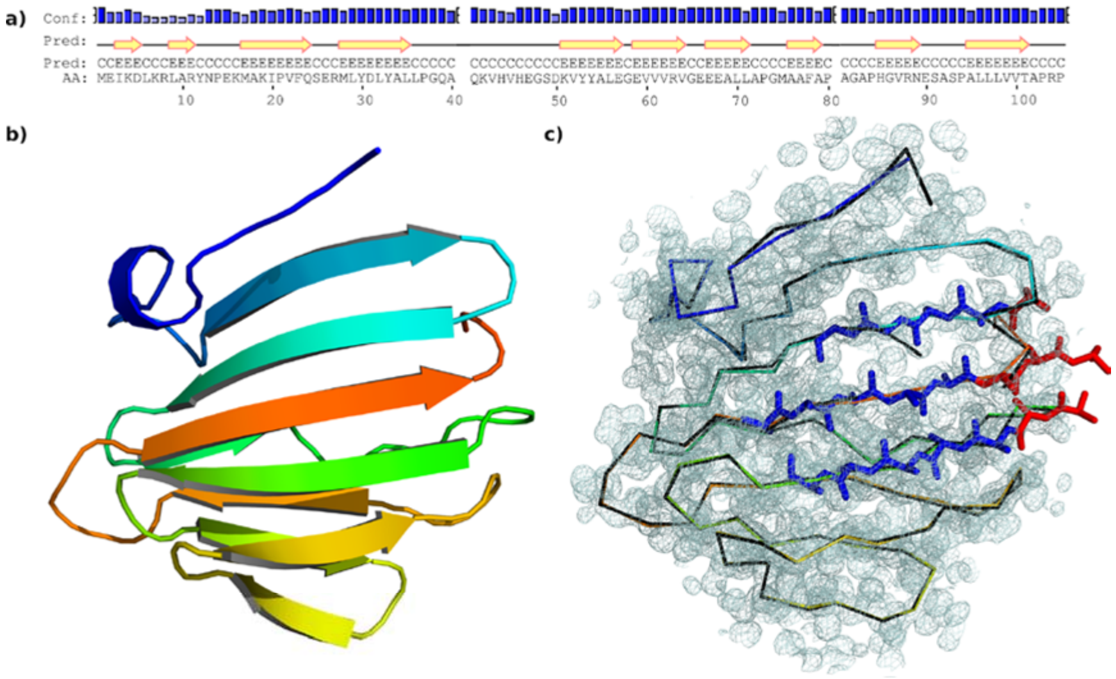

a)
Conf:
Pred:
Pred: CCEEECCCEEECCCCCEEEEEEEECCCEEEEEEEECCCCC CCCCCCCCCCEEEEEEECEEEEEECCEEEEECCCCEEEEC CCCCEEEEECCCCCEEEEEEEECCCC
AA: MEIKDLKRLARYNPEKMAKIPVFQSERMLYDLYALLPGQA QKVHVHEGSDKVYYALEGEVVVRVGEEEALLAPGMAAFAP AGAPHGVRNESASPALLVVTAPRP
10 20 30 40 50 60 70 80 90 100
b)
c)

Figure 6-2 ARCIMBOLDO_BORGES solution of 1V70

a) Secondary structure prediction. b) Structure in rainbow colored cartoon representation, c) SHELXE electron density map of a solution, the trace is shown as coil and the placed antiparallel three-stranded fragment as sticks, with residues eliminated during *pdb* optimization displayed in red.

As in ARCIMBOLDO_LITE, a line in the *bor* file setting parameterization for SHELXE will override any of the inbuilt defaults, but leaving this line unset will apply the resolution- and hardware-tailored values described. Finally, it is important to remark that the outcome naturally depends on the hardware environment in which the program is executed, not only because of the limited number of clusters, prioritized solutions that are automatically scaled to fit the computational resources but also because of memory optimization in SHELXE. As an example, the 105 amino acids structure 2HAZ at 1.7Å was solved automatically on a double Xeon workstation, with similar features to the ones used for benchmarking performance in this study, but endowed with 3GB rather than 2GB RAM per core. This case has not been counted as solved in our table of results.

6.3 Distributed computing and supercomputing generalization

Structure phasing with libraries of folds is a heavy task for a single workstation. Libraries typically contain thousands of models, each producing hundreds of solutions. This can be already enough to justify the need for a powerful system able to spread tokenized jobs over multiple processors. But difficult cases may require more sophisticated searches. Supercomputing facilities are indicated for such jobs. From the developer's point of view, a supercomputer is a collection of shared cores and memory in which job parallelization can be performed by Message Passing Interface (MPI) calls. MPI calls are optimized for parallelization of internal code or functions but our software requires process parallelization, which is more natural in a computer network grid environment. Solutions to this problem should be tailored to the resources provided by the supercomputer in which the program is working.
Our software ports process parallelization on supercomputers, in two steps: allocate a fixed number of cores for an established time period, and internally access the machine nodes where they are located to start new jobs in parallel.

The first step is performed by creating a torque script job and submitting it with the command *qsub*.

An example of a Torque script for an ARCIMBOLDO_BORGES job is given:

```
1.  #! /bin/bash
2.   # Template for TORQUE array job for ARCIMBOLDO-BORGES and
ARCIMBOLDO
3.  #PBS -N namejob
4.  #PBS -q normal
5.  #PBS -r y
6.  #PBS -o /dev/null
7.  #PBS -e /dev/null
8.  #PBS -d /working_directory
9.  #PBS -l nodes=4:ppn=16,walltime=15:00:00
10.
11. cat $PBS_NODEFILE > nodes.txt
12. ARCIMBOLDO_BORGES job.bor > outlog.txt 2> err.txt
```

- Line 1 is used by the bash shell to recognize the file as bash script.
- Lines 3-9 are parsed by the *qsub* command from Torque and interpreted as internal parameters:
- Line 3: sets the name of the Torque job, which will be visible in the queue.
- Line 4: selects the queue name, if not default. In Gordon *normal* queue has a maximum running time of 48h and gives access to a maximum of 64 cores. User occupation of nodes is exclusive.
- Line 5: declares the job as rerunnable. ARCIMBOLDO and ARCIMBOLDO_BORGES can always restart from an interrupted run by parsing intermediate autogenerated steps.
- Lines 6-7: discard standard output and standard error of the *qsub* command.
- Line 8: defines a working directory from which all following paths are starting, and where the output is saved.
- Line 9: requires 4 nodes and 16 cores per each, thus allocating the maximum allowed number of 64 cores, for a maximum of 15h of running time.
- Line 11: redirects to a text file, the value of *$PBS_NODEFILE*, which contains the list of allocated cores and their node hostnames.
- Line 13: starts the ARCIMBOLDO_BORGES binary, inputs the instruction file and redirects output and error in two different text files.

The section *CONNECTION* of the configuration file must contain the keyword *nodefile_path* pointing to the file generated in line 11. Once the binary is started by the scheduler, it parses the nodefile with the allocated cores. It also starts a separate thread to manage the queue, maintaining a list of jobs associated to a core and a path to an output text file from which to determine job completion. To launch a new process an *ssh connection* to the machine of the assigned core is established, the process is detached from the *ssh session* with *nohup* and the remote connection is closed. One remote connection per node is created for a process to set up all jobs associated to the cores available in that node. As a job finishes, the freed slot is occupied by the next job in the node process.

The procedure for starting short and long jobs varies. SHELXE expansions need more computing running time (minutes-hours vs seconds-minutes) and are individually launched as requested, whereas fast jobs are sent in packets. The running time of slower jobs compensates the overhead for remote connection. Multiple PHASER jobs generated by the same step are prepared and collected to be distributed to the nodes in packs, in such way all cores are fully occupied at once. If the number of available cores is lower than the number of jobs to perform, each core will sequentially launch more than one PHASER job.

In contrast to our standard grid usage, where all jobs are simultaneously queued, the procedure has to iterate over each allocated node establishing a remote connection, instructing the *shell* to start the associated jobs, detaching jobs from the remote session and disconnecting. However, ARCIMBOLDO_BORGES can be also employed with a grid network of computers managed by middleware such as Condor, Torque, Open Sun Grid and Moab, where PHASER and SHELXE jobs are distributed and parallelized through the grid. In both implementations, the supercomputer and the grid-distributed network, do not pose any limitations on the number of clusters to evaluate nor on the number of solutions that they may contain while the number of prioritized solutions processed with SHELXE is limited to a fix number: 60 for the grid-computing implementation, and the exact number of cores available for the supercomputer. However, both numbers can be modified through the instruction *bor* file.

6.4 Successful cases in ARCIMBOLDO_BORGES

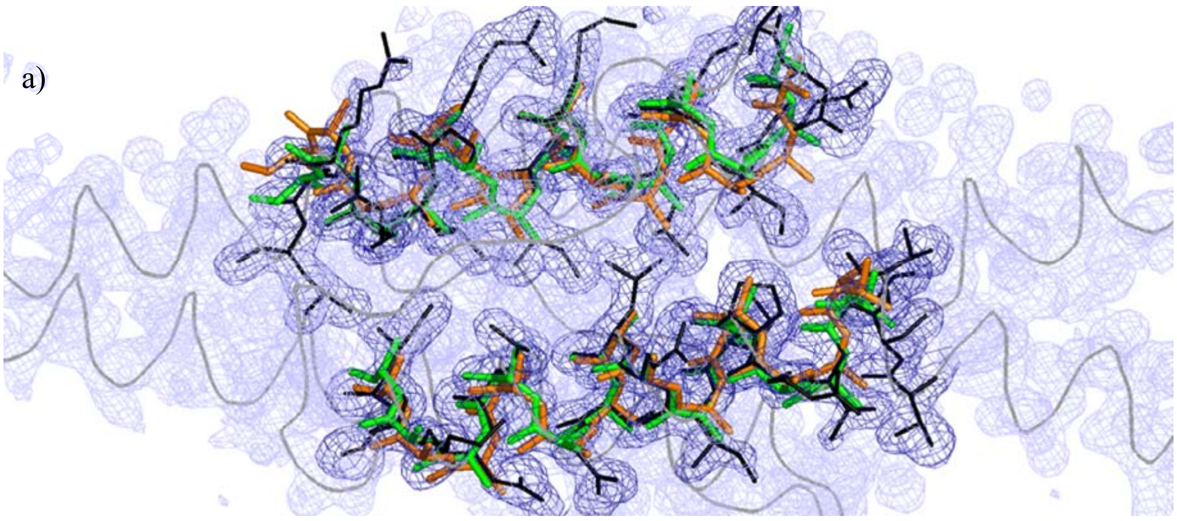
Apart of being tested against a pool of known structures ARCIMBOLDO_BORGES has solved in our hands four previously unknown structures reported in

Data from	Space Group	Residues	Search Fragment(s)	d(Å)	PDBID
S. Fedosyuk, K. Djinovic, T. Skern	C2	150	Library of three antiparallel β-strands for a total of 20 aa	1.55	
J.M. Pereda	C2	240	Library of two antiparallel helices of 17 aa	1.7	4GDO
K.Zeth, A. Lupas	$P2_1$	428	Library of two parallel helices of 16 aa	1.7	4GN0
M. van Breugel	$P2_1$	600	Library of two antiparallel helices of 17 aa	1.97	

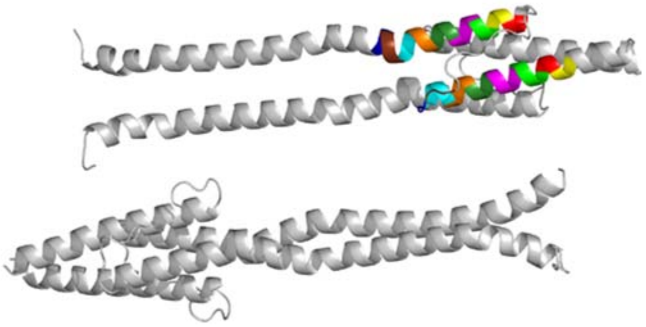
Table 6-3 Structures solved with ARCIMBOLDO_BORGES

So far, only the structures 4GDO and 4GN0 have been published [7].

a)



b)



c)

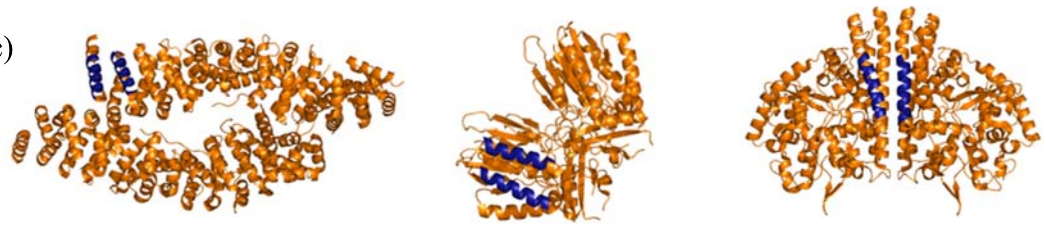


Figure 6-3 Solution of the AF1503 membrane protein

(a) SHELXE electron density map (FwMPE = 42°). The final 4GN0 model is depicted in gray. The closest original fragment (orange) extracted from 2GL7 (ref. 19) has an r.m.s. deviation of 0.90 Å, whereas after refinement (green) the RMSD is 0.54 Å. (b) Cartoon representation of 4GN0; coloured regions display the location of fragments leading to solution. (c) Three of the 13 structures from which models (blue) were extracted to solve the 4GN0 structure: β-catenin–BCL9–Tcf4 complex (2GL7), bacterial Mre11 core (3THN) and cytochrome C nitrite reductase (1QDB).

Figure 6-3 describes the solution of the partial AF1503 membrane protein from *Archaeoglobus fulgidus*. In the solution of this structure it was fundamental the employ of P1 refinement against the rotation function (Section 6.1.4), which improved the model up to an RMSD of 0.54 Å (Figure 6-3 (a)), allowing to phase the structure. This was solved multiple times starting from library models completely unrelated to the structure of the protein (Figure 6-3 (b)). These models would not be extracted by sequence alignments and thus they confirm the validity of the libraries for *ab initio* approach in which ubiquitous folds can be located in unknown protein structures.
The following Section will describe another case, also reported in the same publication. The coiled coil plectin fragment deposited under the *pdb* code 4GDO, was solved through the previously presented library of antiparallel helices of 17 residues. The same library was

used to phase another coiled coil structure from Dr. van Breugel. Finally, the 1.55 Å structure of a virus protein was solved with the library of antiparallel β-strands also employed for testing. This was the first case of *ab initio* phasing for a previously unknown all-β structure through an ARCIMBOLDO method. It was phased employing the Gordon supercomputer in San Diego (California). The case is described later in this Chapter.

6.5 Solving the coiled coil plectin fragment of the Rod domain

The 223-amino-acid structure of a plectin fragment 4GDO at 1.7 Å in C2 is a coiled coil helical structure solved with ARCIMBOLDO_BORGES in 2012. 67 models out of 121 structures (82 unrelated) in the PDB were similar enough to one portion of the final structure (<0.6 Å) for our library of contiguous antiparallel helices to solve, even without further model refinement. The overall pattern of contiguous helices in coiled coil structures is usually visible in the Patterson Map, these structures tend to have multiple copies in the asymmetric unit related by NCS and a self-rotation function can help identifying the general direction of the helices. In this case both Patterson and self-rotation evidenced the presence of helices but an ARCIMBOLDO sequential search for single fragments failed because all solutions at the stage of the second fragment were positioned on top of previously found helices. Packing function discards such solutions and reaching an interpretable map from just one helix was not possible. This justified the use of a library of models of contiguous helices. As a preliminary step, we tested the use of the experimental rotation function to obtain structural information on the crystal contents. The first test we performed was to analyse with ARCIMBOLDO only rotation clusters produced for one ideal helix of 14 residues. We aimed to predict elongation for this helix by collecting rotations around the helical axis (Section 3.1.2). This was done by computing CV distributions for the helix rotated by the different solutions and identifying an elongation in case two rotations would map when shifting one of the two CV distributions for *k* number of CVs. The highest *k* should correspond to the elongation of the helix. An example of this procedure is shown in Figure 6-4 where a small helix of 7 residues produces 5 different rotations that in fact map different overlapping parts of a longer helix. Comparing the 5 rotations by their Euler angles or by matching corresponding CVs (comparisons indicated as dotted line in the Figure 6-5) would not identify that they actually are elongations of the same helix. On the other hand, if rotations are compared in pairs and each CV of the first rotation is compared with the next (right shift) or with the previous one (left shift) of the second rotated helix then the two would be recognized as the same rotation produced by a shift of one residue (blue lines in the figure). The sum of the longest left shift with the longest right one corresponds to the predicted elongation of the helix. However, this procedure can recognize only a limited number of elongated residues for a helix, given the internal periodicity of the secondary structure element origins that after 4 shifts in either direction, the original Euler angles are obtained again reproduced and the rotation cannot be distinguished from the starting one. Also, rotations with lower LLG might indicate errors derived from partial model mismatch, signalling a shorter extent of the elongation.

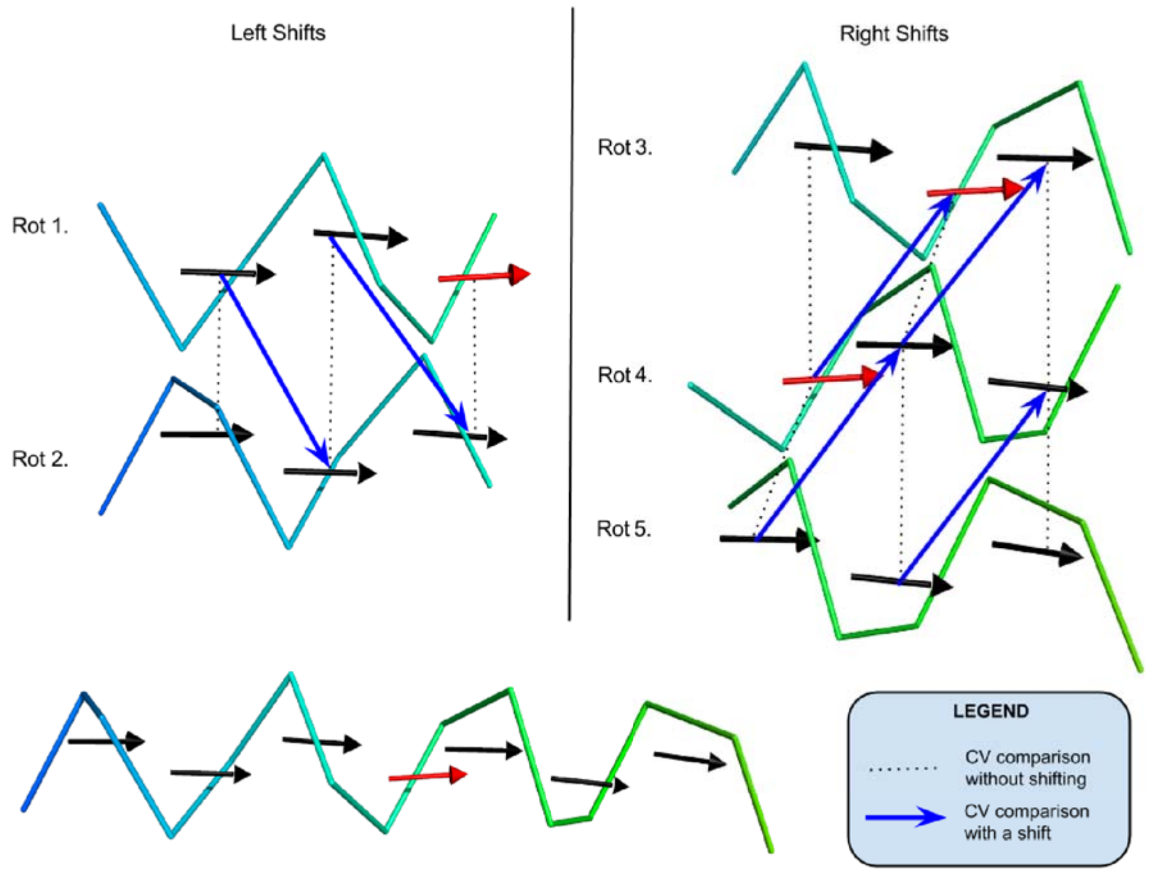


Figure 6-4 Elongation search for an helix

Example of a small helix of 7 residues producing 5 different rotations. These rotations would have different Euler angles and direct comparison does not identify any relationship among them. The dotted lines identify comparisons between corresponding CVs of two rotations: the first CV of the first rotated helix is compared with the first CV of the second rotated helix and so on. The blue arrows display comparisons between shifted CVs. In particular, the second helix is shifted left or right so the first CV of the second rotated helix is compared with the second CV of the first helix and so on. These comparisons help identify elongation as the sum of the longest left shift and the longest right one. A longer helix can then be predicted from the analysis of the rotation function.

For the *plectin* fragment, the procedure generated 6 rotation clusters and for two of them an elongation of six residues was compatible predicting two helices of 20 aa length. Actually, after solving the structure the helices resulted to be 40 aa length but for the reasons detailed, the algorithm would require a longer search helix to probe them. From biological and functional knowledge the expected structure was a parallel coiled coil of helices but due to the indetermination inherent to the rotation function we decided to test both conformations. We had previously extracted libraries of parallel and antiparallel helices but the second library contained models that were longer by two residues than the parallel ones. One residue per helix was trimmed from the models of the antiparallel library to make both comparable and we took a random sample of 51 models from both libraries. We then performed a MR rotation function with PHASER for all the models limiting the resolution to 3.0 Å.

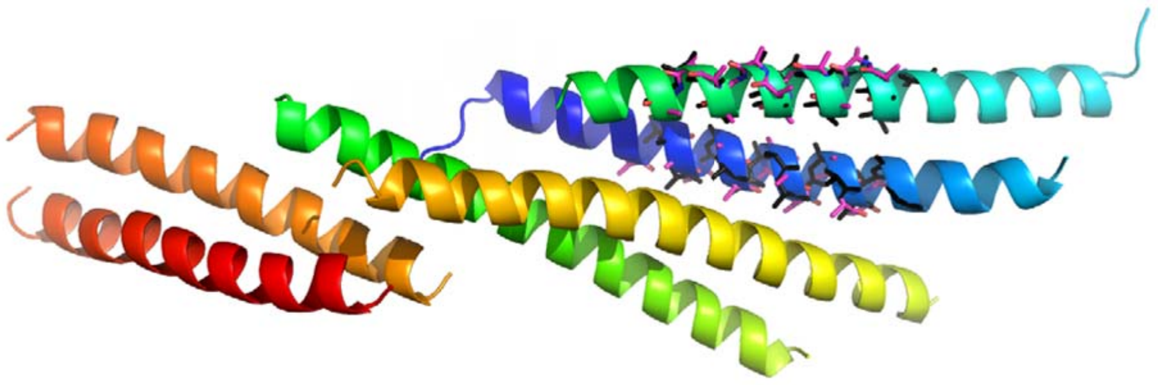
	PARALLEL HELICES of 16 aa	ANTIPARALLEL HELICES of 16 aa
Samples	51	51
Best LLG	62.41	99.59
Best ZSCORE	4.46	4.41

Table 6-4 Parallel vs Antiparallel helical libraries for plectin fragment

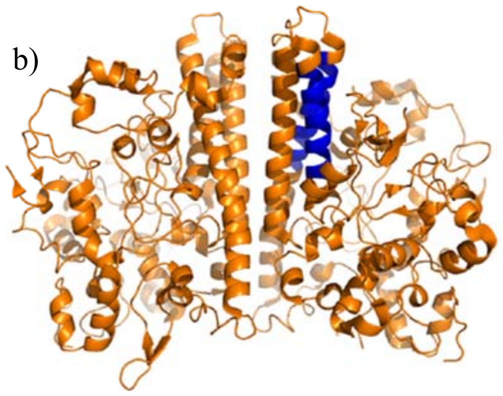
	Residue ranges	RMSD vs plectin (Å)	LLG	ZSCORE	CC (%)	FwMPE (º)
1OAH	A377-393 A407-423	0.46	122.93	7.35	49.88	32.0
3AJW	A10-26 A96-111	0.29	140.80	7.46	50.43	31.9

Table 6-5 FOMs of the solutions for the plectin fragment

a)



b)



c)

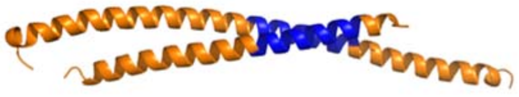


Figure 6-5 Plectin Fragment structure with the solving library models.

(a) The plectin fragment of the Rod domain is composed by six helices in the asymmetric unit. Colours reflect B-factors values, where red is the region of highest B-factors. The libraries models: 1OAH in magenta and 3AJW in black are both located in the same area of lower B-factors. (b) 1OAH structure is the cytochrome c nitrite reductase from Desulfovibrio desulfuricans ATCC 27774, the library model extracted with BORGES is highlighted in blue. (c) 3AJW structure represent the common architecture of the flagellar type III protein export apparatus and F- and V-type ATPases, again the blue region indicates the residue ranges from which the model was extracted.

Rotation ZSCOREs were comparable for both libraries but LLGs were clearly better for the antiparallel library. Extending the test to the full libraries confirmed the outcome, increasing the LLG of the antiparallel library to 110.07 while the LLG of the parallel remained 68.26. Accordingly, we decided to pursue phasing with the antiparallel library.
After running ARCIMBOLDO_BORGES 13 rotation clusters were found and the second most populated cluster, with 6284 models, was the correct rotation cluster containing two models, one cut from the *pdb* 3AJW and another from the *pdb* 1OAH, from which the correct structure of the *plectin* could be reconstructed *ab initio*. Both models produced CC around 50% and FwMPE under 32º, moreover the RMSD against the final structure was lower than 0.5 Å in both cases. The structure 3AJW represents the common architecture of the flagellar type III protein export apparatus and F- and V-type ATPases[162]. Although, the two structures are not homologous, as a local alignment of 162 residues gives a SI of 22%, their similarity of 33% indicates conservation in the secondary structure.
Both the model from 1OAH and from 3AJW were placed by PHASER on the helices with lower B-factors. Refinement of this structure, performed by Dr. Buey, was complicated by the presence of two helices with high B-factors in which the electron density map was not clear. Automatic model building failed reconstructing these helices that have been manually modelled and refined.

6.6 Solving an all-β virus protein structure involved in immunomodulation

Recently our group solved a 150 residue long all-beta structure of a virus protein using ARCIMBOLDO-BORGES. Data were collected from Dr. Fedosyuk, Dr. Djinovic and Dr. Skern. Data resolution reached 1.55 Å and crystals belonged to space group C2. We performed the calculations in the Gordon supercomputer in San Diego (CA, USA) with the collaboration of Prof. Dr. Lynn Ten Eyck.

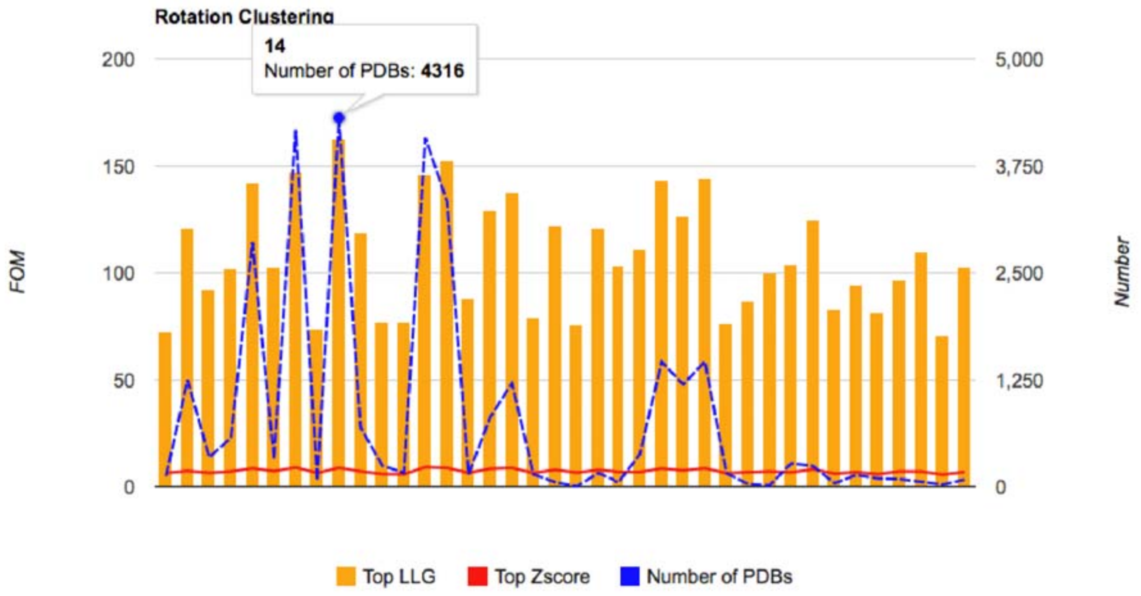


Figure 6-6 MR Rotation Function output

View of the dynamic graph incorporated in the *html* output of the program displaying the clustering and the analysis of the MR rotation function. 38 out of 75 clusters are selected from ARCIMBOLDO_BORGES to be evaluated. The most populated cluster with ID 14, is the first being processed and it also contains the top rotation LLG among all the clusters.

The supercomputer power gave us the possibility to extend the number of parallel jobs. The job was launched to occupy 64 cores in 4 node machines thus we were expanding 64 solutions with SHELXE.

We started with a secondary structure prediction that confirmed the presence of only β-strands but no information about the conformation of the sheets. We decided to start with the most frequent fold of three antiparallel β-strands with models of 20 residue length, and this library was successful. After performing an MR rotation function in PHASER, 75 rotation clusters were identified and the most populated one with 4316 models, which is the one prioritized by the algorithm, was also the correct cluster that led to solutions. The structure was solved with starting phases generated from 5 different models, two of which are homologous structures, thus resulting in 4 unrelated models summarized in the following Table 6-6:

	Residue ranges	RMSD vs traced structure (Å)	LLG	ZSCORE	CC (%)	FwMPE (º)
2QLG	A92-99 A104-108 A175-181	0.42	159.1	5.11	42.20	25.7
2GSK	A218-222 A245-252 A267-273	0.25	137.0	5.26	44.68	25.6
2EFU	A22-29 A35-39 A326-332	0.61	125.55	5.08	45.56	25.5
4DCB	A136-140 A167-174 A200-206	0.35	141.7	4.76	45.69	25.6

Table 6-6 Solutions for the all-β viral structure.

This structure is still unpublished but constitutes an important milestone for our method, being the first all-β structure ever solved with one of the ARCIMBOLDO procedures.

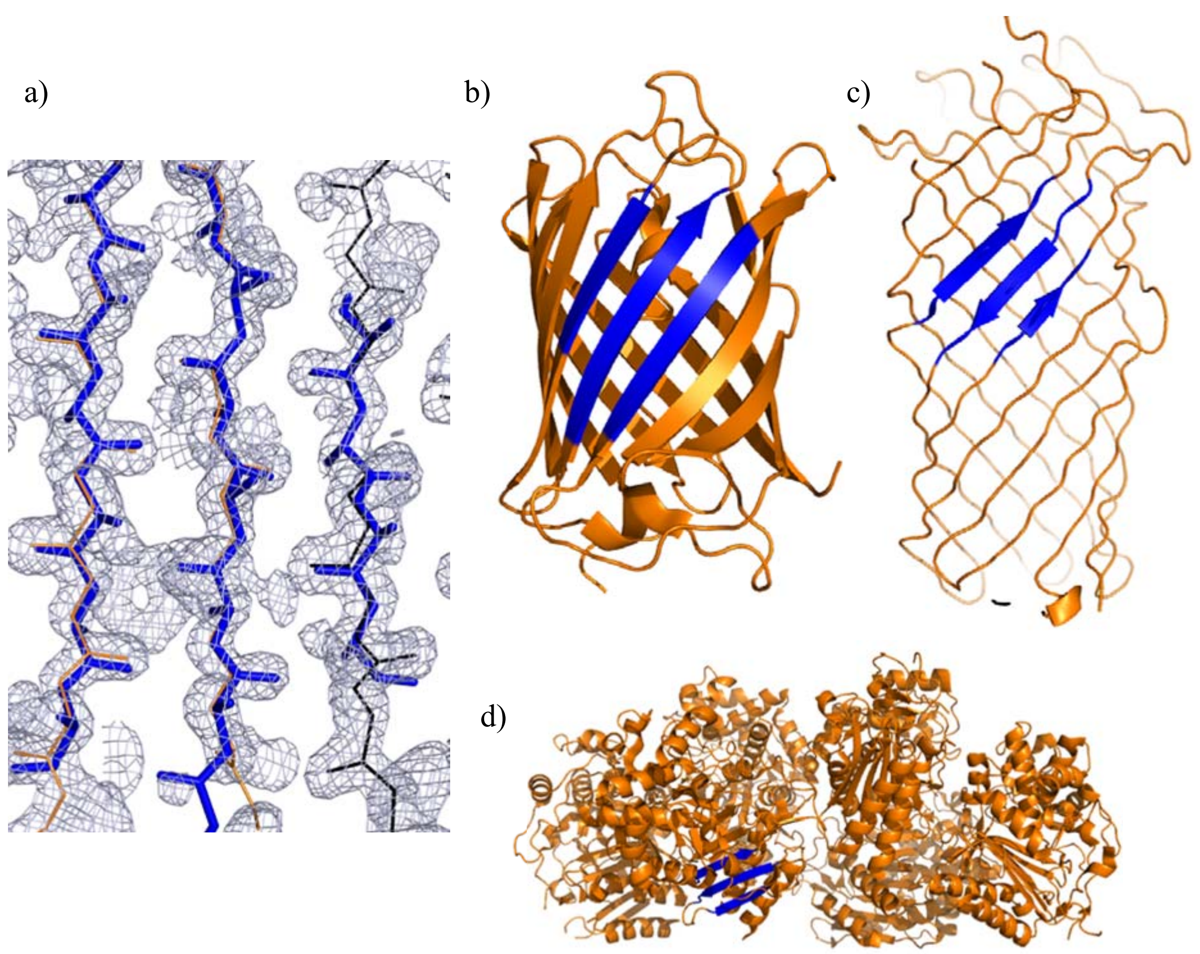


Figure 6-7 View of the all-β viral structure and the extracted library models

(a) Partial view of the all-β viral protein as traced by SHELXE. the region from which models were extracted with BORGES is shown in blue. (b) The mPlum protein, a monomeric red fluorescent protein 2QLQ. (c) The crystal structure of D-amino acid amidase from Ochrobactrum anthropi SV3 complexed with L-phenylalanine 2EFU. (d) Y. pestis Plasminogen Activator Pla 4DCB.

7 SOFTWARE DEVELOPER ENVIRONMENT

In scientific research, software can only develop if it is also capable of continuously evolving so that it can adapt its functions to the knowledge and ideas of the researcher. A classical previous step of requirement collection is almost impossible, because it requires a comprehensive and global view of all the expected results that could not be known at that stage. At the same time, refinement of the software itself would only be possible after producing, analyzing and discussing some results from a prototype version. For these reasons a waterfall development model[163] is unreachable, while an increment model[164] based on the programming and validation of little steps could not satisfy the needs of a researcher that cannot approve the whole software until the final results are produced. The world of research is a dynamic world, where people share ideas, make experiments and revisit the foundations after weeks of work. Therefore, programming software in an environment like this requires a reflection of this dynamicity and the way we have chosen to achieve this goal was using the XP Programming[165] development model. This methodology is based on the active participation of all stakeholders on all the producing phases, generating working prototypes and going back refining them constantly.
All software was programmed under the supervision of the principal investigator and director of this thesis Dr. Isabel Usón.

7.1 Software engineering and design patterns

Programs are coded through modular library design pattern. Object Oriented Programming[166] (OOP) is reserved only to particular aspects in which the benefit of the generalization and hierarchy intrinsic to the OPP paradigm is a real advantage respect to library collections.

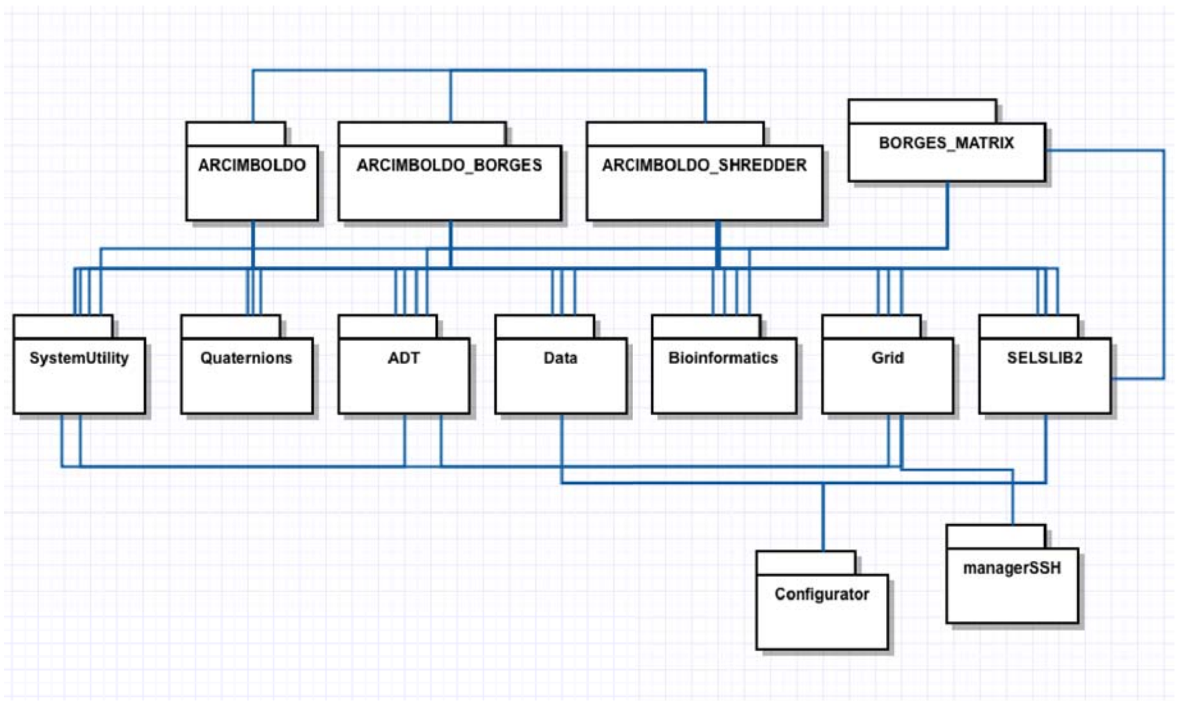


Figure 7-1 UML Schema of the connected modules

Executable programs provided of a Main section are: ARCIMBOLDO, ARCIMBOLDO_BORGES, ARCIMBOLDO_SHREDDER and BORGES_MATRIX, in Figure 7-1 they correspond to the first row of modules. In particular, the first two are also imported as libraries by ARCIMBOLDO_SHREDDER, which automatically connects the analysis of the Shredding function with the evaluation of the generated models through ARCIMBOLDO. The second row in the picture corresponds to libraries of direct access for the main programs:

- **DATA**: contains the defaults configuration for grid middleware and for each one of the main programs deployed. Moreover, it contains all the inbuilt fragments such as an ideal helix of 70 residues and cluster of Fe, Heme group and so on. Currently, libraries of typical folds are not included, and instead these are given through external paths, because their inclusion would generate heavy binaries. However, we are working in a system to compress and encode them to consequentially restore their original coordinates when required by the program.
- **ADT**: the abstract data type module focus on the management of tertiary or inbuilt data type used among the programs and the mathematical functions required for specific issues. The module contain functions to:
    - manipulate tagged instruction files
    - compute statistics employed for peak search and analysis of the Shredder descriptor function and to extract discrete percentiles of distributions.
    - perform mathematical encodings of multiple variables in single integer unit by application of an extended Cantor pairing function.
    - manage matrices, binary min heaps, disjoint sets and a wrapper for atoms and residues used in serialization.
    - define, visit and display trees and graphs. Furthermore, well-known algorithms[167] to find the strongly connected component, perform topological sort and find the shortest path of graph by Dijkstra.
- **Bioinformatics**: collects general functions and procedure for manipulate structures, and compute bioinformatic and crystallographic operations to

- generate subsets or splits of a *pdb* according to sequential or structural criteria; write a *pdb* from a list of residues, atoms or selections; renumber residues or apply offset; substitute B-factors, coordinates or other attributes of a *pdb*; extract atoms by selection criteria from a *pdb*
- check continuity of 2 residues by C-N distance between them
- superpose two structures or compute RMSD of two *pdb* without moving
- K-means Clustering and K-value search by cluster variance and cross validation
- create fragment objects to contain CVs and geometrical descriptors and compute vectorial basic algebra, such as the angle between vectors, normal and dihedral angles to be compared against Ramachandran plot
- perform pairwise local and global alignment and multiple alignments
- check elongation for an helix using CVs and a set of rotation
- get structural information about local folds present in a structure.

- **Quaternions**: contains mathematical description of rotations and for standard crystal setting Sohnke space group.
    - A class is created to define and operate with quaternions; conversions from and to rotation matrices, Euler angles and vector/angle descriptions are included. Moreover, quaternion operations are also implemented such as normalization, conjugate and inverse, point and dot product, rotation of a quaternion, interpolation of two quaternions.
    - Other mathematical, vectorial and crystallographic operations are contained as library functions, i.e collections of symmetry operations and origin positions for standard 65 space group allowed for protein. Conversions between crystallographic and universal orthogonal coordinates (used in standard *pdb* format).
- **SELSLIB2**: contains MR operations through calls to PHASER program and density modification and autotracing via SHELXE. It also includes operations for SHREDDER.
    - Managing input and output for PHASER: Anisotropy correction, rotation function, translation function, symmetry packing, rigid body refinement, Gyre function, normal modal analysis, P1 refinement of individual secondary structure elements in fold libraries.
    - Managing input and output for SHELXE: density modification, autotracing, *pdb* optimization, origin shift search and MPE calculation.
    - Performing checks for software/hardware environment and for input through instruction file *bor*. Also retrieving data from crystallographic data such as space group conformation, resolution, cell dimensions and number of unique reflections.
    - Generating sequential or spherical shreds of *pdb* structures, evaluate LLG of the generate models against rotation function and generate and analyse Shredder descriptor function to compute Shred-LLG.
    - Performing aftermath analysis of solved structures identifying critical FOMs based on MPE and RMSD.
    - Managing rotations and translation MR solutions collecting euler angles, fractional coordinates, relating FOMs. It also performs backtracking analysis of solutions, rotation clustering, solutions filtering, cluster merging and comparing.
    - Writing and reading formatted summary text files to save partial states of the pipeline to continue automatically interrupted/uncompleted jobs.

    - Rotating, translating atom coordinates and generating symmetry equivalent copies of a given model or structure. Also, translating models or symmetry copies in the same unit cell and origin.
    - Managing HTML and XML output and developer logging.
- **SystemUtility**: contains basic interface between OS and the programs.
    - Opening and closing user interface connection to remote workstations.
    - Implementing a system checking guardian to monitor memory, load average of CPU process usage, threads and children processes, process signals.
    - Managing file system operations for a large set of files or directory.
    - Generating new thread or process incorporating block of code to be parallelized.
    - Managing supercomputer, assessing node and generating an internal thread to watch their occupation and internally distribute the load charge of computation through them.
- **Grid**: manages middleware grids interface and user interface for files transfers.
    - An abstract class is created with standard interface to submit single and array jobs and for queue monitoring. This class is then extended with specialized class, each one for a different middleware supported, implementing the interface with the concrete code specific of the middleware. In particular Condor, Sun Grid Engine, Torque and Torque/Moab are currently supported.
    - All middleware implementations support remote access and remote file transfer, in particular it is possible to create remote links, directories, text files; transfer files and directory from local to remote workstation by first archiving and compressing to reduce payload transfer; copy files between remote directories; register specific extensions to automatically retrieving; support for Network File Systems (NFS) and managing of big directories to not overload the system.

The third row of the picture lists support modules with low level implementations of basic OS interfaces and imported from the libraries but not directly accessed from the executable.

- **managerSSH**: contains low level interface for remote connections.
    - Opening and closing remote connection through *ssh* or *sftp* connections. Supporting auto-connection and managing RSA public key identification, coding private ASCII/UNICODE password identification through md5
    - Connecting to an interactive remote shell
    - Generating channels and ssh tunnels.
    - Rsync connections and transfers.

An executable **configurator** is also under development to aid installation of input *bor* files to use the program with a network grid. It will have a graphical user interface and will also test for hardware and software compatibility.

7.2 Deployment and packaging

Binaries of the programs, ARCIMBOLDO_LITE, ARCIMBOLDO_SHREDDER, ARCIMBOLDO_BORGES, BORGES_MATRIX, are generated with PyInstaller a program that converts (packages) Python programs into stand-alone executables, under Windows, Linux, Mac OS X, Solaris and AIX. Its main advantages over similar tools are that PyInstaller works with any version of Python since 2.4, it builds smaller executables owing to transparent compression, it is fully multi-platform, and uses the OS support to load the dynamic libraries. The binaries can be run in the same OS in which they were produced provided an equal or higher python version is accessible. Currently we support Linux, in Ubuntu, CentOS, Debian, openSuse distributions and Mac OX in Mavericks and Yosemite versions.

7.3 Input/Output management

7.3.1 Instruction files: .bor

ARCIMBOLDO programs recognize a formatted text file *bor* as an instruction file from which to read input and configure the run. This file is divided in sections starting with a tagged title (**[title_section]**) followed by all keywords (**key: value**) related to that section for which defaults are not available or for which the user is modifying the default. Starting the program with the command line option –b would display a commented *bor* example. All programs require the section to configure the parallelization modality:

```
1. [CONNECTION]:
2. #    NOTE: following is default
3. distribute_computing: multiprocessing
4. #    NOTE: other modes are:
5. #distribute_computing: local_grid
6. #setup_bor_path: /path/to/setup.bor
7. #    NOTE: if the RSA private key is not found or invalid, 8. a password is
required
9. #distribute_computing: remote_grid
10. #setup_bor_path: /path/to/setup.bor
11. #remote_frontend_passkey: ~/.ssh/id_rsa
12. #
13. #distribute_computing: supercomputer
14. #nodefile_path: /path/to/nodefile.txt
```

Computing can be distributed on a single workstation through multiprocessing (default, line 3) activating the *lite* version of the program, or can be spread to a local or remote grid network, marked respectively in lines 5-6 and lines 9-11. A remote grid requires to provide the RSA identification key to access remotely the submitter machine or to insert the password when prompted by the program. Anyhow, if a grid is used, execution requires the path to the *setup.bor* configuration file installed by the administrator of the grid in which settings parameters are contained to submit correctly jobs. Next section is focused on the description of this special configuration file. The last mode is the one indicated in lines 13-14, to parallelize jobs through a supercomputer; in this case the only additional parameter is the path to the list of hostnames or IP nodes assigned to the ARCIMBOLDO process.

```
15. [GENERAL]:
16. #NOTE: following are mandatory
17. working_directory= /path/to/workdir
18. mtz_path: %(working_directory)s/data.mtz
19. hkl_path: %(working_directory)s/data.hkl
20. #NOTE: following is optional
21. ent_path: %(working_directory)s/structure.ent
```

A second section required for all programs is the GENERAL. As shown at line 18, the keyword defining the path of the working directory is reused as part of the path where data files in *mtz* and *hkl* format are saved. To activate an aftermath post solution analysis, requires the path of the *ent* file that is standard *pdb* file for the final deposited structure.

```
22. [LOCAL]
23. # Third party software paths
24. # Requires PHASER 2.5.7 and SHELXE 2014/4
25. path_local_phaser: phenix.phaser
26. path_local_shelxe: shelxe
27. path_local_arcimboldo: ARCIMBOLDO_BORGES
```

The section LOCAL is only required if multiprocessing or supercomputer modalities are activated otherwise will be ignored. In this section the path to the local PHASER and SHELXE executable are indicated. Moreover, ARCIMBOLDO_BORGES also requires its own path to parallelize the rotation clustering.
The current version of PHASER supported is the 2.5.7 while for SHELXE it is 2014/4, both are the latest released. PHENIX distribution provide an executable version of PHASER called *phenix.phaser* whereas to use the CCP4 distribution it is required to export to the shell all the libraries used by the program. A simple bash script can be written including the following line:

```
export LD_LIBRARY_PATH=/path/to/phaser/lib/:$LD_LIBRARY_PATH
```

libraries required are: libcctbx.so, libcctbx_sgtbx_asu.so, libiotbx_pdb.so, libmmtbx_masks.so, libomptbx.so.

The others sections are alternative and specific for each program. A list with keyword options for ARCIMBOLDO follows:

```
28. [ARCIMBOLDO]
29. name_job: example
30. molecular_weight:
31. f_label: F
32. sigf_label: SIGF
33. #Or alternatively
34. #i_label: I
35. #sigi_label: SIGI
36. number_of_component: 1
37. fragment_to_search: 2
38. helix_length: 14
```

A basic run requires the description of the molecular content of the crystal through the molecular weight and the expected number of copies in the asymmetric unit (number_of_component). Crystallographic data in *mtz* format can contain different set of collected data, for this reason must be provided the labels for structure factors, or alternative for intensities, and their relative errors to employ for the job. Finally the number of helices to search and their length can be specified modifying the default option that search for two helices of 14 residues.

```
39. #NOTE: to use a different helix length for each fragment, use instead
"helix_length_n" specifying the length for each fragment to search
40. #helix_length_1: 15
41. #helix_length_2: 13
42. #NOTE: to use a specific pdb model as search fragment, comment out
"helix_length" and set:
43. #model_file: /path/to/the/file.pdb
44. #NOTE: to use a different model for each search fragment, specify
"model_file_n" for EACH fragment
45. #model_file_1: /path/to/file1.pdb
46. #model_file_2: /path/to/file2.pdb
```

An alternative to searching a number of helices of the same size is to specify for each helix the expected length (lines 40-41). External *pdb* files can be used as search models instead of internally generated helices (lines 42-46).

```
47. rmsd: 0.2
48. rotation_clustering_algorithm: rot_matrices
49. threshold_algorithm: 15
50. resolution_rotation: 1.0
51. sampling_rotation: -1
52. number_of_component: 1
53. resolution_translation: 1.0
54. sampling_translation: -1
55. resolution_refinement: 1.0
56. sampling_refinement: -1
57. exclude_llg: 0
58. exclude_zscore: 0
59. pack_clashes: 0
60. pack_distance: 3.0
```

ARCIMBOLDO jobs searching for helices usually do not require to modify the expected RMSD of the model against the final structure (line 47), indeed main chain of small helices tend to be conserved through unrelated structures, but if an external model is input the user should consider to modify this parameter accordingly. Space group and resolution is read from the data but the user can force a resolution limit for each specific MR operation or override the sampling internally configured by PHASER (lines 50-56). The user can select the rotation clustering algorithm (48-49) employing matrices, quaternions or CV distribution (Section 3.1.2). MR solutions can be discarded by excluding those with LLG or ZSCORE under the limit indicated in lines (57-58). Default is to discard solutions associated to negative scores. In some cases, especially when external models are input, the number of allowed clashes (default is 0) or the relative clashing distance for atoms could be too stringent. Thus, modifying these parameters, solutions with clashes would survive the packing filtering, and could be improved later on with *pdb* optimization in SHELXE.

```
61. TNCS: True
62. #if Intensities
63. #ANISO: False
64. #if Structure Factors
65. ANISO: True
66. nice: 0
```

```
67. #NOTE: the program automatically configure the shelxe_line but you can
decomment it and use your own
68. #shelxe_line:
69. noDMinitcc: True
70. topExp_1: 60
71. topExp_n: 60
72. RNCS_MATL = []
73. force_core = -1
74. force_exp = False
```

The program does not yet automatically treat rotational NCS, but the user can provide a list of 3x3 rotation matrices (line 72: in python or json format) to allow ARCIMBOLDO including them during the clustering procedure. Translational NCS search is activated by default and usually it is a good strategy to include more symmetry constraints before expanding solutions, although the isolation of pseudo-translations peaks in the Patterson map is not always straightforward. Anisotropy correction of structure factors is usually performed at the starting of the program and the new corrected data are then used in all successive steps. However, the current PHASER version does not allow proceeding in the same way if data are Intensities without accessing PHASER python libraries, which is not yet implemented in our programs. For this reason with intensity data ANISO (line 65) is switched off automatically and the anisotropy correction would be performed each time a MR function is required. Finally, the user can control multiprocessing by assigning a nice priority with a POSIX integer value (10 lowest priority – 0 highest priority), and/or forcing the number of parallel process (line 73) that by default is configured equal to the number of cores minus one. Without modifying the number of parallel processes, it is possible to increase the number of solutions to expand with SHELXE by indicating the number of the maximum solutions to test at first fragment search (line 70) and for the subsequent ones (line 71) and then switching on the parameter *force_exp*.

For ARCIMBOLDO_SHREDDER special keywords are added to select the type of Shred algorithm to use and also distinguish between MR rotation parameter configuration of the original distant homolog and MR configuration for the generated shredded models. For simplicity only specific parameterization is listed here, whereas previously introduced parameters are omitted.

```
75. [ARCIMBOLDO-SHREDDER]
76. model_file: /path/to/the/model.pdb
77. rmsd_shredder: 1.2
78. rmsd_arcimboldo: 0.8
79. resolution_rotation_shredder: 1.0
80. resolution_rotation_arcimboldo: 1.0
81. sampling_rotation_shredder: -1
82. sampling_rotation_arcimboldo: -1
```

The distant homolog model is given together with its expected RMSD (lines 76-77). Usually this value is larger than for the shredded models (line 78) that should be improved to resemble the final structure. The user can also introduce a different parameterization for the MR rotation function of the distant homolog and for the shredded models (lines 79-82), the reason is that sometimes cutting the resolution for the MR rotation function search with a model of hundred residues (a typical template in an ARCIMBOLDO_SHREDDER job), can help retrieving the approximately correct rotation without generating a large number of rotations.

```
83. #SHRED_METHOD: spherical
84. sphere_definition: 15 1 fragment
85. #SHRED_METHOD: secondary_structure
86. cut_alpha_comb: 0
87. cut_beta_comb: 0
88. cut_ss_comb: 3
89. SHRED_METHOD: sequential
90. SHRED_RANGE: 4 20 1 omit all
91. #SHRED_RANGE: 4 20 1 omit fix_ah remove_coil
92. #SHRED_RANGE: 4 20 1 omit fix_ah maintain_coil
93. #SHRED_RANGE: 4 20 1 omit fix_bs remove_coil
94. #SHRED_RANGE: 4 20 1 omit fix_bs maintain_coil
95. #SHRED_RANGE: 4 20 1 fragment
96. writePoly: True
97. maintainCys: False
```

For the Shredding methods, the user can select sequential, spherical or secondary structure. Sequential is activated by default (lines 89-95) and requires to define the range of spanning shreds by three values: minimum cut, maximum cut and step (default 4, 20, 1). Then *omit* will remove the shreds while *fragment* will extract them. If *all* is specified then shreds will be generated independently from all the main chain, but if *fix_ah* (or *fix_bs*) is inserted then shreds will be skipped inside an helix (or a β-strand); the user can also require the program to remove coil regions from all the shredded models or retain side chains instead of writing polyalaline models. Alternatively it can generate polyalaline models with the exception of Cysteine side chains retaining disulphide bridges if present (lines 96-97).

The last program that uses the *bor* configuration file to process input parameters is ARCIMBOLDO_BORGES. Again, only specific parameterization is reported.

```
98. [ARCIMBOLDO-BORGES]
99. library_path: /path/to/the/borges/library/
100. n_clusters: 4
101. prioritize_phasers: False
102. f_p1_label:
103. sigf_p1_label:
104. number_of_component_p1:
105. number_cycles_model_refinement: 3
106. NMA: False
```

This program works with libraries of superposed folds extracted with the program BORGES_MATRIX. These libraries must be valid directories containing *pdb* files named with the following format: **pdbid_m_n.pdb** where *pdbid* is the identifier of the deposited structure in the Protein Data Bank, *m* is the number of model in the structure (0 if X-ray structure) from which the fold was extracted, and *n* is a sequential integer associated to the library model. In multiprocessing mode ARCIMBOLDO_BORGES will explore only the 4 top populated clusters unless differently specified (line 100). Moreover, the program will process each cluster completely (including the expanding step with SHELXE), before testing a new one as the correct clusters is usually one of the top prioritized. However, the user can ask to prioritize PHASER jobs of all the clusters before selecting the solutions to expand with SHELXE (line 101). Patterson Correlation refinement or the refinement of the rotation in P1 is available only if the library of folds is composed by helices and if the extended data to triclinic are also provided, in this case labels and number of component for the new cell must be provided either (lines 102-104). This kind of model refinement is not available for β-strands as it is not appropriate. If model refinement is activated it would be performed 3 times by default before expanding, but the user can reduce/increase the number of these cycles (line 105). The user can also activate a Normal Modal Analysis of the model to give original models of the library more degrees of freedom, but the new generated models will be accepted only if they lead to improved INITCC.

7.3.2 Output files: .html and .xml

ARCIMBOLDO programs generate several output files. Some of them are internally used by the program to recognize intermediate steps and to continue interrupted jobs. This is the case of the directory structures containing *.sum* files that are text files with all MR solutions divided in rotation clusters. The advanced user can parse, search, filter and even modify them to personalize the program run but the standard user probably can get confused by the amount of information contained. A log is also printed trough terminal standard output (that can be redirected to a file), again its content is more indicated for advanced users or developers and it results especially useful for bug reporting. The official user readable output is an *html* page carrying its own stylesheet configuration and JavaScript code, which is used to generate dynamic graphs and to allow sorting tables by clicking any column header. The page is automatically generated and updated by the programs during execution and has the name given at the job in the *bor* configuration file. The html is divided in eight sections:

- Header: that displays the name of the program and buttons linking published papers and dedicated webpage in our website.
- Summary: that reports all used configuration, including non modified parameters. Furthermore, it also displays a small description of the data reporting space group, resolution range, and number of unique reflections. If some warning or error occurs regarding the data it will be displayed also in this section.
- MR analysis: this section is only visible for ARCIMBOLDO_BORGES. It contains a table of all rotation clusters together with their respective FOMs. The table can be reduced by clicking a button that filters out all improbable clusters and shows only the one selected by the program coloured in green. Next to the table an dynamic histogram graph is showed reporting the same information. The graph is clickable to read the current value of each bar, point or to modify the scale of the graph.
- FOMs description:
    - in ARCIMBOLDO consists of a series of tables, one per fragment search, describing FOM evolution of the MR location of the model in the unit cell and finally CC against the data before and after tracing. Rotation clusters and their combinations are separated to be explored and compared.
    - in ARCIMBOLDO_BORGES is a single big table in which each row is associated to a rotation cluster. Colours mark the top of each FOM: LLG, ZSCORE and INITCC. A correct solution, with CC higher than 30%, is marked in green.
- Backtracing of the best solution: reporting detailed information about the evolution of the best solution found (the one with the highest CC). In particular for each MR step relative FOMs and their rank are reported.

- Output link: displayed with a message announcing the solved structure or just the best result obtained along with the links to the traced coordinates *pdb* and the produced map *phs*.
- Timings: in which the list of consumed RT in each step is reported, especially useful for developer support.
- Footer: with the list of citations suggested in cases of success of the program.

Although, the *html* is easily displayable and "almost" compatible with all browsers, other developers can find useful the intermediate *xml* format also generated, where the same information of the html is described with the use of a tagged structure easy to inspect from external programs. In particular, we would like to build a new graphical module that will not perform scientific computation but will only render input/output data through the analysis of the *xml* file. This clean design pattern that disconnect the scientific core code from the GUI implementation is widely used in successful application such as iMosflm[168].

7.4 Grid network support

Programs can parallelize jobs through a grid of connected computers. The installation of the middleware software and the network is not always trivial and usually requires expert knowledge. However, many scientific institutions have direct or remote access to a grid to submit batch jobs. The only requirement for our software is that all machines in the grid have access to the same path where PHASER, SHELXE and our binaries are installed. In the following described *setup.bor* can be configured one local grid and one remote grid, the user will activate either one through the instruction *bor* file.
The section LOCAL refers to a local grid, that is, a grid that is reachable from the machine where the main program is started, that machine must be also a submitter for the grid.

```
[LOCAL]
# Third party software paths
# New phaser stands for phaser 2.5.7 or higher
python_local_interpreter:
path_local_phaser: phenix.phaser
path_local_shelxe: shelxe
path_local_borges: BORGES_MATRIX
path_local_arcimboldo: ARCIMBOLDO_BORGES
```

In this section the path for the programs PHASER, SHELXE, BORGES_MATRIX and ARCIMBOLDO_BORGES are specified. The program automatically reads these paths and will configure grid job to execute locally the third party programs.

```
[CONDOR]
# Parameters for each executable under Condor (memory constraints, CPU speed ...)
requirements_shelxe:
requirements_phaser:
requirements_borges:
```

If a Condor grid is installed, be it a local grid or a remote one, job requirements can set for a specific type of job the pool of machines in which to calculate. The requirement syntax is specific of the Condor middleware and should be consulted from its manual. All these keywords are optional and can be let blank if they should be ignored by the programs.

```
15. [SGE]
16. qname: name_queue
17. fraction: 1.00
```

For Sun Grid Engine grid the queue name must be specified and the fraction of the free machines in that queue to dynamically occupy with the generated jobs.

```
18. [TORQUE]
19. qname: arcimboldo
20. [MOAB]
21. partition: uc1
```

Similarly to SGE grid, also Torque and Moab middleware just require indicating the queue name and the partition name respectively.

```
22. # Grid environment parameters
23. [GRID]
24. # BORGES parameters for "borges" database generation
25. number_of_pdbs_per_tar: 5
26. number_of_parallel_grid_jobs: 7000
27. type_remote: SGE
28. type_local: Condor
29. path_remote_sgepy:
30. #REMOTE IMPLEMENTATION
31. path_remote_phaser: phenix.phaser
32. path_remote_shelxe: shelxe
33. path_remote_borges:
34. python_remote_interpreter:
35. remote_frontend_username:
36. remote_frontend_host:
37. path_remote_sgepy:
38. home_frontend_directory:
39. remote_frontend_port:
40. remote_fylesystem_isnfs:
41. remote_frontend_prompt:
```

The GRID section contains all parameters to activate a particular remote configuration. In particular it is possible to link the local grid configuration to one among CONDOR, SGE or MOAB and link the remote one to another one (lines 27-28). When a BORGES_MATRIX run is started against a grid it is possible to decide the number of models composing a compressed packed sequentially evaluated from one machine of the grid (line 25) and the number of parallel jobs that BORGES_MATRIX can maintain in its internal queue (line 26). In fact, process output from single packet jobs of BORGES_MATRIX can require time and to avoid increasing of memory occupation and to control the size of the queue, this number can be limited to a maximum value. Finally, the last lines 29-39 collect access information to the remote frontend of the grid, such as username, hostname and port number and possibly a second connection step can be also included.

OUTLOOK

The work on the development and implementation of ARCIMBOLDO, detailed in this thesis has led to identify many opportunities for improvement and extensions that should be addressed in the future. Related projects are also suggested.

In all programs:

- Internal data quality check may be extended, enhancing data characterization and strategy prediction prior to the execution of the program: for example analysing completeness and signal to noise ratio at the highest resolution shell can provide valuable indication to set the program to better address the limitations of the case.
- Rotation analysis and clustering should be consistently analysed. Our current experience of the implemented algorithms is the result of empirical experiments while a formal statistical and analytical test should be conducted to derive the best strategy.
- K-means clustering requires *a priori* knowledge of the number of clusters (the value k). Currently this value is found by probing different values until variance within clusters does not significantly decrease by adding more clusters. More sophisticated approaches as the Gap statistic[169] are available and should be tested.
- Helix elongation search can be optimized exploiting the crystallographic and geometrical constraints. Quaternions can be used to define interpolated rotations around the helix axis and Patterson peaks may be analysed in search for helical features
- Non Crystallographic Symmetry Rotations are input to the program as external matrices but internal analysis of rotational NCS is possible and should be conducted. For example the tool Polarrfn[81] can be used to identify them and generate rotation matrices.
- Solution prioritization by INITCC or LLG has been a good approximation of selecting the most promising solutions, although in some test cases MPE revealed that correct solutions were discarded by the algorithm. More complex sorting functions may be generated taking into account various FOMs and enhancing the likelihood of expanding best solutions.
- Expanding solutions at lower resolutions (> 2.5 Å) should require a different SHELXE parameterization and should be investigated systematically, as we have done for high and medium resolution.

In ARCIMBOLDO:

- The power of supercomputer can be exploited to evaluate random fragment positioning to be tested against the LLG. In this way borderline cases may have the possibility to be correctly placed.

In ARCIMBOLDO_SHREDDER:

- Spherical and Secondary structure shredding modalities should be tested deeply and improved to define properly strategies reflecting the different nature of the models
- A complete battery of tests should be conducted for ARCIMBOLDO_SHREDDER as has been done for ARCIMBOLDO_LITE and ARCIMBOLDO_BORGES in order to determine a general parameterization.
- Peak analysis can be more sophisticated and make use of more complex statistics, as Personal Component Analysis (PCA), to determine the region and their elongation to cut out.

In BORGES_MATRIX:

- A systematic comparison between the first prototype and the new version should be also conducted to ensure all features of the first version are preserved in the second.
- The scoring algorithm of CV distributions describing fragments can be more sophisticated. At present, a simple ranked-sum over the different parameters is conducted, evenly weighting all geometrical parameters: distance, angles, CV length. In the future, the algorithm should weight the contributions of different regions of the fragment, as example extremities of a fragment are intrinsically more mobile than internal regions.
- A database containing CVs of all the deposited structure can be implemented and maintained to allow BORGES_MATRIX to query it without recomputing constant values.
- Superposition and clustering parameterization is appropriate for phasing, but detailed studies must be conducted to establish a new configuration for the statistical analysis of local folds.

In ARCIMBOLDO_BORGES:

- For helices, we employ P1 refinement to allow the composite models internal degrees of freedom and thus better approximate search substructures to the real structure. We should investigate different alternatives to local model refinement of other tertiary conformations such as beta sheets, that can be expressed with few degrees of freedom. Structure properties lead to the expectation that their variation is rooted in backbone torsion angles and cannot be expressed through rigid body movement of the single strands.

CONCLUSIONS

Analysing the results from the studies described in this thesis we can conclude that all proposed objectives have been satisfactorily completed. In particular new macromolecular phasing methods for data with resolution up to 2 Å, based on the enforcement of secondary and tertiary structure have been developed and implemented in distributed software:

- The program ARCIMBOLDO has been redesigned incorporating past experience from a prototype and extensive testing in the course of this work. It has been implemented in Python, introducing new algorithms for analysis, clustering and solution filtering
- ARCIMBOLDO has been made scalable to different hardware, ranging from single workstations to clusters, grids and supercomputers. Alternative strategies have been tailored to the various hardware scenarios.
- Algorithms for *a posteriori* analysis of phasing on test structures have been included and automated.
- ARCIMBOLDO_LITE was tested over a pool of 294 structures, succeeding in 139 out of the 230 helical cases and in 4 cases where chemical moieties of known structure, such as clusters or cofactors could be used as search fragments. This test led to a default, optimized parameterization in the program, which has increased the rate of success.
- At least 22 previously unknown structures of sizes ranging from 50 to 360 aa and resolutions comprised between 2.12 and 1.4 Å have been solved with ARCIMBOLDO method by us as well as by independent users.
- In particular, the solution of a 13-fold superhelix has been studied and the effects of the helix length have been explored in this context.
- A new method named shredder has been proposed to improve and evaluate search models relying on the experimental data. In particular, exploiting the rotation function.
- ARCIMBOLDO_SHREDDER has been developed including algorithms to generate sequential, spherical or secondary structure filtered shred models from distant homologs.
- The Shred-LLG has been introduced to evaluate and exploit conserved regions and by the interpretation of the Shred-descriptor function various models are produced and tested with ARCIMBOLDO.
- The novel structure MltE, originally solved with multiple ARCIMBOLDO runs on fragments cut from a distant homolog has been solved with the ARCIMBOLDO_SHREDDER algorithm.
- Other previously unknown structures have been phased with SHREDDER one diffracting at 2.35 with 1450 aa and another diffracting at 2.5 Å with 532 aa.
- In the course of this work, BORGES was implemented as a first prototype in python, as an algorithm to define and extract customized libraries of local folds to be used for phasing. In detail:
    - Characteristic Vector distributions were introduced as geometrical descriptors for local folds. CVs have been proven to match DSSP predictions.
    - Mathematical operations among CVs were designed to spatially compare fragments and to locally describe distortions.
    - Non parametric statistic tests were applied to 4 pools of different fragment size for both helical and β-strands fragments, establishing a dependence of the CVL to the size of fragment resolved with the general definition of CV distributions of overlapping tripetides.

- BORGES_MATRIX was implemented as a revisited pattern recognition algorithm based on the theory of Complete Graphs. The program introduced:
    - Superposition algorithms based on the search of the best core
    - Clustering of intermediate solutions trough K-means and clustering of the global extracted models through reference and RMSD.
- Libraries of local folds have been extracted, superposed and clustered. Represented fragments comprise helical DNA binding motifs and other general motifs of parallel, antiparallel and parallel-antiparallel β-sheets, parallel and antiparallel contiguous helices and polypeptides linked by disulphide bridges.
- ARCIMBOLDO_BORGES was written to exploit the libraries of local folds previously computed.
- Even without the use of supercomputing, 38 test cases were solved out of 151 with the single workstation implementation of ARCIMBOLDO_BORGES. This test enlightens the relevance of the *pdb* optimization when using libraries and the dependence of the success from the hardware configuration.
- At least 4 novel structures have been solved with ARCIMBOLDO_BORGES. In particular the method proved to be successful both for helical structures as in the case of the solution of a plectin fragment and for all-β structures such as the immunomodulator virus protein, despite the intrinsic difficulty associated to structures with no α-helices.
- All software has been ported to both Linux and Mac OSX Operating Systems and made compatible with a broad choice of middleware, in particular Condor, Opengrid/SGE, Torque and MOAB. Running times for the test structures quoted typically amounted to hours for ARCIMBOLDO_LITE and days for ARCIMBOLDO_BORGES.
- User-friendly access to the input/output user-interface has been introduced by the creation of tagged configuration *bor* files, and the generation of graphical output through *html.*
- Computation has been parallelized for different environments ranging from a single workstation machine, to a distributed grid computer network or supercomputer facilities.

REFERENCES

1. Rodríguez, D.D., Grosse, C., Himmel, S., González, C., de Ilarduya, I.M., Becker, S., Sheldrick, G.M., and Usón, I., *Crystallographic ab initio protein structure solution below atomic resolution.* Nature methods, 2009. **6**: p. 651-3.
2. Rodríguez, D., Sammito, M., Meindl, K., De Ilarduya, I.M., Potratz, M., Sheldrick, G.M., and Usón, I., *Practical structure solution with ARCIMBOLDO.* Acta Crystallographica Section D Biological Crystallography, 2012. **68**: p. 336-343.
3. Millán, C., Sammito, M., and Usón, I., *Macromolecular ab initio phasing enforcing secondary and tertiary structure.* IUCrJ, 2015. **2**: p. 95-105.
4. Sammito, M., Millán, C., Frieske, D., Rodríguez-Freirea, E., Borges, R.J., and Usón, I., *ARCIMBOLDO_LITE: single workstation implementation and use.* Acta Crystallographica Section D Biological Crystallography, 2015.
5. Mccoy, A.J., Grosse-kunstleve, R.W., Adams, P.D., Winn, M.D., Storoni, L.C., and Read, R.J., *Phaser crystallographic software.* Journal of applied crystallography, 2007. **40**: p. 658-674.
6. Sheldrick, G.M., *Macromolecular phasing with SHELXE.* Zeitschrift für Kristallographie, 2002. **217**: p. 644-650.
7. Sammito, M., Millán, C., Rodríguez, D.D., de Ilarduya, I.M., Meindl, K., De Marino, I., Petrillo, G., Buey, R.M., de Pereda, J.M., Zeth, K., Sheldrick, G.M., and Usón, I., *Exploiting tertiary structure through local folds for crystallographic phasing.* Nature methods, 2013. **10**: p. 1099-101.
8. Sammito, M., Meindl, K., de Ilarduya, I.M., Millán, C., Artola-Recolons, C., Hermoso, J.A., and Usón, I., *Structure solution with ARCIMBOLDO using fragments derived from distant homology models.* FEBS Journal, 2014. **281** p. 4029-45
9. Schoch, G.A., Sammito, M., Millán, C., Usón, I., and Rudolph, M.G., *Structure of a 13-fold superhelix (almost) determined from first principles.* IUCrJ, 2015. **2**: p. 177-187.
10. Friedrich, W., Knipping, P., and von Laue, M., Sitzungsber. K. Bayer. Akad. Wiss., 1912: p. 303-322.
11. von Laue, M., *Eine quantitative prüfung der theorie für die interferenz-erscheinungen bei Röntgenstrahlen.* Sitzungsberichte der Königlich Bayerische Akademie der Wissenschaften, 1912: p. 363-373.
12. Bragg, W.L., *The Structure of Some Crystals as indicated by their Diffraction of X-rays.* Proc. Roy. Soc., 1913b. **89**: p. 248-277.
13. W.H., B. and W.L., B., *The structure of the diamond.* Nature, 1913b. **91**: p. 557.
14. Watson, J.D. and Crick, F.H., *Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid.* Nature, 1953. **171**(4356): p. 737-8.
15. Bernal, J.D. and Crowfoot, D., *X-ray photographs of crystalline pepsin.* Nature, 1934. **133**(3369): p. 794-795.
16. Kendrew, J.C., Bodo, G., Dintzis, H.M., Parrish, R.G., Wyckoff, H., and Phillips, D.C., *A three-dimensional model of the myoglobin molecule obtained by x-ray analysis.* Nature, 1958. **181**(4610): p. 662-6.
17. Perutz, M.F., Rossmann, M.G., Cullis, A.F., Muirhead, H., Will, G., and North, A.C., *Structure of haemoglobin: a three-dimensional Fourier synthesis at 5.5-A. resolution, obtained by X-ray analysis.* Nature, 1960. **185**(4711): p. 416-22.
18. Blake, C.C., Koenig, D.F., Mair, G.A., North, A.C., Phillips, D.C., and Sarma, V.R., *Structure of hen egg-white lysozyme. A three-dimensional Fourier synthesis at 2 Angstrom resolution.* Nature, 1965. **206**(4986): p. 757-61.
19. Blundell, T.L., Cutfield, J.F., Cutfield, S.M., Dodson, E.J., Dodson, G.G., Hodgkin, D.C., Mercola, D.A., and Vijayan, M., *Atomic positions in rhombohedral 2-zinc insulin crystals.* Nature, 1971. **231**(5304): p. 506-11.

20. Wiley, D.C. and Skehel, J.J., *Crystallization and x-ray diffraction studies on the haemagglutinin glycoprotein from the membrane of influenza virus.* J Mol Biol, 1977. **112**(2): p. 343-7.
21. Wilson, I.A., Skehel, J.J., and Wiley, D.C., *Structure of the haemagglutinin membrane glycoprotein of influenza virus at 3 A resolution.* Nature, 1981. **289**(5796): p. 366-73.
22. Weissenhorn, W., Dessen, A., Harrison, S.C., Skehel, J.J., and Wiley, D.C., *Atomic structure of the ectodomain from HIV-1 gp41.* Nature, 1997. **387**(6631): p. 426-30.
23. Karle, J.t. and Hauptman, H., *The phases and magnitudes of the structure factors.* Acta Crystallographica, 1950. **3**(3): p. 181-187.
24. Anfinsen, C.B., *Principles that govern the folding of protein chains.* Science, 1973. **181**(4096): p. 223-30.
25. Huber, R., *Die automatisierte Faltmolekülmethode.* Acta Crystallographica, 1965. **19**: p. 353-356.
26. Rossmann, M.G., *The molecular replacement method.*, in *Acta Crystallographica Section A: Foundations of Crystallography.* 1990.
27. Crowther, R.A., *The Molecular Replacement Method, edited by MG Rossmann.* New York: Gordon & Breach, 1972: p. 173-178.
28. Scapin, G., *Molecular replacement then and now.* Acta crystallographica. Section D, Biological crystallography, 2013. **69**: p. 2266-75.
29. Urzhumtsev, A. and Podjarny, A., *On the solution of the molecular-replacement problem at very low resolution: application to large complexes.* Acta Crystallogr D Biol Crystallogr, 1995. **51**(Pt 6): p. 888-95.
30. Chothia, C. and Lesk, A.M., *The relation between the divergence of sequence and structure in proteins.* EMBO J, 1986. **5**(4): p. 823-6.
31. Illergard, K., Ardell, D.H., and Elofsson, A., *Structure is three to ten times more conserved than sequence--a study of structural response in protein cores.* Proteins, 2009. **77**(3): p. 499-508.
32. Mao, B., Guan, R., and Montelione, G.T., *Improved technologies now routinely provide protein NMR structures useful for molecular replacement.* Structure, 2011. **19**(6): p. 757-66.
33. Hong, X. and Hao, Q., *Combining solution wide-angle X-ray scattering and crystallography: determination of molecular envelope and heavy-atom sites.* J Appl Crystallogr, 2009. **42**(Pt 2): p. 259-264.
34. Xiong, Y., *From electron microscopy to X-ray crystallography: molecular-replacement case studies.* Acta Crystallogr D Biol Crystallogr, 2008. **64**(Pt 1): p. 76-82.
35. Amunts, A., Brown, A., Bai, X.-c., Llácer, J.L., Hussain, T., Emsley, P., Long, F., Murshudov, G., Scheres, S.H.W., and Ramakrishnan, V., *Structure of the yeast mitochondrial large ribosomal subunit.* Science (New York, N.Y.), 2014. **343**: p. 1485-1489.
36. Qian, B., Raman, S., Das, R., Bradley, P., McCoy, A.J., Read, R.J., and Baker, D., *High-resolution structure prediction and the crystallographic phase problem.* Nature, 2007. **450**(7167): p. 259-64.
37. DiMaio, F., Terwilliger, T.C., Read, R.J., Wlodawer, A., Oberdorfer, G., Wagner, U., Valkov, E., Alon, A., Fass, D., Axelrod, H.L., Das, D., Vorobiev, S.M., Iwaï, H., Pokkuluri, P.R., and Baker, D., *Improved molecular replacement by density- and energy-guided protein structure optimization.* Nature, 2011. **473**: p. 540-543.
38. Bibby, J., Keegan, R.M., Mayans, O., Winn, M.D., and Rigden, D.J., *AMPLE: A cluster-and-truncate approach to solve the crystal structures of small proteins*

using rapidly computed ab initio models. Acta Crystallographica Section D: Biological Crystallography, 2012. **68**: p. 1622-1631.

39. Storoni, L.C., McCoy, A.J., and Read, R.J., *Likelihood-enhanced fast rotation functions.* Acta Crystallographica Section D: Biological Crystallography, 2004. **60**: p. 432-438.

40. McCoy, A.J., Grosse-Kunstleve, R.W., Storoni, L.C., and Read, R.J., *Likelihood-enhanced fast translation functions.* Acta crystallographica. Section D, Biological crystallography, 2005. **61**: p. 458-64.

41. Wang, B.C., *Resolution of phase ambiguity in macromolecular crystallography.* Methods Enzymol, 1985. **115**: p. 90-112.

42. Zhang, K.Y.J. and Main, P., *Histogram matching as a new density modification technique for phase refinement and extension of protein molecules.* Acta Crystallographica Section A: Foundations of Crystallography, 1990. **46**(1): p. 41-46.

43. Cowtan, K. and Main, P., *Miscellaneous algorithms for density modification.* Acta Crystallogr D Biol Crystallogr, 1998. **54**(Pt 4): p. 487-93.

44. Terwilliger, T.C., *Maximum-likelihood density modification.* Acta Crystallogr D Biol Crystallogr, 2000. **56**(Pt 8): p. 965-72.

45. Sheldrick, G.M., *Experimental phasing with SHELXC/D/E: combining chain tracing with density modification.* Acta Crystallogr D Biol Crystallogr, 2010. **66**(Pt 4): p. 479-85.

46. Thorn, A. and Sheldrick, G.M., *Extending molecular-replacement solutions with SHELXE.* Acta crystallographica. Section D, Biological crystallography, 2013. **69**: p. 2251-6.

47. Read, R., *Improved Fourier coefficients for maps using phases from partial structures with errors.* Acta Crystallographica Section A, 1986. **42**(3): p. 140-149.

48. Usón, I. and Sheldrick, G.M., *Advances in direct methods for protein crystallography.* Current opinion in structural biology, 1999. **9**: p. 643-648.

49. Miller, R., DeTitta, G.T., Jones, R., Langs, D.A., Weeks, C.M., and Hauptman, H.A., *On the application of the minimal principle to solve unknown structures.* Science (New York, N.Y.), 1993. **259**: p. 1430-1433.

50. Millán, C., Sammito, M., Garcia-Ferrer, I., Goulas, T., Sheldrick, G.M., and Usón, I., *Combining phase information in reciprocal space for ARCIMBOLDO.* Acta Crystallogr D Biol Crystallogr, 2015. **Submitted and recommended for publication**.

51. Caliandro, R. and Carrozzini, B., *Phasing at resolution higher than the experimental resolution.* Acta Crystallogr D Biol Crystallogr, 2005. **61**: p. 556-565.

52. Caliandro, R., Carrozzini, B., Cascarano, G.L., De Caro, L., Giacovazzo, C., and Siliqi, D., *Ab initio phasing at resolution higher than experimental resolution.* Acta Crystallogr D Biol Crystallogr, 2005. **61**(Pt 8): p. 1080-7.

53. Lunin, V.Y., Urzhumtsev, A.G., and Podjarny, A., *Ab initio phasing of low-resolution Fourier syntheses*, in *International Tables for Crystallography*, A. E., H. D.M., and R. M.G., Editors. 2011, Wiley & Sons: Chichester. p. 437-442.

54. Saunders, R. and Deane, C.M., *Synonymous codon usage influences the local protein structure observed.* Nucleic Acids Res, 2010. **38**(19): p. 6719-28.

55. Murzin, A.G., Brenner, S.E., Hubbard, T., and Chothia, C., *SCOP: a structural classification of proteins database for the investigation of sequences and structures.* J Mol Biol, 1995. **247**(4): p. 536-40.

56. Cuff, A.L., Sillitoe, I., Lewis, T., Clegg, A.B., Rentzsch, R., Furnham, N., Pellegrini-Calace, M., Jones, D., Thornton, J., and Orengo, C.A., *Extending CATH:*

increasing coverage of the protein structure universe and linking structure with function. Nucleic Acids Res, 2011. **39**(Database issue): p. D420-6.

57. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E., *The Protein Data Bank.* Nucleic Acids Res, 2000. **28**(1): p. 235-42.
58. Golovin, A. and Henrick, K., *MSDmotif: exploring protein sites and motifs.* BMC Bioinformatics, 2008. **9**: p. 312.
59. Holm, L. and Rosenstrom, P., *Dali server: conservation mapping in 3D.* Nucleic Acids Res, 2010. **38**(Web Server issue): p. W545-9.
60. Cowtan, K., *Fast Fourier feature recognition.* Acta Crystallogr D Biol Crystallogr, 2001. **57**(Pt 10): p. 1435-44.
61. Oldfield, T.J., *Creating structure features by data mining the PDB to use as molecular-replacement models.* Acta Crystallogr D Biol Crystallogr, 2001. **57**(Pt 10): p. 1421-7.
62. Nicholls, R.A., Long, F., and Murshudov, G.N., *Low-resolution refinement tools in REFMAC5.* Acta Crystallogr D Biol Crystallogr, 2012. **68**(Pt 4): p. 404-17.
63. Holm, L. and Sander, C., *Mapping the protein universe.* Science, 1996. **273**(5275): p. 595-603.
64. Simons, K.T., Ruczinski, I., Kooperberg, C., Fox, B.A., Bystroff, C., and Baker, D., *Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins.* Proteins, 1999. **34**(1): p. 82-95.
65. Cortes, C. and Vapnik, V., *Support-vector networks.* Machine learning, 1995. **20**(3): p. 273-297.
66. Yadav, A. and Jayaraman, V.K., *Structure based function prediction of proteins using fragment library frequency vectors.* Bioinformation, 2012. **8**(19): p. 953.
67. Dikaiakos, M.D., Katsaros, D., Mehra, P., Pallis, G., and Vakali, A., *Cloud computing: Distributed internet computing for IT and scientific research.* Internet Computing, IEEE, 2009. **13**(5): p. 10-13.
68. Allen, F., Almasi, G., Andreoni, W., Beece, D., Berne, B.J., Bright, A., Brunheroto, J., Cascaval, C., Castanos, J., and Coteus, P., *Blue Gene: a vision for protein science using a petaflop supercomputer.* IBM systems journal, 2001. **40**(2): p. 310-327.
69. Saha, A., *Parallel programming in C and Python.* Linux Journal, 2012. **2012**(217): p. 4.
70. Matloff, N. and Hsu, F., *Tutorial on threads programming with Python.* University of California, 2007.
71. Bhaniramka, P., Robert, P.C.D., and Eilemann, S. *OpenGL Multipipe SDK: A toolkit for scalable parallel rendering*. 2005. IEEE.
72. Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., and Volkov, V., *Parallel computing experiences with CUDA*. IEEE micro, 2008(4): p. 13-27.
73. Tannenbaum, T., *Beowulf Cluster Computing with Linux*. 2002.
74. Gentzsch, W. *Sun grid engine: Towards creating a compute power grid.* 2001. IEEE.
75. Adaptive, C. and Green, C. *Torque resource manager (http://www. adaptivecomputing. com)*. 2012.
76. Lynch, P., *The origins of computer weather prediction and climate modeling.* Journal of Computational Physics, 2008. **227**(7): p. 3431-3444.

77. Kehtarnavaz, N. and Gamadia, M., *Real-time image and video processing: from research to reality.* Synthesis Lectures on Image, Video & Multimedia Processing, 2006. **2**(1): p. 1-108.
78. Shaw, D.E., Maragakis, P., Lindorff-Larsen, K., Piana, S., Dror, R.O., Eastwood, M.P., Bank, J.A., Jumper, J.M., Salmon, J.K., Shan, Y., and Wriggers, W., *Atomic-level characterization of the structural dynamics of proteins.* Science, 2010. **330**(6002): p. 341-6.
79. Gropp, W., Lusk, E., Doss, N., and Skjellum, A., *A high-performance, portable implementation of the MPI message passing interface standard.* Parallel computing, 1996. **22**(6): p. 789-828.
80. Nilsen, J.K., Cai, X., Høyland, B., and Langtangen, H.P., *Simplifying the parallelization of scientific codes by a function-centric approach in Python.* Computational Science & Discovery, 2010. **3**(1): p. 015003.
81. Winn, M.D., Ballard, C.C., Cowtan, K.D., Dodson, E.J., Emsley, P., Evans, P.R., Keegan, R.M., Krissinel, E.B., Leslie, A.G., McCoy, A., McNicholas, S.J., Murshudov, G.N., Pannu, N.S., Potterton, E.A., Powell, H.R., Read, R.J., Vagin, A., and Wilson, K.S., *Overview of the CCP4 suite and current developments.* Acta Crystallogr D Biol Crystallogr, 2011. **67**(Pt 4): p. 235-42.
82. Adams, P.D., Afonine, P.V., Bunkóczi, G., Chen, V.B., Davis, I.W., Echols, N., Headd, J.J., Hung, L.W., Kapral, G.J., and Grosse-Kunstleve, R.W., *PHENIX: a comprehensive Python-based system for macromolecular structure solution.* Acta Crystallographica Section D: Biological Crystallography, 2010. **66**(2): p. 213-221.
83. StataCorp. *Stata Statistical Software: Release 12. College Station, TX: StataCorp LP.* 2011.
84. Sheldrick, G.M., Gilmore, C.J., Hauptman, H.A., Weeks, C.M., Miller, R., and Usón, I., *International Tables for Crystallography*, ed. E. Arnold, D.M. Himmel, and M.G. Rossmann. 2011, Chichester: Wiley. 413-429.
85. Hauptman, H.A. and Karle, J., *ACA Monograph No. 3*. 1953, Ohio: Polycrystal Book Service.
86. Karle, J. and Hauptman, H., *A THEORY OF PHASE DETERMINATION FOR THE 4 TYPES OF NON-CENTROSYMMETRIC SPACE GROUPS 1P222, 2P22, 3P12, 3P22.* Acta Crystallographica, 1956. **9**(7): p. 635-651.
87. Evans, P., *Scaling and assessment of data quality.* Acta Crystallogr D Biol Crystallogr, 2006. **62**(Pt 1): p. 72-82.
88. Evans, P.R. and Murshudov, G.N., *How good are my data and what is the resolution?* Acta Crystallographica Section D Biological Crystallography, 2013. **69**: p. 1204-1214.
89. Shoemake, K., *Euler Angle Conversions*, in *Graphics Gems IV*. 1994. p. 222-229.
90. Diebel, J., *Representing attitude: Euler angles, unit quaternions, and rotation vectors.* Matrix, 2006. **58**: p. 1-35.
91. Crowther, R.A., *The Fast Rotation Function*, in *The Molecular Replacement Method*, M.G. Rossmann, Editor. 1972, Gordon and Breach: New York.
92. Christopher, M.B., *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006: Springer-Verlag New York, Inc.
93. Mardia, K.V., Kent, J.T., and Bibby, J.M., *Multivariate Analysis (Probability and Mathematical Statistics)*. 1980: Academic Press. 521.
94. Sheldrick, G.M. and Gould, R.O., *Structure solution by iterative peaklist optimization and tangent expansion in space group P1*, in *Acta Crystallographica Section B Structural Science*. 1995. p. 423-431.

95. Read, R.J., Adams, P.D., and McCoy, A.J., *Intensity statistics in the presence of translational noncrystallographic symmetry.* Acta Crystallographica Section D: Biological Crystallography, 2013. **69**: p. 176-183.
96. Sliwiak, J., Jaskolski, M., Dauter, Z., McCoy, A.J., and Read, R.J., *Likelihood-based molecular-replacement solution for a highly pathological crystal with tetartohedral twinning and sevenfold translational noncrystallographic symmetry.* Acta Crystallographica Section D: Biological Crystallography, 2014. **70**: p. 471-480.
97. Shiono, M. and Woolfson, M.M., *Direct-space methods in phase extension and phase determination. I. low-density elimination.* Acta Crystallographica Section A: Foundations of Crystallography, 1992. **48**: p. 451-456.
98. McCoy, A.J. and Read, R.J. http://www.phaser.cimr.cam.ac.uk/index.php/Keywords. 2014.
99. Lunin, V.Y. and Woolfson, M.M., *Mean phase error and the map-correlation coefficient.* Acta Crystallographica Section D: Biological Crystallography, 1993. **49**(6): p. 530-533.
100. Bieniossek, C., Schütz, P., Bumann, M., Limacher, A., Uson, I., and Baumann, U., *The Crystal Structure of the Carboxy-Terminal Domain of Human Translation Initiation Factor eIF5.* Journal of Molecular Biology, 2006. **360**: p. 457-465.
101. Hissa, D.C., Vasconcelos, I.M., Carvalho, A.F.U., Nogueira, V.L.R., Cascon, P., Antunes, A.S.L., de Macedo, G.R., and Melo, V.M.M., *Novel surfactant proteins are involved in the structure and stability of foam nests from the frog Leptodactylus vastus.* The Journal of experimental biology, 2008. **211**: p. 2707-2711.
102. Keegan, R.M., Bibby, J., Thomas, J., Xu, D., Zhang, Y., Mayans, O., Winn, M.D., and Rigden, D.J., *Exploring the speed and performance of molecular replacement with AMPLE using QUARK ab initio protein models.* Acta Crystallographica Section D Biological Crystallography, 2015. **71**: p. 338-343.
103. Anderson, D.H., Sawaya, M.R., Cascio, D., Ernst, W., Modlin, R., Krensky, A., and Eisenberg, D., *Granulysin crystal structure and a structure-derived lytic mechanism.* J Mol Biol, 2003. **325**(2): p. 355-65.
104. Krivov, G.G., Shapovalov, M.V., and Dunbrack, R.L., *Improved prediction of protein side-chain conformations with SCWRL4.* Proteins, 2009. **77**: p. 778-795.
105. Seitz, T., Thoma, R., Schoch, G.A., Stihle, M., Benz, J., D'Arcy, B., Wiget, A., Ruf, A., Hennig, M., and Sterner, R., *Enhancing the stability and solubility of the glucocorticoid receptor ligand-binding domain by high-throughput library screening.* J Mol Biol, 2010. **403**(4): p. 562-77.
106. Newton, R., *Molecular mechanisms of glucocorticoid action: what is important?* Thorax, 2000. **55**(7): p. 603-13.
107. Bledsoe, R.K., Montana, V.G., Stanley, T.B., Delves, C.J., Apolito, C.J., McKee, D.D., Consler, T.G., Parks, D.J., Stewart, E.L., Willson, T.M., Lambert, M.H., Moore, J.T., Pearce, K.H., and Xu, H.E., *Crystal structure of the glucocorticoid receptor ligand binding domain reveals a novel mode of receptor dimerization and coactivator recognition.* Cell, 2002. **110**(1): p. 93-105.
108. Kobe, B. and Kajava, A.V., *When protein folding is simplified to protein coiling: the continuum of solenoid protein structures.* Trends Biochem Sci, 2000. **25**(10): p. 509-15.
109. Stroud, J.C., Liu, C., Teng, P.K., and Eisenberg, D., *Toxic fibrillar oligomers of amyloid-beta have cross-beta structure.* Proc Natl Acad Sci U S A, 2012. **109**(20): p. 7717-22.
110. Rost, B., *Twilight zone of protein sequence alignments.* Protein Eng, 1999. **12**(2): p. 85-94.

111. Abergel, C., *Molecular replacement: tricks and treats.* Acta Crystallogr D Biol Crystallogr, 2013. **69**(Pt 11): p. 2167-73.
112. Bunkoczi, G. and Read, R.J., *Improvement of molecular-replacement models with Sculptor.* Acta Crystallogr D Biol Crystallogr, 2011. **67**(Pt 4): p. 303-12.
113. Gruene, T., *mrtailor: a tool for PDB-file preparation for the generation of external restraints.* Acta Crystallogr D Biol Crystallogr, 2013. **69**(Pt 9): p. 1861-3.
114. McCoy, A.J., Nicholls, R.a., and Schneider, T.R., *SCEDS: protein fragments for molecular replacement in Phaser.* Acta Crystallographica Section D Biological Crystallography, 2013. **69**: p. 2216-2225.
115. Bunkoczi, G., Echols, N., McCoy, A.J., Oeffner, R.D., Adams, P.D., and Read, R.J., *Phaser.MRage: automated molecular replacement.* Acta Crystallogr D Biol Crystallogr, 2013. **69**(Pt 11): p. 2276-86.
116. Bibby, J., Keegan, R.M., Mayans, O., Winn, M.D., and Rigden, D.J., *Application of the AMPLE cluster-and-truncate approach to NMR structures for molecular replacement.* Acta Crystallographica Section D Biological Crystallography, 2013. **69**: p. 2194-2201.
117. Artola-Recolons, C., Llarrull, L.I., Lastochkin, E., Mobashery, S., and Hermoso, J.A., *Crystallization and preliminary X-ray diffraction analysis of the lytic transglycosylase MltE from Escherichia coli.* Acta Crystallogr Sect F Struct Biol Cryst Commun, 2011. **67**(Pt 1): p. 161-3.
118. Fibriansah, G., Gliubich, F.I., and Thunnissen, A.M., *On the mechanism of peptidoglycan binding and cleavage by the endo-specific lytic transglycosylase MltE from Escherichia coli.* Biochemistry, 2012. **51**(45): p. 9164-77.
119. Kraft, A.R., Templin, M.F., and Holtje, J.V., *Membrane-bound lytic endotransglycosylase in Escherichia coli.* J Bacteriol, 1998. **180**(13): p. 3441-7.
120. Artola-Recolons, C., Carrasco-Lopez, C., Llarrull, L.I., Kumarasiri, M., Lastochkin, E., Martinez de Ilarduya, I., Meindl, K., Uson, I., Mobashery, S., and Hermoso, J.A., *High-resolution crystal structure of MltE, an outer membrane-anchored endolytic peptidoglycan lytic transglycosylase from Escherichia coli.* Biochemistry, 2011. **50**(13): p. 2384-6.
121. van Asselt, E.J., Thunnissen, A.M., and Dijkstra, B.W., *High resolution crystal structures of the Escherichia coli lytic transglycosylase Slt70 and its complex with a peptidoglycan fragment.* J Mol Biol, 1999. **291**(4): p. 877-98.
122. Larkin, M.A., Blackshields, G., Brown, N.P., Chenna, R., McGettigan, P.A., McWilliam, H., Valentin, F., Wallace, I.M., Wilm, A., Lopez, R., Thompson, J.D., Gibson, T.J., and Higgins, D.G., *Clustal W and Clustal X version 2.0.* Bioinformatics, 2007. **23**(21): p. 2947-8.
123. Fujinaga, M. and Read, R.J., *Experiences with a new translation-function program.* Journal of Applied Crystallography, 1987. **20**: p. 517-521.
124. Soding, J., Biegert, A., and Lupas, A.N., *The HHpred interactive server for protein homology detection and structure prediction.* Nucleic Acids Res, 2005. **33**(Web Server issue): p. W244-8.
125. Joosten, R.P., te Beek, T.A., Krieger, E., Hekkelman, M.L., Hooft, R.W., Schneider, R., Sander, C., and Vriend, G., *A series of PDB related databases for everyday needs.* Nucleic Acids Res, 2011. **39**(Database issue): p. D411-9.
126. Cowtan, K., *Completion of autobuilt protein models using a database of protein fragments.* Acta Crystallogr D Biol Crystallogr, 2012. **68**(Pt 4): p. 328-35.
127. Kabsch, W. and Sander, C., *Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features.* Biopolymers, 1983. **22**(12): p. 2577-637.

128. Kabsch, W. and Sander, C., *DSSP: definition of secondary structure of proteins given a set of 3D coordinates.* Biopolymers, 1983. **22**: p. 2577-2637.
129. Doornik, J.A. and Hansen, H., *An omnibus test for univariate and multivariate normality*.* Oxford Bulletin of Economics and Statistics, 2008. **70**(s1): p. 927-939.
130. Kruskal, W.H. and Wallis, W.A., *Use of ranks in one-criterion variance analysis.* Journal of the American statistical Association, 1952. **47**(260): p. 583-621.
131. Pavelcik, F. and Pavelcikova, P., *Conformation families of protein fragments in multidimensional torsion-angle space.* Acta Crystallographica Section D: Biological Crystallography, 2007. **63**: p. 1162-1168.
132. Milbradt, A.G., Kerek, F., Moroder, L., and Renner, C., *Structural characterization of hellethionins from Helleborus purpurascens.* Biochemistry, 2003. **42**(8): p. 2404-11.
133. West, D.B., *Introduction to graph theory.* 2ed ed. 2001, Upper Saddle River: Prentice Hall.
134. Read, R.J. and Chavali, G., *Assessment of CASP7 predictions in the high accuracy template-based modeling category.* Proteins, 2007. **69 Suppl 8**: p. 27-37.
135. Zhang, Y. and Skolnick, J., *TM-align: a protein structure alignment algorithm based on the TM-score.* Nucleic Acids Res, 2005. **33**(7): p. 2302-9.
136. Golub, G.H. and Loan, C.F.v., *Matrix Computations.* 3rd ed. 1996, Baltimore and London: Johns Hopkins University Press. 728.
137. Golub, G.H. and Reinsch, C., *Singular value decomposition and least squares solutions*, in *Handbook for Automatic computation, 2, (Linear Algebra).* 1971, Springer-Verlag: New York. p. 134-151.
138. Cock, P.J.A., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M.J.L., *Biopython: freely available Python tools for computational molecular biology and bioinformatics.* Bioinformatics, 2009.
139. Theobald, D.L. and Wuttke, D.S., *Accurate structural correlations from maximum likelihood superpositions.* PLoS Comput Biol, 2008. **4**(2): p. e43.
140. Smith, T.F. and Waterman, M.S., *Identification of common molecular subsequences.* J Mol Biol, 1981. **147**(1): p. 195-7.
141. Krissinel, E. and Henrick, K., *Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions.* Acta Crystallogr D Biol Crystallogr, 2004. **60**(Pt 12 Pt 1): p. 2256-68.
142. Emsley, P., Lohkamp, B., Scott, W.G., and Cowtan, K., *Features and development of Coot.* Acta Crystallogr D Biol Crystallogr, 2010. **66**(Pt 4): p. 486-501.
143. Schrödinger, L., *The {PyMOL} Molecular Graphics System, Version~1.3r1.* 2010.
144. Nilges, M., Clore, G.M., and Gronenborn, A.M., *A simple method for delineating well-defined and variable regions in protein structures determined from interproton distance data.* FEBS Letters, 1987. **219**(1): p. 11-16.
145. Papaleo, E., Mereghetti, P., Fantucci, P., Grandori, R., and De Gioia, L., *Free-energy landscape, principal component analysis, and structural clustering to identify representative conformations from molecular dynamics simulations: the myoglobin case.* J Mol Graph Model, 2009. **27**(8): p. 889-99.
146. Ortiz, A.R., Strauss, C.E., and Olmea, O., *MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison.* Protein Sci, 2002. **11**(11): p. 2606-21.
147. de Hoon, M.J., Imoto, S., Nolan, J., and Miyano, S., *Open source clustering software.* Bioinformatics, 2004. **20**(9): p. 1453-4.

148. Shortle, D., Simons, K.T., and Baker, D., *Clustering of low-energy conformations near the native structures of small proteins.* Proc Natl Acad Sci U S A, 1998. **95**(19): p. 11158-62.
149. Ginalski, K., Elofsson, A., Fischer, D., and Rychlewski, L., *3D-Jury: a simple approach to improve protein structure predictions.* Bioinformatics, 2003. **19**(8): p. 1015-8.
150. Kaufmann, K.W., Lemmon, G.H., Deluca, S.L., Sheehan, J.H., and Meiler, J., *Practically useful: what the Rosetta protein modeling suite can do for you.* Biochemistry, 2010. **49**(14): p. 2987-98.
151. Li, S.C., Bu, D., and Li, M., *Clustering 100,000 protein structure decoys in minutes.* IEEE/ACM Trans Comput Biol Bioinform, 2012. **9**(3): p. 765-73.
152. Rawat, N. and Biswas, P., *Shape, flexibility and packing of proteins and nucleic acids in complexes.* Phys Chem Chem Phys, 2011. **13**(20): p. 9632-43.
153. Vymetal, J. and Vondrasek, J., *Gyration- and inertia-tensor-based collective coordinates for metadynamics. Application on the conformational behavior of polyalanine peptides and Trp-cage folding.* J Phys Chem A, 2011. **115**(41): p. 11455-65.
154. Luscombe, N.M., Austin, S.E., Berman, H.M., and Thornton, J.M., *An overview of the structures of protein-DNA complexes.* Genome Biol, 2000. **1**(1): p. REVIEWS001.
155. Pröpper, K., Meindl, K., Sammito, M., Dittrich, B., Sheldrick, G.M., Pohl, E., and Usón, I., *Structure solution of DNA-binding proteins and complexes with ARCIMBOLDO libraries.* Acta crystallographica. Section D, Biological crystallography, 2014. **70**: p. 1743-1757.
156. Huffman, J.L. and Brennan, R.G., *Prokaryotic transcription regulators: more than just the helix-turn-helix motif.* Curr Opin Struct Biol, 2002. **12**(1): p. 98-106.
157. Burley, S.K., *The TATA box binding protein.* Curr Opin Struct Biol, 1996. **6**(1): p. 69-75.
158. Schuetz, A., Nana, D., Rose, C., Zocher, G., Milanovic, M., Koenigsmann, J., Blasig, R., Heinemann, U., and Carstanjen, D., *The structure of the Klf4 DNA-binding domain links to self-renewal and macrophage differentiation.* Cell Mol Life Sci, 2011. **68**(18): p. 3121-31.
159. Houbaviy, H.B., Usheva, A., Shenk, T., and Burley, S.K., *Cocrystal structure of YY1 bound to the adeno-associated virus P5 initiator.* Proc Natl Acad Sci U S A, 1996. **93**(24): p. 13577-82.
160. Buehler, A., Urzhumtseva, L., Lunin, V.Y., and Urzhumtsev, A., *Cluster analysis for phasing with molecular replacement: a feasibility study.* Acta Crystallogr D Biol Crystallogr, 2009. **65**(Pt 7): p. 644-50.
161. DeLano, W.L. and Brunger, A.T., *The direct rotation function: Patterson correlation search applied to molecular replacement.* Acta Crystallogr D Biol Crystallogr, 1995. **51**(Pt 5): p. 740-8.
162. Ibuki, T., Imada, K., Minamino, T., Kato, T., Miyata, T., and Namba, K., *Common architecture of the flagellar type III protein export apparatus and F- and V-type ATPases.* Nat Struct Mol Biol, 2011. **18**(3): p. 277-82.
163. Royce, W.W. *Managing the development of large software systems.* 1970. Los Angeles.
164. Larman, C. and Basili, V.R., *Iterative and incremental development: A brief history.* Computer, 2003. **36**(6): p. 47-56.
165. Beck, K., *Embracing change with extreme programming.* Computer, 1999. **32**(10): p. 70-77.

166. Kindler, E. and Krivy, I., *Object-oriented simulation of systems with sophisticated control.* International Journal of General Systems, 2011. **40**(3): p. 313-343.
167. Leiserson, C.E., Rivest, R.L., Stein, C., and Cormen, T.H., *Introduction to algorithms*. 2001: MIT press.
168. Battye, T.G.G., Kontogiannis, L., Johnson, O., Powell, H.R., and Leslie, A.G.W., *iMOSFLM: a new graphical interface for diffraction-image processing with MOSFLM.* Acta Crystallographica Section D: Biological Crystallography, 2011. **67**(4): p. 271-281.
169. Tibshirani, R., Walther, G., and Hastie, T., *Estimating the number of clusters in a data set via the gap statistic.* Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2001. **63**(2): p. 411-423.

APPENDICES

1 SCIENTIFIC PRODUCTION

Book chapter:

1. I. Usón, C. Millán, M. Sammito, K. Meindl, I. M. de Ilarduya, I. De Marino, D. D. Rodríguez, "Phasing Through Location of Small Fragments and Density Modification with ARCIMBOLDO", In: Advancing Methods for Biomolecular Crystallography, R. Read, A. G. Urzhumtsev, V. Y. Lunin (Eds.), Springer, Dordrecht, The Netherlands, (2013) pp. 123-132.

Scientific Articles:

2. M. Sammito, C. Millán, D. Frieske, E. Rodríguez-Freirea, R. J. Borges, I. Usón. ARCIMBOLDO_LITE: single workstation implementation and use. Acta Cryst D. (2015) Submitted
3. C. Millán, M. Sammito, I. Garcia-Ferrer, T. Goulas, G. M. Sheldrick and I. Usón. Combining phase information in reciprocal space for ARCIMBOLDO. Acta Cryst D. (2015) Submitted and recommended for publication.
4. C. Millán, M. Sammito, I. Usón. Macromolecular ab initio phasing enforcing secondary and tertiary structure. IUCrJ (2015) 2 p. 95-105.
5. G. A. Schoch, M. Sammito, C. Millán, I. Usón, M.G. Rudolph. Structure of a 13-fold superhelix (almost) determined from first principles. IUCrJ (2015) 2: p. 177-187.
6. M. Sammito, K. Meindl, I. M. de Ilarduya, C. Millán, C. Artola-Recolons, J. A. Hermoso, I. Usón. Structure solution with ARCIMBOLDO using fragments derived from distant homology models. FEBS J. (2014), 281 p. 4029-45.
7. K. Pröpper, K. Meindl, M.Sammito, B. Dittrich, G.M. Sheldrick, E. Pohl, I. Usón. Acta Cryst D. (2014), 70 p. 1743-57
8. M. Sammito, C. Millán, D.D. Rodríguez, I. M. de Ilarduya, K. Meindl, I. De Marino, G. Petrillo, R. M. Buey, J. M. de Pereda, K. Zeth, G. M. Sheldrick and I. Usón. “Exploiting tertiary structure through local folds for crystallographic phasing”. Nature Methods (2013) 10 p. 1099- 101.

- Selected for the Special Nature Milestone in Crystallography in 2014 http://www.nature.com/milestones/milecrystal/library/structural-biology/index.html

9. D.D. Rodríguez, M. Sammito, K. Meindl, I. M. de Ilarduya, M. Potratz, G. M. Sheldrick and Isabel Usón, "Practical structure solution with ARCIMBOLDO", Acta Crys. D. 68 336-43

Teaching:
Reader at Molecular Biotechnology Master's Degree, University of Barcelona: I prepared and held Structural Bioinformatics course, for a total of 21h with lectures given both in Spanish and English and practical tutorials on homology modelling, structural prediction and computational crystallography.

2 POSTERS AND TALKS

- C.L. Millán, M. Sammito, D.D. Rodríguez, K. Meindl, I. M. de Ilarduya, G.M. Sheldrick, I. Usón. "Clustering for Arcimboldo". ZCAM-Daresbury Collaborative Tutorial Zaragoza, (ES)

- C.L. Millán, M.Sammito, K. Meindl, I. Martínez de Ilarduya. "Reciprocal space clustering of BORGES-ARCIMBOLDO partial solutions: Practical cases" ECM28 (28th European Crystallography Meeting) ▪ Awarded with the IUCr Biology Poster Prize

- G. Schoch, M. Sammito, C. Millán, I. Usón and M. Rudolph "Structure of a thirteen-fold superhelix (almost) determined from first principles" The Biophysical Society Annual Meeting

- C.L. Millán, M. Sammito, D.D. Rodríguez, K. Meindl, I. M. de Ilarduya, G.M. Sheldrick, I. Usón. "Clustering fragments for ARCIMBOLDO phasing". International School of Crystallography 45th Course Present and Future Methods for Biomolecular Crystallography, Erice (IT)

- M. Sammito, C.L. Millán, D.D. Rodríguez, K. Meindl, I. M. de Ilarduya, G.M. Sheldrick, I. Usón. "ARCIMBOLDO structure solution with customised and clusterised fragment libraries from Borges". International School of Crystallography 45th Course Present and Future Methods for Biomolecular Crystallography and BioCrys2012 School, Erice (IT)

- M. Sammito, D.D. Rodríguez, K. Meindl, I. M. de Ilarduya, I. Usón. "BORGES a tool to generate customised, secondary structure libraries for phasing", XXII Congress and General Assembly of International Union of Crystallography, Madrid (ES)

- D.D. Rodríguez, M. Sammito, I. M. de Ilarduya, K. Meindl, I. Usón. "New features in ARCIMBOLDO: a tutorial", XXII Congress and General Assembly of International Union of Crystallography, Madrid (ES)

- Usón, D.D. Rodríguez, M. Sammito, K. Meindl, I. M. de Ilarduya. "ARCIMBOLDO goes super: ab Initio phasing on the supercomputer Calendula FCSCL ", XXII Congress and General Assembly of International Union of Crystallography, Madrid (ES)

- K. Propper, K. Meindl, D.D Rodríguez, M. Sammito, B. Dittrich, G.M. Sheldrick, E. Pohl, I. Usón. "DNA - Protein Complex Structure Prediction",Acta Cryst A. A68, s121

3 SCHOOLS AND MEETINGS

- Attendance to IRB Barcelona BioMed Conference on "Transporters and other Molecular Machines" (Barcelona, 2014)
- Invited Speaker at BAC2014 (Biotech annual Congress 2014), (Barcelona, 2014)
- Attendance to The Campus Gutenberg, (Barcelona, 2014)
- Attendance to MCS2014. Macromolecular Crystallography School, (Madrid, 2014)
- Attendance and grant awarded at Introduction to Software Development for Crystallographers (Warwick, 2013)
- Invited Speaker at ECM28 (28th European Crystallography Meeting) (Warwick, 2013)
- Attendance with poster presentation and grant awarded at the school BioCrys2012: Fundamental of Modern Methods of BioCrystallography (FEBS) (Oeiras, 2012)
- Attendance with poster presentation and grant awarded at 45th Course of the International School of Crystallography "Present and Future Methods for Biomolecular Crystallography" (Erice, Sicily, 2012)
- Attendance with poster presentation and grant awarded at the XXII Congress and General Assembly of International Union of Crystallography (Madrid, 2011)
- Attendance with grant awarded at the course Bologna Winter School 2010: Computational Methods for System Biology (Bologna, 2010)