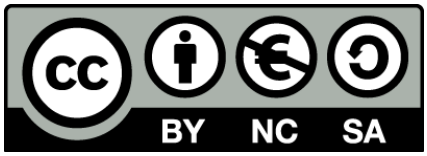
UNIVERSITAT DE BARCELONA

Algorithmic and Technical Improvements for Next Generation Drug Design Software Tools

Víctor A. Gil Sepúlveda



Aquesta tesi doctoral està subjecta a la llicència ***Reconeixement- NoComercial – CompartirIgual 4.0. Espanya de Creative Commons***.

Esta tesis doctoral está sujeta a la licencia ***Reconocimiento - NoComercial – CompartirIgual 4.0. España de Creative*** **Commons**.

This doctoral thesis is licensed under the ***Creative Commons Attribution-NonCommercial-ShareAlike 4.0. Spain License***.

Universitat de Barcelona

Facultat de Farmàcia

Algorithmic and Technical Improvements for Next Generation Drug Design Software Tools

VÍCTOR A. GIL SEPÚLVEDA, 2016

Universitat de Barcelona

Facultat de Farmàcia

Programa de Doctorat en Biomedicina

Algorithmic and Technical Improvements for Next Generation Drug Design Software Tools

Memòria presentada per Víctor A. Gil Sepúlveda per
optar al títol de doctor per la Universitat de Barcelona

Autor:
Víctor A. GIL

Tutor:
Josep Ll. GELPÍ

Director:
Víctor GUALLAR

A mi madre, mi padre y
mi hermano
A Marghe

Agradecimientos

El doctorado es un largo viaje. Un viaje de descubrimiento científico, y también de descubrimiento personal. Y, en contra de lo que algunos puedan pensar, no es un viaje nada fácil. Afortunadamente, a lo largo de su periplo, el viajero encuentra personas que le guían, le abren caminos y le dan apoyo. Quisiera dedicar estas líneas a esas personas.

Me gustaría empezar agradeciendo al Dr. Víctor Guallar, mi supervisor, el haberme dado la oportunidad sin la cual nada de esto hubiera sido posible. Mi más sincera gratitud a las demás personas que me han permitido conocer un poco más el mundo y como se trabaja fuera de España, especialmente: al Dr. Juan Cortés, del LAAS-CNRS, por haberme acogido en mi primera estancia en el extranjero; al Dr. Scott Callaghan, uno de los organizadores del programa *HPC summer school*, al cual le debo el haber "cruzado el charco" por primera vez; y al Dr. Anders Hogner por acogerme en su grupo de investigación en AstraZeneca y haberme hecho sentir como si fuera uno más del equipo.

Todo viaje largo se disfruta más en buena compañía. Considero haber tenido mucha suerte en ese aspecto, tanto en el terreno personal como en el laboral. Durante todos estos años, he podido gozar de la compañia de mis colegas del Barcelona Supercomputing Center. Con ellos he compartido reflexiones (a veces científicas), alegrías y penas. Sería difícil escribir los nombres de todas las personas que he conocido y aprecio sin que la memoria me acabara

traicionando. Gracias a todos!

Agradezco también el apoyo incondicional de mi madre, mi padre, y mi hermano. Ellos no han escrito una tesis doctoral, pero sin duda tienen madera para hacerlo. Finalmente, doy las gracias a la Dra. Margherita Taffarel, que ha recorrido conmigo este camino. Su excelente trabajo contrareloj ha ayudado a que esta obra sea mucho mejor de lo que podría haber sido.

A todos, muchísimas gracias.

Coraggio!!

Sant Cugat, 11 de Abril 2016

Contents

List of Figures vi

List of Tables xi

Abbreviations xiii

1 Introduction 1
- 1.1 The innovation crisis in pharmaceutical industry 1
  - 1.1.1 Possible causes of the crisis 2
  - 1.1.2 Solutions from the pharmaceutical industry 4
- 1.2 The drug discovery pipeline 4
- 1.3 HTS and VHTS 6
  - 1.3.1 How does VHTS work? 8
  - 1.3.2 Fast docking software 9
- 1.4 Flexibility 10
  - 1.4.1 Flexibility and binding 10
  - 1.4.2 Flexibility models 11
    - 1.4.2.1 Molecular Dynamics and flexibility 11
    - 1.4.2.2 Normal Mode Analysis (NMA) as an alternative 14
    - 1.4.2.3 ANM 15
- 1.5 PELE: Achievements and limitations 16
  - 1.5.1 The Metropolis Monte Carlo (MMC) Algorithm . . . 16
  - 1.5.2 The PELE software 19
    - 1.5.2.1 Treatment of flexibility in PELE 19
    - 1.5.2.2 Backbone flexibility 21
    - 1.5.2.3 Minimization 22
    - 1.5.2.4 Major achievements to the date 22
  - 1.5.3 Limitations 22
    - 1.5.3.1 Force field and solvation model 22
    - 1.5.3.2 Metropolis MC 24

1.5.3.3 ANM methodology 24

2 Objectives **29**
2.1 Objective: Technical improvement of PELE 30
2.2 Objective: Algorithmic improvement of PELE 30
2.3 Objective: Efficient and reliable analysis of huge conformational ensembles 30
2.4 Summary of the objectives 31

3 Articles **33**
3.1 Technical improvement of PELE 33
3.1.1 Computer Engineering Degree Project 33
3.2 Algorithmic improvement of PELE 34
3.2.1 Role of the authors 34
3.2.2 Computer Engineering Degree Project 34
3.3 Efficient and reliable analysis 35
3.3.1 Computer Engineering Degree Project 36
3.4 Derived publications 36
3.5 Supplementary materials: Enhancing sampling 73
3.5.1 Are CC and IC modes equivalent? 73
3.5.1.1 Collectivity of CC and IC modes 73
3.5.1.2 Comparison between the CC and IC mode spaces 76
3.6 Supplementary materials: pyRMSD 81
3.6.1 Algorithm performance comparison 81
3.6.2 QCP performance 83
3.6.3 Input size response of QCP implementation 84
3.6.4 Accuracy check 86
3.6.5 OpenMP scalability 87
3.6.6 Comparison with existing packages 87
3.7 Supplementary materials: pyProCT 99
3.7.1 Implemented clustering algorithms 99
3.7.1.1 DBSCAN 99
3.7.1.1.1 Parameters generation strategy . . 99
3.7.1.1.2 Parameters object structure 100
3.7.1.2 GROMOS 100
3.7.1.2.1 Parameters generation strategy . . 101
3.7.1.2.2 Parameters object structure 101
3.7.1.3 Hierarchical clustering 101
3.7.1.3.1 Parameters generation strategy . . 101
3.7.1.3.2 Parameters object structure 102

3.7.1.4 K-Medoids . . . 102
3.7.1.4.1 Parameters generation strategy . . 103
3.7.1.4.2 Parameters object structure 103
3.7.1.5 Spectral Clustering . . . 104
3.7.1.5.1 Parameters generation strategy . . 104
3.7.1.5.2 Parameters object structure 105
3.7.1.6 Random Grouping . . . 105
3.7.1.7 Parameters generation strategy . . . 105
3.7.1.8 Parameters object structure . . . 105
3.7.2 Clustering properties and quality functions . . . 106
3.7.2.1 General definitions . . . 106
3.7.2.2 Properties . . . 107
3.7.2.3 Quality functions . . . 108
3.7.2.3.1 Cohesion . . . 108
3.7.2.3.2 Separation . . . 109
3.7.2.3.3 Compactness . . . 109
3.7.2.3.4 Gaussian Separation . . . 110
3.7.2.3.5 Davies-Bouldin . . . 110
3.7.2.3.6 Dunn . . . 110
3.7.2.3.7 Calinski-Harabasz . . . 111
3.7.2.3.8 Silhouette . . . 111
3.7.2.3.9 PCAanalysis . . . 112
3.7.2.4 Graph cut indices . . . 112
3.7.2.4.1 Degree of a node . . . 112
3.7.2.4.2 Internal volume . . . 112
3.7.2.4.3 Cut . . . 112
3.7.2.4.4 Normalized Cut . . . 113
3.7.2.4.5 MinMaxCut . . . 113
3.7.2.4.6 RatioCut . . . 113
3.7.3 Input script . . . 113
3.7.4 Results file . . . 115
3.7.4.1 Best clustering . . . 115
3.7.4.2 Files . . . 115
3.7.4.3 Trajectories . . . 116
3.7.4.4 Workspace . . . 116
3.7.4.5 Scores . . . 116
3.7.4.6 Timing . . . 117
3.7.4.7 Clustering information . . . 117
3.7.5 Clustering of a long trajectory (proof of concept) . . . 119
3.7.5.1 The trajectory . . . 120
3.7.5.2 Redundancy elimination . . . 120

3.7.5.3 Results . . . 120
3.7.5.3.1 Performance . . . 120
3.7.5.3.2 Clustering . . . 122
3.7.6 2D Validation . . . 122
3.7.6.1 Datasets . . . 122
3.7.6.2 Protocol validation . . . 124
3.7.6.3 Results . . . 127

4 Summary of the results 129
4.1 Technical improvement of PELE . . . 129
4.2 Algorithmic improvement of PELE . . . 129
4.3 Efficient and reliable analysis . . . 130
4.3.1 Implementation of an efficient solution for the calculation of collective superimposition operations . . . 130
4.3.2 Implementation of a reliable cluster analysis protocol . 131

5 Discussion 133
5.1 Technical improvement of PELE . . . 133
5.1.1 From PELE to PELE++ . . . 133
5.1.2 Performance vs. maintainability . . . 135
5.1.3 Optimization and Parallelization . . . 135
5.1.3.1 Initial profilings . . . 137
5.1.3.2 Non-bonding energy parallelization . . . 139
5.1.3.3 Solvent parallelization . . . 142
5.1.3.3.1 Solvent parallelization: OpenMP . 144
5.1.3.3.2 Solvent parallelization: OpenCL . 144
5.1.3.4 Montblanc and other projects . . . 145
5.2 Algorithmic improvement of PELE . . . 147
5.2.1 Switching to a different coordinates space . . . 147
5.2.2 Coarse grain model . . . 148
5.2.3 icNMA theory . . . 148
5.2.3.1 Hessian calculation . . . 148
5.2.3.2 Calculation of the metric tensor . . . 151
5.2.4 From internal to Cartesian coordinates and back . . . 152
5.2.5 Description of the Internal coordinate NMA-based algorithm . . . 153
5.2.6 Alternative implementation . . . 154
5.3 Obtention of the best set of parameters . . . 154
5.3.1 Characterization of the NMA step . . . 154
5.3.2 Obtention of the best simulations and comparison with MD . . . 160

5.3.2.1 Best values for the parameters at 300 K . . . 160
5.4 Comparison with MD . . . 161
5.4.1 Advantages of the new method . . . 163
5.5 Limitations and perspectives . . . 164
5.6 Efficient and reliable analysis . . . 167
5.6.1 Efficient calculation of collective superimposition operations . . . 167
5.6.2 Superimposition algorithms . . . 169
5.6.3 Parallelization and performance . . . 169
5.6.4 Distribution . . . 170
5.7 A reliable cluster analysis protocol . . . 171
5.7.1 Looking for the best clustering . . . 171
5.7.2 pyProCT . . . 171
5.7.3 Hypothesis refinement . . . 173
5.7.4 Software flow detail . . . 173
5.7.5 Scoring criteria . . . 174
5.7.6 Use cases . . . 175
5.7.7 Graphical User Interface (GUI) . . . 175
5.7.8 Distribution . . . 175

6 Conclusions **177**
6.1 Technical improvement of PELE . . . 177
6.2 Algorithmic improvement of PELE . . . 178
6.3 Efficient and reliable analysis . . . 178

List of Figures

1.1 Cost and time needed to develop and market one NME (including failures). Cost data were obtained from (in plot order) [11] [4] [7] [12] [13] [12] [9] [14] [15] [16] [17]. Time data were obtained from [18] [18] [13] [9]. 3

1.2 Drug discovery pipeline phases (pre-discovery and target identification phases have been omitted). The number of compounds used at each step [3, 25], the number of people involved in the clinical trials [25], the percentage of the total development cost per phase [9], and the percentage of time invested in each phase [10, 16, 26] are illustrated. 7

1.3 Example of the interactions described by a force field. From them the potential energy of the system ($U = E_{bond} + E_{angle} + E_{torsion} + E_{nb}$) as well as the forces, accelerations and velocities of each single particle (atom) can be calculated. 12

1.4 ANM study of an open conformation of adenylate kinase (PDB id: 4AKE). From the starting structure (A) the elastic network is calculated (B). Once the normal modes are obtained (C) the structure can be modified so that the final conformation is similar to the closed one (PDB id: 1AKE) (D). In this case, these open and close states may be present without the need of ligand interaction [105]. 17

1.5 Schematic representation of PELE flux. The initial conformation goes through four perturbation/relaxation steps, after which the acceptance criterion is tested. If the final conformation is accepted, it will be used as the initial conformation of a new iteration. If it is rejected, a new iteration will be started with the same initial conformation. 20

1.6 A) Two normal modes of a triatomic molecule. B) Application of those normal modes following PELE algorithm. A linear combination of the modes (here with weights 1,1) defines the directions of the conformational change (1). These directions are scaled in order to obtain the new positions for the particles of the system (2). Harmonic constraints to target positions are added and C_α atoms are moved through a minimization (3). Finally, in the last global minimization step, weak harmonic constraints are added to C_α the current position of the atoms so that the movement is not undone (4). 23

1.7 A) 7 frames from an MD simulation of a porcine adenylate kinase showing different inter-domain distances. Below each frame, its elastic network has been reproduced. The EN changes with the inter-domain distance, being especially perceptible from the 17 Å. B) Cumulative overlap and maximum value of the overlap (including the index of its related mode) between the modes of the first conformation and all the others. C) Cumulative overlap between the linear displacement from the open to the closed conformations and the modes of each frame and the maximum overlap (again, the index of the mode with maximum overlap is also shown). 27

3.1 Degree of collectivity for each structure and calculation method. 75

3.2 Detailed study of the degree of collectivity per mode, structure and method. Cutoff has been set to 9 Å . The Hessian modification method (m.H.) produces only slight improvements, generally concentrated in higher frequency modes. 75

3.3 Average cumulative overlap for different cutoff distances, methods, and structures in our test set. Standard deviations are not shown for the sake of clarity. 77

3.4 Detailed study of the cumulative overlap per mode, structure and method. Cutoff has been set to 9 Å . In general, the rightmost modes (higher frequencies) are the ones with worst overlap. 78

3.5 RMSIP of CC and IC mode spaces. The x-axis shows the upper limit of the mode space tested (e.g, 10 means that the first ten modes are to be used to obtain the RMSIP). Both spaces look to be very similar, at least for the 5 lowest frequency modes. The similarities decrease as we move to modes of higher frequencies, with the only exceptions already commented in the cumulative overlap study. 78

3.6 Comparison of the execution of three methods for the three available algorithm implementations (serial). 82
3.7 Comparison of the execution of three methods for the three available algorithm implementations (OpenMP). 82
3.8 Calculation time of a pairwise matrix from a 30k frames trajectory. 83
3.9 Performance comparison of serial, OpenMP and CUDA (single-precision) implementations of the QCP algorithm. 84
3.10 Performance comparison of CUDA implementations. 'Mem' versions hold the entire matrix into memory, (s) versions use single-precision arrays and (d) versions use double-precision arrays. 85
3.11 Input size response of the OpenMP and CUDA versions of the QCP algorithm. 86
3.12 Time needed to compute a pairwise matrix from a 30k frames trajectory using a different number of threads. 88
3.13 Percentage of speedup per thread added to the calculation. Speedup is almost linear with number of threads. 88
3.14 A) Global reduction of the size of the dataset vs. merging local reductions. B) Different levels of compression including the number of frames used in each level. 121
3.15 Representative conformations for the 4 most populated clusters, holding a 34%, 13%, 8 % and 8% of the elements of the dataset. 123
3.16 Results of the application of pyProCT to nine 2D datasets. Clusters are plotted using different colors and symbols. 125
3.17 An incorrect choice of the ICVs to express the desired resulting clustering traits can drastically modify the results. In this case the criteria was changed from "graph_criteria" to "default_criteria", favoring one of the clusterings generated by the K-Medoids algorithm. 126

5.1 Pseudo-UML diagram showing the most relevant classes in PELE++ core: the AtomSet tree which allows the definition of different types of molecules, the topology subsystem and the energy/potential subsystem. The last uses geometry (atom coordinates) and topological information to calculate the energy of an AtomSet. 136
5.2 A set of profiles was performed for different protein sizes and using OBC and SGB solvent. This bar plot shows the percentage of time spent in non-bonding and solvent-related calculations. 138

5.3 Comparison of the energy and gradient functions speedup for different protein sizes and the two programming models used. One of the reasons why the speedup for the medium protein is higher is because the relative weight of the non-bonding calculations has increased with the number of atoms. 141
5.4 The global speedups have been calculated using a model. As parameters N and M range from 1 to 65 and from 20 to 50 respectively, we decided to study the best case (faster serial execution, where N = 1 and M = 20) and the worst case (slower serial execution, where N = 65 and M = 50). Theoretical speedup increases to decrease afterwards. This happens as a result of the changes in the relative weight of the non-bonding calculations: the weight first increases due to the increment in the number of atoms and then decreases since other functions, such as the covalent energy calculations, start to require more time. The difference between implementations is not significative. . 143
5.5 Kernel speedup for each of the methods and proteins tested. Methods 2 and 3 seem to obtain an equivalent efficiency improvement. 145
5.6 CG model of a peptide (A) using the $[C_\alpha]^3$ distribution of atoms. Each residue is composed of two units which are delimited by the ϕ and ψ torsions. The case of proline residues (B) is special, as they only form one unit. 148
5.7 Representation of the rotation of two rigid bodies (green and purple) around axis q_α. 149
5.8 Plot showing the relationship of the two studied parameters (*steeringForce* and *MinimumRMS*) with the RMSD and energy increments of the Cartesian coordinate ANM step. Each point shows the average and standard deviation of the RMSD and energy increments for a given combination of parameters. . . . 157
5.9 Pseudo-UML (Unified Modelling Language) diagram showing some key classes of pyRMSD as well as the three-layer design (Python classes, Python C interface and C++ classes). 168
5.10 Four cluster analysis have been performed over the same data set. Results can change dramatically depending on the algorithm (k-medoids or spectral clustering here) and parameters used (k = 2 or k = 3). 172

List of Tables

3.1 Root Mean Square of the RMSD array differences for each of the algorithms. 87
3.2 Comparison of the time, lines of code, speedup and integration complexity (I.C.) needed to complete an RMSD collective operation. 90
3.3 Around 40k clusterings were produced in almost 58h (1 clustering each 5s). 122
3.4 Clustering hypothesis for each of the datasets. 127
3.5 Details of the results. Last column indicates the criteria that obtained the best score. 128

5.1 Size-related details of the systems used in the initial profilings. 137
5.2 Choice of parameters affecting the mode application step in the CC and IC methods, including the values that will be used in characterization tests. 155
5.3 Association strength of the studied parameters and simulation features. Each value has been colored depending on its category: Green for high association, yellow for medium association, orange for low association and red for no association. . . 157
5.4 Association strenght of the RSMD and energy increments, and step time between themselves and the chosen parameters. . . 159
5.5 Values for the parameters that produced simulations with acceptance between 20 and 40%. 162

Abbreviations

ADME/Tox Absorption, Distribution, Metabolism, Excretion / Toxicological.

AMBER Assisted Model Building with Energy Refinement.

ANM Anisotropic Network Model.

ARM Advanced RISC (Reduced Instruction Set Computing) Machine.

BNM Block Normal Mode.

CC Cartesian Coordinates.

ccNMA Cartesian Coordinate Normal Mode Analysis.

CG Coarse Grain.

CPU Central Processing Unit.

DNA Deoxyribonucleic Acid.

EN Elastic Network.

FDA Food and Drug Administration.

FEP Free Energy Perturbation (method).

GPGPU General-Purpose computing on Graphics Processing Units.

GPU Graphics Processing Unit.

HTS High-Throughput Screening.

IC Internal Coordinates.

icNMA Internal Coordinate Normal Mode Analysis.

IDP Intrinsically Disordered Protein.

LBVS Ligand-Based Virtual Screening.

MC Monte Carlo.

MD Molecular Dynamics.

MIC Many Integrated Core.

MM/PBSA Molecular Mechanics / Poisson-Boltzmann Surface Area.

MMC Metropolis Monte Carlo.

MSM Markov State Model.

NMA Normal Mode Analysis.

NME New molecular Entity.

NMR Nuclear Magnetic Resonance.

NVT (constant) Number (of particles), Volume and Temperature.

OBC Onufriev-Bashford-Case.

OOP Object-Oriented Programming.

OPLS Optimized Potential for Liquid Simulations.

PCA Principal Component Analysis.

PCR Polymerase Chain Reaction.

PDB Protein Data Bank.

PELE Protein Energy Landscape Exploration.

PMF Potential of Mean Force.

R&D Research and Development.

RMSD Root Mean Square Deviation.

RMSF Root Mean Square Fluctuation.

RNA Ribonucleic Acid.

RTB Rotational-Translational Block.

SASA Solvent-Accessible Surface Area.

SBVS Structure-Based Virtual Screening.

SGB Surface Generalized Born.

VHTS Virtual High-Throughput Screening.

VS Virtual Screening.

1

Introduction

Human biology seats in a weak equilibrium. The breakdown of this homeostasis, whether it is because of external agents or deficiencies in the internal regulations, produces illness and death. Therefore, it is not surprising that, throughout its history, humankind has always been looking for treatments and medicines to fight disease. Today, more than ever before, we need to keep perfecting health methods. Industrial excesses and lax laws are creating a hostile environment that affects us directly or through the food chain in unexpected ways. Also, the same advances that have allowed the rise of the world's population life expectancy [1] are being put to the test to cure the increasing number of aging-related illnesses that reduce our quality of life.

1.1 The innovation crisis in pharmaceutical industry

Nowadays, the role of discovering and manufacturing new and better drugs is being played mainly by the pharmaceutical industry. With a 3.9% of the gross value added in manufacturing worldwide in 2011 [2], more than 690,000 direct employees in Europe (2013) [3] and more than 810,000 in the United States of America (USA) (2013) [4], its economical importance is out of doubt. Most big pharmaceutical companies have good financial results and are able to attract investors. However,these good results are mainly because of their incremental innovation model, that is, their ability to improve old products [5]. This is an efficient strategy, since it can help milden the profit loss coming from expiring

patents at a time when consumers tend to buy generics over brand name drugs[1].

However, if we focus on an innovation indicator like the number of NMEs (New Molecular Entities) approved every year by the USA's Food and Drug Administration[2], the pharmaceutical industry does not look so prosperous. The number of approved NMEs has been very low during the last decade (with an average of 24 per year [6]).This is a matter of concern, since , the cost of marketing one of these drugs has increased three times in the same amount of time. Also, the drug development time span has not shortened and it usually lasts 15 years [7], but can exceed 30 years [8]. The discovery of a single NME is becoming more difficult and costly, and the revenues returned from them rarely exceed Research and Development (R&D) investments. That is why, currently, the pharmaceutical industry is considered to be faced with a deep innovation crisis.

1.1.1 Possible causes of the crisis

Numerous efforts have been made to understand the causes of this crisis in order to improve R&D efficiency and effectiveness. From what is known, the main problem is not the lack of investments, since the annual spending in R&D has been around $40-$50 billion [4, 9] during the last ten years. Indeed pharmaceutical companies reinvest a 12.4 per cent of gross domestic sales on R&D, of which 9.3 to 12.4% (1.2 / 2.4% of total) go to fundamental research [10].

Possible causes of the innovation problem include [19]:

- The increasingly difficult requirements from regulatory agencies to accept new drugs. New standards of safety make clinical trials longer and more costly.

- The "saturation of low hanging fruits" theory, which says that pharmaceutical industry has already discovered all that is possible with our current knowledge.

- R&D teams give a sharp focus to revolutionary instead of evolutionary technologies. For instance, the use of genomics-based candidates was so promising that was prioritized over clinically validated drug targets. Still, the repercussions of including revolutionary technologies in pharmaceutical pipelines are not always negative. For example PCR[3] (1983)

[1] The balance increased from 49% in 2000 to 88% in 2013 [4].

[2] All data refers to USA agencies unless otherwise stated.

[3] Polymerase Chain Reaction

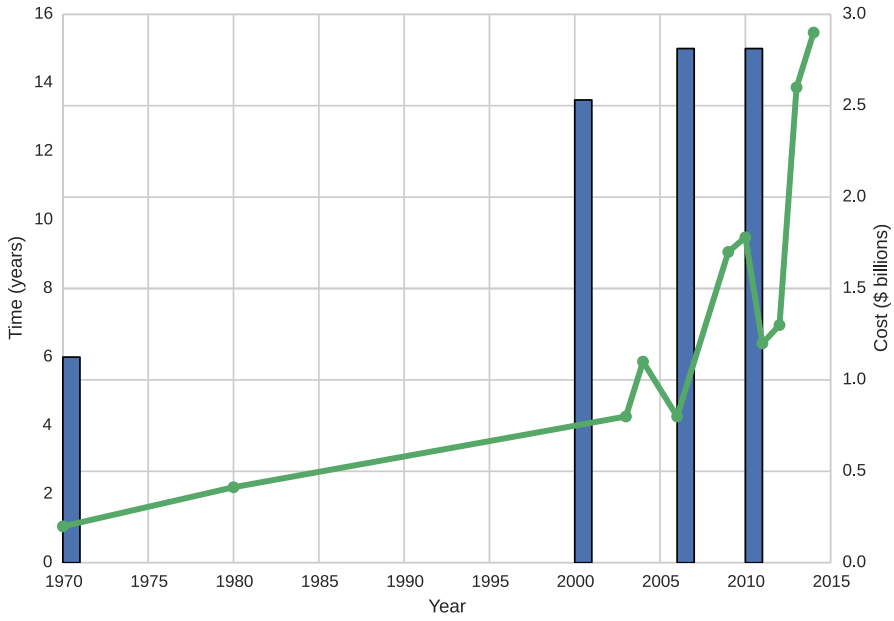


Figure 1.1: Cost and time needed to develop and market one NME (including failures). Cost data were obtained from (in plot order) [11] [4] [7] [12] [13] [12] [9] [14] [15] [16] [17]. Time data were obtained from [18] [18] [13] [9].

for DNA[4] replication or gene chips for RNA[5] in the early 1990s allowed researchers to perform more and faster experiments.

- Pharmaceutical companies are extremely big, and they spend great amounts of resources on non-productive work.

1.1.2 Solutions from the pharmaceutical industry

Pharmaceutical companies have reacted to this situation immediately in order to improve the outcome of their R&D investments. One of the consequences has been the forge of strategic alliances (especially with biotechnology firms) or even company mergers[6]. The industry has also changed its marketing strategies in an attempt to overcome this delicate situation. For instance focusing on rare diseases, or opting for chronic diseases instead of acute diseases make clinical trials more challenging but give good economic results in the long term. Finally, drug repurposing seems a good solution as it can reduce the costs of bringing a drug to the market by almost 40% [20].

Special attention has been paid to boosting the efficiency of the drug discovery pipeline steps, as this has direct repercussion on the final cost and on the time invested on the drug . For instance, an outsourcing of early steps such as screening, lead identification or lead optimization has shown to decrease the total costs noticeably. Interestingly, if several companies share the same service providers, they will act as knowledge and expertise exchange nodes, which seems to be a very beneficial side effect in such a hermetic environment .

1.2 The drug discovery pipeline

The drug discovery pipeline can be divided into two consecutive main phases. In the first one, the aim of researchers is to understand the illness and find a molecule that can cure it (or, at least, improve its symptoms). During the second main phase, which starts right after regulatory agency clearance, the drug is tested in human patients.

Nowadays Bioinformatics and Computational Biology software play an im-

[4]Deoxyribonucleic acid

[5]Ribonucleic acid

[6]For instance, the two giants Pfizer and Allergan, have entered into a merger agreement (announced on the 23th of November of 2015), after several hostile takeover bids to AstraZeneca in 2014.

portant role in the discovery process[7]. It is hard to quantify the impact of software on efficiency because, unfortunately, there are few studies on this topic. However, it is quite obvious that the improvements made on this software would also lead to improvements in the overall effectiveness of the pipeline.

The phases of the pipeline are:

1. Discovery process

   Pre-discovery The goal of this phase is to study a disease in order to understand its causes. This phase is usually neglected in pharmaceutical industry reports because of two main factors. First, the time and budget needed are often very variable and this phase generally contributes to drug discovery in the long term. Second, this kind of research is usually performed at non-corporate tax-funded institutions, such as universities or government research centers, and do not have a direct impact on companies budgets. Fostering basic research and the technology transfer between these institutions and the industry is of utmost importance in order to shorten the time of this phase.

   Target identification During this phase, a druggable molecule involved in the disease, usually a protein, is searched . Improvements in this stage come from the use of gene chips and bioinformatics techniques, as well as proteomics [21]. The sequencing of the human genome also seemed to be an abundant source of drug targets [22, 23], but, in general, it has not met the initial expectations yet [24].

   Target validation During this process, the previous target is tested in single cells and animal models. In-vitro validation can be performed by disrupting target expression through the use of gene knockouts.

   Lead identification At this point, researchers look for the chemical leads (small drug-like molecules capable of altering the function of the target proteins) and early pharmacokinetics tests (ADME/Tox[8]) are performed. This includes the target-to-hit phase, a preliminary screening to filter non-active compounds, and the hit-to-lead phase, a secondary screening where the compounds with the higher potential to become a drug are chosen. High-throughput screening (HTS) and

[7] The success of Shrödinger is a good example of the increasing importance of computational methods in drug discovery. In 2015 they signed a $120M deal with Sanofi (with tasks including target analysis, validation, lead identification and lead optimization). More recently (2016) they have entered a research collaboration agreement with Pfizer.

[8] Absorption, Distribution, Metabolism, Excretion and Toxicological

virtual high throughput screening (VHTS) supported by chemoinformatics seem to help speed up this phase.

Lead optimization Leads are further optimized to act as a non-toxic therapeutic drug. Rational drug design and combinatorial chemistry work can be used to improve the properties of the drug candidate.

Preclinical testing Drug toxicity is determined using in-vitro and in in-vivo (animal models) experiments.

2. Development process

Phase 1 clinical trial Initial tests in healthy volunteers (20-100).

Phase 2 clinical trial Test in a small group of patients (100-500).

Phase 3 clinical trial Safety and efficacy tests in a large group of patients (1000-5000).

Phase 4 monitoring After this steps the drug is commercialized, but it is still monitored in search of possible undocumented side effects.

Several new technologies have been used to lower the cost and time required for clinical trials. Genomics, for instance, would allow subdividing patients according to their drug response, and pharmacogenetics and expression pharmacogenomics can enhance the predictions of patient's drug response by analyzing their DNA variability or determining the levels of gene expression. The use of collaborative management software in clinical trials has shown to have positive effects on the efficiency as it makes researchers less prone to commit errors.

1.3 HTS and VHTS

The early steps of the pipeline are crucial for the success. Target selection, for instance, is one of the most important determinants of R&D productivity as an error during this phase can potentially compromise the entire process.

The lead identification step is a very sensitive point too. This step is often performed using HTS. With this technique, lead molecules are identified through automated individual chemical assays using libraries of millions of compounds. Almost a 40-45% of the drugs currently being tested in human clinical trials come from HTS systems. This is more than four times the percentage found ten years ago [13].

Instead of finding hits using experimental assays, screening experiments can also be conducted in-silico through VHTS protocols. These high-speed

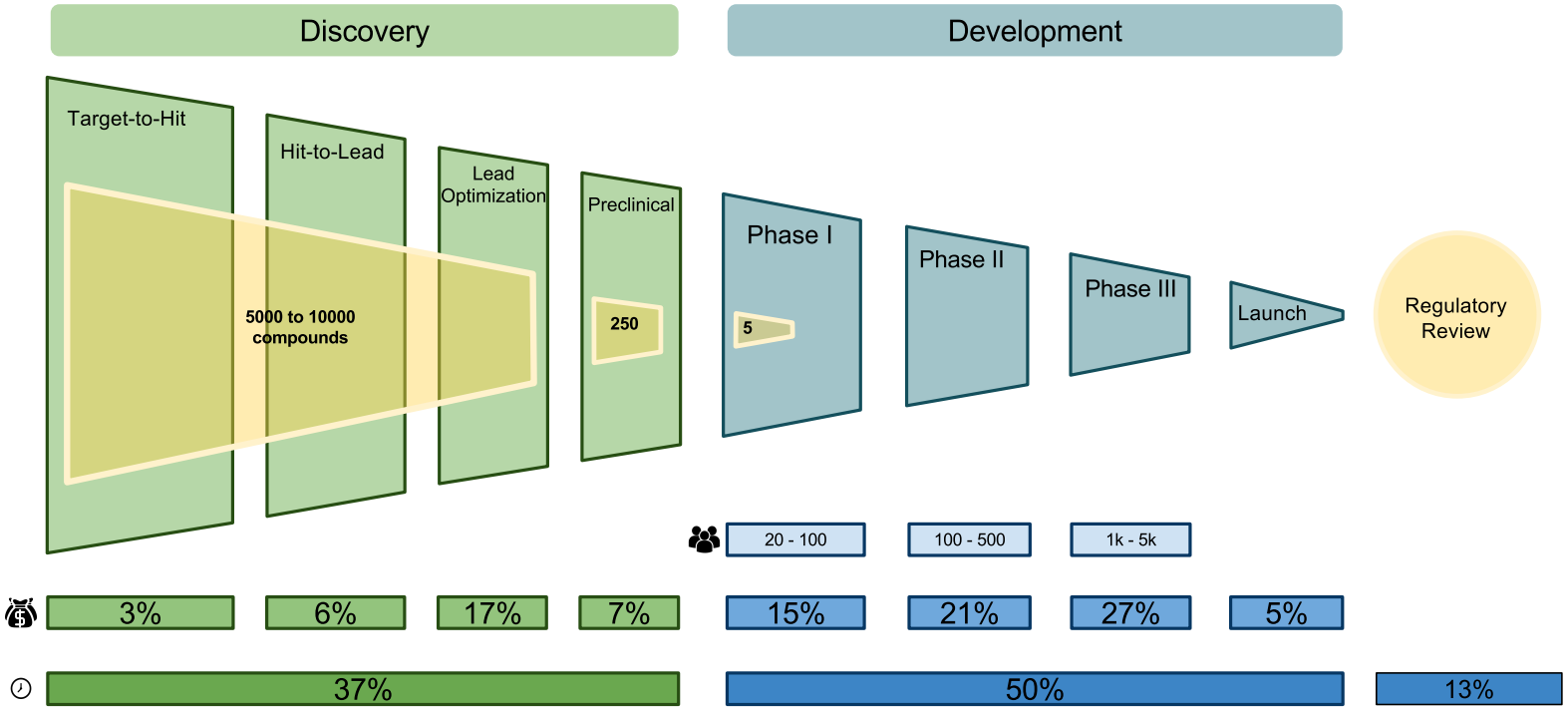


Figure 1.2: Drug discovery pipeline phases (pre-discovery and target identification phases have been omitted). The number of compounds used at each step [3, 25], the number of people involved in the clinical trials [25], the percentage of the total development cost per phase [9], and the percentage of time invested in each phase [10, 16, 26] are illustrated.

computer screenings consume only a small percentage of the R&D costs and time and can potentially save up to one year and 15% of the total development cost (according to the Boston Consulting Group [27]).

1.3.1 How does VHTS work?

In order to start the VHTS process, researchers first need to obtain libraries of compounds. The following steps will vary depending on the approach adopted. The first one is ligand-based virtual screening (LBVS). It rests on the idea that molecules with similar structures have similar properties. It uses the information of compounds known to bind the protein and does not need the structural data of the target [28, 29]. It may use Quantitative Structure-Activity Relationship methods, pharmacophore modeling ("the largest common denominator") and database mining. The second approach, that is structure-based virtual screening (SBVS), needs the tridimensional structure of the target and uses docking algorithms to select the drugs that bind best. The choice of one method or the other depends on the availability of data and on the type of problem. LBVS methods are still dominating the VS field because lead information is usually more easily available than structural information [30]. However SBVS is gaining reputation, probably because nowadays there is a huge amount of structural data available for researchers (e.g. the Protein Data Bank (PDB) [31] stores more than 100k structures, and in-house industry resources also store extensive data), and it looks to be increasing at a good pace thanks to the advances of structure determination techniques. Moreover, SBVS is able to find allosteric binding sites, which is more difficult in LBVS approaches [32]. SBVS and LBVS are not mutually exclusive and can improve final results if used together [33–35].

The phases of SBVS usually are:

Prefiltering Libraries are prefiltered using their ADME/Tox properties and a physicochemical profile (if a lead profile is available). Binding site discovery: When crystal structures with bound ligands are not available, cavities can be located computationally using either geometric approaches (e.g. fpocket [36]), grid-based approaches (e.g. LigSite [37]) or energy-based approaches.

Docking A high throughput docking over thousand to millions [38] of compounds is performed. Each single compound is positioned into the target's previously discovered binding pockets using docking software (e.g. DOCK [39], AUTODOCK [40, 41], GOLD [42], GLIDE [43], FlexX [44], ARTIST [45], ICM [46], and Surflex-Dock [47] among others).

Scoring Finally the poses are ranked using a scoring function, and the best results are kept. These scoring functions include: force field scoring, empirical scoring, knowledge-based scoring and consensus scoring (which uses more than one scoring function in order to balance their errors). The calculation of binding free energies as a scoring function by using, for example, endpoint (MM/PBSA[9]), pathway (PMF[10], Metadynamics) or free energy perturbation methods is more rigorous, but also more computationally expensive [48, 49] and barely used.

VHTS works as a filtering mechanism that benefits the following stages by reducing the number of compounds to be tested experimentally[11] (e.g. inactive compounds). However, its use can become a burden if too many false positives are generated, thus "contaminating" the pipeline; errors will be paid in next steps at a high cost [50]. That is why it is desirable and more productive to "fail fast" and unambiguously. The sources of error in SBVS are numerous: incorrect protonation [51], incorrect assignment of side chain rotamers, low sequence identity in homology modelling (if the structure is not available), absence of water in the binding site [52], incorrect/no handling of flexibility or, simply, inaccuracies of the docking procedure or in the scoring functions. As a consequence, it is still safer to use SBVS as a complement [53] to empirical screening rather than as a complete replacement (e.g. to increase hit-rate of HTS).

1.3.2 Fast docking software

The keystone of the SBVS process is the high throughput docking software. Docking is one of the most significant challenges of computational biology and represents an area of intense academic research [54]. Docking algorithms can be classified in several categories, depending on the types of molecules (protein-protein and protein-ligand docking), the treatment of flexibility (flexible ligand, receptor or both), or the algorithmic details (matching or simulation). Given receptor and ligand molecules, the goal of a docking program is to predict if they interact and find their relative positions and conformations in order to generate the resulting complex.

[9] Molecular Mechanics (combined with) Poisson-Boltzmann Surface Area

[10] Potential of Mean Force

[11] This need has been lately acknowledged with the approval of the 4-year project ADDoPT (Advanced Digital Design of Pharmaceutical Therapeutics) in the UK. One of its goals is to improve the productivity of drug discovery processes through the early detection of non-viable drugs by using mainly computer tools and big data approaches. The academia and the industry (including big pharmaceutical companies such as AstraZeneca, Bristol-Myers Squibb, GlaxoSmithKline and Pfizer) will cooperate in order to bring the process to a successful conclusion.

VHTS-compatible docking software must be fast. The time it works with each compound cannot exceed a few seconds, as libraries may contain thousand to millions of them [33]. The combinatorial explosion of possible conformations/poses due to the enormous number of internal and external degrees of freedom involved makes it hard to perform a systematic exploration of the solution space. As a consequence, many simplifications must be done in order to attain reasonable execution times. To this end, the docking problem usually becomes an optimization problem over certain fitness function where a rigorous simulation of the biophysical properties of the system, as well as the handling of ligand and receptor flexibility, is often sacrificed.

1.4 Flexibility

1.4.1 Flexibility and binding

Flexibility is known to play a major role in molecular recognition processes and is a long-studied topic. Nowadays, we know that it contributes to favorable changes in the binding free energy [55] by optimizing the noncovalent interactions between the receptor and the ligand, or by increasing the entropy upon binding by releasing interfacial water and increasing flexibility in parts of the protein or ligand [56]. However, the importance of flexibility has not always been present in binding theories. Models like Fisher's "lock and key" [57] (1984), described binding as an entirely rigid interaction in which ligands and receptors must be complementary in shape in order to bind. This model prevailed for more than 60 years until Koshland [58] formulated its induced fit hypothesis in 1959. In this second model, the ligand is able to produce a deformation in the receptor's active site: complete complementarity is not needed for binding to happen. Another hypothesis, the conformational selection model [59, 60], asserts that all possible conformers of the receptor coexist in solution, including the binding one. The interaction with the ligand produces a population shift towards the binding conformation. The consequences of this model can affect drug design strategies as the knowledge of these relative populations could be used to create drugs with different degrees of binding affinities.

Currently, both the induced fit and conformational selection models are accepted. Both agree on the fact that binding is a dynamic event where ligand and receptor may change their structures dynamically. Although these models may not be complete, as both are defied by intrinsically disordered proteins (IDPs[12]), experimental evidences support them (e.g. the findings on HIV-1

[12] A fourth model "coupled folding and binding" [61]

protease [62], DHFR [63] or aldose reductase [64]); this may mean that they are not exclusive.

1.4.2 Flexibility models

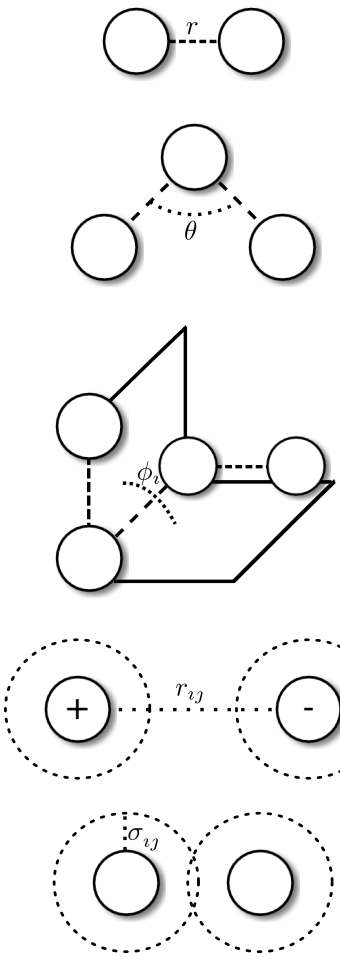
Neglecting the treatment of flexibility reduces the degrees of freedom of the search space dramatically, thus lowering the time needed to find solutions. However it can severely limit the accuracy of results. Some studies, for instance, show that the consequences of incorrectly treating flexibility drive a success rate drop of ~25%-55% in binding simulations [65] being this drop proportional to the number of rotatable bonds of the ligand and correlated with the degree of the protein movement in the active site. In order to improve the accuracy of docking software, a restricted treatment of flexibility must be applied, trying to reach a good compromise between a robust theoretical treatment and computational performance.

1.4.2.1 Molecular Dynamics and flexibility

The best-known method to sample the flexibility of ligands and receptors is molecular dynamics (MD). In MD, the positions of the particles of the system are predicted by integrating Newton's equations of movement. This method offers a detailed vision of the dynamics of molecules, allowing to understand how systems evolve at atomic level.

Despite its renown, MD is not completely flawless. For instance, the force fields it uses (see Fig. 1.3) are known to be biased towards certain types of secondary structures [66, 67], and it has been reported that simulations can end trapped into sink free energy states [68, 69], reducing the sampling quality. Also, the discretization error of integrators advises against running single long simulations in favour of running multiple short ones [68, 69]. Despite this, MD simulations have shown to have a good correlation with experiments (for example in comparisons with Residual Dipolar Coupling data [70]) and has been successfully used on many occasions to unveil molecular mechanisms that could not be studied in other manners. The reviews of Karplus [71], Dodson [72] and Dror [73] illustrate many of these success stories.

The integration step in MD must be small enough to guarantee the algorithm's stability and is frequently set on the femtosecond scale. Given that functional changes, which may be related to binding mechanisms, often occur at the μs/ms scale, obtaining an informative simulation requires computing a considerable number of steps. Besides, the atomic detail of simulations, which usually includes an explicitly modeled solvent, means that studied systems have



$$E_{bond} = \sum_{bonds} K_r(r - r_{eq})^2$$

$$E_{\text{angle}} = \sum_{\text{angles}} K_\theta(\theta - \theta_{\text{eq}})^2$$

$$E_{torsion} = \sum_i \sum_{j=1}^{3} \left(\frac{V_j^i}{2} 1 + (-1)^{j+1} \cos(j\phi_i + f_j^i) \right)$$

$$E_{\text{nb}} = \sum_{i<j} \left(\frac{q_i q_j}{r_{ij}} + 4\epsilon_{ij} \left(\frac{\sigma_{ij}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}^{6}}{r_{ij}^{6}} \right) \right) f_{ij}$$

Figure 1.3: Example of the interactions described by a force field. From them the potential energy of the system ($U = E_{bond} + E_{angle} + E_{torsion} + E_{nb}$) as well as the forces, accelerations and velocities of each single particle (atom) can be calculated.

a huge amount of particles, making the calculation of each step harder. These are the main reasons why MD is such a computationally demanding method.

Even with the current algorithmic and hardware progress (that includes the intensive use of accelerators), calculations cannot routinely exceed the μs total time. The construction of single-purpose dedicated-hardware machines (e.g. FASTRUN [74], MD Engine [75], MDGRAPE [76] and ANTON [77]) has allowed researchers to perform longer simulations more easily. An example of this is a recent work by Shan [78], who has calculated a 20 microsecond simulation showing the free diffusion and binding process of dasatinib and the kinase inhibitor PP1 with Src kinase. Unfortunately, these specialized computers are usually not publicly accessible, and even if they were, the pace at which these machines produce results is still impractical for high throughput methodologies like VS[13].

Derived methods aiming to speed MD simulations, like replica exchange, are still too slow to be applied in VS protocols. The need to overcome these computational limitations led to the creation of the first rigid ligand / rigid receptor algorithms. However, the need for some forms of computationally lightweight flexibility was acknowledged shortly afterwards, giving place to other well-known techniques such as:

Incremental construction A rigid part of the ligand is docked first and then the flexible elements are incrementally added so that the ligand adapts to the binding site. Ex. GROWMOL [81].

Multiple conformations Can be applied to both ligand and receptor. In the case of the ligand, new conformations can be obtained by sampling the angles of rotatable bonds and keeping the lower energy solutions, using them for the docking. The main problem is that the complexed ligand is not always in a low energy state [82, 83]. In the case of a protein receptor, different conformations can be obtained from Nuclear Magnetic Resonance (NMR) or X-ray structure determination experiments.

Other types of methods perform a conformational search in order to sample different receptor and ligand structures. Some of these methods are simulated annealing (Yue's algorithm [84]), Genetic Algorithms (ex. DARWIN [85]), tabu search (Ex. PRO_LEADS [86]) and path planning (MIAX [87]).

Due to their inherent flexibility, the representation of side chains is a challenge on its own. Several techniques have been proposed. For example, the "soft receptors" method allows the interpenetration between atoms to a certain

[13]With the exception of generating ensembles for "multiple receptor conformations" methods [79, 80]

extent [88]. It is more frequently used in fully rigid docking setups. Another possibility is the use of rotamer libraries. This is a systematic method in which a collection of possible rotamers for side chain torsional angles is randomly or heuristically applied.

1.4.2.2 Normal Mode Analysis (NMA) as an alternative

At this point, it may be clear that a reasonable compromise between the correct modeling of flexibility and its computational requirements must be found: accuracy and detail in the representation of flexibility have an expensive price in CPU[14] cycles, but if a less accurate treatment of flexibility is chosen, the simulation can be so inexact to become useless. NMA-based methods might offer a reasonable trade-off, since they add flexibility in the receptor at a low computational cost.

The basis of NMA [89] is to simplify the potential energy surface by means of a harmonic approximation. To this end, a potential well is built around a stable conformation (here with general coordinates q). The molecule is represented by a set of coupled harmonic oscillators and motion is restricted to small fluctuations around that single minimum. The Taylor expansion to the second order of the potential and kinetic energy around the minimum is:

$$V = V_0 + \sum_{\alpha=1}^{N} \left(\frac{\partial V}{\partial q_\alpha} \right)_0 q_\alpha + \frac{1}{2} \sum_{\alpha=1}^{N} \sum_{\beta=1}^{N} \left(\frac{\partial^2 V}{\partial q_\alpha \partial q_\beta} \right)_0 q_\alpha q_\beta + \cdots \tag{1.1}$$

$$K = K_0 + \sum_{\alpha=1}^{N} \left(\frac{\partial K}{\partial \dot{q}_\alpha} \right)_0 \dot{q}_\alpha + \frac{1}{2} \sum_{\alpha=1}^{N} \sum_{\beta=1}^{N} \left(\frac{\partial^2 K}{\partial \dot{q}_\alpha \partial \dot{q}_\beta} \right)_0 \dot{q}_\alpha \dot{q}_\beta + \cdots \tag{1.2}$$

Under the assumption of equilibrium, the kinetic and potential energy depend only on the second order terms.

From the resulting Hessian H (where $H_{\alpha\beta} = \frac{\partial^2 V}{\partial q_\alpha \partial q_\beta}$) and metric tensor K where $K_{\alpha\beta} = \frac{\partial^2 K}{\partial \dot{q}_\alpha \partial \dot{q}_\beta}$ the equations of motion can be derived from Lagrange's equation, with the Lagrangian $L = K - V$, so that:

$$-\sum_{\beta}^{N} H_{\alpha\beta} q_\beta = \sum_{\beta}^{N} K_{\alpha\beta} \dot{q}_\beta \tag{1.3}$$

The N harmonic solutions to the equations of motion ($q_\alpha = \sum_k^N A_{\alpha k} \alpha_k cos(\omega_k t + \phi_k)$) can be calculated by solving the eigenproblem:

[14] Central Processing Unit

$$KA\Lambda = HA \tag{1.4}$$

where A and Λ are the eigenvector and eigenvalues matrices respectively. The resulting modes are an orthonormal basis of all possible deformations of the molecule around the equilibrium structure. For sufficiently small displacements, the motion of the particles in the system is proportional in amplitude to the magnitude of the eigenvectors and its frequency is proportional to the square root of the eigenvalue. Also, each eigenvalue represents the energetic cost of displacing the system by one length unit along its eigenvector. In general, we are only interested in the subset of modes of lower frequencies i.e. the ones that require less energy, as this slower but wider displacements are usually the ones encoding functional information.

1.4.2.3 ANM

In 1996, Tirion [90] introduced an NMA model where the molecule is modeled as an EN of Hookean springs of equal strength with potential:

$$V = \sum_{\alpha,\beta} \frac{k}{2}(r_{\alpha\beta})^2 \tag{1.5}$$

One of the advantages of the simplification of the potential is that, unlike regular NMA, it does not need to minimize the initial structure to fulfil the assumption of equilibrium: the initial description of the network is already in equilibrium.

Bahar and coworkers [91] extended the model by using only the C_α atoms to represent each residue in an isotropic NMA version. Moreover, the number of atomic interactions was reduced by using a cutoff distance. They showed that, even with these dramatic simplifications, the beta factors derived from the new model were in good agreement with experiments. The model was further improved by Atilgan *et al.* [92], who used it to extract anisotropic information from the fluctuations. This development, and more generally the NMA models using elastic networks and coarse-grained representations, are also known as Anisotropic Network Models (ANM, see Fig. 1.4).

Most of the efforts to enhance the model have focused on finding a definition of force constants that is able to improve the modeling of the atomic interactions. First attempts come from Hinsen's work [93], whose distance-dependent force with exponential decay was fitted using the AMBER force field. More recent studies include the analysis of different formulations for the force constant [94] and MD-based parameterizations [95].

The most important computational improvements of ANM come from the

use of a cutoff and from the coarse-grained representation. The cutoff reduces the number of iterations needed to calculate the Hessian and the reduction of degrees of freedom from the reduced representation makes it smaller and easier to diagonalize. Since the Hessian calculation represents the computational bottleneck, finding efficient methods to diagonalize it has been the focus of several research projects. These efforts have resulted in the creation of new algorithms such as the RTB (Rotational-Translational Block) [96] or BNM (Block Normal Mode) [97], as well as more efficiently parallelizable techniques using the Krylov subspace and Cholesky factorization [98]. Besides, the use of mass-weighted coordinates further simplifies calculations by avoiding the need to calculate the Kinetic tensor (the equation $\hat{H}\hat{A} = \hat{A}\hat{\Lambda}$ is to be solved instead).

These simplifications, however, do not compromise the predictive capabilities of the ANM method. This is demonstrated by the good agreement with atomistic simulations [99, 100] and essential dynamics from ensembles of experimental structures (e.g. HIV-1 protease [101]). ANM has shown to have good correlation with experimental beta factors (e.g. of DNA-dependent polymerases [102]) and anisotropic temperature factors of X-Ray and NMR ensembles [103]. Also, it has been able to reproduce experimentally solved domain movements in several proteins, such as the Aspartate transcarbamylase [104].

The success of ANM modeling the dynamics of proteins despite its simplifications leads to two main conclusions. First, the irregular energy surface of biomolecules, which contains several local minima, can be approximated by a quadratic function. Second, protein collective movements are insensitive to sequence details or underlying force field, and are, to a large degree, topology dependent.

In any case, the drastic reduction of the degrees of freedom involved in the ANM approximation introduces serious technical problems in its implementation: translating the motion to the rest of atoms not present in the coarse grain (CG) model is not a trivial task.

1.5 PELE: Achievements and limitations

1.5.1 The Metropolis Monte Carlo (MMC) Algorithm

Monte Carlo algorithms are a class of stochastic algorithms frequently used in physics, engineering, economy and several other disciplines. The MMC [106] algorithm is a Markov Chain Monte Carlo technique that, as other techniques of the same family, can be used to sample high-dimensional probability distributions and obtain statistical estimates that would not be feasible otherwise

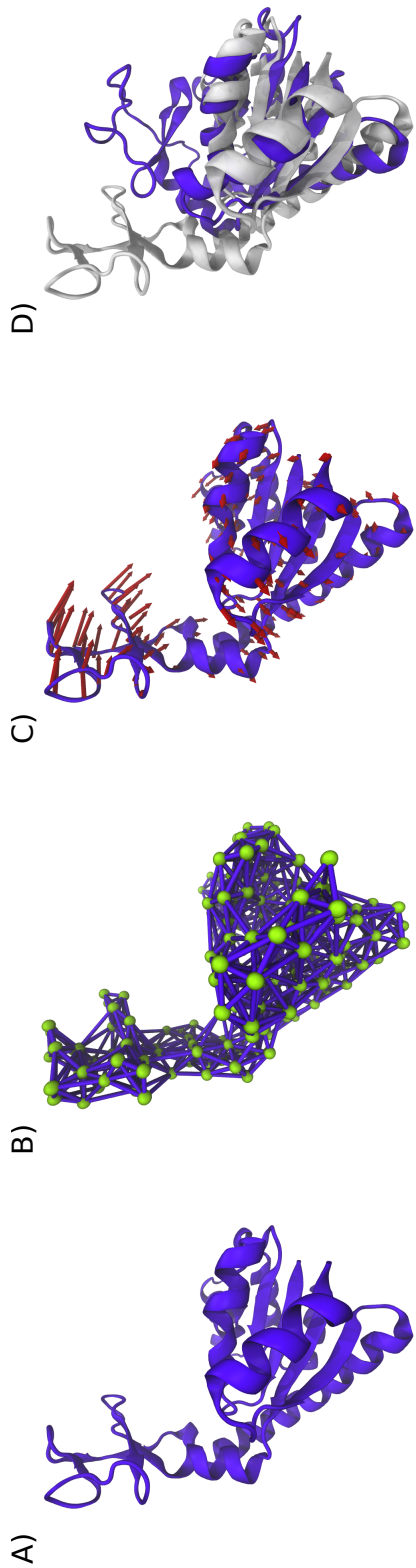


Figure 1.4: ANM study of an open conformation of adenylate kinase (PDB id: 4AKE). From the starting structure (A) the elastic network is calculated (B). Once the normal modes are obtained (C) the structure can be modified so that the final conformation is similar to the closed one (PDB id: 1AKE) (D). In this case, these open and close states may be present without the need of ligand interaction [105].

[107]. MMC generates an ergodic[15] Markov Chain whose stationary distribution is proportional to the probability distribution of interest.

The algorithm looks as follows:

- First, initialize x_0 to a value in the domain.
- For i in $0..n$ do:
  - Generate a proposal x_p (choose from the proposal distribution q).
  - Generate a random value u from the uniform distribution $[0, 1]$.
  - Calculate the acceptance probability:

$$\alpha(x_i, x_p) = min\left(1, \frac{p(x_p)q(x_i|x_p)}{p(x_i)q(x_p|x_i)}\right), \tag{1.6}$$

    which, if q is chosen to be symmetric becomes:

$$\alpha(x_i, x_p) = min\left(1, \frac{p(x_p)}{p(x_i)}\right). \tag{1.7}$$

  - Accept or reject the proposal so that if $u < \alpha(x_i, x_p)$ then $x_{i+1} = x_p$ or $x_{i+1} = x_i$ instead.

If we want to get samples from systems following the canonical distribution (constant number of particles, volume and temperature, NVT), we need to remember that, according to statistical mechanics, the probability that a system is found in a given energy state with energy E_i is proportional to the Boltzmann factor: $\exp(-\frac{E_i}{k_B T})$

If this probability distribution is used, then the acceptance probability becomes:

$$\alpha(x_i, x_p) = min\left(1, e^{-\frac{\Delta E}{k_B T}}\right), \tag{1.8}$$

where $\Delta E = E(x_p) - E(x_i)$.

The algorithm guarantees that, eventually, all states accessible for a given temperature are visited, no matter which state we started in.

The MMC method allows us to sample the Boltzmann distribution, being able to calculate thermodynamic properties by ensemble averaging. It is often used to simulate the behaviour of biomolecules and has become part of many ligand-protein docking solutions like ICM [46], Prodock [108], AutoDock [41] or MCDOCK [109]. Some of the drawbacks of MMC are:

[15] Ergodic implies aperiodicity, i.e. there is a path to move to any state from any other state, and irreducibility, which means that all probabilities to move are positive.

- As there is no temporal relationship between samples, the dynamics of the system cannot be studied.
- The simulation needs to converge as the ensemble of initial samples may not follow the desired probability distribution[16].
- The jump size needs to be adjusted in order to avoid excessive correlation of the samples.
- The probability of rejection increases exponentially with the number of dimensions, unless very small jump sizes are used.

Overall, the application of MC methods in large biological systems is scarce (compared to other sampling techniques like MD). Development of new heuristic MC methods, such as PELE, aimed at addressing this point.

1.5.2 The PELE software

PELE (Protein Energy Landscape Exploration) [110, 111] was designed as an alternative to protein conformation sampling and ligand-binding simulations. It implements an MMC scheme where each iteration consists of a perturbation and a relaxation step followed by a Metropolis test. In the perturbation step, the ligand is translated and rotated to a new position, and its conformation is changed, if needed. Afterwards, the protein backbone is modified according to an NMA-based algorithm. During the relaxation step, the side chains with higher potential energies are changed using a rotamer library. A modified Truncated Newton algorithm [112] is then used in order to further lower the energy of the system. Finally, the global potential energy is calculated, and a Metropolis test is performed. If the new system state is accepted, it will be used as the initial configuration of the next iteration, otherwise it will be discarded .

PELE implements the OPLS (Optimized Potential for Liquid Simulations) [113, 114] and the AMBER (Assisted Model Building with Energy Refinement) force fields [115], as well as the Surface Generalized Born (SGB) [116, 117], Variable Dielectric Generalized Born [118] and the Onufriev-Bashford-Case (OBC) [119] implicit solvent models.

1.5.2.1 Treatment of flexibility in PELE

We can classify the movements induced by molecular flexibility depending on the scale at which they act. We can find, for instance, subtle local side chain

[16]This initial ensemble is also called ”the burn-in period” and is frequently discarded, even if this practice is not justified by the MCMC theory.

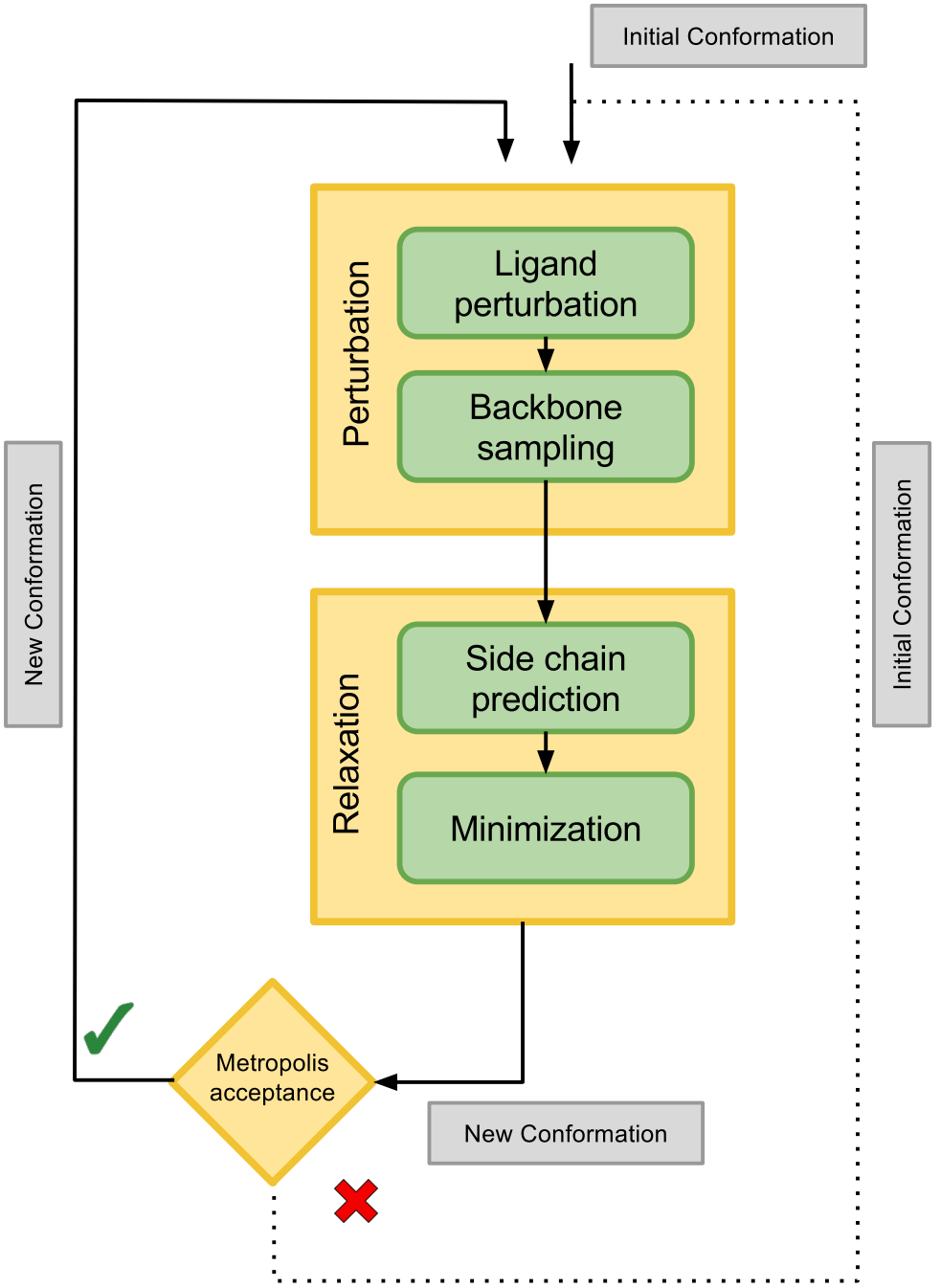


Figure 1.5: Schematic representation of PELE flux. The initial conformation goes through four perturbation/relaxation steps, after which the acceptance criterion is tested. If the final conformation is accepted, it will be used as the initial conformation of a new iteration. If it is rejected, a new iteration will be started with the same initial conformation.

movements, medium scale motions performed by loops and, finally, large-scale collective motions made by domains. Through its four substeps, PELE is able to reproduce conformational changes mainly in the local and global levels of detail.

In PELE, local molecular flexibility is handled during the side chain prediction and ligand perturbation steps. In brief, the side chain prediction methodology [120, 121] implemented in PELE uses precalculated rotamer libraries [122] to modify the torsion angles of a given percentage of the most energetic side chains so that the overall energy is lowered (and possible clashes are eliminated). Chosen side chains might be selected based on their distance to the ligand or as a result of their increase in energy during the perturbation step. In the case of the ligand, a core chemical group (typically the largest rigid group) is determined so that the length of the flexible groups attached is minimum. Finally, rotamer libraries for the ligand are built on the flight and applied, to select a more energetically favorable conformation.

1.5.2.2 Backbone flexibility

An ANM-based technique [123] is used to reproduce the backbone flexibility (see Fig. 1.6). The coarse-grained EN defines each node as a particle centered in each residue alpha carbon position. The spring force constant used for the Hookean potential follows the formulae and parameterizations described by Atilgan [92] and Eyal [124]. As all nodes are required to have equal mass, the lowest frequency modes can be obtained by using the mass-weighted version of Eq. 1.4. Large amplitude motions can be described using only a set of the lower frequency modes [125, 126], and that is why no more than six modes are usually calculated.

The direction of the conformational change is computed as a linear combination of the eigenvectors. The magnitudes of these translations are eventually scaled in order to allow a maximum displacement and a sense for the translation is chosen; the application of this translation will produce the conformation proposal. It is worth saying that several of the parameters used in this calculations can be defined by the user.

As previously mentioned, the way modes are applied to the initial structure in order to reproduce protein dynamics is not trivial, and several techniques have been suggested [127]. The most popular technique is moving the atoms following a direction that comes from a combination of modes (as illustrated above). The drawback of these interpolation methods is that the details of the movement are only known for the atoms included in the EN, making it unfeasible for all-atom approaches unless the movement of the excluded atoms is approximated. This last problem has been circumvented in PELE by adding

harmonic constraints between the initial alpha carbon positions and the target positions to perform a global minimization. In this way, all atoms follow the alpha carbons, which are pulled gently to their new positions without compromising the covalent structure.

1.5.2.3 Minimization

The global minimization step also plays an important role in the flexibility representation; it emphasizes induced fit effects and adds an anharmonic factor through the use of the overall force field and implicit solvent. Since minimization could potentially revert changes in the backbone, a set of weak harmonic constraints is added to restrain the movement of the C_α atoms.

1.5.2.4 Major achievements to the date

PELE has proven to be useful in all atom protein studies [123] thanks to its efficient conformational sampling. It has also been successfully used to unveil the mechanism of protein-ligand interactions [128–130], including the analysis of mutational effects on ligand delivery [131]. Finally, it has been used to describe ligand thermodynamics with a lower computational cost than other more consolidated methods like MD [132, 133]. More recently PELE has been used to rationalize enzymatic hydroxylation of steroids [134] and vitamin D [135] as well as directed evolution experiments [136].

1.5.3 Limitations

Although the usefulness of PELE has been amply demonstrated, the weak points of the methods used in each step cause some weaknesses in the software. Therefore, before suggesting any improvements, we needed to find these limitations and evaluate how they could affect its performance.

1.5.3.1 Force field and solvation model

Potential energy is usually calculated using the OPLS or AMBER force fields. It is well known that empirical force fields are biased towards certain types of secondary structure [66, 67, 69]. The impact of this bias on PELE has not been studied yet. However, as the improvement of force field parameterizations is under continuous research and revisions are made available to the public every few years, using updated parameterizations would be recommended.

Moreover, the implicit solvation model used seems to bias the equilibrium towards compact structures. In this scenario, the use of a minimization acts as a

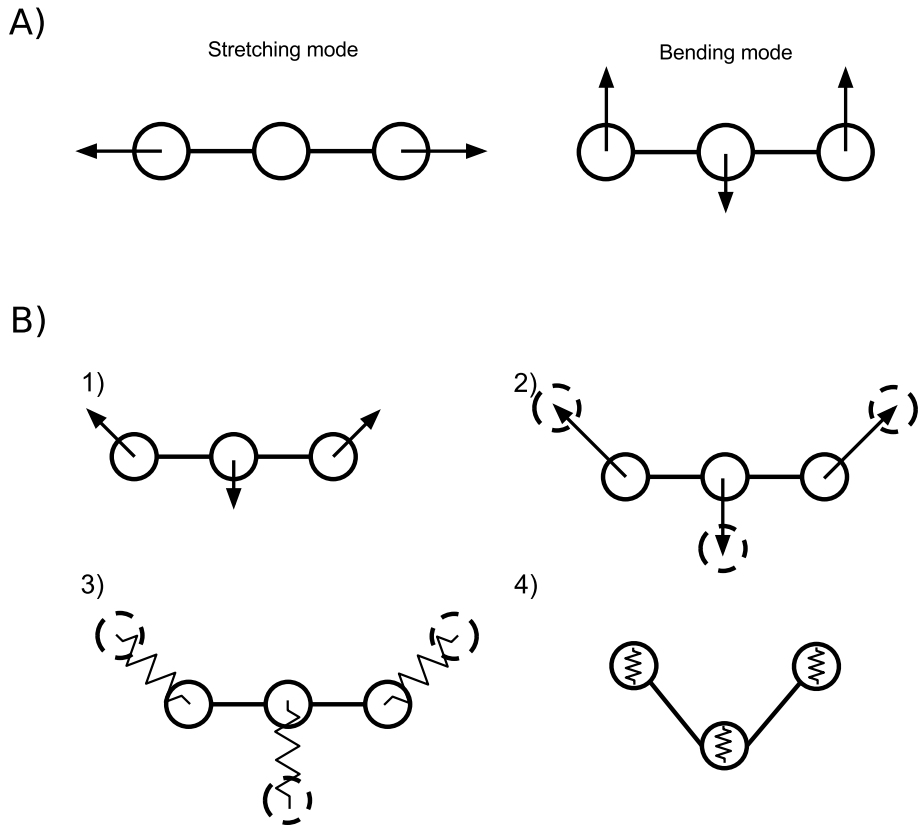


Figure 1.6: A) Two normal modes of a triatomic molecule. B) Application of those normal modes following PELE algorithm. A linear combination of the modes (here with weights 1,1) defines the directions of the conformational change (1). These directions are scaled in order to obtain the new positions for the particles of the system (2). Harmonic constraints to target positions are added and C_α atoms are moved through a minimization (3). Finally, in the last global minimization step, weak harmonic constraints are added to C_α the current position of the atoms so that the movement is not undone (4).

bias amplifier that can eventually confine sampling to certain metastable states. It is unclear to which extent the minimization algorithm is contributing to this behaviour and whether switching to algorithms that have shown to perform better, like the quasi-Newton method BFGS (Broyden-Fletcher-Goldfarb-Shanno) [137], would help to lessen the bias.

1.5.3.2 Metropolis MC

Under equilibrium conditions, the probability T of going to state j from state i must be equal to the probability to return from j to the initial state ($T_{ij} = T_{ji}$).This is known as microscopic reversibility. The equilibrium is kept by balancing the flux between these states ($f_i T_{ij} = f_j T_{ji}$). This is known as detailed balance and it is sufficient to guarantee ergodicity.

As an illustrative example, a positive increment of about 1.3 kcal at 300 K would have an acceptance probability of only 10%. As PELE is moving numerous degrees of freedom at the same time (it does atomic-detail simulations), the energy increments are bound to be bigger. Having good acceptance rates could be tough without minimization. However, using minimizations breaks microscopic reversibility, which is a necessary requirement for detailed balance. Attractive states can potentially appear so that $f_i T_{ij} \neq f_j T_{ji}$. Therefore, the convergence towards Boltzmann distribution cannot be warranted and no calculated thermodynamical average quantity should be trusted.

Despite that, PELE has had undoubtful success simulating protein flexibility mechanisms and ligand-protein interactions. This is due to the combination of MC techniques with protein structure prediction algorithms, which allow to move between distant important regions of the conformational space.

1.5.3.3 ANM methodology

The ability of NMA-related methodologies to predict collective conformational changes has been already discused in Section 1.4.2.3, however, there is still room for improvement:

- The theory restricts atomic translations to differential movements around the potential minimum. In practice, this restriction is not honored, and atomic translations tend to be very wide with results that are still in agreement with experimental data. A more correct way of using NMA-based atomic translations would be applying tiny displacements and recalculate the modes before starting a new iteration.

- As the deformation of the protein progresses, the EN evolves, and the updated normal modes may not contain the translational information of

interest. To illustrate this we have downloaded a 10ns MD simulation[17] of porcine adenylate kinase (Protein Data Bank (PDB) id.: 3ADK) from the MODEL [138] database. In this trajectory, the protein performs a conformational change from the initial open form to the closed form (see Fig. 1.7A). One can measure how close the domains are by measuring the distance between the atoms LYS64:CA and THR136:CA. We have extracted seven frames showing different stages of the "closing" process, and then, we have calculated their normal modes using VMD [139] and Prody [140]. We have also obtained the translational vectors that would move the protein directly from its open conformation (33 Å) to the closed one (4 Å). In Fig. 1.7B, we can see the cumulative overlap between the modes of the open conformation (33 Å) and all the modes of all the other conformations, as well as the maximum value of the overlap between the modes of the first conformation and the modes of all the other conformations. Again, as the distance between domains decreases, so does the ability of the modes to explain the modes of the first conformation.

$$O_{ij} = \frac{|P_i M_j|}{||P_i||||M_j||} \tag{1.9}$$

$$CO(k) = \left(\sum_{j=1}^{k} O_{ij}^2 \right)^{\frac{1}{2}} \tag{1.10}$$

Modes change noticeably as the EN evolves (see Fig. 1.7A) which enforces the idea (coming from the theoretical basis of NMA) that a given set of modes is only valid while the structure has not undergone a big change. If the modes of the first conformation are in better agreement with the desired change, as in this example, the mode calculation will be limited to the first step, and the initial conformation will be the "open" one (already discussed elsewhere [141]). If this is not the case, the method loses its ability to predict well studied dynamic behaviours like the open-close transition mentioned above, thus diminishing the usefulness of the approach. If we calculate the maximum value for the overlap (Eq. 1.9 [141]) of the modes of each conformation with the open to close translation vector, we can observe that the first mode is usually the one that better explains this conformational change. However, this similarity decreases as the protein closes and the EN changes. The cumulative overlap (Eq. 1.10 [142]) of the translational vector with all the modes of each conformation shows to which extent modes find it difficult to repro-

[17] Using the AMBER 8.0 force field and explicit solvent.

duce the conformational transition (see Fig. 1.7C). However, using the same modes for too long is not in agreement with the theoretical basis of NMA, as seen above.

- Protein dynamics is known to be highly anharmonic [143], but NMA is completely harmonic by definition. This can lead to an underestimation of the mean square fluctuation of residues [144]. The frequent recalculation of modes would help to add anharmonicity to simulations. In PELE, the minimization of the energy function, which includes the force field and solvation term, also adds anharmonicity.

- The first low-energy modes coming from ANM calculations are usually enough to describe wide domain collective motions. However, it lacks the information needed to model local flexibility, like folding/unfolding events, which may be a major issue if such conformational changes are part of a binding mechanism.

- NMA methodologies do not treat solvation effects explicitly.

- There is no time information in NMA-based simulations, and it is not possible to know the time scale of the modeled conformational transitions.

- Side chains are considered as rigid bodies, which may help to generate conformations with steric clashes. Also, regular ANM is amino-acid type agnostic and it is difficult to use to study mutations. This last issue can be solved by using improved coarse grain models [145] that take this information into account.

- Selecting the modes to be used is not trivial. The number of unweighted choices is proportional to 2^m, being m the number of modes. Also, not all mode combinations match with the direction of a real conformational transition. PELE allows users to define which modes to use, how to combine them and when to change directions. However, with the exception of random choices, the user must know the dynamics of the system beforehand in order to take profit of these options.

- The linear combination of ANM modes can produce many possible movements, but not all combinations are necessarily correct. Also, the application of the modes through linear interpolations can destroy the covalent structure of the protein. In PELE, this is handled by applying the modes through a minimization (however this can worsen the bias introduced by the potential energy definition).

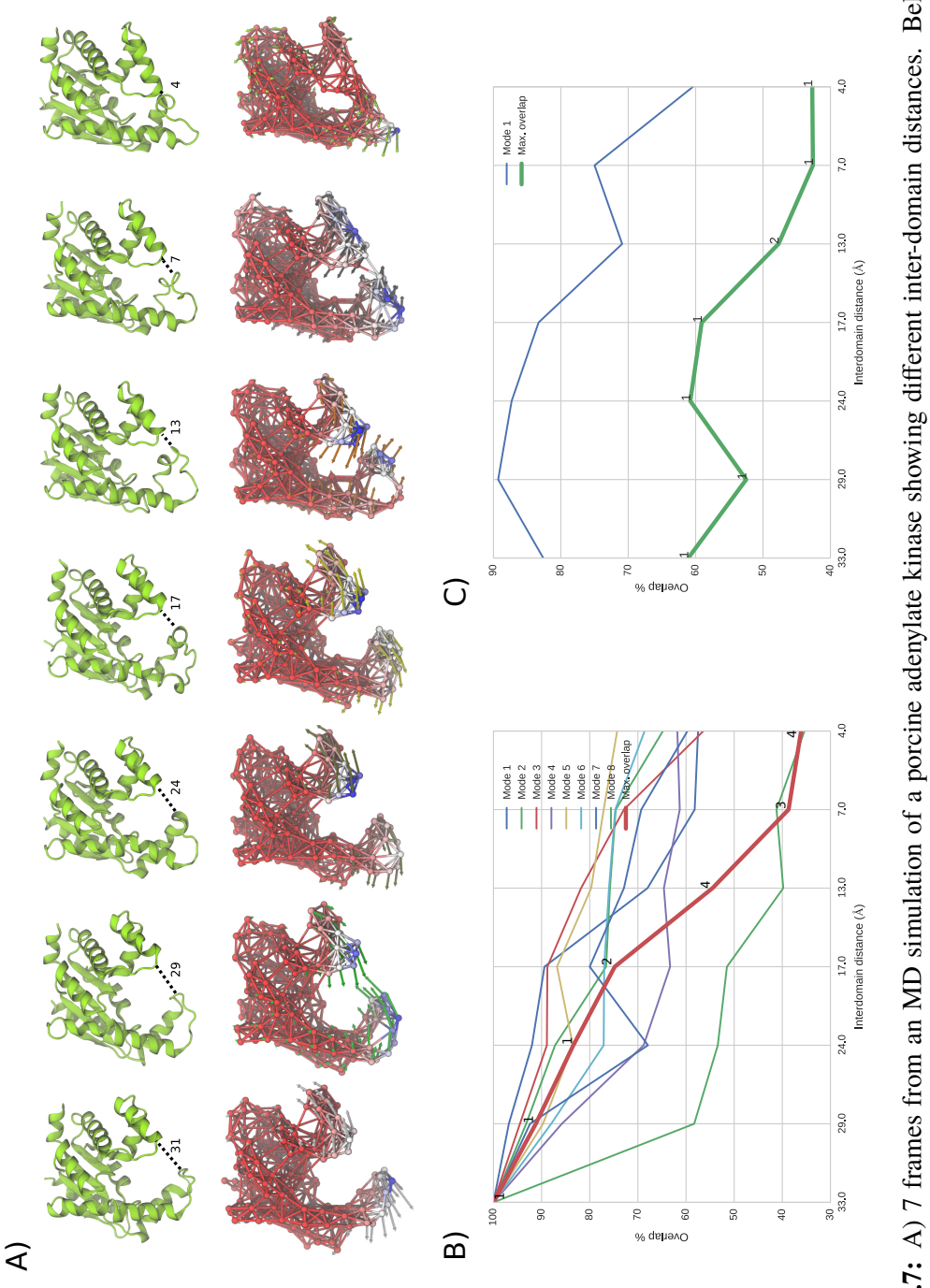


Figure 1.7: A) 7 frames from an MD simulation of a porcine adenylate kinase showing different inter-domain distances. Below each frame, its elastic network has been reproduced. The EN changes with the inter-domain distance, being especially perceptible from the 17 Å. B) Cumulative overlap and maximum value of the overlap (including the index of its related mode) between the modes of the first conformation and all the others. C) Cumulative overlap between the linear displacement from the open to the closed conformations and the modes of each frame and the maximum overlap (again, the index of the mode with maximum overlap is also shown).

- The so-called ‘tip effect’ is an artifact happening in proteins where there are different packing density zones. Less dense regions, like loose loops at the beginning or at the end of the protein, will be considered highly flexible. As the magnitude of the movement is determined by this relative flexibility, structured zones are very likely to lose mobility, reducing the sampling of those parts. Recently, there have been some efforts to reduce the ‘tip effect’ by taking advantage of the Hessian robustness [146].

2

Objectives

As we have seen in the introduction, the pharmaceutical industry is actively looking for new ways of boosting the efficiency and effectiveness of their R&D programmes. The use of computational modeling tools in the drug discovery pipeline is having a positive impact on research performance, since in silico experiments are usually faster and cheaper that their real counterparts.

We can envisage a scenario where almost all steps of the drug discovery pipeline are performed by fast and specialized software[1] running on custom hardware architectures.

This vision can only be achieved through technical improvements, both in hardware and software, and through the research of new and more robust algorithms that can improve the accuracy and quality of results. In particular, we believe that these developments will be important in improving conformational sampling in VHTS, where current techniques do not allow screening thousands of compounds accurately. This thesis aims to work in this line, turning PELE into a faster and more efficient tool to add receptor flexibility. Besides, we have addressed the difficulties of analyzing extensive data associated with massive simulation production.

In this work, we will focus on the improvements achieved in PELE.

[1]The company Nimbus Therapeutics is currently one of the best examples of this vision. Its drug discovery processes rely heavily in computational tools. It is currently in partnership with Monsanto Growth Ventures and Shire HTG (Human Genetics Group) and has attracted top players in the pharmaceutical (Pfizer) and technological industries (Bill Gates, Shrödinger).

2.1 Objective: Technical improvement of PELE

PELE software is currently well established in the academic environment and is already penetrating the pre-discovery phase thanks to its atomic-detail conformational sampling and protein-ligand binding prediction capabilities. This software is able to perform faster simulations than MD and is more accurate but still slower than regular VHTS software. Unfortunately, performance is a matter of concern to VHTS protocols, as the size of compound libraries can be vast.

PELE could clearly earn a place in VS protocols if execution times werelowered. Improving its performance would imply optimizing the code, and adapting it to take full advantage of the newest parallel hardware architectures. One of the goals of this work is to **enhance the technical features of PELE by performing a complete rewriting of its code in order to optimize and parallelize its most computationally demanding parts**.

2.2 Objective: Algorithmic improvement of PELE

As highlighted in the introduction, in the context of protein-ligand interactions the way algorithms model flexibility is one of the keys to success. Probably, this is the reason why finding the balance between accurate flexibility modelling and computational performance has become a matter of concern for the development of new software. This is especially important for the tools typically used in VHTS pipelines, where a detailed simulation of flexibility is usually sacrificed for the sake of speed. However, improving the reliability of solutions would help to produce less false positives and less false negatives, thus favoring the successive stages of the drug discovery pipeline. **Our goal is to perfect and speed up PELE flexibility handling (with a minimum performance impact) in order to improve the quality of its results and convert it into an alternative to current VHTS software**.

2.3 Objective: Efficient and reliable analysis of huge conformational ensembles

Performance improvements are usually translated into a shortening of execution time. Scientists working with conformational sampling and ligand binding simulation software often make the most of this extra time in three ways: i) us-

ing more accurate and computationally intensive algorithms that increase the theoretical correctness and quality of results, ii) increasing the size of the systems studied or, iii) choosing to run simulations over longer periods of time. In this last scenario, it is very likely that the size of the output grows considerably.

Analyzing large sets of conformations is highly demanding due, mainly, to the own nature of this data. Indeed, structural superimposition is the requirement of many popular conformational analysis methods, and this does require a significative amount of computational power and time. Moreover, software implementing superimposition algorithms are often limited to the pairwise case, which makes them an underperforming solution when applied to ensembles. **In order to avoid this, we aim to implement an efficient solution for the calculation of collective superimposition operations**.

Also, as the size of results becomes larger, so does the difficulty of analyzing them and the chances of making errors. Cluster analysis techniques, which are unsupervised machine learning methods, have become a standard solution to this issue. However, its results can be unpredictable if used as a black box and no further validation checks are performed. **We want to address this challenge through the implementation of a reliable cluster analysis protocol**.

2.4 Summary of the objectives

1. Technical improvement of PELE
   (a) Complete rewrite of the code
   (b) Optimization and parallelization of the most computationally demanding parts
2. Algorithmic improvement of PELE
   (a) Perfect PELE's flexibility handling in order to improve results quality
3. Efficient and reliable analysis of large conformational ensembles
   (a) Implementation of an efficient solution for the calculation of collective superimposition operations
   (b) Implementation of a reliable cluster analysis protocol

3

Articles

In this chapter, we present the scientific production relevant to the three objectives we proposed previously. The three articles presented here include supporting materials that complement the reading. We have decided to add them here and to make some enhancements in order to improve their integration with the rest of the document. Some of the changes we have introduced are: correction of minor spelling and grammar errors, rework of figures (whenever possible), and addition of their references to the main bibliography.

The author also proposed and directed three Computer Engineering Degree Projects, all of them for the Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya, which are also related to the objectives.

3.1 Technical improvement of PELE

Due to the complexity of the project, which is in continuous evolution, no publication has yet been issued related to Objective 1.a and Objective 1.b. Despite this, as the software is nowadays reaching its maturity, we do not discard to release a publication concerning its new features and technical improvements in the near future. A benchmark using the new version of the code has been recently published [147].

3.1.1 Computer Engineering Degree Project

Paral.lelització del software de simulació PELE++ utilitzant GPUs by Xavier Orò Gay *et al.* [148] (2012)

3.2 Algorithmic improvement of PELE

We present the draft of an article regarding Objective 2.a, which will be submitted as soon as possible to a scientific journal. Again, we have included its supplementary materials.

Enhancing backbone sampling in Monte Carlo simulations using Internal Coordinates Normal Mode Analysis

Author : Víctor A. Gil
Affiliation: Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain
Author : Daniel Lecina-Casas

Affiliation: Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain
Author : Christoph Grebner

Affiliation: Department of Medicinal Chemistry, CVMD iMed, AstraZeneca, S-43183 Mölndal, Sweden
Author: Víctor Guallar

Affiliation: Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain and Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig Lluís Companys 23, E-08010 Barcelona, Spain

3.2.1 Role of the authors

The author of this thesis was the main responsible for: developing the methods introduced in each publication, planning and running the analysis required for the evaluation of the methods, as well as writing the articles themselves. The collaborators were involved in the following processes: suggesting theoretical improvements, relating the analyses with the biological background of the systems, and adding contents to and proofreading the articles.

3.2.2 Computer Engineering Degree Project

Análisis vibracional de proteínas en coordenadas internas mediante el modelo ANM by Alba Rincón Muñoz *et al.* [149] (2014)

3.3 Efficient and reliable analysis of large conformational ensembles

Finally, we present two publications concerning objectives 3.a and 3.b. These articles are reproduced in the next sections. Their details can be found below:

pyRMSD: a Python package for efficient pairwise RMSD matrix calculation and handling.

Author : Víctor A. Gil
Affiliation: Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain
Author: Víctor Guallar

Affiliation: Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain and Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig Lluís Companys 23, E-08010 Barcelona, Spain
Journal: Bioinformatics (15th September 2013)

Journal impact factor: 5.498 (02/03/2016)

pyProCT: Automated Cluster Analysis for Structural Bioinformatics

Author : Víctor A. Gil
Affiliation: Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain
Author: Víctor Guallar

Affiliation: Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain and Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig Lluís Companys 23, E-08010 Barcelona, Spain
Journal: Journal of Chemical Theory and Computation (18th July 2014)

Journal impact factor: 4.981 (02/03/2016)

The permission to reproduce these articles is granted by Oxford Open license [1] in the first case, and the ACS permission [2] in the second.

[1] http://www.oxfordjournals.org/our_journals/bioinformatics/for_authors/creativecommons.pdf

[2] http://pubs.acs.org/userimages/ContentEditor/1218205107465/

3.3.1 Computer Engineering Degree Project

Optimization of the cluster analysis tool pyProCT with pyCOMPSs by Pol Alvarez Vecino *et al.* [150] (2015)

3.4 Derived publications

During the development of this thesis, the author also contributed in the elaboration of other scientific publications. In these works he programmed supporting software, carried out the analysis of the results, and performed writing tasks:

- *Monte Carlo free ligand diffusion with Markov state model analysis and absolute binding free energy calculations* by Ryoji Takahashi *et al.* [132] (2013)
- *Nucleoside inhibitors of tick-borne encephalitis virus* by Eyer *et al.* [151] (2015)
- *Computational Prediction of HIV-1 Resistance to Protease Inhibitors* by Ali Hosseini *et al.* [152] (2016)

The thesis director, **Víctor Guallar Tasies**, certifies that all the information above is accurate and that the articles presented are not part of other theses or works of any kind.

At Barcelona, ... 2016:

Enhancing backbone sampling in Monte Carlo simulations using Internal Coordinates Normal Mode Analysis

Victor A. Gil,[†] Daniel Lecina-Casas,[†] Christoph Grebner,[‡] and Victor Guallar[*,†,¶]

[†]*Joint BSC-CRG-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain*

[‡]*Department of Medicinal Chemistry, CVMD iMed, AstraZeneca, S-43183 Mölndal, Sweden*

[¶]*Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig Lluís Companys 23, E-08010 Barcelona, Spain*

E-mail: victor.guallar@bsc.es

Abstract

Normal mode methods are becoming a popular alternative to sample the conformational landscape of proteins. In this study, we describe the implementation of an internal coordinate normal mode analysis method and its application in exploring protein flexibility by using the Monte Carlo method PELE. This new method alternates two different stages, a perturbation of the backbone through the application of torsional normal modes, and a resampling of the side chains. We have evaluated the new approach using two test systems, ubiquitin and c-Src kinase, and the differences to the original ANM method are assessed by comparing both results to reference molecular dynamics simulations. The results suggest that the sampled phase space in the internal coordinate approach is closer to the molecular dynamics phase space than the one coming from a Cartesian coordinate anisotropic network model. In addition, the new

method shows a great speedup (∼5-7x), making it a good candidate for future normal mode implementations in Monte Carlo methods.

1 Introduction

Computational experiments are, in general, easier to prepare, and faster and cheaper to perform than their real-life counterparts. Furthermore, they allow scientist to have a privileged view of the systems under study, providing spatial and (often) timescale resolution that cannot be achieved by any other mean.

Of all the features that can be simulated in biopolymers, flexibility is one of the most difficult to reproduce and still remains a major challenge. The biological role of protein flexibility is highly relevant, being tightly related to protein function and ligand binding mechanisms, the understanding of which is a central aspect in computer-aided drug design. The correct handling of ligand and receptor flexibility is increasingly seen as critical for the success of virtual screening experiments.[1] To this aim, several approaches have been proposed to introduce flexibility: docking on multiple receptor conformations,[2,3] rotamer sampling on side chains and ligands,[4] molecular dynamics,[5] stochastic techniques,[6,7] etc.

The most known and used technique to sample protein flexibility, with great success, is molecular dynamics (MD). Long scale MD simulations can be used to describe protein folding or even to capture ligand binding events efficiently.[8,9] However, these simulations require large computational resources or specialized hardware, which limits their use. In order to be applicable in computer-aided drug design, the used sampling techniques need to be fast and efficient allowing for reasonable short timelines. This is also the main reason why a detailed handling of flexibility is greatly simplified in methods where speed is desirable, such as the ones typically used in virtual high-throughput screening (VHTS). Recent hardware improvements, especially in accelerators such as GPUs, are spreading the use of MD simulations in drug design projects,[10] however, they cannot be applied routinely in VHTS pipelines, where

thousands of compounds have to be tested.

An alternative to MD are Monte Carlo (MC) techniques.[11] These methods have the capability of generating uncorrelated samples in two subsequent steps, allowing to quickly traverse the conformational space. This property permits, theoretically, to sample the conformational space more effectively and in a computationally tractable way. However, depending on the problem, the difficulty of generating new likely poses may largely drop its performance.[12] In particular, when modeling large biomolecules, MC methods have difficulties in efficiently sampling all degrees of freedom;[13,14] most successful MC studies (and methods) focus on small-medium polypeptides and/or local sampling, such as loops and side chains.[15,16] The Protein Energy Landscape Exploration (PELE) software[7] aims to overcome these sampling problems by adding protein structure prediction techniques in the MC sampling iteration.

PELE has shown to be successful in many tasks like sampling protein flexibility,[17] protein-ligand interactions,[18–20] enzyme engineering[21,22] and to describe ligand thermodynamics.[23]

1.1 The Anisotropic Network Model (ANM) in PELE

PELE is implemented as an iterative procedure where each MC iteration is composed of a perturbation and a relaxation phase. In the perturbation phase, the protein backbone is modified using an approach based on the ANM, a type of Normal Mode Analysis (NMA) first introduced by Atilgan and coworkers.[24] This model simplifies the protein potential as a harmonic elastic network of coarse grain units (typically C_α atoms) oscillating around a stable equilibrium conformation. These kinds of methods have shown to be successful in the study of large amplitude transitions in proteins, which are often involved in biological functions. Despite their simplicity, they have been shown to have good predictive capabilities and correlate well with experimental data.[25,26]

Once the elastic network has been defined and the Hessian ($H_{ij} = \frac{\partial^2 V}{\partial q_i \partial q_j}$) is calculated, eigenvectors and eigenvalues can be obtained by solving $HA = A\Lambda$.

Eigenvalues (Λ) are associated with the frequencies of the modes and the energetic cost of

moving the system along its related eigenvector (A). Usually, only the first lower frequency modes are retrieved, as they are enough to describe wide conformational movements.[27] Keep in mind that higher frequency modes, associated with C_α -C_α stretching, do not have physical meaning. These lower frequency modes have shown to have a high degree of correlation with MD essential space.[28]

In order to apply the ANM modes in PELE, a subset of the lower energy eigenvectors is chosen (either randomly or as defined by the user) and a linear combination of them is calculated. The translation of the current coordinates by the resultant vector defines a set of target coordinates: the ANM conformation proposal. The conformational change is eventually calculated by adding harmonic constraints between the initial position of each C_α atom and their position in the proposal to finally perform a minimization that will move the atoms close to their target positions.

Because of the chosen strategy to apply the modes during the ANM step, the covalent geometry of the protein can be slightly distorted, producing large energy increments. In order to overcome this, PELE enters a relaxation stage where the energy is lowered. First, the torsional angles of the most energetic side chains are changed using a side chain prediction algorithm and a library of predefined rotamers. Second, a global minimization is performed in order to further lower the energy. In this system-wide minimization, a weak constraint is added to the C_α atoms to prevent the minimization from undoing the perturbation backbone move.

Finally, a metropolis acceptance step is performed using the Boltzmann criterion, and the current conformation is accepted or rejected. Thus, the overall approach is quite different from most MC implementations: each MC step involves a significant number of protein structure prediction techniques, requiring a remarkable amount of computational time ($\sim$1 minute on average), but is capable of introducing a large collective displacement. Notice that, by using these methods, we introduce importance sampling toward feasible structures (also adding limitations, see below).

One of the consequences of the use of minimizations is the loss of detailed balance, which implies that a complete exploration of all accessible states is not guaranteed. Also, we observe that, under certain circumstances, the combination of the profuse use of minimizations (twice every MC iteration) and the implicit solvent model adds a bias towards compact conformations.

1.2 Internal coordinates conformational sampling

Internal coordinates (IC) are a set of interdependent coordinates that include the distance, angle and dihedral angle between atoms, and are supposed to be a more natural way of representing and manipulating chemical entities.

Conformational sampling in torsional space is, in general, more efficient than in Cartesian coordinate (CC) space. The main reason is that changes in the torsional degrees of freedom often show low energy barriers, while bending and angular changes have higher energy barriers. This idea has often been used to enhance the sampling of MC algorithms.[29–32]

It is also possible to calculate normal modes in the internal coordinates space. Our preliminary analysis of torsional modes showed that they are more collective (see Section S1). This is an indicator of a less severe "tip effect", produced by a concentration of the modes in loose loops that can potentially nullify the movements in more structured regions, slowing down sampling. Besides, their degree of collectivity is less dependent on the topology of the elastic network (EN). When comparing them with CC modes, we have observed that both mode spaces do not always contain the same information (their overlaps range from 65% to 95%) possibly due to their differences in collectivity.

Internal coordinates NMA (icNMA) has already been used as a feasible alternative to handle protein flexibility. However, its presence in literature is incidental compared to its CC counterpart, maybe because the implementation of the first is harder. We would like to highlight the methods published by Noguti and Go,[33] Kidera *et al.*[32] and the more recent iMC method proposed by López-Blanco *et al.*[34]

In this article, we will focus on describing the improvements carried out on the PELE conformational sampling features; our long-term goal is to provide a fast and reliable backbone sampling technique to add into virtual screening refinement steps. To this end, we implemented an icNMA algorithm in PELE and adapted it to perform the backbone perturbation and thus handling protein flexibility. The resiliency to changes in the description of the elastic network, and the higher collectivity of IC modes, make them a perfect alternative for the current Cartesian coordinate NMA (ccNMA) step in PELE. Also, the way torsional changes are applied supposes a fundamental advantage compared to ccNMA: as it does not distort the covalent structure, the complex relaxation protocol is no longer needed. In order to understand the protein motion provided by the new method, as well as to assess its performance, we ran CC and IC NMA simulations for two different systems widely used in benchmarking: ubiquitin and c-Src kinase, and we compared our results with reference MD trajectories.

2 Materials and Methods

2.1 Internal coordinates normal mode analysis

The implemented IC NMA method starts by describing an elastic network of rigid units that encompass all the heavy atoms among rotatable backbone torsions. This means that two rigid units per residue are defined instead of only one as described by ANM (see Fig. 1A).

The potential of this spring network is the sum of all Hookean interactions between the units. If expressed using generalized internal coordinates (backbone dihedrals in this case) the potential can be written as:

$$V = \frac{1}{2}(q - q^0)H(q - q^0)^T, \tag{1}$$

where H is the Hessian, that in terms of q[35] can be written as:

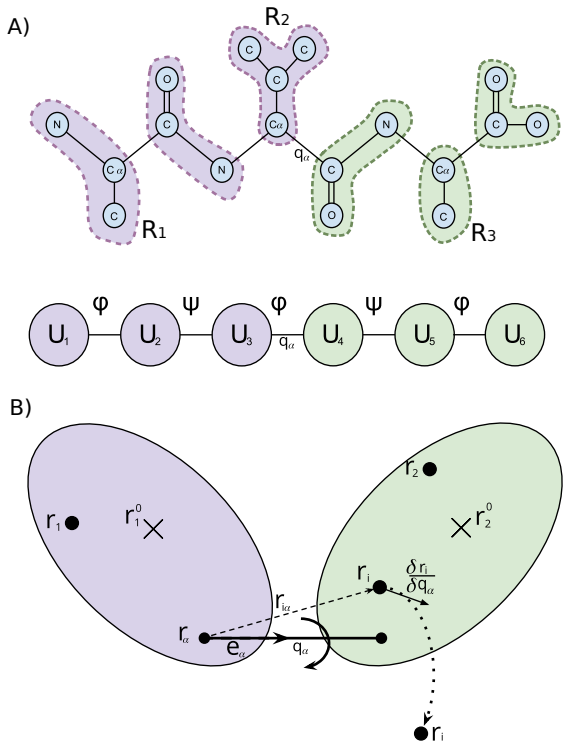


Figure 1: A) The coarse grain model defines units encompassed by the torsion angles *phi* and *psi*. B) Schematic representation of the rotation of units 4-6 (green) around torsion q_α, including the notation used in the formulae.

$$H_{\alpha,\beta} = \frac{\partial^2 V}{\partial q_\alpha \partial q_\beta} = \sum_{i<j} \frac{f_{ij}}{|r_{ij}|^2} \left\langle r_i j, \frac{\partial r_i - \partial r_j}{\partial q_\alpha} \right\rangle . \left\langle r_i j, \frac{\partial r_i - \partial r_j}{\partial q_\alpha} \right\rangle . \tag{2}$$

By imposing Eckart conditions[36] and that the origin of the molecule is the center of mass, Noguti and Go proposed an analytical solution for the partial derivatives[37] so that

$$\frac{\partial r_1}{\partial q_\alpha} = e_\alpha \times \left(\frac{M_2}{M} r_\alpha + \frac{M_1}{M} r_1^0 \right) - r_1 \times \frac{M_1 r_1^0 \times (e_\alpha \times r_\alpha) + I_2 e_\alpha)}{I} \tag{3}$$

$$\frac{\partial r_2}{\partial q_\alpha} = -e_\alpha \times \left(\frac{M_1}{M} r_\alpha + \frac{M_2}{M} r_2^0 \right) + r_2 \times \frac{M_2 r_2^0 \times (e_\alpha \times r_\alpha) + I_1 e_\alpha)}{I}, \tag{4}$$

where symbols without subscript refer to global quantities and symbols with superscript refer to the set of units to the left (1) or to the right (2) of the rotation axis, M is the mass, I the inertia, r^0 the center of mass and r are the atom positions. These derivatives describe how Cartesian coordinates change upon rotation around the e_α axis (torsion q_α), in such a way that the momentum is conserved (see Fig. 1B).

The recursive method proposed by Noguti and Go[33] and by Abe and coworkers[38] lowers memory consumption as well as computational complexity, which decreases from $\theta(n^4)$ to $\theta(n^2)$. The kinetic energy can than be expressed in terms of the generalized coordinates q:

$$K = \frac{1}{2} \dot{q}^T K \dot{q} \tag{5}$$

and the metric tensor K:

$$K_{\alpha,\beta} = \frac{\partial^2 K}{\partial \dot{q}_\alpha \partial \dot{q}_\beta} = \sum_i^n m_i \left\langle \frac{\partial r_i}{\partial q_\alpha}, \frac{\partial r_i}{\partial q_\beta} \right\rangle , \tag{6}$$

which can be calculated as:[37]

$$\begin{aligned} K_{\alpha\beta} = \frac{M_1 M_3}{M} \left[e_\alpha \times (r_\alpha - r_1^0) \right] \left[e_\beta \times (r_\beta - r_3^0) \right] + \\ \left[M_1 r_1^0 \times (e_\alpha \times r_\alpha) - I_1 e_\alpha \right] I^{-1} \left[M_3 r_3^0 \times (e_\beta \times r_\beta) - I_3 e_\beta \right] \end{aligned} \tag{7}$$

The eigenvectors and eigenvalues are obtained solving the eigenproblem $HA = KA\Lambda$. On this occasion the meaning of a eigenvector is no longer a displacement in Cartesian coordinates, but a set of differential rotations around the ϕ and ψ torsions of the protein backbone. It is worth noting that, as torsional NMA uses less degrees of freedom than ANM ($\sim$2 vs. 3 per residue), the Hessian matrix is smaller and therefore its diagonalization is faster. However, this is not going to affect the overall performance of the proposed method, as frequently modes are calculated only few times along the simulation.

2.2 IC-based sampling method

The new implemented method can be divided in two independent stages: backbone perturbation and side chain perturbation, both implemented as MC algorithms. Each iteration of the backbone perturbation stage (icNMA step) is analogue to an ANM step in regular PELE (ccNMA step). First, the increments for ϕ and ψ torsions of the backbone are calculated using the eigenvectors. Then, the resulting differential rotations are rescaled using the maximum amplitude chosen from a user-defined range $[a_{min}, a_{max}]$. The value is drawn from a normal distribution with mean $a_{min}+a_{max}/2$ and standard deviation $a_{min}-a_{max}/4$. The distribution is truncated so that, when the chosen amplitude is outside the range, the draw is repeated. The angular increments are applied using Choi's method.[39]

In general, small torsional displacements can give place to large linear displacements, favouring the appearance of steric clashes. The energy increment caused by the clashes will most likely ensure rejection of the backbone move, decreasing acceptance dramatically. As this issue is especially evident in side chains, we have chosen to minimize the energy of those side chains whose atoms collide with other atoms of the system, using PELE implementation of a Truncated Newton minimizer.[40] While this minimization breaks detailed balance, the following stage (see below) aims at recovering proper side chain sampling.

The side chain perturbation stage, also implemented as an MC algorithm, is performed right after the backbone perturbation stage. At each iteration, a residue with a side chain

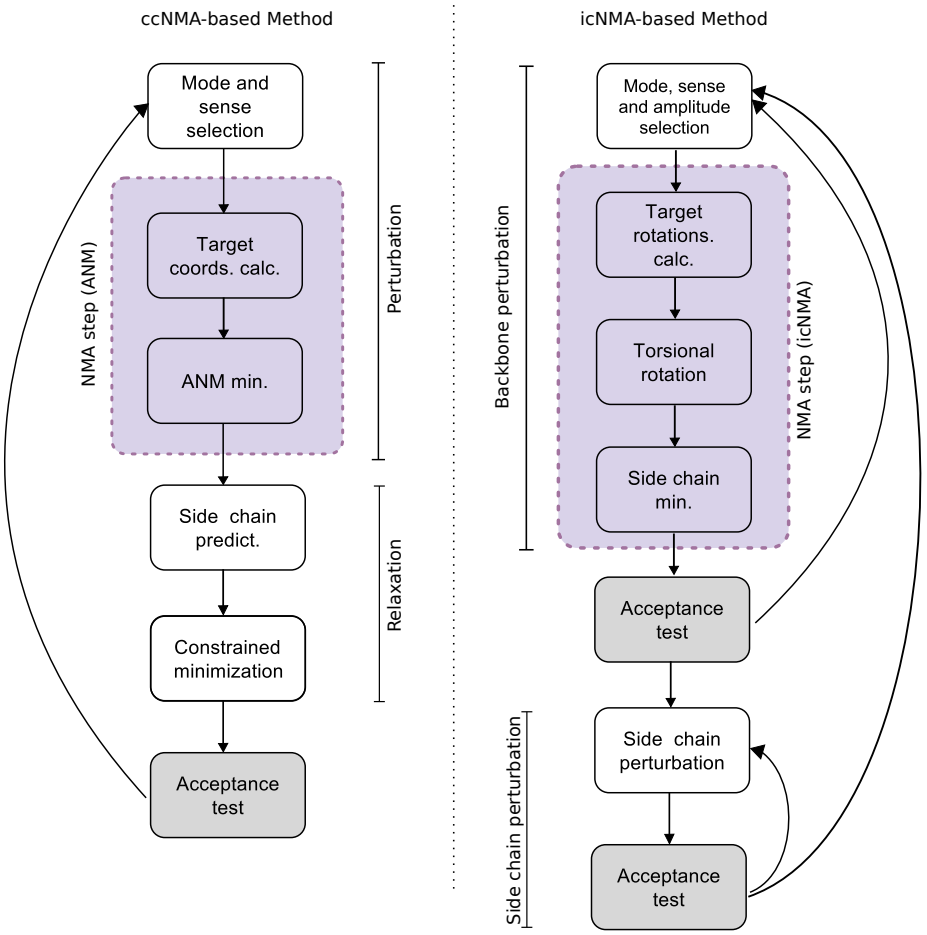


Figure 2: Diagram of the two methods currently implemented in PELE. In the ccNMA-based method, each iteration is composed of two parts: the ANM perturbation and the relaxation. The new icNMA-based method consists of two independent perturbation stages, both implemented as MC algorithms. In the first stage (backbone perturbation), the backbone is modified, in the second (side chain perturbation), a proper sampling of the side chains is recovered.

with rotatable bonds is randomly chosen. Then a random increment for each of its rotatable bonds is sampled from a truncated normal distribution (defined as before), a sense for the rotation is randomly selected, and the new side chain conformation is built. After this, the energy increment of the system is calculated and the proposal is accepted or rejected depending on the outcome of the Metropolis test.

2.2.1 Test systems and reference simulations

We have decided to use MD as a reference in order to compare the two backbone sampling methods. MD simulations are a standard approach for investigating protein dynamics, even though assessing full convergence is hard to achieve.[41] We assume that a long enough MD simulation (i.e. beyond the μs) should be capable of sampling the relevant conformational space.[42,43]

We have selected Ubiquitin (PDB id: 1UBQ) and c-Src-kinase (PDB id: 1Y57, residues 258:534) as test systems because they show very different structural and dynamic properties. Ubiquitin is a small globular protein with only 76 amino acids which can be found in nearly all eukaryotic cells. It plays a crucial role in post-translational modifications and modifies the function of substrate proteins.[44–46] It is very well investigated, both experimentally[47–52] and theoretically.[53–55] The structure is well characterized by X-ray (PDB-id: 1UBQ[47]) and NMR (e.g. PDB-id: 2MSG[52]). This makes it an ideal test system for new methods and it is often used as part of benchmark sets.[56–58] The reference MD simulation for ubiquitin is taken from a previous benchmark of PELE.[59] The trajectory was calculated with GROMACS 4.0.5[60] using the OPLSAA force field[61] and explicit solvent (a cubic box of water molecules). It was run at 300 K and 1 atm, using periodic boundary conditions. The production run was performed for 1 μs. We analysed one snapshot every 100 ps. In the PELE simulations, we omitted the last three residues of the n-terminal loop, as its high flexibility is a source of "tip effect" and masks the flexibility of the rest of the protein.

c-Src is a cytoplasmic tyrosine kinase that catalyzes the transfer of a phosphate group

from ATP to the hydroxyl group of the tyrosine residue. Src plays an important role in cellular proliferation, survival, migration, and angiogenesis.[62] It shows wide inter-domain displacements between its three well-defined domains, the N-lobe, C-lobe and hinge. Our reference MD simulation for c-Src is a 20 μs MD simulation describing the binding of PP1 (pyrimidine-type) inhibitor, a Src-selective tyrosine kinase inhibitor binding to the ATP-binding site of the kinase,[63] presented by Shan *et al.*[8] It was parameterized using a corrected[64,65] Amber99SB force field and the simulation was run using Desmond 2.2.2.1 software[66] in the Anton specialized hardware[67] with explicit water solvent. Although the simulation was performed in the presence of a ligand, we assume that only the populations of visited states[68,69] change, but the accessible conformational space does not change considerably.

2.3 Comparison analyses

In order to assess the performance of the different approaches, we carried out different analyses and compared them against our reference classical MD simulations. The chosen analyses are:

2.3.1 Root mean square fluctuation (RMSF)

The RMSF is an isotropic measure of the displacement over time of an atom from a reference position (typically the average structure after superposition of all frames). It is a common measure of conformational sampling efficiency, however it cannot be used as the only source to compare different methods, as the exploration of very different zones could lead to the same RMSF. It can be calculated as

$$RMSF \equiv \sqrt{\langle (x - \langle x \rangle)^2 \rangle} = \sqrt{\frac{1}{F} \sum_{n=1}^{F} (x(n) - \langle x \rangle)^2} \tag{8}$$

where F is the number of structures of the conformational ensemble, and $\langle x \rangle$ is the average position.

2.3.2 Solvent-accessible surface area (SASA)

The SASA describes the accessible surface for a spherical probe which is rolling over the molecule of interest. The probe, with a typical radius set to 1.4 $\mathring{A}$, mimics a water molecule, giving a measure of the exposition of the molecule to the solvent and, therefore, its compactness.[70]

2.3.3 Radius of gyration

The radius of gyration is a measure of the dispersion of a set of atoms to its center of mass.

$$R_g \equiv \sqrt{\langle (r - r_{COM})^2 \rangle} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (r_i - r_{COM})^2} \tag{9}$$

Thus, it is a measure of its compactness.[71] By measuring the radius of gyration at different times, one can assess protein conformational changes. Both SASA and the radius of gyration have been calculated using the VMD software.[72]

2.3.4 Conformational Space Overlap

In order to perform a purely geometrical comparison of the ensembles produced by each method, it is convenient to calculate the overlap between the conformational space of the reference MD and the normal mode driven trajectories. We used a similar approach to the ones shown in Lymann's and Lindorff-Larsen's articles:[73,74]

- First, we want to obtain a partition in k regions of our reference conformational space $(R_1, R_2, \ldots, R_k)$. To this end, we use the cluster analysis software pyProCT[75] and apply a k-medoids algorithm using the RMSD between conformations as the distance. The medoids of each of these clusters $(r_1, r_2, \ldots, r_k)$ are the representative conformations of each region R_i of the conformational space.

- Then, for each conformation c_j belonging to the CC or IC NMA ensembles, we look for

its most similar MD representative r_i and we add it to the cluster R_i. In other words, this means that conformation c_i is assigned to the region R_i of the conformational space.

- For all the methods, we calculate the population of all the clusters. Afterwards, values are normalized in order to get a probability density distribution.

- Finally, the square root of the Jensen-Shannon divergence (JSD) of the distributions is calculated as

$$O = JSD(P||Q)^{\frac{1}{2}} = \left(\frac{1}{2}KLD(P||M) + \frac{1}{2}KLD(Q||M)\right)^{\frac{1}{2}}, \tag{10}$$

where KLD is the Kullback-Liebler divergence and Q is the average distribution. The square root of JSD can be used as a metric[76] to assess the degree of overlap of the probability distributions, and, therefore, of each one of the methods with MD. Values are confined into the range [0,1], where a value of 1 means that the population distributions diverge completely and a value of 0 that the distributions are analogous.

2.3.5 Computational efficiency of deformation

Another goal of this work is to assess the temporal performance of the new method. We are mainly interested in two aspects, the time needed to complete an iteration, and the amount of deformation produced in each iteration (calculated as the RMSD of the structures before and after the iteration). We will combine both measures to calculate the computational efficiency of the deformation process as $^{RMSD_{step}}/_{t_{step}}$, the time needed to deform the backbone 1 $\mathring{A}$.

3 Results and discussion

3.1 Simulation setup

Both normal mode methods are customizable, letting the user control most aspects of the simulation. Since we are introducing a new methodology, we tuned the values of some simulation parameters in order to obtain the best performance and to compare both methods. First, we set the values of the common parameters like the temperature (300 K), or the cutoff distance of the elastic network (set to a 9 $\mathring{A}$, a value that is in agreement with the literature[77] and produces those highly collective modes; see Section S1). We decided to calculate up to 10 modes and to not combine them linearly. The mode and the sense of the movement are randomly changed at each MC iteration, which is a general and system-agnostic solution. Also, we chose to calculated the modes once at the beginning of the simulation, assuming that the potential surface does not change too abruptly.

Second, we determined the most sensitive parameters for each method. The first one, the displacement magnitude, controls the maximum translation ($\mathring{A}$) or the maximum torsional rotation (rad) performed in the NMA step. The second one, the minimum root mean square gradient (RMSG), is the threshold of convergence of the minimizations and can be understood as the "strength" of the minimization. In the ANM method, it modules the closeness of the final conformation to the proposal, whereas in the IC NMA method, it regulates the intensity of the side chain minimization, and it is related to the success releasing steric clashes.

Third, we performed a first round of simulations setting these parameters with all the combinations of the values found in Table 1. In order to find the best settings, we discarded the results with acceptances out of our working range (20-40%[78]) and we ranked them according to the similarity of their RMSF profiles with the reference MD simulation and favouring large backbone deformations. Then, we performed a second round of simulations, fine tuning the best displacements found in the previous round.

For the ubiquitin system, we found that the optimal values for our working parameters were a displacement magnitude of 1.08 $\mathring{A}$ and an RMSG of 0.1 $^{kcal}/_{mol\mathring{A}}$for the CC simulations, and a displacement magnitude in the range of 0.07 to 0.16 rad and an RMSG of 0.1 $^{kcal}/_{mol\mathring{A}}$for the IC simulations. Likewise, for c-Src kinase, the best values for the CC simulations were 0.66 $\mathring{A}$ for the displacement magnitude and and 0.1 $^{kcal}/_{mol\mathring{A}}$for the RMSG. A displacement range of 0.07 to 0.14 rad and RMSG of 0.05 $^{kcal}/_{mol\mathring{A}}$was found to be the best fitted parameters for the icNMA simulation.

As for the side chain perturbation stage in the icNMA method, we chose to perform 2000 side chain changes every ten steps of icNMA. In this case, we performed dihedral rotations in the range of 0.02 to 0.024 rad. These values are again optimized so that they yield acceptances between 20 and 40%.

Finally, we performed 12 independent 24 h production runs for each method and system.

Table 1: Choice of parameters affecting the mode application step in both studied methods and the values that will be used in the characterization tests.

Method	Displacement magnitude ($\mathring{A}$ rad)	Relaxation strength ($^{kcal}/_{mol\mathring{A}}$)
ccNMA	0.25, 0.66, 1.08, 1.5, 1.92	0.01, 0.02, 0.04, 0.05, 0.1
icNMA	0.02, 0.05, 0.075 ,0.10, 0.12, 0.15	0.01, 0.02, 0.04, 0.05, 0.1

3.2 Energetic cost of the NMA perturbation

We calculated the energy increments of the NMA step in the IC simulations as well as the energy increments of the ANM step and full iteration (perturbation+relaxation) of the CC simulations. The results show that icNMA is able to make MC proposals that are energetically favorable in both systems (see Table 2 and 3), while the ccNMA-based method cannot (the energy increments of this step are almost always positive). Without a way to make these energies decrease, it would be almost impossible to accept any step. It is only thanks to the relaxation phase of the CC algorithm, that the energy of the MC proposals

can be lowered. It is worth noting that the IC method can result in large increments in energy (considerably larger than in CC) due to unresolved backbone clashes from dihedral rotation. It happens more often in ubiquitin than in the c-Src kinase, since the first is more globular than the latter, and steric clashes can be introduced more easily.

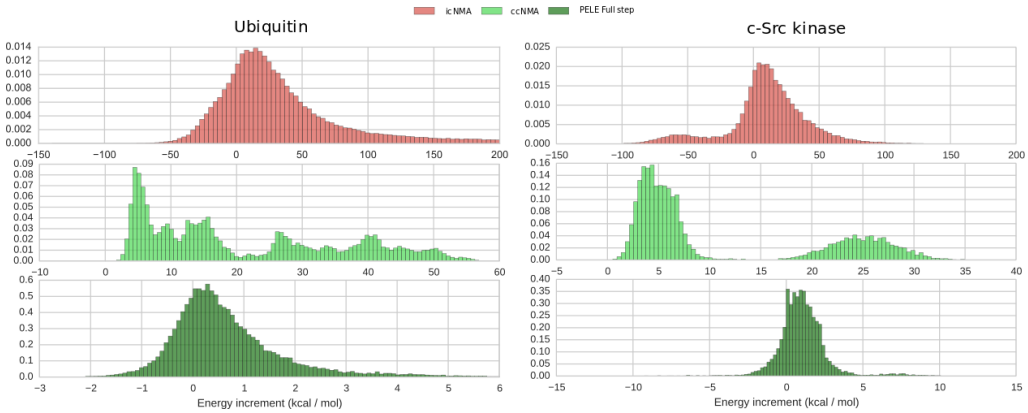


Figure 3: Distribution of the energy increments produced on each iteration for each method and protein system. In red, we show the energy increments in the icNMA step. In light and dark green we show the distributions of energy increments related to the ccNMA step and the complete PELE step, respectively. Note the changes in axis scales.

Table 2: Acceptance and percentage of energetically favorable proposals (EFP) generated by each algorithm in our simulations. In the CC, we can find two values: the first value belongs to the EFP of the ANM step, the second, between parentheses is the EFP of the whole PELE step (i.e. including the relaxation phase).

	Ubiquitin		c-Src kinase	
	CC	IC	CC	IC
Acceptance (%)	45.3	19.4	37.5	28.2
EFP (%)	<0.1 (26.6)	18.7	<0.1 (21)	27.2

3.3 Compactness of the protein (SASA and radius of gyration)

As it can be seen in Fig. 4, both ensembles generated by NMA-based methods are more compact than the ones obtained using MD. Comparing both normal mode methods, we see

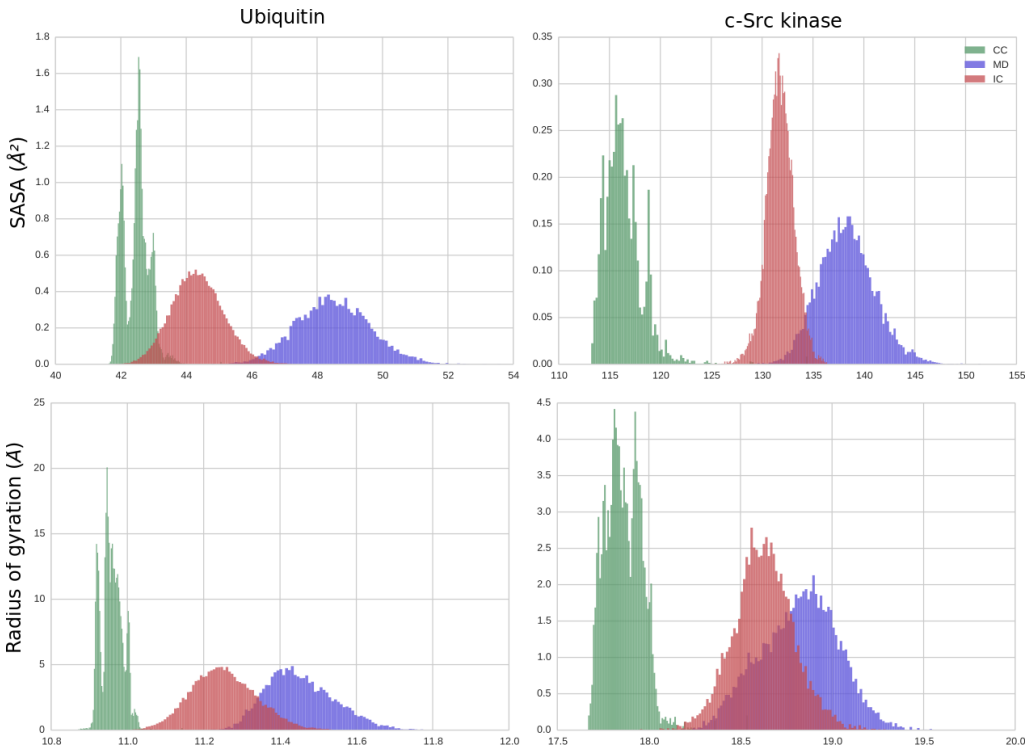


Figure 4: SASA and radius of gyration distributions for all the studied ensembles and systems. In general the measures obtained from the ensembles produced by the IC method look similar to our reference MD simulations.

that protein structures obtained by applying icNMA modes have larger SASA values and, therefore, seem to be less compact and collapsed than those obtained using the ccNMA method. We can draw similar conclusions by looking at the radius of gyration distribution.

We could expect differences, since the NMA-based methods are using an implicit solvation model (the Onufriev, Bashford, and Case, OBC[79] model), whereas the MD simulations were run using explicit solvent. The bias of some implicit solvent models, including OBC, to compact structures is well studied.[79–83] This bias is mainly caused by the over-stabilization of nonpolar interactions[84] and the increase in number and stability of hydrogen bonds.[85] This effect is further emphasized in the CC model by the two minimization procedures (see below). In current PELE simulations this is typically avoided by adding a weak ($\sim$0.1 $^{kcal}/_{mol\mathring{A}^2}$) harmonic constraint every 10 alpha carbons, for example, to the initial model.

3.4 Protein fluctuations (RMSF)

RMSF plots of the ubiquitin simulations show small overall fluctuations for all methods (see Fig. 5A), between 1 and 2 $\mathring{A}$. This is expected, as ubiquitin is a relatively stiff protein. The movement is concentrated on the less structured parts: the loops between beta-sheets and alpha-helices (see Fig. 5D). In general, icNMA fluctuations are in closer agreement with MD than ccNMA fluctuations and also show a higher baseline. The scaled RMSF (see Fig. 5B) allows us to have a clearer view of the relative magnitudes of fluctuations. The β2-α-helix loop is more stable in the MD and IC simulations, while it shows a prominent peak in the RMSF for the CC simulation. This peak belongs to a temporary backbone rearrangement that is not present in the IC and MD simulations. CC clearly populates a further state with larger RMSD than IC and MD (See Fig. 5C). However, ccNMA clearly underestimates the movement of the β-4-β5 loop, which is nicely captured by icNMA.

Our c-Src kinase simulations are again producing smaller overall fluctuations than MD (with the exception of the P-loop). The rescaling of the RMSF plots shows that the ANM method is able to capture the flexibility of the A-loop with slightly more success (see 6 B).

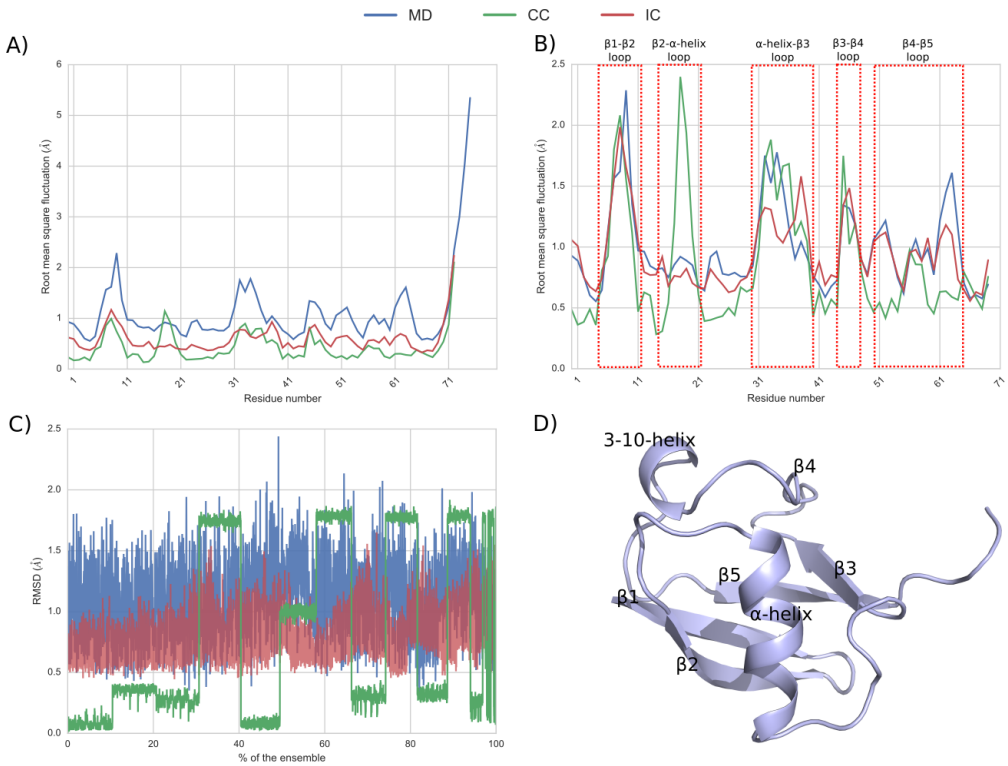


Figure 5: A) RMSF profile of the ubiquitin NMA simulations compared with MD. The last three residues have been excluded in order to milden the "tip effect". B) We have superimposed the RMSF plots so that relative fluctuations can be checked. In order to do this, we have scaled them so that the root mean square error was minimum. This yielded scaling factors of 2.10 (CC) and 1.7 (IC). The most flexible parts of the protein (mainly the loops connecting the secondary structure) have been highlighted in this scaled representation of the RMSF. C) RMSD plot of the $\beta 2$-α-helix loop referred to the first CC simulation frame. D) Representation of ubiquitin identifying its secondary structure elements.

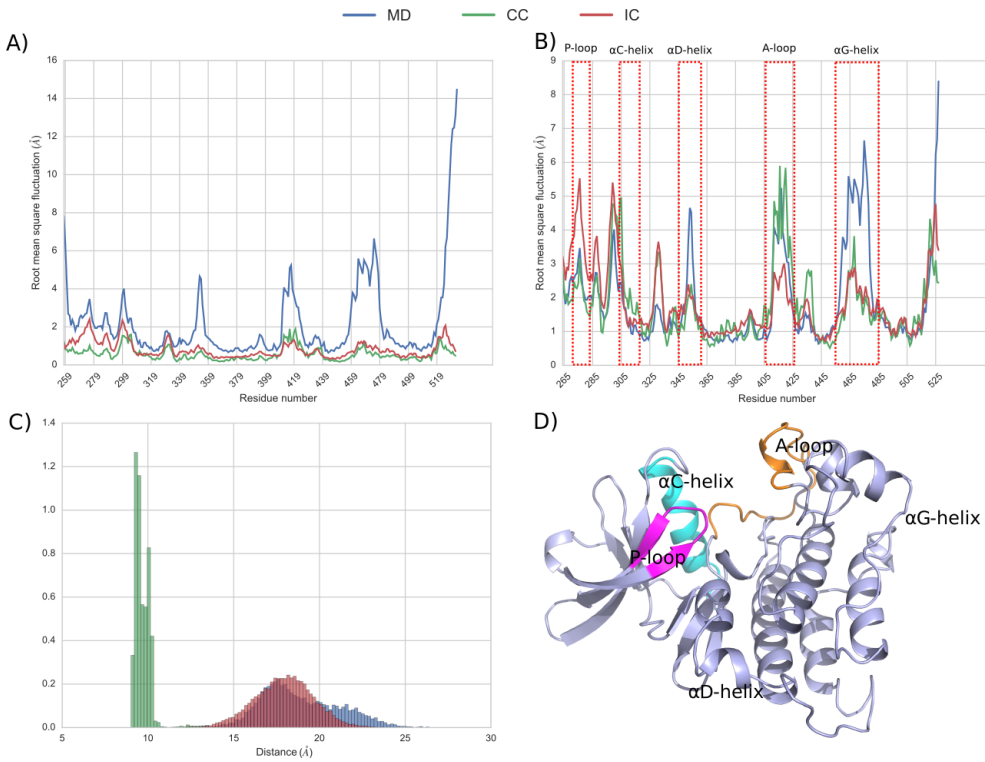


Figure 6: A) RMSF plot of the c-Src kinase NMA simulations compared with MD. N and C-terminal loops are omitted B) Scaled RMSF plot with highlighted functionally relevant elements. The scale factors are 3.10 (CC) and 2.3 (IC). C) Distributions of the distances between CYS:277:CA and LEU:387:CA for each simulation. D) Representation of the protein identifying its more relevant secondary structure elements.

Major differences can be found in the fluctuations of the residues belonging to the αD-helix, A-loop and αG-helix (see 6A, B and D). These structures show wide local rearrangements, in essence folding-unfolding events, that are hard to capture using NMA-based techniques.

The correct sampling of the P-loop dynamics is of utmost importance, as it is directly involved in the binding process.[86] We decided to investigate the opening and closing of this loop in more detail by measuring the distance between a central residue in the P-loop (CYS:277:CA) and a second residue on the other side of the binding site (LEU:387:CA) (see Fig. 6C). This distance shows a similar fluctuation range in the MD and icNMA simulations ($\sim$15-25 $\mathring{A}$ and $\sim$13-23 $\mathring{A}$ respectively), whereas the range of distances sampled by the ANM-based simulation is significantly smaller ($\sim$9.5-10.5 $\mathring{A}$). This is related to the increase of compactness observed in the SASA and rgyr analyses: the protein collapses quickly and the inter-domain distance does not oscillate much (see Fig. 4).

The lower overall fluctuations of both normal mode methods, as well as the low baselines, suggest that the proteins are moving less. This is partly due to the lack of local motion which can only be mapped with higher frequency modes, not present in our simplified NMA procedure. In addition, in the icNMA method there are no anharmonic backbone movements, further limiting its comparison with MD; anharmonicity has been studied to play an important role in the modulation of the amplitude of fluctuations.[77,87]

3.4.1 Effect of the minimization in the compaction process

In order to gain more insight on the compaction process and the effect of minimizations on it, we study how the distance between CYS:277:CA and LEU:387:CA changes in the different stages of the algorithms in the closing regime. It is convenient to define $r_d \equiv \frac{\sum_{i\in D_-} d_i}{\sum_{i\in D_+} d_i}$, where D_+ and D_- are the domains defined by positive and negative increments respectively, and d_i are the distance increments.

We will focus on the three parts of the ccNMA-based algorithm related to the change of

the backbone (see Figure 2): the target coordinates calculation, where we make a proposal, the ANM minimization, where we apply the minimization with spring constraints toward the proposals; and the system-wide constrained minimization. In the icNMA-based algorithm, we will focus on the target rotation calculation (i.e. proposal rotations) and the torsional rotation, where we apply the rotation.

In the proposal stage (see Table 3), $r_d \simeq 1$, since proposals are symmetric by construction (random choice of modes and senses). In the application of the ANM modes, there is symmetry again, which means that the ccNMA step is successfully generating conformations close to the proposal. However, the constrained minimization shows a recurrent bias toward negative increments ($r_d \simeq 1.23$), leading to more compact structures. In the icNMA-based algorithm, proposals are again symmetric by construction, and torsional rotations are set to those values, giving no overall bias ($r_d \simeq 1$).

This agrees with the results previously seen in Figure 4, where ccNMA generated structures tend to be more compact, since backbone movement is minimization-driven. That would explain the tendency to over-close the protein, whereas in the icNMA step this does not happen, since minimizations do not apply to the backbone. These results demonstrate that the use of minimizations plays an important role in the compaction process.

Table 3: r_d values in the different stages of the ccNMA and icNMA-based methods.

	CC	IC
Target coordinates	0.98	0.95
ANM application	1.02	0.95
Constrained minimization	1.23	

3.5 Conformational space overlap

We have applied the algorithm described in Section 2.3.4 for partitions of 1 to 1k clusters. In panel A of Figure 7, we can see that, for all the studied numbers of clusters, the JSD is always lower for icNMA than for ccNMA. This means that the exploration of the configurational

space performed by icNMA is in closer agreement with MD. The JSD can tell us whether NMA methods are populating the same regions as MD trajectories, however it does not give us information about the structural similarity of the conformations. In order to analyze it, we have calculated the average C_α RMSD of each structure in the NMA ensembles with the most similar MD conformation inside its cluster (see lower subplots of Fig. 7A). The RMSD results for ubiquitin are similar for both methods, showing an average RMSD value of $\sim$0.8 $\mathring{A}$. In the c-Src kinase case, however, results are significantly different between methods: the RMSD difference of 0.8 $\mathring{A}$ is indicating that icNMA is not only populating similar regions of the space, but also generating similar conformations to MD. Taking $k = 10$ as a case study (see Fig. 7B), we can observe that:

Ubiquitin Both NMA methods are visiting only a fraction of the possible regions of the conformational space sampled by MD (3 out of 10). The exploration performed by the ccNMA method is clearly even less than the one performed using the icNMA method: more than 90% of the population is concentrated in one cluster. This can be explained with a quick look to the RMSD matrix of the combined ensembles (Fig. 7C) : the submatrices of the NMA methods are more similar between themselves than to the MD submatrix. This is a consequence of the NMA methods failing to model the flexible loops connecting the secondary structure.

c-Src kinase The populations of the icNMA ensemble is distributed more evenly than the ccNMA ensemble populations. However, the icNMA method is still overpopulating one state ($\sim$60% of its population). The ccNMA algorithm is visiting only 2 regions, a poor result compared with the 7 out of 10 regions that the icNMA is able to populate. In the subplot, we depicted the average interdomain distance for the structures of each cluster. icNMA has better agreement with MD, capturing the different stages of the P-loop opening/closing process. The RMSD matrix shows, again, notable differences between the ccNMA and the MD ensembles. However, the differences between icNMA and MD are less noticeable, and this can be related to the successful modelling of the

P-loop behaviour .

3.6 Computational performance of the methods

We have measured the time required to complete different tasks: an icNMA step, a ccNMA step, and a full PELE iteration (ccNMA step + relaxation phase). We have also calculated the extent of the perturbations performed in each task by calculating the RMSD of the structure before and after the task. We observe that the average time required to perform a ccNMA step is lower than the average time needed to perform an icNMA step (see Table 4), however, their efficiency distributions are pretty similar (see Fig. 8). This indicates that, although the icNMA step is slightly slower, the perturbations performed are wider than the ones applied by the ccNMA step. Nevertheless, the perturbations performed in the ccNMA step are, in general, not energetically favorable (already seen in Section 3.2), which forces PELE to add a relaxation phase. If we take into account the time needed to run the relaxation phase, the IC methodology clearly outperforms the CC methodology (the speedups per step are $\sim$5x and $\sim$7x, depending on the system). This explains the dramatic differences in their efficiencies illustrated in Fig. 8.

As the relaxation phase is an essential part to the ccNMA-based algorithm, so is the side chain perturbation stage to the icNMA method. In order to study the impact of this stage on the overall time, we have added the time needed to run 200 iterations of side chain perturbation to the icNMA time (remember that the ratio of iterations between the first and second stages is 10:2000). As expected, the resultant distributions show a shift to the left (less efficiency). This effect is more significant in the ubiquitin case than in the c-Src kinase case, mainly because the amortized side chain perturbation time ($\sim$2 to 3s) is of the order of the time spent in an icNMA iteration. However, in both cases, it still shows a better overall efficiency than the CC-based method.

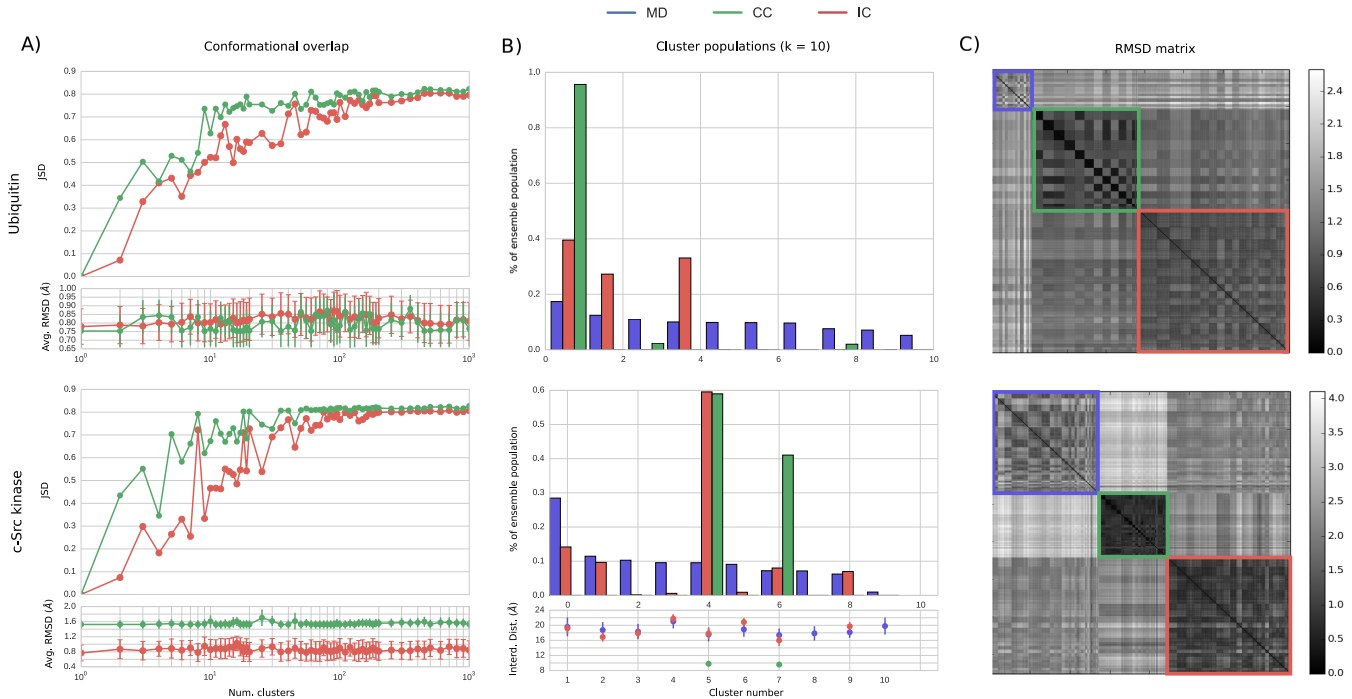


Figure 7: A) JSD of the NMA methods and MD for a different number of clusters ($1 \leq k \leq 1000$). B) Detail of the distributions per cluster for $k = 10$. C) C_α RMSD matrices for the whole ensemble of simulations. The submatrices of MD and NMA simulations have been highlighted.

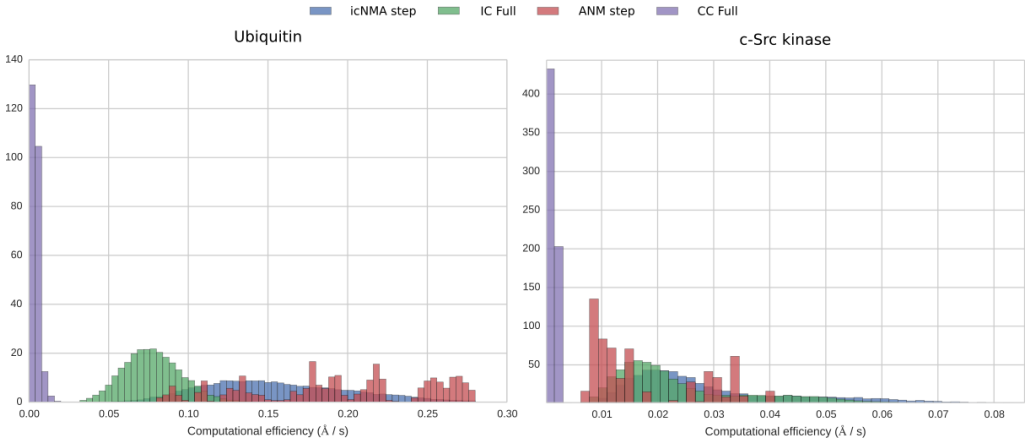


Figure 8: Distribution of the computational efficiency for both methods and systems. Both NMA steps (red and blue) look to be able to perform similarly. However, when comparing with PELE full iteration (purple), it can be clearly seen that the IC-based method has a larger efficiency, even when the amortized side chain perturbation time is considered (green).

Table 4: Average time values of an icNMA iteration, the side chain perturbation step, the ANM step, and a full MC iteration of PELE. The need of the relaxation phase in the CC-based PELE algorithm makes the overall time of an iteration clearly slower than an icNMA iteration. The simulations were run in AMD Opteron 6238 @ 2.60GHz nodes with 4Gb of RAM per node. Standard deviations are included between parentheses.

System	icNMA step (s)	Side. Perturb. All steps (s)	ccNMA step (s)	PELE iteration (s)
Ubiquitin	2.945 (1.456)	21.327 (2.503)	1.376 (0.704)	20.065 (5.624)
c-Src kinase	11.648 (3.273)	27.917 (2.958)	6.106 (0.679)	57.9 (9.233)

4 Conclusions

There is a high interest in developing faster sampling techniques for modeling backbone flexibility in proteins, with normal mode approximations such as ANM becoming a popular alternative. The modes obtained using ANM, however, describe the movement of only one atom per residue (the C_α atom) and applying this movement to the remaining atoms is not trivial. In PELE, this is achieved by applying a minimization. However, in this step the covalent topology of the protein is often unphysically distorted, which requires the introduction of a relaxation phase where a system-wide minimization is performed.

In this article we have presented a new MC method that handles the protein backbone changes using IC NMA. The application of the internal modes through a geometrical manipulation of torsions does not distort the covalent topology, allowing us to to generate more energetically favorable conformation proposals than our previous method. Another fundamental advantage of the use of torsional modes is their increased collectivity and robustness, which alleviates the "tip effect" problem, facilitating the fast traversal of the conformational space.

The elimination of the ANM minimization and the unneeded relaxation phase has two major consequences. First, as both the side chain prediction step and the system-wide constrained minimization are computationally costly processes, the new method shows a great speedup ($\sim$5-7x) without decreasing the RMSD of the perturbations. Second, as these minimizations amplify the solvent model bias towards compact structures, the new method shows SASA and radius of gyration distributions which are closer to those observed in MD. Moreover, our analyses have also shown improvements in the exploration of the conformational space: the icNMA algorithm is able to explore similar regions to MD and generate structures with closer RMSD than the CC-based method.

The main drawback we have found is that, although the icNMA method seems to produce relative fluctuations that are in better accordance with MD than the ANM-based method, these fluctuations are smaller and show a lower baseline than MD. This is the consequence

of one of the main limitations of applying NMA-based techniques, where local flexibility is not well represented by higher frequency modes.

Overall, implementing icNMA results in better agreement with MD explicit solvent simulations. Using internal coordinates seems to be a promising technique for speeding up the induced fit studies in PELE and supposes a step forward in terms of the quality of the flexibility handling.

Acknowledgement

The authors thank D. E. Shaw Research lab. for providing the kinase MD coordinates and Dr. López Blanco for sharing the code developed in his thesis. This work was supported by the CTQ-48287-R projects of the Spanish Ministry of Economy and Competitiveness (MINECO) and the grant SEV-2011-00067 of Severo Ochoa Program, awarded by the Spanish Government.

References

(1) Erickson, J. A.; Jalaie, M.; Robertson, D. H.; Lewis, R. A.; Vieth, M. *J. Med. Chem. 47*, 45–55.

(2) Huang, S.-Y.; Zou, X. *Proteins 66*, 399–421.

(3) Totrov, M.; Abagyan, R. *18*, 178–184.

(4) Sherman, W.; Day, T.; Jacobson, M. P.; Friesner, R. A.; Farid, R. *J. Med. Chem. 49*, 534–553.

(5) Legge, F. S.; Budi, A.; Treutlein, H.; Yarovsky, I. *Biophys. Chem. 119*, 146–157.

(6) Jorgensen, W. L.; Tirado-Rives, J. *J. Comput. Chem. 26*, 1689–1700.

(7) Borrelli, K. W.; Vitalis, A.; Alcantara, R.; Guallar, V. *J. Chem. Theory Comput. 1*, 1304–1311.

(8) Shan, Y.; Kim, E. T.; Eastwood, M. P.; Dror, R. O.; Seeliger, M. A.; Shaw, D. E. *J. Am. Chem. Soc. 133*, 9181–9183.

(9) Kresten Lindorff-Larsen1, R. O. D., Stefano Piana; Shaw, D. E. *334*, 517–520.

(10) De Vivo, M.; Masetti, M.; Bottegoni, G.; Cavalli, A. *J. Med. Chem.*

(11) Cole, D. J.; Tirado-Rives, J.; Jorgensen, W. L. *Biochim. Biophys. Acta, Gen. Subj 1850*, 966–971.

(12) Frenkel, D.; Smit, B. *Understanding molecular simulation: from algorithms to applications*, 2nd ed.; Computational science series 1; Academic Press.

(13) Rapaport, D. C.; Scheraga, H. A. *Macromolecules 14*, 1238–1246.

(14) Paine, G. H.; Scheraga, H. A. *Biopolymers 24*, 1391–1436.

(15) Lotan, I.; Schwarzer, F.; Latombe, J.-C. In *Algorithms in Bioinformatics*; Benson, G., Page, R. D. M., Eds.; Lecture Notes in Computer Science 2812; Springer Berlin Heidelberg, pp 354–373.

(16) Tang, K.; Zhang, J.; Liang, J. *PLoS Comput. Biol. 10*.

(17) Esteban-Martín, S.; Fenwick, R. B.; Ådén, J.; Cossins, B.; Bertoncini, C. W.; Guallar, V.; Wolf-Watz, M.; Salvatella, X. *PLoS Comput. Biol. 10*, e1003721.

(18) Hosseini, A.; Espona-Fiedler, M.; Soto-Cerrato, V.; Quesada, R.; Pérez-Tomás, R.; Guallar, V. *PLoS ONE 8*, e57562.

(19) Eyer, L.; Valdés, J. J.; Gil, V. A.; Nencka, R.; Hřebabecký, H.; Šála, M.; Salát, J.; Černý, J.; Palus, M.; Clercq, E. D.; Růžek, D. *Antimicrob. Agents Chemother. 59*, 5483–5493.

(20) Grebner, C.; Iegre, J.; Ulander, J.; Edman, K.; Hogner, A.; Tyrchan, C. *J. Chem. Inf. Model.*

(21) Monza, E.; Lucas, M. F.; Camarero, S.; Alejaldre, L. C.; Martínez, A. T.; Guallar, V. *J. Phys. Chem. Lett. 6*, 1447–1453.

(22) Acebes, S.; Fernandez-Fueyo, E.; Monza, E.; Lucas, M. F.; Almendral, D.; Ruiz-Dueñas, F. J.; Lund, H.; Martinez, A. T.; Guallar, V. *ACS Catal. 6*, 1624–1629.

(23) Takahashi, R.; Gil, V. A.; Guallar, V. *J. Chem. Theory Comput. 10*, 282–288.

(24) Atilgan, A. R.; Durell, S. R.; Jernigan, R. L.; Demirel, M. C.; Keskin, O.; Bahar, I. *Biophys. J. 80*, 505–515.

(25) Rueda, M.; Chacón, P.; Orozco, M. *Structure 15*, 565–575.

(26) Ahmed, A.; Villinger, S.; Gohlke, H. *Proteins 78*, 3341–3352.

(27) Kitao, A.; Hirata, F.; Gō, N. *158*, 447–472.

(28) Skjaerven, L.; Martinez, A.; Reuter, N. *Proteins 79*, 232–243.

(29) Northrup, S. H.; McCammon, J. A. *Biopolymers 19*, 1001–1016.

(30) Li, Z.; Scheraga, H. A. *Proc. Natl. Acad. Sci. U.S.A. 84*, 6611–6615.

(31) Jorgensen, W. L.; Tirado-Rives, J. *J. Phys. Chem. 100*, 14508–14513.

(32) Kidera, A. *Int. J. Quantum Chem 75*, 207–214.

(33) Noguti, T.; Gō, N. *Biopolymers 24*, 527–546.

(34) Lopez-Blanco, J. R.; Garzón, J. I.; Chacón, P. *Bioinformatics 27*, 2843–2850.

(35) Kovacs, J. A.; Cavasotto, C. N.; Abagyan, R. *J. Comput. Theor. Nanosci. 2*, 354–361.

(36) Eckart, C. *Phys. Rev. A 47*, 552–558.

(37) Noguti, T.; Gō, N. *J. Phys. Soc. Jpn. 52*, 3283–3288.

(38) Abe, H.; Braun, W.; Noguti, T.; Gō, N. *Comput. Chem. 8*, 239–247.

(39) Choi, V. *J. Chem. Inf. Model. 46*, 438–444.

(40) Tamar Schlick, A. L. F. *18*, 71–111.

(41) Genheden, S.; Ryde, U. *Phys. Chem. Chem. Phys. 14*, 8662–8677.

(42) Henzler-Wildman, K.; Kern, D. *Nature 450*, 964–972.

(43) Plattner, N.; Noé, F. *Nat. Commun. 6*, 7653.

(44) Glickman, M. H.; Ciechanover, A. *82*, 373–428.

(45) Schnell, J. D.; Hicke, L. *J. Biol. Chem. 278*, 35857–35860.

(46) Mukhopadhyay, D.; Riezman, H. *315*, 201–205.

(47) Vijay-Kumar, S.; Bugg, C. E.; Cook, W. J. *J. Mol. Biol. 194*, 531–544.

(48) Bang, D.; Makhatadze, G. I.; Tereshko, V.; Kossiakoff, A. A.; Kent, S. B. *Angew. Chem. Int. Ed. 44*, 3852–3856.

(49) Liu, G.; Forouhar, F.; Eletsky, A.; Atreya, H. S.; Aramini, J. M.; Xiao, R.; Huang, Y. J.; Abashidze, M.; Seetharaman, J.; Liu, J.; Rost, B.; Acton, T.; Montelione, G. T.; Hunt, J. F.; Szyperski, T. *J. Struct. Funct. Genomics 10*, 127–136.

(50) Alexeev, D.; Barlow, P. N.; Bury, S. M.; Charrier, J.-D.; Cooper, A.; Hadfield, D.; Jamieson, C.; Kelly, S. M.; Layfield, R.; Mayer, R. J.; McSparron, H.; Price, N. C.; Ramage, R.; Sawyer, L.; Starkmann, B. A.; Uhrin, D.; Wilken, J.; Young, D. W. *ChemBioChem 4*, 894–896.

(51) Levin-Kravets, O.; Shohat, N.; Prag, G. *Biochemistry 54*, 4704–4710.

(52) Fasshuber, H. K.; Lakomek, N.-A.; Habenstein, B.; Loquet, A.; Shi, C.; Giller, K.; Wolff, S.; Becker, S.; Lange, A. *Prot. Sci. 24*, 592–598.

(53) Kony, D. B.; Hünenberger, P. H.; van Gunsteren, W. F. *Prot. Sci. 16*, 1101–1118.

(54) Piana, S.; Lindorff-Larsen, K.; Shaw, D. E. *Proc. Natl. Acad. Sci. U.S.A. 110*, 5915–5920.

(55) Ganoth, A.; Tsfadia, Y.; Wiener, R. *J. Mol. Graphics Modell. 46*, 29–40.

(56) Beauchamp, K. A.; Lin, Y.-S.; Das, R.; Pande, V. S. *J. Chem. Theory Comput. 8*, 1409–1414.

(57) Leherte, L.; Vercauteren, D. P. *Sci. China Chem. 57*, 1340–1354.

(58) Raval, A.; Piana, S.; Eastwood, M. P.; Shaw, D. E. *Prot. Sci. 25*, 19–29.

(59) Cossins, B. P.; Hosseini, A.; Guallar, V. *J. Chem. Theory Comput. 8*, 959–965.

(60) Van Der Spoel, D.; Lindahl, E.; Hess, B.; Groenhof, G.; Mark, A. E.; Berendsen, H. J. C. *J. Comput. Chem. 26*, 1701–1718.

(61) Jorgensen, W. L.; Tirado-Rives, J. *J. Am. Chem. Soc. 110*, 1657–1666.

(62) Aleshin, A.; Finn, R. S. *Neoplasia 12*, 599–607.

(63) Schindler, T.; Sicheri, F.; Pico, A.; Gazit, A.; Levitzki, A.; Kuriyan, J. *Mol. Cell 3*, 639–648.

(64) Hornak, V.; Abel, R.; Okur, A.; Strockbine, B.; Roitberg, A.; Simmerling, C. *Proteins 65*, 712–725.

(65) Lindorff-Larsen, K.; Piana, S.; Palmo, K.; Maragakis, P.; Klepeis, J. L.; Dror, R. O.; Shaw, D. E. *Proteins 78*, 1950–1958.

(66) Bowers, K. J.; Chow, D. E.; Xu, H.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Moraes, M. A.; Sacerdoti, F. D.; Salmon, J. K.; Shan, Y.; Shaw, D. E. Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters. Proceedings of the ACM/IEEE SC 2006 Conference. pp 43–43.

(67) Shaw, D. E.; Chao, J. C.; Eastwood, M. P.; Gagliardo, J.; Grossman, J. P.; Ho, C. R.; Lerardi, D. J.; Kolossváry, I.; Klepeis, J. L.; Layman, T.; McLeavey, C.; Deneroff, M. M.; Moraes, M. A.; Mueller, R.; Priest, E. C.; Shan, Y.; Spengler, J.; Theobald, M.; Towles, B.; Wang, S. C.; Dror, R. O.; Kuskin, J. S.; Larson, R. H.; Salmon, J. K.; Young, C.; Batson, B.; Bowers, K. J. *51*, 91.

(68) Boehr, D. D.; Nussinov, R.; Wright, P. E. *Nat. Chem. Biol. 5*, 789–796.

(69) Csermely, P.; Palotai, R.; Nussinov, R. *Trends Biochem. Sci 35*, 539–546.

(70) Moret, M. A.; Santana, M. C.; Zebende, G. F.; Pascutti, P. G. *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys. 80*, 041908.

(71) Lobanov, M. Y.; Bogatyreva, N. S.; Galzitskaya, O. V. *Mol. Biol. 42*, 623–628.

(72) Humphrey, W.; Dalke, A.; Schulten, K. *J. Mol. Graph. Model. 14*, 33–38.

(73) Lyman, E.; Zuckerman, D. M. *Biophys. J. 91*, 164–172.

(74) Lindorff-Larsen, K.; Ferkinghoff-Borg, J. *PLoS ONE 4*, e4203.

(75) Gil, V. A.; Guallar, V. *J. Chem. Theory Comput. 10*, 3236–3243.

(76) Endres, D. M.; Schindelin, J. E. *49*, 1858–1860.

(77) Zheng, W. *Biophys. J. 98*, 3025–3034.

(78) Gelman, A.; Roberts, G.; Gilks, W. *Bayesian statistics, 5 (Alicante, 1994)*; Oxford Univ. Press, pp 599–607.

(79) Onufriev, A.; Bashford, D.; Case, D. A. *Proteins 55*, 383–394.

(80) Zhang, W.; Ganguly, D.; Chen, J. *PLoS Comput. Biol. 8*, e1002353.

(81) Foloppe, N.; Chen, I.-J. *Bioorg. Med. Chem.*

(82) Jaramillo, A.; Wodak, S. J. *Biophys. J. 88*, 156–171.

(83) Formaneck, M.; Cui, Q. *J. Comput. Chem. 27*, 1923.

(84) Chen, J.; Iii, C. L. B. *Phys. Chem. Chem. Phys. 10*, 471–481.

(85) Stultz, C. M. *J. Phys. Chem. B 108*, 16525–16532.

(86) Boggon, T. J.; Eck, M. J. *Oncogene 23*, 7918–7927.

(87) Hayward, S.; Kitao, A.; Gō, N. *Prot. Sci. 3*, 936–943.

3.5 Supplementary materials for: Enhancing backbone sampling in Monte Carlo simulations using Internal Coordinates Normal Mode Analysis

3.5.1 Are Cartesian coordinate and internal coordinate normal modes equivalent?

We have calculated the (Cartesian coordinate) ANM modes and the internal coordinates NMA modes of a set of structures and compared them. Our test set comprises the proteins with PDB ID: 1ubq, 2lzm, 1ex6, 1ddt, 4ake, 1ggg, and the src kinase domain of 1y57. Most of these proteins have been used in NMA benchmarks, as they present wide inter domain movements. We have added two alternative structures for 1ubq and 1y57: 1ubq_cut, a copy of 1ubq that does not contain the last 3 residues from the C-terminal loop and 1y57_MD, a randomly picked frame from an MD simulation of 1y57.

The first thing needed in order to compare both sets of modes is to convert the IC modes to CC modes. This can be achieved calculating the Jacobian (J, inverse of Wilson's B matrix) as

$$J_{i,\alpha} = \frac{\partial r_i}{\partial q_\alpha}, \tag{3.1}$$

and applying the following equation:

$$\Delta\vec{r}_{i,\alpha} = \sum_{\alpha}^{N} \vec{J}_{i,\alpha} v_i^\alpha. \tag{3.2}$$

As all heavy atoms are involved in the calculation of the IC modes and in the conversion, the resulting CC modes will have 3H elements (being H the number of heavy atoms).

3.5.1.1 Collectivity of CC and IC modes

One of the most compelling features of NMA-based protein simulations is that the modes of lower frequency are able to mobilize big groups of atoms that perform large displacements (i.e. large collective motions). It would be interesting

to know if this assumption is true for both models, and if the performance of both is similar. In order to do this we have calculated the first ten ANM and IC NMA modes for and all the structures in our test, modifying the cutoff distance that modulates the density of springs in the elastic network. Then, we have calculated the degree of collectivity of each mode. This gives us information about how the modes change when the elastic network and the shape of the protein change.

We have also performed a second batch of calculations of IC modes applying the method described by Lu *et al.* [146]. In his work, they add an extra term to the NMA potential, $^{\omega}/_{2}\sum_{\alpha}(\phi_{\alpha}-\phi_{\alpha}^{0})^{2}$, where $\omega = 3min(H_{\alpha\alpha}^{0})$. This extra term modifies the Hessian diagonal, presumably lowering the so-called "tip effect" problem.

To quantify the differences of collectivity, we will calculate the degree of collectivity. This measure, first proposed by Bruschweiler [153], quantifies the number of atoms that are affected by a mode and the relative magnitude of the induced displacement. Its value can be calculated as

$$\kappa_i = \frac{1}{N} exp\left(-\sum_{j}^{N} \alpha\Delta R_j^2 log\left(\alpha\Delta R_j^2\right)\right), \tag{3.3}$$

where N is the number of atoms and ΔR_j is the atomic displacement described by mode i on atom j. The degree of collectivity is proportional to the exponential of the "information entropy" embedded in vector ΔR [141]. Its lower and upper bound is known (N^{-1} and 1 respectively). This allows us to normalize its value in the range $[0, 1]$, meaning 1 that the conformational change the mode produces is maximally collective.

As we can see in Fig. 3.1, the average collectivities of the CC modes are always lower than their IC counterparts. Furthermore, the degree of collectivity seems to decline as the cutoff varies and the elastic network becomes more dense. Conversely, the IC modes remain almost unaffected by the changes of the elastic network. It is worth noting that the ones calculated using the Hessian modification method have slightly higher values of collectivity and are, again, almost immune to the EN changes.

From the plot we can see that 9 Å is a good choice for the cutoff: it yields good collectivity values and it is in agreement with the conclusions of other studies [144].

The detailed view of mode collectivities for a cutoff of 9 Å is shown in Fig. 3.2. We can see how the collectivity of IC modes is higher, especially that of the lower frequency modes. The differences between 1ubq_cut and 1ubq are of particular interest. In these two cases, the collectivity values of the lower frequency modes of 1ubq are pretty small for all three methods, and they increase

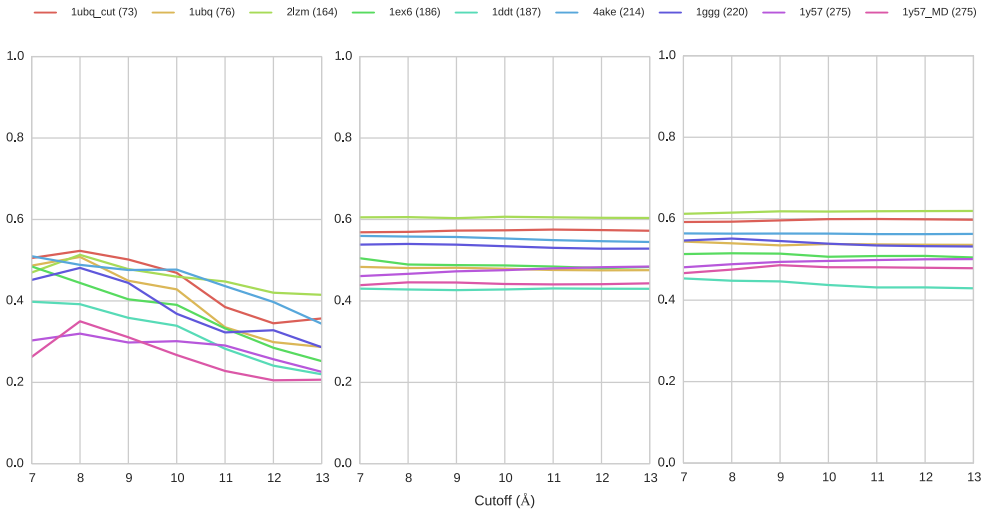


Figure 3.1: Degree of collectivity for each structure and calculation method. .

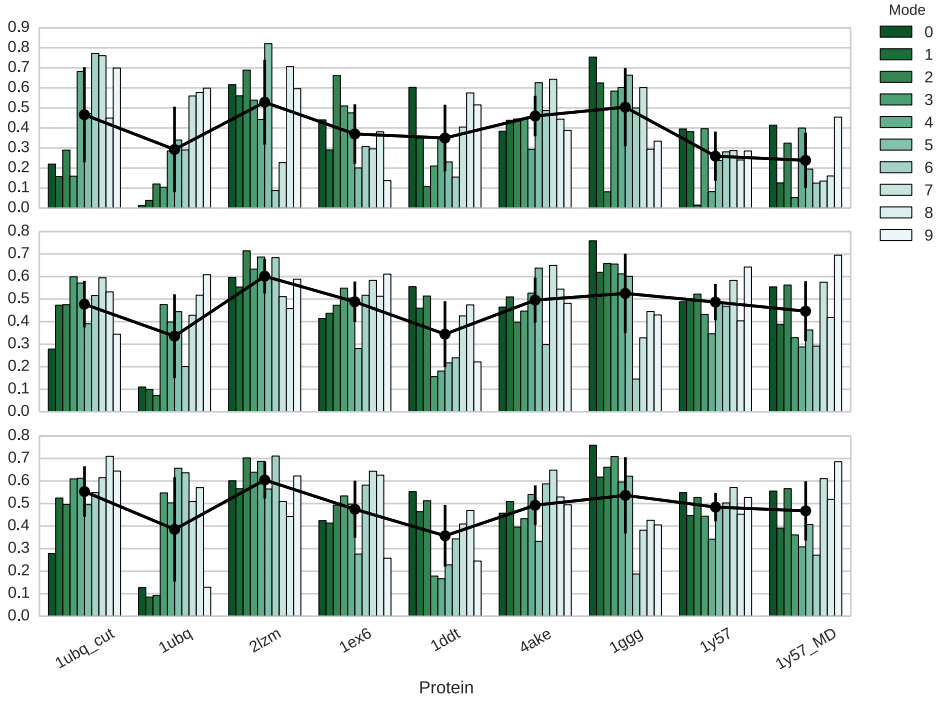


Figure 3.2: Detailed study of the degree of collectivity per mode, structure and method. Cutoff has been set to 9 Å . The Hessian modification method (m.H.) produces only slight improvements, generally concentrated in higher frequency modes.

perceptibly in 1ubq_cut. This must be caused by the only difference between these two structures: the appearance of the C-terminal loop. The structure with the loop (1ubq) is severely suffering from the "tip effect" due to the flexibility of the final loop, perfectly illustrating how this effect can worsen the collectivity of the modes.

From the analyses performed on this data set, we can conclude that IC modes have higher collectivity, and that this collectivity is more robust to changes in the elastic network.

3.5.1.2 Comparison between the CC and IC mode spaces

We also wonder to which extent the mode space spanned by the CC and IC modes is similar. To this end we will use two measures: the cumulative overlap and the root mean square inner product (RMSIP). Both of them are based on the mode overlap operation, which measures the projection of one mode over the other. It can be calculated as:

$$O_{ij} = \frac{\left|P_i.M_j\right|}{||P_i||||M_j||}. \tag{3.4}$$

Its value ranges from 0 to 1, meaning 1 a perfect overlap.

The Cumulative overlap [101] measures to which extent a range of modes can capture the motion of a single mode. It is calculated as

$$CO_i(k) = (\sum_{j}^{k} O_{ij}^2)^{\frac{1}{2}}, \tag{3.5}$$

where i is the mode we are checking and $[j,k]$ is the range of modes we will use to explain the first. Its value is, again, in the range from 0 to 1, meaning 1 a perfect match (assuming perfect orthogonality of the modes).

Finally, the RMSIP [142, 154] measures how the normal mode space spanned by a range of modes overlaps with another range of modes. It is calculated as

$$RMSIP(l,m) = \left(\frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{m}(P_i.M_j)^2\right)^{\frac{1}{2}}. \tag{3.6}$$

Its value is independent of the mode order and ranges from 0 to 1, meaning 1 that both normal mode spaces are identical.

It is important to note that the modes coming from the IC conversion and

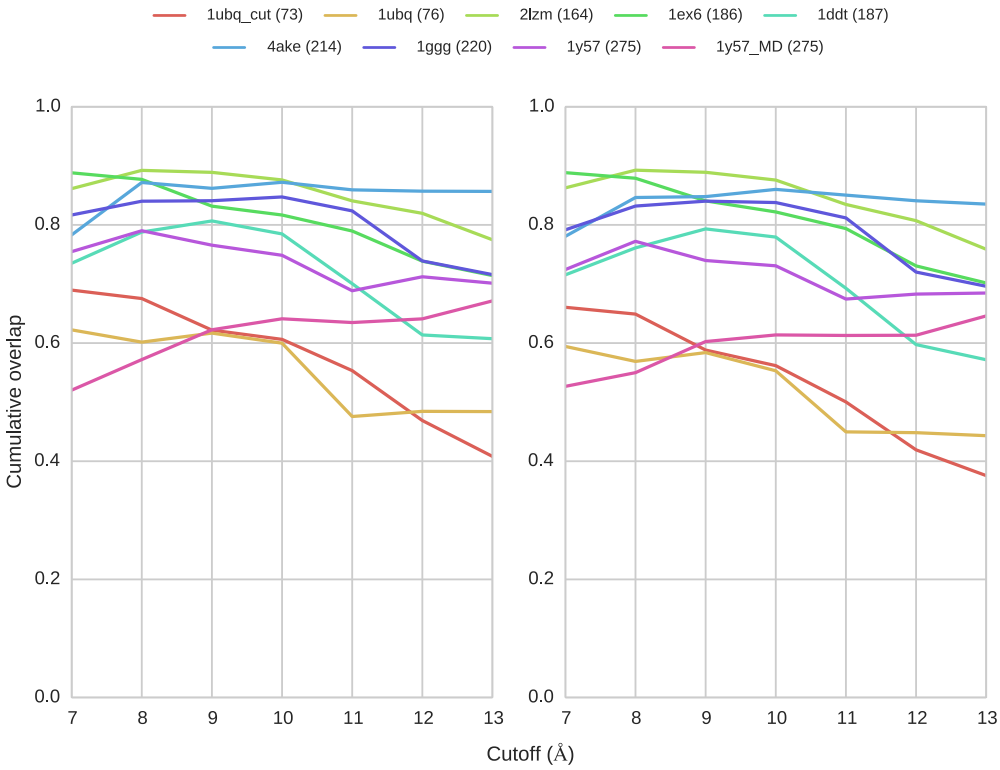


Figure 3.3: Average cumulative overlap for different cutoff distances, methods, and structures in our test set. Standard deviations are not shown for the sake of clarity.

PELE ANM model are defined for a different number of atoms (all heavy atoms in the first case, C_αs in the second) and, therefore, they represent very different mode spaces. In order to make the comparisons possible, we need to calculate the ANM modes for all heavy atoms.

The cumulative overlaps shown in Figs. 3.3 and 3.4 indicate that both mode spaces can explain each other successfully, being 1ubq, 1ubq_cut and 1y57_MD the only exceptions. In general, increasing the cutoff makes the differences between mode spaces more noticeable. The calculations performed with cutoff distance equal to 9 Å (Fig. 3.4), show that lower frequency modes are generally the ones that find a best correspondence with the modes of the other space.

Regarding the RMSIP for the modes calculated using a cutoff distance of 9 Å , the mode space overlap is higher for the subspace of the low frequency modes, and decreases when more high frequency modes are added to the calculation (see Fig. 3.5), which correlates well with the observations made for Fig. 3.4.

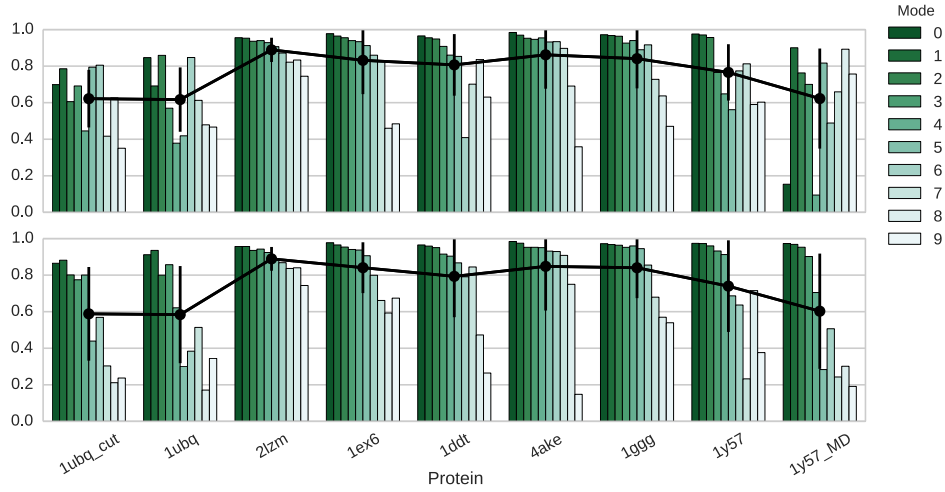


Figure 3.4: Detailed study of the cumulative overlap per mode, structure and method. Cutoff has been set to 9 Å . In general, the rightmost modes (higher frequencies) are the ones with worst overlap.

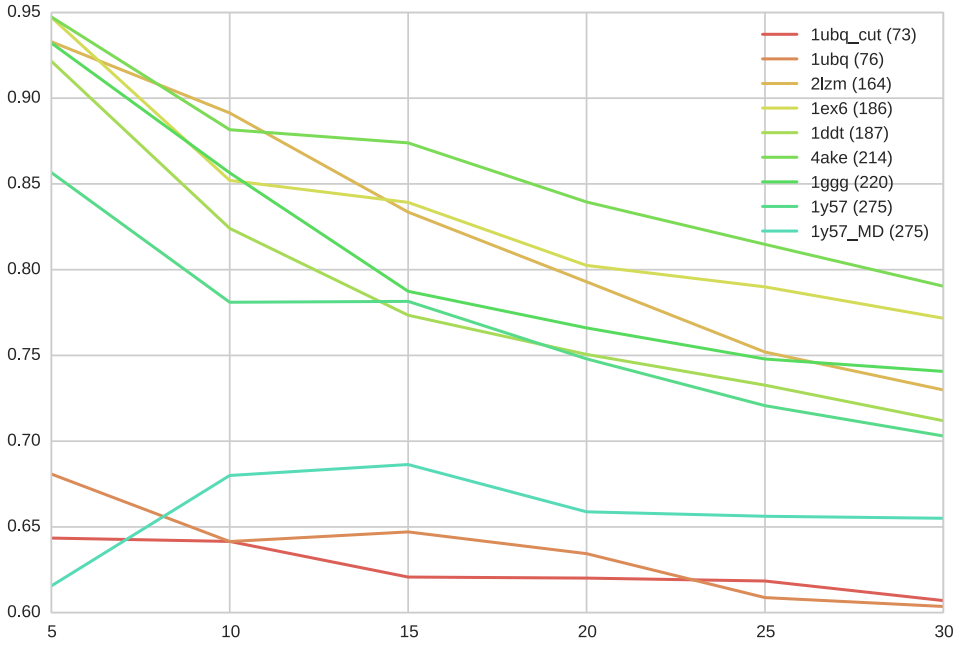


Figure 3.5: RMSIP of CC and IC mode spaces. The x-axis shows the upper limit of the mode space tested (e.g, 10 means that the first ten modes are to be used to obtain the RMSIP). Both spaces look to be very similar, at least for the 5 lowest frequency modes. The similarities decrease as we move to modes of higher frequencies, with the only exceptions already commented in the cumulative overlap study.

BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 18 2013, pages 2363–2364
doi:10.1093/bioinformatics/btt402

Structural bioinformatics

Advance Access publication July 10, 2013

pyRMSD: a Python package for efficient pairwise RMSD matrix calculation and handling

Víctor A. Gil[1] and Víctor Guallar[1,2,*]

[1]Joint BSC-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, 08034 Barcelona, Spain and [2]Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig Lluís Companys 23, E-08010 Barcelona, Spain

Associate Editor: Anna Tramontano

ABSTRACT

Summary: We introduce pyRMSD, an open source standalone Python package that aims at offering an integrative and efficient way of performing Root Mean Square Deviation (RMSD)-related calculations of large sets of structures. It is specially tuned to do fast collective RMSD calculations, as pairwise RMSD matrices, implementing up to three well-known superposition algorithms. pyRMSD provides its own symmetric distance matrix class that, besides the fact that it can be used as a regular matrix, helps to save memory and increases memory access speed. This last feature can dramatically improve the overall performance of any Python algorithm using it. In addition, its extensibility, testing suites and documentation make it a good choice to those in need of a workbench for developing or testing new algorithms.

Availability: The source code (under MIT license), installer, test suites and benchmarks can be found at https://pele.bsc.es/ under the tools section.

Contact: victor.guallar@bsc.es

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on April 24, 2013; revised on June 11, 2013; accepted on July 5, 2013

1 INTRODUCTION

As molecular modeling keeps expanding, obtaining the Root Mean Square Deviation (RMSD) with optimum superposition for a large set of structures in an efficient and fast manner is a necessity. Clustering methods, for example, which are becoming increasingly popular as trajectory analysis and compression tools (Karpen *et al*., 1993; Phillips *et al*., 2011), can benefit from the use of a pre-calculated pairwise distance matrix or even totally depend on it, e.g. Spectral Clustering (Luxburg, 2007). However, as hardware and algorithms improve, the output size of simulations grows bigger, and the calculation of the distance matrix becomes the bottleneck in any process depending on it. There are several implementations of the different superposition algorithms, which are written in wide spectra of programming languages. Almost all Molecular Dynamics packages and biomolecule handling software include their own RMSD calculation tools. Every time programmers need to use an external RMSD solution in a project, they have two options. The first one is to use an external source or library, which requires previous knowledge of the language in which it was written and its dependencies. A second option is to use a precompiled tool with a bigger scope, which means creating an interface with their own application by writing wrappers and output converters (with the consequent performance loss). In general, the main problems to face are fragmentation, excess of or missing features, bad documentation, lack of sources and the intrinsic difficulty of the languages used. pyRMSD is a Python package that overcomes all the above problems in the following way:

- It is totally focused on the calculation of RMSD. It provides solutions for all the usual RMSD problems and is specially tuned for RMSD collective calculations, like pairwise RMSD matrices, a feature that is usually missing in most utilities.
- Python (www.python.org) is an easy to learn and use programming language, which has an extensive library pool that includes wrappers for almost all libraries used in science. This makes it one of the better languages for scientific software prototyping and development.
- As pure Python implementations have a poor performance (even when using fine tuned packages as numpy), pyRMSD uses Python C extensions with OpenMP and CUDA code, allowing the full use of multicore machines and Graphics Processing Units (GPU).
- It implements the most important superposition algorithms in the same place.
- It is documented, well tested and open source; therefore, it can be the perfect workbench for any experienced user who wants to develop and test their own superposition algorithms.

2 IMPLEMENTATION

2.1 Features

pyRMSD is built around two main classes: the *RMSDCalculator* and the *CondensedMatrix*. The *RMSDCalculator* class provides a straightforward interface to three superposition algorithm implementations, as well as some convenience methods to set up their options: Kabsch's superposition algorithm (Kabsch, 1978), QTRFIT (Heisterberg, 1990) and the Quaternion Characteristic Polinomial method (QCP) (Theobald, 2005). All have been written as Python C extensions, with serial and parallel (OpenMP) versions for the first two and an additional CUDA version for the last.

*To whom correspondence should be addressed.

© The Author 2013. Published by Oxford University Press. All rights reserved. For Permissions, please e-mail: journals.permissions@oup.com

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on February 7, 2016

RMSDCalculator's methods cover all the usual scenarios for superposition and RMSD calculation:

(1) Pairwise RMSD calculation.
(2) Reference versus the rest of the set.
(3) Reference versus following conformations.
(4) Calculation of a pairwise RMSD matrix of the whole set.
(5) Iterative superposition of a set of conformations.

Moreover, it offers two additional options that further extend the previous methods. The first one is allowing the modification of input coordinates to obtain the superposed conformations. The second one is to the use of different coordinate sets for superposition and RMSD calculation.

The *CondensedMatrix* class models a symmetric squared matrix. It allows the same row/column access of a regular matrix, storing only the upper triangle and thus saving half of the memory. The class has been completely written in C, allowing access times which are up to 6× faster compared with its Python counterpart. As a consequence, any algorithm that requires intensive matrix read access improves its performance. For instance, our cardinality function benchmark, available in the benchmarks folder, shows a 100× free speedup just by using it.

pyRMSD also provides two small helper classes that make the process of generating a pairwise RMSD matrix easier. The *Reader* class obtains the coordinate sets by means of a simple and fast C written PDB reader. Finally, the *MatrixHandler* class is capable of creating a distance matrix from a set of coordinates and managing its persistence, with functions to load and save matrices from disk.

2.2 Usage

The following code snippet illustrates the creation and access of a pairwise RMSD matrix of a 35 k frames trajectory, available in the 'benchmark/data' folder, using the QCP superposition algorithm, in its CUDA version:

```
from pyRMSD.matrixHandler import MatrixHandler
from pyRMSD.utils.proteinReading import Reader
cords = Reader().readThisFile('amber_35k.pdb')\
                .gettingOnlyCAs().read()
matrix = MatrixHandler()\
                .createMatrix(coords, 'QCP_OMP_CALCULATOR')
```

Here, we can find a minimum subset of all the features of pyRMSD and of the *MatrixHandler* class itself. However, it is a good example of how this class nicely encapsulates all the steps of creating a matrix and of the succinct interface presented to the user.

2.3 Performance

Using pyRMSD, we have coded a set of benchmark scripts to understand the performance differences between the three implemented algorithms. We have observed that, in all studied scenarios, QCP is the faster method.

We have also compared the performance of our four QCP implementations. Compared with the serial code, our OpenMP version is 5× faster; our CUDA-based implementation shows a 11× speedup (see Fig. 1). This leads us to conclude that GPU implementations can really make the difference in this kind of problems.

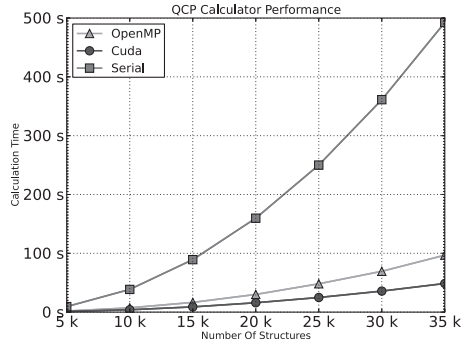


Fig. 1. QCP calculator performance over a Ubiquitin trajectory (only CAs) using a 6 cores Intel Xeon E5649 CPU with an NVIDIA M2090 GPU. OpenMP version reaches a 5× speedup. CUDA version gets a maximum 11× speedup (almost 12 million RMSD calculations per second)

These and other benchmarks, as well as a comparison with other packages, are discussed in depth in the Supplementary Data.

3 CONCLUSIONS

We have created pyRMSD, a user-friendly RMSD focused Python package, which allows, besides other functionalities, the efficient creation of RMSD pairwise matrices. Its design provides a natural way of accessing its functionalities making it a good candidate to be used in bigger packages to replace slower RMSD functions. This is specially true for those who need to calculate and access large pairwise RMSD matrices, as clustering-related packages.

Funding: European project PELE (ERC-2009-Adg 25027).

Conflict of Interest: none declared.

REFERENCES

Heisterberg,D.J. (1990) QTRFIT algorithm for superimposing two similar rigid molecules. The Ohio Supercomputer Center, Ohio State University, Columbus, OH.

Kabsch,W. (1978) A discussion of the solution for the best rotation to relate two sets of vectors. *Acta. Crystallogr. A*, **34**, 827–828.

Karpen,M.E. *et al.* (1993) Statistical clustering techniques for the analysis of long molecular dynamics trajectories: analysis of 2.2-ns trajectories of YPGDV. *Biochemistry*, **32**, 412–420.

Luxburg,U. (2007) A tutorial on spectral clustering. *Stat. Comp.*, **17**, 395–416.

Phillips,J.L. *et al.* (2011) Validating clustering of molecular dynamics simulations using polymer models. *BMC Bioinformatics*, **12**, 445.

Theobald,D.L. (2005) Rapid calculation of RMSDs using a quaternion-based characteristic polynomial. *Acta. Crystallogr. A*, **61**, 478–480.

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on February 7, 2016

3.6 Supplementary materials for: pyRMSD: a Python package for efficient pairwise RMSD matrix calculation and handling

Comparing the performance of different algorithms is not straightforward. We can deduce some theoretical bounds, but these might be too general to have any practical value, especially when comparing similar algorithms. When this happens, the only way to go is to compare implementations.

In our implementations, good code design (especially reusability) had a bigger priority than optimality. All three algorithm implementations have been coded sharing the same framework, reusing as many pieces of code as possible. This enhances the comparability of our implementations, which may be suboptimal in the same degree, giving us a unique opportunity of making a more fair performance comparison between them.

All benchmark tests were performed in BSC's Minotauro [155], which has been built with Intel Xeon E5649 CPUs and NVIDIA M2090 GPUs. Only 1 GPU was reserved for CUDA runs. OpenMP runs used six threads at most.

3.6.1 Algorithm performance comparison

In order to get a good overview of each algorithm's performance, we have checked the time that each of the three algorithms need to complete the execution of each one of pyRMSD's basic methods (oneVsFollowing, pairwiseMatrix and iterativeSuperposition). A 30k trajectory of Ubiquitin (reading only C_α atoms) was used. The oneVsFollowing method has been tested without and with input coordinates rotation, as this last adds overhead to QCP implementation due to the mandatory rotation matrix calculation (QCP does not need to calculate it otherwise).

For basic linear operations (see Fig. 3.6), QCP has the best performance of all three, even when the rotation matrix is calculated (oneVsFollowing(r) and iterativeSuperposition). The use of OpenMP smooths differences so much that choosing one algorithm over the other becomes a mere matter of preference (see Fig. 3.7).

However, things change when comparing the performance of the pairwise matrix generation (see Fig. 3.8). Small performance differences get amplified because of the quadratic nature of the problem. In this case, we observe that QCP excels by achieving a 2x/4x speedup with respect to the other two, in both serial and OpenMP modes.

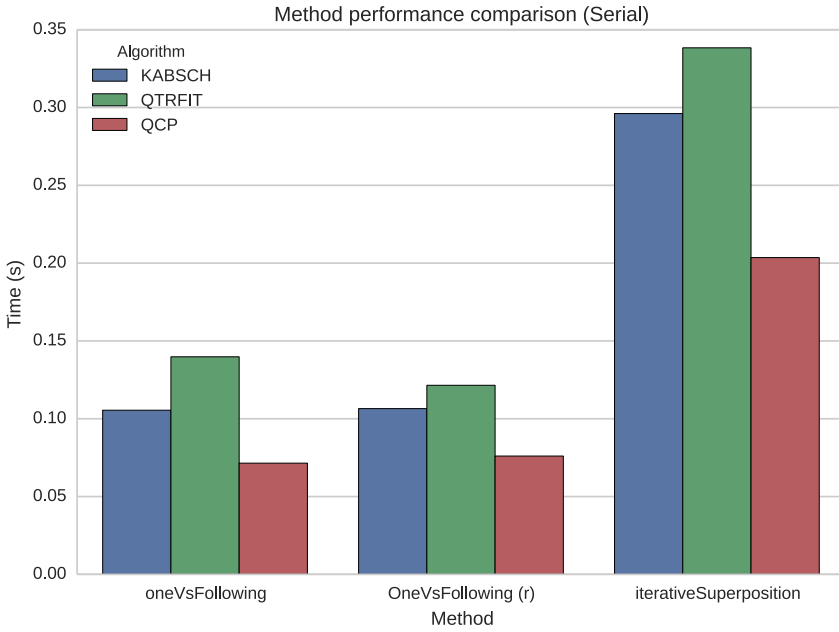


Figure 3.6: Comparison of the execution of three methods for the three available algorithm implementations (serial).

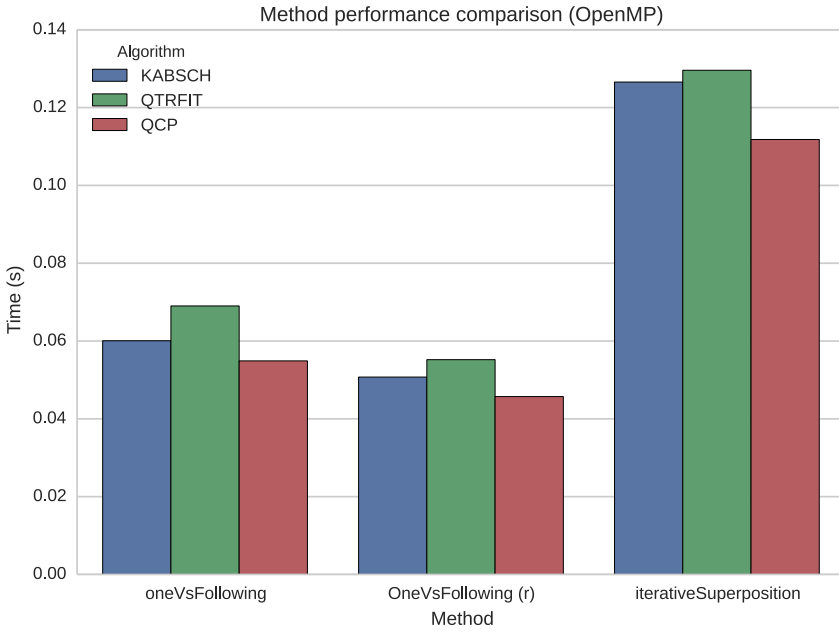


Figure 3.7: Comparison of the execution of three methods for the three available algorithm implementations (OpenMP).

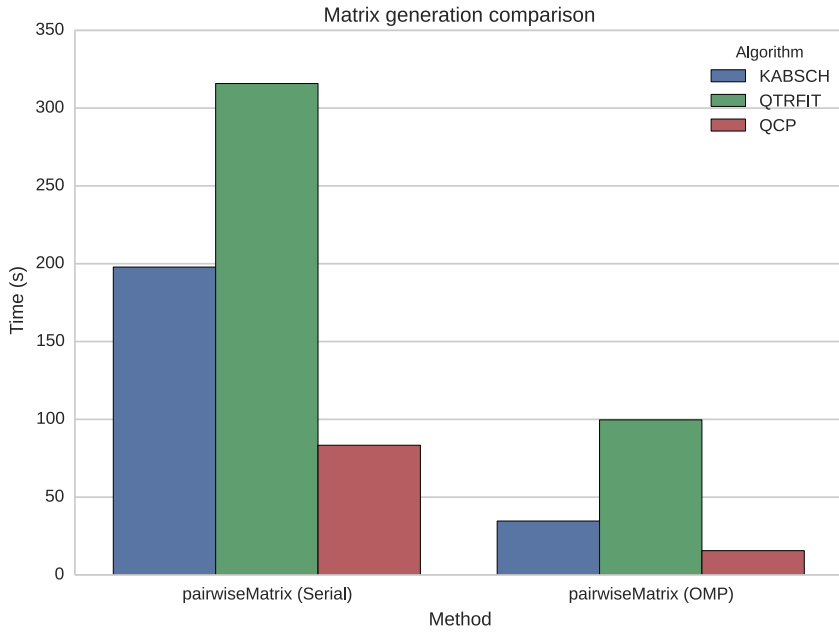


Figure 3.8: Calculation time of a pairwise matrix from a 30k frames trajectory.

3.6.2 QCP performance

We want to compare all four implementations of QCP algorithm: serial, OpenMP, CUDA and CUDA with full matrix memory allocation into the device. To this end, we will use the pairwiseMatrix method over Ubiquitin trajectories of 5, 10, 15, 20, 25, 30 and 35k frames.

We can see in the resulting plot (Fig. 3.9) that OpenMP version is about 5x faster than the serial one, and CUDA version is about 8,5x faster.

After profiling the CUDA version, we saw that a considerable part of the time was spent in memory transactions from the device to the host. The RMSD matrix is calculated line by line and, after each one of this calculations, the host matrix representation is updated. This method can be successfully used in a broad range of GPUs, as it does not require cards with big amounts of RAM. We implemented another method that holds the entire matrix in GPU's RAM. The speedup is greater (11x compared with serial code), as memory transactions are performed only once (Fig. 3.10).

Finally, there are two things worth mentioning. The first one is that our CUDA implementation can be further improved by enhancing work balance and memory coalescence. The second is that the use of single point precision in our QCP CUDA implementation, contrary to what is expected, does not per-

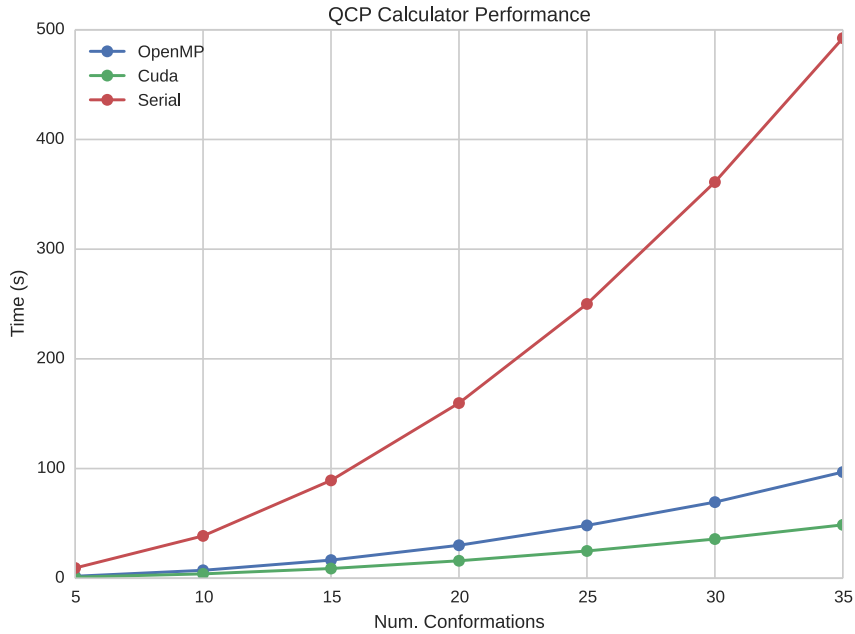


Figure 3.9: Performance comparison of serial, OpenMP and CUDA (single-precision) implementations of the QCP algorithm.

form substantially better than the double precision implementation. The reason for this behavior is again the big effort put into generalizing the code. pyRMSD uses internally double-precision arrays to store coordinates and RMSD values. This implies that, when using GPUs without double-precision support, a single-precision temporary buffer is to be filled at every host to device memory move.

3.6.3 Input size response of QCP implementation

The last benchmark established a clear relationship between the size of a trajectory (in frames) and the time needed to do calculations. In this benchmark we want to fix the number of frames and test the impact of biomolecule sizes in performance. The number of frames of the trajectory will be 10k, and the number of atoms of each one of the conformers will be artificially increased at every step in order to calculate the pairwise RMSD matrix.

Both CUDA and OpenMP implementations, show a linear increase of the time needed to calculate the matrix (Fig. 3.11). Incrementing conformer size does not increase or decrease performance.

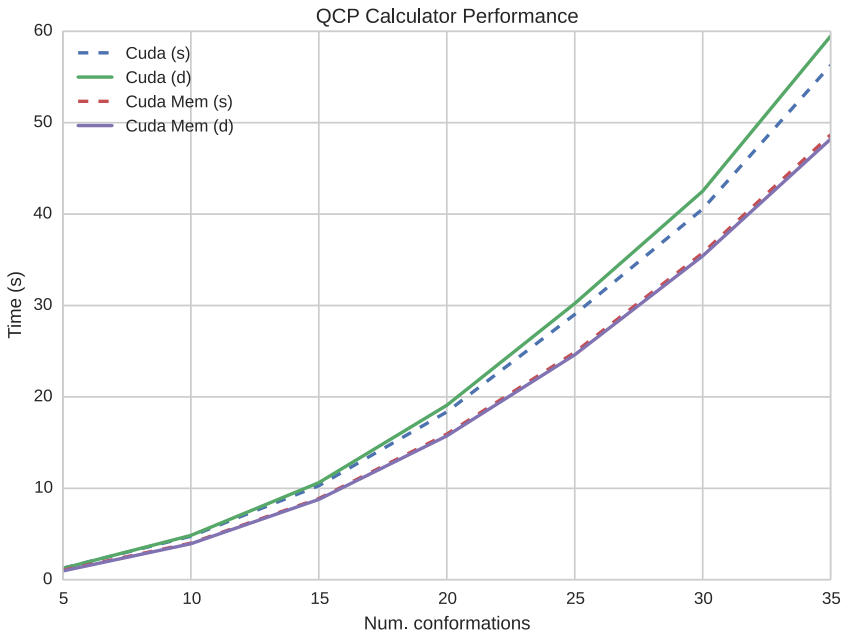


Figure 3.10: Performance comparison of CUDA implementations. 'Mem' versions hold the entire matrix into memory, (s) versions use single-precision arrays and (d) versions use double-precision arrays.

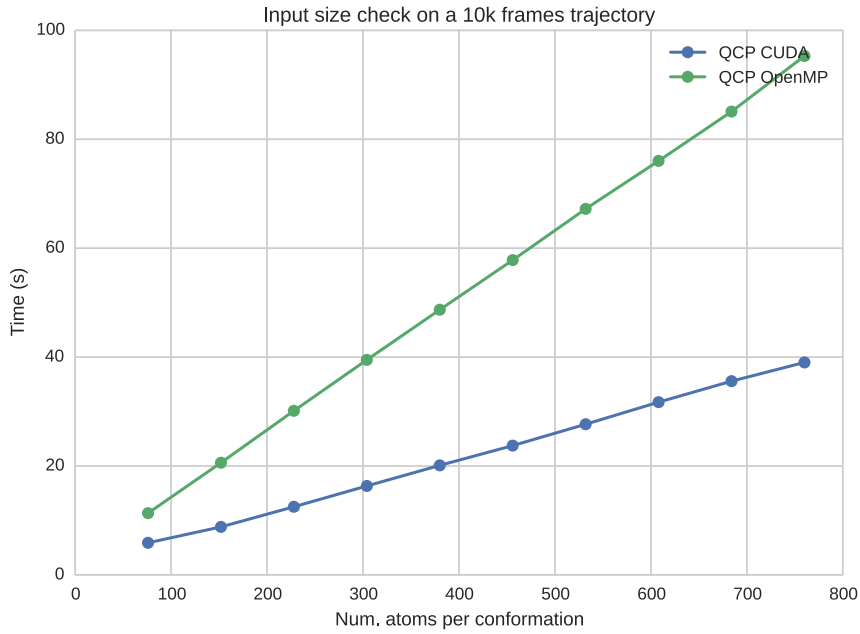


Figure 3.11: Input size response of the OpenMP and CUDA versions of the QCP algorithm.

3.6.4 Accuracy check

While KABSCH algorithm tries to find an optimal rotation matrix, QTRFIT and QCP will use quaternions in order to get this rotation. Does the base method affect accuracy? In this test, we have applied the oneVsFollowing method over the first frame of a 10k frames Ubiquitin trajectory. Then we have calculated the root mean square of the differences, which will be our index of RMSD value variation.

In Table 3.1, we can see that all implementations have an RMS different than 0, which means that all algorithms have calculated different RMSD values. Kabsch's algorithm and QTRFIT have close results in spite of their different approaches to calculating the superposition. QCP CUDA single-precision floating point version differs the most, being the double-precision version the most accurate of both.

It is important to note that, although QCP solves the problems of prior methods like Diamond's [156] algorithm instability in rotations close to 180ž, it can suffer from convergence problems derived from the use of Newton-Raphson root-finding method.

However, differences are generally small, and RMSD is often used qual-

Method	KABSCH	QTRFIT	QCP	QCP CUDA (Single)	QCP CUDA (Double)
KABSCH		2.308e-12	2.678e-12	1.028e-03	2.689e-12
QTRFIT			1.568e-12	1.028e-03	1.547e-12
QCP				1.028e-03	8.926e-13

Table 3.1: Root Mean Square of the RMSD array differences for each of the algorithms.

itatively, which means that, unless we require high precisions, all algorithm implementations are interchangeable.

3.6.5 OpenMP scalability

We want to study the scalability of the OpenMP version of each of the algorithms. To achieve it we have calculated the pairwise RMSD matrix of a 30k frames trajectory of Ubiquitin, using a different number of threads for each execution (from 1 to 6).

It is not difficult to see that we obtain a linear speedup, with a maximum of 5.6-5.8x for six threads (compared with the one thread run), which is close to the theoretical maximum speedup (6x for six threads) (Fig. 3.12, 3.13).

3.6.6 Comparison with existing packages

As a final test, we will compare pyRMSD with other software packages that include RMSD calculation features. We have chosen four publicly available open source packages:

g_rms Is a C-written command line program part of the Gromacs [157] suite. Its main feature is the fast creation of distance matrices from trajectories.

Prody [140] A Python package which offers very interesting features to load and analyze biomolecule trajectories, including a complete PDB parser and a powerful selection language.

Biopython [158] A mature Python package which offers numerous bioinformatic computational tools.

PyVib2 [159] A pure Python package used to analyze vibrational motion and spectra of molecules.

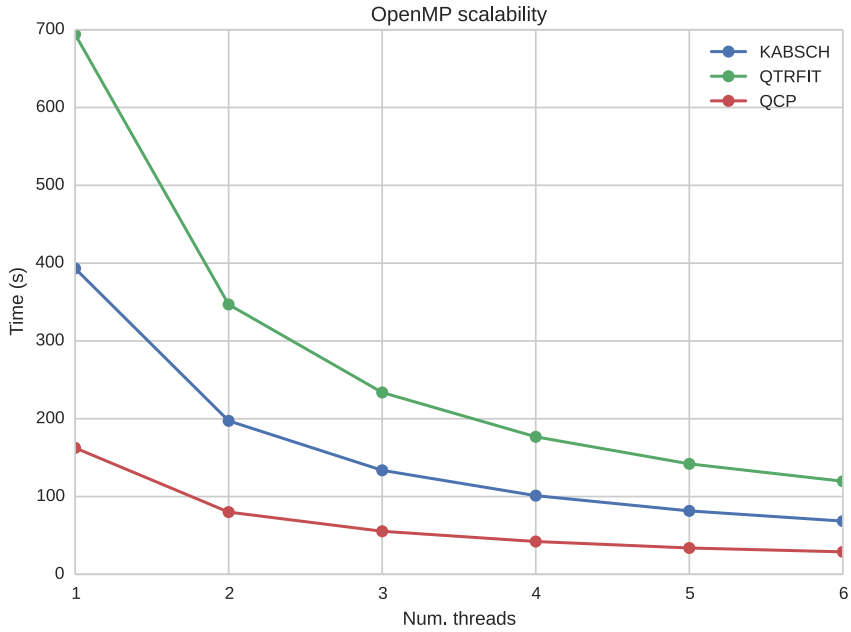


Figure 3.12: Time needed to compute a pairwise matrix from a 30k frames trajectory using a different number of threads.

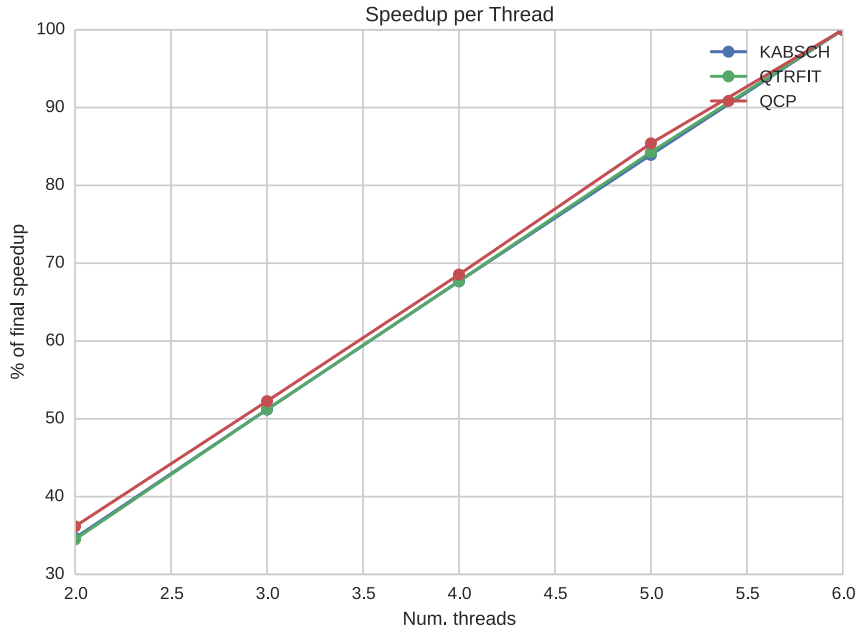


Figure 3.13: Percentage of speedup per thread added to the calculation. Speedup is almost linear with number of threads.

Our comparison will be based on two measures: calculation time and integration complexity. To this aim we have created 4 scripts [3]. Every script calculates the RMSD matrix of a trajectory twice, first using one of the aforementioned packages, and then replicating the same calculation using pyRMSD (serial). Integration complexity has been measured as the ratio between the number of effective code lines necessary to program the task with the tested package and pyRMSD. Performance has been calculated as the ratio of the time needed to complete the task by the tested package over the time required by pyRMSD. All tests were performed on a workstation with an Intel Xeon W3530 CPU (four cores at 2.80GHz) with 12GB of RAM.

Table 3.2 shows that pyRMSD is always faster than the other packages. In the last three cases, this is because that RMSD calculations are also written in pure python (which includes the use of numpy), while in pyRMSD these have been written as C extensions. These packages have not been specifically designed to handle RMSD collective operations, which explains why more lines of code are needed. Biopython and PyVib2 RMSD-related features are indeed constrained to the pairwise RMSD case, while Prody can calculate rows of the matrix using only one function. This makes the last faster and easier to use in this scenario. g_rms is considerably faster than the others, as it has been written in C and because of its narrower scope. However, to use it within a Python script, the user will need to create a wrapper to control the program execution (here simplified by the utilization of the 'expect' command), and a method to parse program results. This adds extra complexity to the code as well as a performance penalty.

[3] The scripts can be found in `https://github.com/victor-gil-sepulveda/pyRMSD-Comparison.git`

Table 3.2: Comparison of the time, lines of code, speedup and integration complexity (I.C.) needed to complete an RMSD collective operation.

Method	Frames	Time (s)	Time pyRMSD(s)	Lines of code	Lines of code (pyRMSD)	Speedup	I. C.
g_rms	10	0.0564	0.0466	24	3	1.21	8
	100	0.7421	0.488			1.52	
	1000	62.5245	14.6778			4.26	
Prody	10	0.1698	0.0008	6	1	212.25	6
	100	3.408	0.0587			58.06	
	1000	312.7806	6.0187			51.97	
PyVib2	10	4.069	0.0001	7	1	40690.00	7
	100	450.1247	0.0995			4523.87	
	1000	30332.6163	10.372			2924.47	
Biopython	10	2.012	0.0008	8	1	2515.00	8
	100	219.6425	0.0572			3839.90	
	1000	22003.5187	6.0966			3609.15	

JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

pyProCT: Automated Cluster Analysis for Structural Bioinformatics

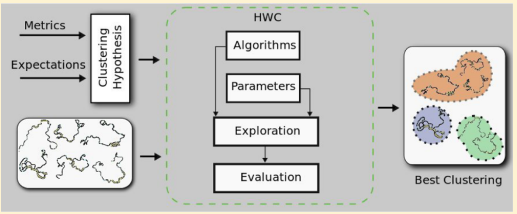
Víctor A. Gil[†] and Víctor Guallar*[,†,‡]

[†]Joint BSC-CRG-IRB Research Program in Computational Biology, Barcelona Supercomputing Center, Jordi Girona 29, 08034 Barcelona, Spain

[‡]Institució Catalana de Recerca i Estudis Avançats (ICREA), Passeig Lluís Companys 23, E-08010 Barcelona, Spain

Supporting Information

ABSTRACT: Cluster analysis is becoming a relevant tool in structural bioinformatics. It allows analyzing large conformational ensembles in order to extract features or diminish redundancy, or just as a first step for other methods. Unfortunately, the successfulness of this analysis strongly depends on the data set traits, the chosen algorithm, and its parameters, which can lead to poor or even erroneous results not easily detected. In order to overcome this problem, we have developed pyProCT, a Python open source cluster analysis toolkit specially designed to be used with ensembles of biomolecule conformations. pyProCT implements an automated protocol to choose the clustering algorithm and parameters that produce the best results for a particular data set. It offers different levels of customization according to users' expertise. Moreover, pyProCT has been designed as a collection of interchangeable libraries, making it easier to reuse it as part of other programs.



1. INTRODUCTION

In structural bioinformatics, data from computational experiments is produced and consumed at great speed. Improvements in software, together with the development of CPUs and accelerators have contributed to increase this data production. Conformational sampling software, such as molecular dynamics or Monte Carlo techniques, have specially enjoyed the benefits of this fast hardware evolution.[1,2] Since raw data is rarely useful per se, it must be processed to extract useful information. Thus, as data sets increase, so does the importance of having unsupervised and computationally cheap tools to analyze them.

Cluster analysis is one of the most successful and widespread tools to postprocess and analyze ensembles of conformations. It has been successfully used to discover near-native structures,[3] to study peptide folding,[4] or as a first step to extract kinetics information[5] (e.g., reaction pathways[6] or relative free energies[7]) among others. As a preprocessing step, it can be used to reduce data sets by extracting the most representative conformations, so that any subsequent computationally expensive postprocessing becomes more affordable.

There are many general use clustering algorithms, all having their own advantages and drawbacks. Some others, such as the ART-2′ algorithm[8] or the fuzzy clustering algorithm presented by Gordon et al.,[9] were specifically created to improve the geometrical analysis of conformational ensembles; others even account for the ensemble's kinetic properties.[5,10]

Implementations of the most traditional algorithms are included as tools into larger multipurpose suites, such as the well-known "ptraj",[11] which can be found in AMBER's suite or GROMACS' "g_cluster".[12] Others, such as "Wordom",[13] offer some clustering options along with tools to analyze MD trajectories. However, there are few software packages that are specifically written to perform cluster analysis over conformation ensembles.

Such diversification gives users plenty of options to choose the best suited algorithm for their problem types and data sets, in order to obtain the best quality clustering. Nevertheless, cluster analysis techniques are not bullet-proof and can be easily misused. This can be dangerous, as unexpected results will go unnoticed, contaminating any subsequent process. Indeed, the successfulness of a cluster analysis process depends mainly on two basic factors, which, if overlooked, can lead to poor or erroneous results:

- Used algorithm: Every clustering algorithm makes its own assumptions about the data set. It is well-known that k-means will perform better when applied to data sets composed of convex clusters and that hierarchical algorithms will obtain better results if the underlying structure is composed of elongated clusters. DBSCAN[14] will have problems clustering data sets with big density differences. The intrinsic characteristics of the data set in terms of cluster size, shape, and density can be a determinant factor to generate a good quality clustering. This reasoning also works the other way round: the clustering algorithm will, at the same time, impose some characteristics to the clusters it produces.
- Parametrization: Many algorithms need to be parametrized before using them. Clustering results can change if these parameters are not tuned carefully. Good examples are the cutoff in hierarchical algorithms

Received: April 10, 2014
Published: July 18, 2014

 © 2014 American Chemical Society dx.doi.org/10.1021/ct500306s | J. Chem. Theory Comput. 2014, 10, 3236−3243

or the 'Eps' and 'MinPts' parameters in DBSCAN (widely studied in the original article[14] and in more recent ones[15]). One parameter, the number of final clusters, has always deserved special attention. Algorithms that need this parameter will usually partition the data set in exactly the number of clusters they are told to use. This can result in a totally artificial partition that will not capture the underlying structure (if any).

Unfortunately, this selection and tuning step requires extensive knowledge of cluster analysis. Users, often experts in other problem domains, have tight time constraints that prevent them from acquiring this knowledge. Therefore, algorithm choice is typically based upon its appearance in other publications, the algorithm reputation, or simply the availability of its implementations. Parameters are, in turn, chosen either randomly or just using implementation default values, which are not always the best suited for the data set under study (an example of the consequences of a bad algorithm/parameters choice can be seen in Figure 4 of Supporting Information). Finally, once the target clustering has been produced, users do not have enough tools to asses its quality, thus reducing the chances of improving it by trial and error.

Here, we want to propose a novel cluster analysis methodology, the Hypothesis-driven Clustering Exploration (HCE), that will help to overcome the aforementioned problems. HCE's approach first converts the user's problem domain knowledge into a clustering hypothesis. Then, an exploration of the clustering space using up to five different algorithms is performed in order to obtain the best clustering fitting the hypothesis. By using HCE, users do not need to be familiar with cluster analysis techniques and its caveats but only to be able to define their problems in terms of a clustering hypothesis. The fact of not having to deal with each algorithm's distinctive features and parameters can potentially improve the quality of the cluster analysis results, which will also better fit the user's expectations and purpose.

HCE methodology has been implemented into pyProCT, an open source (MIT licensed) Python (http://www.python.org) toolkit. This software extends HCE by implementing two of the most common use cases: discovering hidden features of the data set (cluster analysis with prototype extraction) and trajectory compression. pyProCT can also be a powerful tool for more experienced clustering users. Thanks to its highly configurable input script, most of the software behavior can be customized, allowing expert users to better exploit their knowledge. Moreover, the object oriented implementation will allow any developer to easily add new modules or modify the provided ones, allowing them to interface with already written packages, which will be able to benefit from any of the implemented cluster analysis tools.

Finally, pyProCT offers a built-in graphical user interface (GUI) that greatly enhances its usability. Among other features, it allows users to visually review the cluster analysis results so that possible errors can be rapidly detected and the hypothesis improved.

2. METHODS AND PROTOCOL

pyProCT implements the HCE methodology by performing a four-step protocol: distance matrix calculation, clustering space exploration, clustering evaluation, and data extraction (see Figure 1).

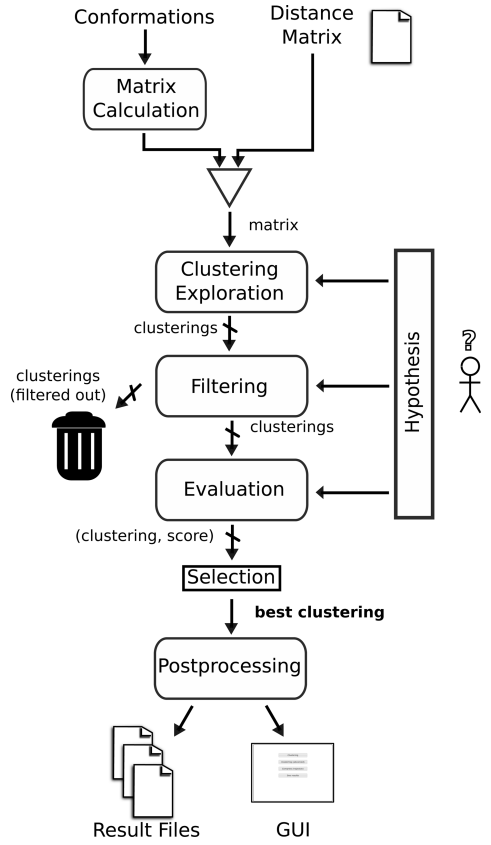


Figure 1. Implementation of HCE methodology in pyProCT.

2.1. Distance Matrix Calculation. Clustering algorithms require to evaluate several times a similarity function that is to be applied to all pairs of elements of the data set. Being one of the computational bottlenecks, its performance can speed up notably by precalculating a pairwise distance matrix. pyProCT offers three matrix generation options:

- Root mean square distance (RMSD) matrix: The RMSD matrix is generated by pairwise superposing all conformations to then calculate their coordinates RMSD. Different parts of the system can be selected to do the superposition and RMSD calculation step. This kind of matrix is the most used in conformational clustering projects (e.g., to cluster protein conformations based on their α carbon RMSD).
- Euclidean distance matrix: Its calculation also involves a two-step process. First, all conformations will be iteratively superimposed (all their coordinates or just a selected subset). Second, the geometric centers of the part of the system we want to analyze are calculated. The pairwise euclidean distances of these centers form the distance matrix. This method is preferred for scenarios where users want to extract information about the relative placement of one part of the system in motion

with respect to a point of reference (e.g., ligand−receptor systems).

- External matrix loading: As the distance metrics included may not be the best for all systems[16,17] and users may want to implement their own similarity functions, the third option allows them to load an external pregenerated matrix.

2.2. Clustering Space Exploration. The number of ways of grouping the elements of a data set, the clustering space, is huge (superexponential in n[18]). Within this space, it is accepted that only few clusterings have enough quality to attain valuable information; the existence of a unique clustering is a subject of debate.[19,20] Obtaining these representative clusterings might involve an exhaustive search of the clustering space assessing their quality. This approach is, in most cases, computationally unaffordable due to the combinatorial explosion.

pyProCT's approach uses a heterogeneous set of clustering algorithms as exploration agents that perform a guided search of the clustering space restricted to the area containing meaningful (not random) clusterings. The more different the used algorithms are, the wider the exploration is, thus increasing the probability of visiting the best quality clustering. Currently, pyProCT implements one representative of each of the most important clustering algorithm families, namely, hierarchical clustering,[21] k-medoids (partitive algorithms), DBSCAN,[14] gromos[4] (both density-based algorithms), and spectral clustering[19] representing connectivity/spectra-based clustering family.

Since the choice of the algorithm's parameters affects notably the result, pyProCT exploits it as another source of variability to yet widen the explored area of the clustering space. In our implementation, the working hypothesis is also involved in the strategies to obtain the parameters for each algorithm.

A brief description of the algorithms and the parametrization methods implemented are further explained in Supporting Information section 1.

2.3. Clustering Evaluation. Some authors warn that 'good quality' clusterings are those that allow users to accomplish their goals.[22] Therefore, it is not possible to asses the quality of a clustering without a previous definition of user's expectations. In pyProCT, this is achieved through the definition of a working hypothesis that is composed of a mixture of subjective and objective elements. Subjective elements (e.g., an estimation of the final number of clusters or the noise the user thinks the data set has) add contextual and domain-based information, directly derived from the users' purpose and expectations as well as their expertise in the system under study.

In pyProCT, the best solution is the one that best fits the clustering hypothesis. This hypothesis is used twice in the evaluation step: to eliminate clusterings too different from user's expectations and to score the remaining ones.

- Filtering: All unsuitable clusterings generated during the exploration are rejected. In this step, the expected number of clusters, the amount of produced noise, and the cluster size (number of conformations inside that cluster) are checked. The purpose of this filtering is twofold: it enforces the clustering hypothesis and improves the overall efficiency of the process, as it reduces the number of clusterings to be evaluated in the next step.
- Clustering evaluation: The specification of metric-based quality criteria adds an objective dimension to the hypothesis by describing measurable traits of each solution (e.g., cluster separation). Clustering quality is usually measured using two approaches. External cluster validation indices compare clusterings with an already known correct answer.[23−25] Instead, internal cluster validation indices (ICVs) only use the internal information on the clustering to evaluate it (e.g., distance between elements) to evaluate it. Given that the correct answer is not known a priori, pyProCT only uses ICVs. Unfortunately, these indices tend to impose their assumptions about the data set layout and characteristics, penalizing those that do not fit them. It has been observed, however, that combining some ICVs can help to overcome this problem.[26] One or more criteria (i.e., a scoring function based on a linear combination of normalized ICVs) must be defined. The clustering with the highest score for any of them is chosen to be the best clustering. Users are also ultimately responsible for choosing the ICVs that compose the criteria. Similar strategies have been used previously in other works, using the silhouette coefficient,[27] or Pal et al.[28] approach, which is closer to the method discussed here.

pyProCT's default behavior defines a simple criteria that fosters both cluster compactness and separation. More information about the implemented ICVs and other query types can be found in Supporting Information section 2.

The exploration and evaluation steps are the core of the HCE strategy. It prevents users from having to learn about clustering algorithms and the meaning and use of their parameters, thus making the whole cluster analysis process less prone to generate unexpected/undesired results. However, the success of the process will still depend on the ability of users to express their hypothesis using the currently implemented tools. We do not recommend to start with too specific hypotheses unless the user has good knowledge of the data set.

2.4. Postprocessing. Finally, the best clustering will be used to extract information from the data sets. Currently, pyProCT implements two use cases.

- Cluster analysis: A set containing all the cluster prototypes (the prototype is the central conformation of a cluster, i.e., the conformation with the minimum distance from all other conformations in the cluster) of the best clustering will be generated. It will also record which conformations were assigned to each of the clusters, so that users can know which conformation belongs to each cluster.
- Compression: In this case, the user will define a target number of conformations (lower than the size of the input ensemble). pyProCT will use the best clustering in order to eliminate the per-cluster redundancy so that the reduced input ensemble contains the chosen target number of conformations. The compression protocol first calculates the target number of elements of each cluster so that it is proportional to the initial population. Then, it applies a k-medoids algorithm to each cluster to obtain a number of 'local clusters' equal to the target number of elements. The prototypes of each of the new 'local clusters' are stored as elements of the compressed set. This way the space is fairly partitioned and we ensure that all new elements are good representatives of the input data.

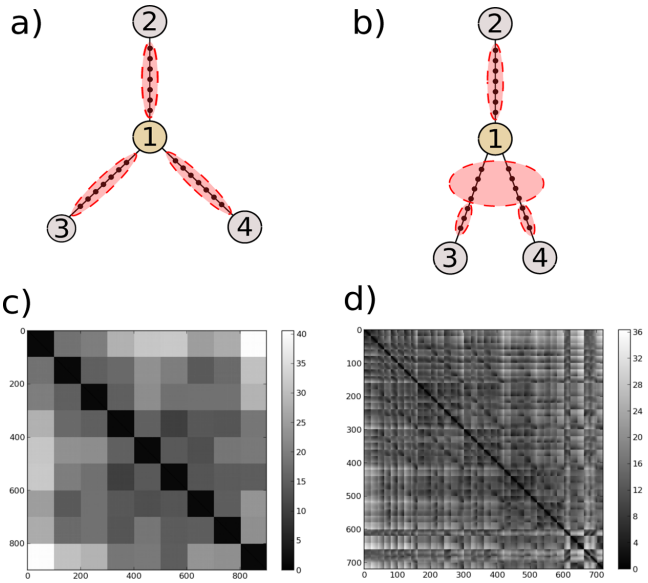


Figure 2. (a and b): New conformations (black dots) are created by interpolating from a base conformation (light-brown colored) to a target conformation (in gray), forming the interpolation sets (in red). All pairs of initial conformations without repetition are used. (a) If all target conformations are equally separated, interpolation clusters are easily separable and can be organized in multiple ways to form interpolation metaclusters. (b) The existence of distance asymmetries will difficult the partitioning into metaclusters. (c) RMSD matrices of the 900 conformations ensemble and (d) the 720 one. Clusters in matrix c are well-defined due to the small magnitude of the added noise. The small visible structures in d correspond to the interpolation clusters.

2.5. Code and Interface. pyProCT is a complete clustering toolkit written mainly in Python. It can be used as a standalone program or, thanks to its high modularity, as part of other software packages.

pyProCT offers different degrees of control according to the method chosen to run the clustering job. The easiest way to use it is through the included browser-based GUI, organized as a wizard. It includes visualization tools, allowing the user to preview the selections needed to superimpose and calculate distances. Finally, it can generate a complete graphical representation of the clustering results. Among other features, it allows visualizing the most representative structures of each of the generated clusters and includes tools to visually validate the results. It also allows plotting the ICV values of all accepted clusterings, being this a tool of utmost importance for users who want to better understand the behavior of applied ICVs and criteria in order to improve the working hypothesis. When used locally, the GUI will connect to an installed pyProCT instance to run the analyses and show the results. It also allows users to download the generated script, so that they can further modify it or use it in remote machines. The GUI has been written in Python (the Web server) and HTML with Javascript (the front end). A detailed list of all the Javascript libraries used can be found in the documentation.

pyProCT can also be used as a command line program that receives a JSON (JavaScript Object Notation) string as its input script. Inside the script the different actions and their parameters are defined; algorithms can be manually selected and their parameters specified at will. The use of this script represents the finest type of control that a user can have over pyProCT (without modifying the code).

Complete specifications of the control script are detailed in the Supporting Information section 3.

2.6. Performance. PyProCT uses the pyRMSD[29] package in order to efficiently calculate pairwise RMSD matrices. Furthermore, the use of the C-written condensed matrix module included in the same package has also helped us to considerably speed up the code thanks to its faster access times and its implementation of some simple neighboring queries. Cython (http://cython.org) generated code has been used to enhance the calculation speed of some quality functions. A Python+C implementation of the hierarchical algorithm has been also used.[30]

Finally, both the clustering search and the clustering evaluation steps have been parallelized by using a parallel task scheduler. One of the two available scheduling types uses Python's 'multiprocessing' module to allow to fully use the cores of a single machine. The second scheduling type is based on MPI (through the mpi4py interface[31]) and is better suited for distributed environments.

2.7. Testing and Validation. The whole framework has been tested using the unit testing technique, with a good testing coverage. The tests of some of the most consolidated parts have been changed to regression tests. A simple validation script of the HCE method over 2D data sets is also available into the code repository. A short discussion about the validation can be found in Supporting Information section 6.

3. APPLICATIONS AND RESULTS

3.1. RMSD-Based Cluster Analysis over Synthetic Conformational Ensembles. *3.1.1. Ensemble Generation.* From an initial Ubiquitin conformation, we have performed an

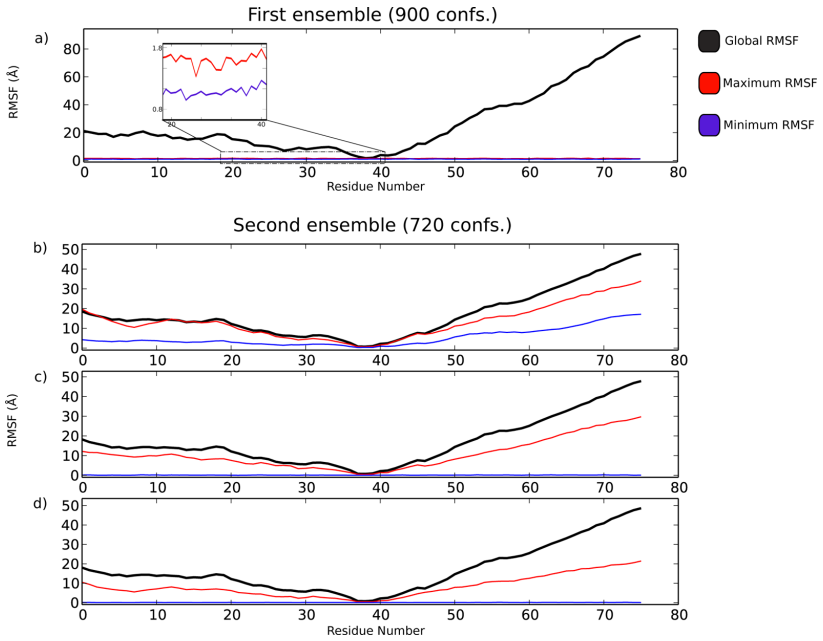


Figure 3. Global, maximum, and minimum RMSF plots. In all cases, the global RMSF plot high values (especially in the head and tail of the protein) are a result of the big global residue movements induced by the torsional changes. For the first ensemble (a), the resulting clustering is able to successfully separate the conformations, which is reflected by low values of $RMSF_{min}$ and $RMSF_{max}$, that only capture the added noise. For the second ensemble (b,c,d), $RMSF_{min}$ and $RMSF_{max}$ show a decrease in its values as the software is able to generate better clusterings in terms of the separation of residue movement.

iterative process in which a torsion angle is randomly chosen and modified at each step. We have produced a set of nine conformations (including the initial nonmodified one) that have great pairwise RMSD differences (17.75 Å mean, 8.85 Å std. dev.).

Using these nine initial conformations, we have generated two ensembles:

- A 900 conformations ensemble was produced by generating 100 random noise structures, using a [−1, 1] Å range for each coordinate, applied to each initial conformation. The resulting ensemble contains nine well-defined clusters with sharp boundaries that are easily identifiable in its RMSD matrix plot (Figure 2c).
- A 720 conformations ensemble was created by interpolating 20 new structures for each of the possible 36 pairs (without repetition) of the initial set, without including any of the original conformations. Each intermediate conformation is calculated using the formula $b_i = b + ((0.2 + (0.6/20)i)(t - b))$, where b and t are the base and target conformations, respectively. The resulting ensemble contains up to 36 small clusters of 20 conformations each: the interpolation sets (Figure 2a and b). The resulting RMSD matrix is shown in Figure 2d.

3.1.2. First Ensemble Cluster Analysis. For the cluster analysis on the first ensemble, we pretend to have superficial knowledge of its generation process so that the exact data layout is unknown, although we know that some well-defined clusters may exist. The first step was to choose which algorithms to use as well as their parameters. In this case, we will use the default options, that is, pyProCT will try all the algorithms and calculate different parameter sets for each of them. The second step is to define the clustering hypothesis. Given the limited information we have about the ensemble, it has to be loose enough to allow pyProCT to effectively explore the clustering space. The clustering hypothesis we have chosen instructs pyProCT to limit its search to clusterings with at least 3 clusters and at most 50 with a minimum size of 20 conformations. The default evaluation criteria has been used, which means that clustering score is calculated summing the 40% of its cohesion index value plus the 60% of its silhouette index value (that gives separation and compactness information at the same time), stressing in this way the contribution of compactness to the score. In terms of the clustering hypothesis, this means that we prefer clusterings where clusters are compact over clusterings where clusters are very separated.

The definition of a clustering size range in the hypothesis prevents the user from having to specify a concrete value for the desired number of clusters, a mandatory parameter in some algorithms, helping to avoid the creation of artificial partitions. Indeed, after applying pyProCT, the resulting clustering contains nine equally sized clusters, which is the expected clustering size. To check whether every cluster contains the expected elements, we can inspect the results file (The file format is described in Supporting Information section 4), where all generated clusterings have been saved in human-readable form. Another option is to use the results viewer included in the GUI: in some cases, especially when a geometrical similarity metric is used and a good balance of

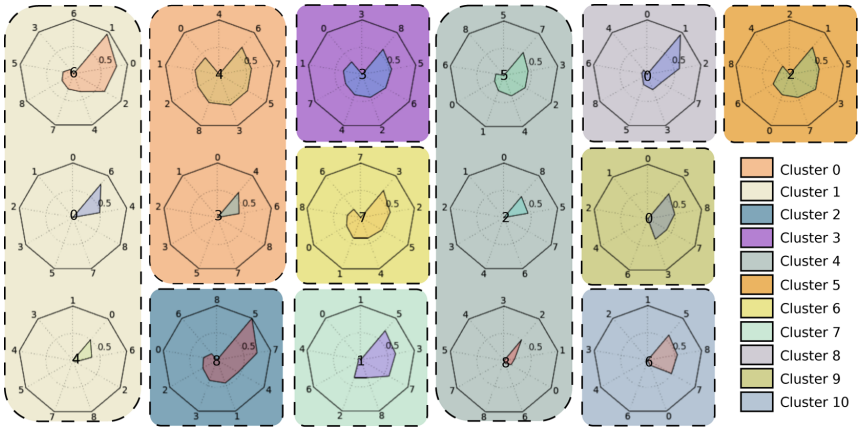


Figure 4. Each shadowed rectangle shows the contents of each of the 11 clusters for the first clustering. Each radial plot shows the percentages of all interpolation sets generated from the base conformation (which id is written in its center) to each of the other target conformations. All clusters contain elements from more than one interpolation set. In particular, clusters 1, 2, and 8 contain an almost complete interpolation set (from conformation 1 to 6, from 5 to 8, and from 0 to 1, respectively).

compactness/separation is needed, cluster contents can be indirectly checked by inspecting the global and per-cluster root-mean-square fluctuation (RMSF) plots. In order to calculate the RMSF of a set of m conformations of n_{res} residues, all conformations are iteratively superimposed. Then RMSF for each residue is calculated following eq 1, where r_{ref} is the mean conformation, $i \in [1,n_{res}]$ is the number of residue and j is the number of conformation. The residue position is represented by its α carbon position.

$$\mathrm{RMSF}^i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(r_j^i - r_{\mathrm{ref}}^i)^2} \quad (1)$$

When applied to the ensemble, the RMSF plot gives us an intuition of the global per-residue mobility. If applied to the conformations inside a cluster it shows how much of this global movement was captured by it. In this work, we will also use RMSF_{max} and RMSF_{min}, two derived measures that account for the maximum and minimum RMSF values of all clusters (see eqs 2 and 3). These two measures have been used to compare the collective per-cluster RMSF with the global one. If a clustering does not succeed to correctly capture structural variance, the values of local RMSF will be closer to the global RMSF and so will be the value of RMSF_{max}. Lower values of local RMSF will be reflected as lower values of RMSF_{min}. Both RMSF_{max} and RMSF_{min} give us a fast way of comparing per-cluster and global RMSF, even in situations when the clustering size (the number of clusters) is big. In Figure 3a, we can see how the global RMSF plot captures wide displacements in α carbon positions, induced by the torsional changes in the initial conformation. However, in this easy case, RMSF_{max} and RMSF_{min} plots show fluctuations between 0.9 and 1.1 Å, meaning that clusters are just capturing the noise added to each of the conformations of the initial ensemble, and indirectly confirming that cluster contents are correct (elements were separated correctly).

$$\mathrm{RMSF}_{\mathrm{min}}^i = \min(\mathrm{RMSF}_1^i, \mathrm{RMSF}_2^i, \ldots, \mathrm{RMSF}_{n_{\mathrm{clusters}}}^i) \quad (2)$$

$$\mathrm{RMSF}_{\mathrm{max}}^i = \max(\mathrm{RMSF}_1^i, \mathrm{RMSF}_2^i, \ldots, \mathrm{RMSF}_{n_{\mathrm{clusters}}}^i) \quad (3)$$

3.1.3. Second Ensemble Cluster Analysis. Again, we want to recreate a scenario in which the user does not have full information about how the data set was generated. For our first cluster analysis attempt, we will use pyProCT's default hypothesis so that we can learn more about the data set.

The resulting clustering contains 11 clusters. The fact that their sizes are all bigger than a single interpolation set, indicates that we are obtaining interpolation metaclusters (clusters containing elements from more than one interpolation set). The percentage of each interpolation set in each cluster (Figure 4) shows that only 3 out of the 11 clusters contain a complete interpolation set; the 8 remaining clusters contain partial interpolation sets that, in most cases, share the same base conformation. These can be explained by the distance asymmetries present in the original 9 conformations ensemble (see Figure 2b). The RMSF plot in Figure 3b shows large RMSF_{max} and RMSF_{min} values, meaning that clusters are capturing wide residue fluctuations. These values warn us about the quality of the clustering, pointing us to reevaluate the validity of the default hypothesis. In this second attempt, we will loosen the hypothesis by allowing more and smaller clusters. The number of clusters range has been maintained, but the minimum cluster size has been lowered to 15. Furthermore, we increase the noise to 5% (unclustered elements), giving pyProCT more freedom to explore. Finally, we change the default evaluation criteria so that the final score is calculated using a 66% of silhouette index value and a 33% of cohesion value. This gives a slightly higher weight to the separation/compactness ratio than the default criteria. After performing the exploration, we have found a clustering composed of 35 clusters with 20 conformations each, and one with 15. If we look for them in the results file, we can see that each of these clusters corresponds to one of the interpolation sets (the missing five elements being interpreted as noise). The RMSF plot in Figure 3c reflects a remarkable decrease of the RMSF_{min} value, meaning that one or more clusters were able to capture the local fluctuations. The RMSF_{max} value, however, only decreases

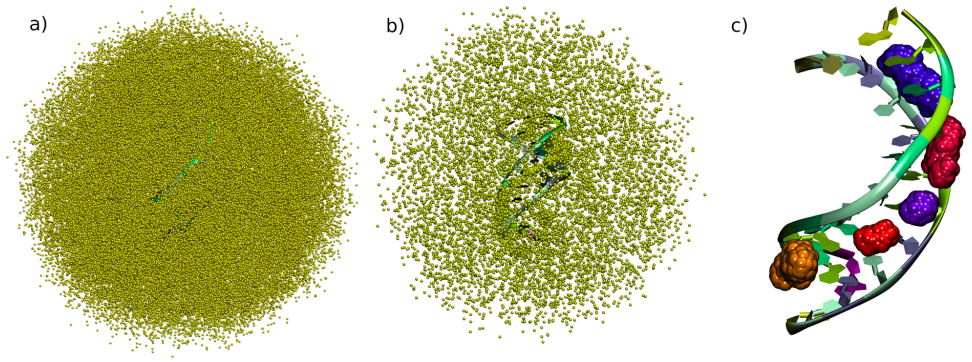


Figure 5. Atomic level representations of the MD trajectory data set. In part a, the whole data set is shown. It is worth noting that the cluster formed by the bulk solvent covers the clusters of interest (around the binding sites). This data set's redundancy was reduced, being part b, the resulting compressed data set. Finally, in part c, the five retrieved clusters are shown along with the DNA strand.

to some extent, suggesting that many clusters are still capturing wide fluctuations: the clustering may still be improved.

The last attempt of improving our cluster is to further loosen the working hypothesis. This time, we will let pyProCT produce even more clusters (in the range 3 to 100). Larger number of clusters involves lowering the minimum cluster size, set here to only 10 elements. Moreover, generating a larger number of clusters (of smaller size) makes the clustering more prone to producing noise. To address this issue, we raised the allowed clustering noise to 10% of the elements.

The resulting clustering contains up to 49 clusters with an average size of 14 elements. With a layout beyond our intuition, this last clustering is the one that best separates the ensemble conformations. As a consequence, its $RMSF_{max}$and $RMSF_{min}$values are the lowest of the three clusterings we have generated (see Figure 3d).

This illustrates how pyProCT can find good quality clusterings if enough freedom is given to it. What is more, the iterative scheme shown here (Figure 4) helped us to gain more insight about the data and its layout by creating three hypotheses adapted to our increasing knowledge of the data set.

3.2. Clustering of an MD Trajectory. In our last application example, we want to cluster analyze the cisplatin−DNA interaction data set used by Lucas et al.[32] This data set holds a 3.7 s molecular dynamics MD simulation stored in 126 857 frames (Figure 5a). It shows the electrostatic preassociation of cisplatin, a clinically relevant drug used to treat several types of cancer, with a DNA strand.

When performing a cluster analysis on this data set the first obstacle we find is its huge size. To handle its pairwise RMSD matrix, at least 29 GB of RAM would be needed. This makes the analysis computationally unfeasible in common workstations and nonspecialized hardware architectures. To overcome this issue, we have first divided the trajectory into 12 chunks of 10 572 frames (the 12th part is 7 frames smaller). Then, we have used the compression feature of pyRMSD to reduce the size of each piece below 1500 frames. Finally, all compressed partial trajectories have been merged. Since the addition of local compressions could have added redundancy, we have compressed the resulting trajectory again in order to have about 15 000 frames (Figure 5b).

A quick visualization of the data set reveals a second difficulty: relevant clusters are hidden by a large convex cluster that corresponds to random posotions of the drug away in bulk solvent space (Figure 5a). This bulk cluster has different density, size, and shape than the clusters we are interested in, thus complicating the analysis. This data set represents a good example of how a clustering analysis that is trivial for the human brain can be difficult to automatize.

The clustering hypothesis we have used reflects what we have learned from this visual inspection. We have allowed a huge noise generation (maximum noise of 80%) in order to permit the classification of the bulk cluster as noise. In addition, we have decided to let pyProCT choose clusterings containing from 3 to 20 clusters, with at least 50 elements inside each cluster. All other parameters were left with their default values.

The resulting clustering (Figure 5c) contains five clusters in which positions and relative populations correlate well with the ones shown in the kinetic analysis performed by Lucas et al.[32] To obtain this final clustering, pyProCT generated 314 clusterings, from which 164 were directly rejected. The whole exploration process took 90 min using five working threads plus a nonproductive control thread in a 4-cores (8 threads) Intel Xeon W3530 @ 2.80 GHz workstation with 12 GB of RAM. Thanks to the use of pyRMSD,[29] the matrix calculation step, one of the bottlenecks of cluster analysis software, took only 19 s (0.35% of the total time).

4. CONCLUSIONS

pyProCT is an open source Python cluster analysis toolkit that can work as a standalone program or as part of other projects. Its implementation of the HCE method can help users with little or no previous experience in cluster analysis to produce more reliable results. In addition, the high level input customization makes it a powerful tool when used by experts.

We have shown how it can be applied to common use cases of cluster analysis: feature extraction and redundancy elimination. In the first conformational clustering case presented, pyProCT's default hypothesis was enough to guess the algorithm and parametrization (including the number of clusters) that produced the best clustering. In the second proposed example, we showed an intuitive iterative scheme that allows us to refine the default hypothesis in order to improve the results and to gain insight into our data set structure. Finally, we have shown how to use it in a spatial cluster analysis

scenario, where a large and noisy data set was compressed to an easy to handle ensemble.

pyProCT is available at https://pele.bsc.es/pele.wt/tools.

■ ASSOCIATED CONTENT

Supporting Information

Implemented clustering algorithms; clustering properties and quality functions; input and output file formats. This material is available free of charge via the Internet at http://pubs.acs.org.

■ AUTHOR INFORMATION

Corresponding Author

*Email: victor.guallar@bsc.es.

Notes

The authors declare no competing financial interest.

■ REFERENCES

(1) Shaw, D. E.; Chao, J. C.; Eastwood, M. P.; Gagliardo, J.; Grossman, J. P.; Ho, C. R.; Lerardi, D. J.; Kolossváry, I.; Klepeis, J. L.; Layman, T.; McLeavey, C.; Deneroff, M. M.; Moraes, M. A.; Mueller, R.; Priest, E. C.; Shan, Y.; Spengler, J.; Theobald, M.; Towles, B.; Wang, S. C.; Dror, R. O.; Kuskin, J. S.; Larson, R. H.; Salmon, J. K.; Young, C.; Batson, B.; Bowers, K. J. *Commun. ACM* **2008**, *51*, 91.

(2) Stone, J. E.; Hardy, D. J.; Ufimtsev, I. S.; Schulten, K. *J. Mol. Graphics Modell.* **2010**, *29*, 116−125.

(3) Zhang, Y.; Skolnik, J. *J. Comput. Chem.* **2004**, *25*, 865−871.

(4) Daura, X.; Gademann, K.; Jaun, B.; Seebach, D.; van Gunsteren, W. F.; Mark, A. E. *Angew. Chem., Int. Ed. Engl.* **1999**, *38*, 236−240.

(5) Prinz, J.-H.; Wu, H.; Sarich, M.; Keller, B.; Senne, M.; Held, M.; Chodera, J. D.; SchÃOEtte, C.; NoÃ©, F. *J. Chem. Phys.* **2011**, *134*, 174105.

(6) Noe, F.; Schutte, C. *Proc. Natl. Acad. Sci. U.S.A.* **2009**, *106*, 19011−6.

(7) Takahashi, R.; Gil, V. A.; Guallar, V. *J. Chem. Theory Comput.* **2013**, *10*, 282−288.

(8) Karpen, M. E.; Tobias, D. J.; Brooks, C. L. *Biochemistry* **1993**, *32*, 412−420.

(9) Gordon, H. L.; Somorjai, R. L. *Proteins* **1992**, *14*, 249−264.

(10) Haack, F.; Fackeldey, K.; Röblitz, S.; Scharkoi, O.; Weber, M.; Schmidt, B. *J. Chem. Phys.* **2013**, *139*, 194110.

(11) Shao, J.; Tanner, S. W.; Thompson, N.; Cheatham, T. E. *J. Chem. Theory Comput.* **2007**, *3*, 2312−2334.

(12) Berendsen, H. J. C.; Spoel, D. V. D.; Drunen, R. V. *Comput. Phys. Commun.* **1995**, *91*, 43−56.

(13) Seeber, M.; Cecchini, M.; Rao, F.; Settanni, G.; Caflisch, A. *Bioinformatics* **2007**, *23*, 2625−2627.

(14) Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. *Kdd* **1996**, 226−231.

(15) Zhou, H.; Wang, P.; Li, H. *J. Inf. Comput. Sci.* **2012**, *9* (7), 1967−1973.

(16) Cossio, P.; Laio, A.; Pietrucci, F. *Phys. Chem. Chem. Phys.* **2011**, *13*, 10421−10425.

(17) McGibbon, R. T.; Pande, V. S. *J. Chem. Theory Comput.* **2013**, *9*, 2900−2906.

(18) Meila, M. Comparing Clusterings: An Axiomatic View. *Proceedings of the 22nd International Conference on Machine Learning* **2005**, 577−584.

(19) Luxburg, U. *Stat. Comput.* **2007**, *17*, 395−416.

(20) Kleinberg, J. *Adv. Neural Inf. Process. Syst.* **2002**, 446−453.

(21) Ward, J. H. *J. Am. Stat. Assoc.* **1963**, *58*, 236.

(22) Guyon, I.; Luxburg, U. V.; Williamson, R. C. *Adv. Neural Inf. Process. Syst.* **2009**.

(23) Rand, W. M. *J. Am. Stat. Assoc.* **1971**, *66*, 846.

(24) Reichart, R.; Rappoport, A. The NVI Clustering Evaluation Measure. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*; Stroudsburg, PA, 2009; pp 165−173.

(25) Meila, M. Comparing Clusterings by the Variation of Information. In *Learning Theory and Kernel Machines*; Scholkopf, B., Warmuth, M. K., Eds.; Lecture Notes in Computer Science 2777; Springer: Berlin Heidelberg, 2003; pp 173−187.

(26) Kryszczuk, K.; Hurley, P. Estimation of the Number of Clusters Using Multiple Clustering Validity Indices. In *Multiple Classifier Systems*; Gayar, N. E., Kittler, J., Roli, F., Eds.; Lecture Notes in Computer Science 5997; Springer: Berlin Heidelberg, 2010; pp 114−123.

(27) Ng, R. T.; Han, J. Efficient and Effective Clustering Methods for Spatial Data Mining. *Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, 1994; pp 144−155.

(28) Pal, N.; Biswas, J. *Pattern Recogn.* **1997**, *30*, 847−857.

(29) Gil, V. A.; Guallar, V. *Bioinformatics* **2013**, *29*, 2363−2364.

(30) Mullner, D. J. *Stat. Soft.* **2013**, *53*, 1−18.

(31) Dalcin, L.; Paz, R.; Storti, M.; D'Elia, J. *J. Parallel Distrib. Comput.* **2008**, *68*, 655−662.

(32) Lucas, M. F.; Cabeza de Vaca, I.; Takahashi, R.; Rubio-Martinez, J.; Guallar, V. *Biophys. J.* **2014**, *106*, 421−429.

3.7 Supplementary materials for: pyProCT: Automated Cluster Analysis for Structural Bioinformatics

3.7.1 Implemented clustering algorithms

In this section, we will review the algorithms currently implemented in pyProCT. For each of these, we will briefly describe how it works, how its parameters are generated (if automatic generation is triggered) and the structure of each parameter objects, so that users can define them inside the script.

3.7.1.1 DBSCAN [160]

DBSCAN[4] is a density-based clustering algorithm. Density is given by the use of 2 parameters: 'eps', that defines a distance radius centered on one element, and 'minpts', that specifies the minimum number of neighboring points for that element to be considered part of a cluster.

The algorithm classifies elements into three categories: not classified, noise and core elements. All elements are initially set to 'not classified'. The 'core' elements are those that have at least 'minpts' elements in an 'eps' radius and also the elements inside the 'eps' radius of a 'core' element (which will be border points). All not 'core' elements are classified as 'noise' and will not be part of the clustering.

One of the key issues of DBSCAN is the proper selection of its 'eps' and 'minpts' parameters. High density combinations can produce very noisy clusterings (which, depending on the context can be an issue or a feature), and low density combinations can easily lead to the singleton clustering (a clustering with only one cluster encompassing all elements). Because of its dependency on the element density, it can have problems detecting clusters if there are different density regions in the dataset.

3.7.1.1.1 Parameters generation strategy

The implemented parameters generator is based on the details given in the original articles [160, 161] as well as the method of Ankerst extitet al. [162]. Briefly:

[4] Density-Based Spatial Clustering of Applications with Noise

1. pyProCT will recreate k-distance lists for some k values ($k < \log(|D|)$ as indicated in the literature).

2. For each of the k-distance lists, an 'eps' value will be chosen so that the generated noise falls between 0% and the maximum allowed noise (plus a 2.5% margin). The value of k in that k-distance list will be used as the value of 'minpts' in that parameter pair.

In addition, we add the parameter choice suggested by Zhou extitet al.[163].

3.7.1.1.2 Parameters object structure

```
{
        "eps": Real,
        "minpts": Integer
}
```

3.7.1.2 GROMOS

First found in the work of Daura extitet al. [164, 165], it is also a density-based algorithm. As it looks that the algorithm has no name, we named it after the flag that the g_cluster tool[157] receives to use it. With a simpler behaviour than DBSCAN, it was thought to adapt to the ideal expected characteristics of datasets coming from the conformational search of small peptides (well-separated regions centered in metastable states).

The algorithm repeats a loop that ends when all elements have been clustered:

Algorithm 1 GROMOS algorithm

Input: D, the data set
Input: $cutoff$, the cutoff radius
Output: C, a set of clusters

$D_{tmp} \leftarrow D$
while $D_{tmp} \neq \emptyset$ **do**
    $e \leftarrow$ mostDense(D_{tmp})
    $neig \leftarrow$ neighbours(e,$cutoff$)
    $c \leftarrow \{e, neig\}$
    addCluster(c, C)
    $D_{tmp} \leftarrow D_{tmp} \setminus c$
end while
return C

mostDense() Find the element with more neighbors inside the given cutoff radius (to find the densest zone).

3.7.1.2.1 Parameters generation strategy

The goal here is to find a finite range of cutoffs, exactly as it is done with the 'k' parameter in k-medoids or 'num_clusters' in the random grouping algorithm. The minimum value for the cutoff parameter is 0, generating a trivial clustering in this case (a clustering where each element is a cluster). In order to get an estimation of the maximum value, we choose the 3 most separated elements of the dataset. We can ensure that the second longest side of the triangle with this 3 elements as vertices will be the minimum radius that encompasses all elements of the dataset, producing a singleton clustering, and thus it is the upper limit of the cutoff range. Once the maximum and minimum are known, we can obtain equidistant values for the cutoff within this range.

3.7.1.2.2 Parameters object structure

```
{
        "cutoff": Real
}
```

3.7.1.3 Hierarchical clustering

pyProCT uses an external package [166] that implements an agglomerative hierarchical clustering algorithm. Agglomerative hierarchical algorithms start with all elements of the dataset forming single clusters (a trivial clustering) and every step it merges the closest clusters by means of a proximity function (only 'single linkage' and 'complete linkage' options can be used). The agglomerative step represents the dataset as a tree (dendrogram) that has to be cut in order to retrieve the clustering. The distance at which the cut is performed is called the cutoff distance.

3.7.1.3.1 Parameters generation strategy

Getting the dendogram cut is computationally cheap compared to the hierarchical matrix calculation. In this case, pyProCT will generate clusterings until it finds the range of cutoffs in which the resulting clustering falls within the range of allowed number of clusters. This range will be refined in order to get more clustering candidates.

3.7.1.3.2 Parameters object structure

```
{
        "cutoff": Real,
        "method": String in ["single",
                             "complete"]
}
```

method If its value is 'single', the single linkage method is used to calculate the proximity of clusters. This proximity is then defined as the minimum distance between any two points in that clusters. If 'complete' is used instead, the proximity of two clusters is measured as the maximum distance between any two elements of different clusters. Various combinations of method-cutoff are possible in the parameters list, but due to performance reasons, only the 'method' value of the first parameter object in the list will be used.

3.7.1.4 K-Medoids

Is a partitional algorithm similar in concept to k-means. Because of its beautiful simplicity, our implementation is based on Lloyd's k-means algorithm [167] instead that on other k-medoids specific algorithms (like PAM [168]).Roughly the algorithm works as follows:

Algorithm 2 K-Medoids algorithm

Input: k, number of clusters
Output: C, a set of clusters

```
1: medoids ← initialSeeding(k)
2: while ¬ convergence() ∧¬ maxStepsReached() do
3:     C ← labelElements()
4:     medoids ← calculateMedoids(k)
5: end while
6: return C
```

convergence() Checks if how the labeling of current iteration has changed with respect to the last iteration.

labelElements() Labels each element of the dataset with the cluster of its closer medoid.

K-Means-like algorithms will perform better detecting convex clusters.

3.7.1.4.1 Parameters generation strategy

A globally-defined or used-defined maximum number of parameter sets ('max') will be generated. Each of the parameter sets will have its 'method' field set to 'EQUIDISTANT' and its 'k' field will is calculated as $(min\_clusters + step \times n$ where $0 <= n <= max$ and $step$ is $^{(max\_clusters - min\_clusters)}/_{max}$.

3.7.1.4.2 Parameters object structure

```
{
        "k": Integer,
        "seeding_type": String in ["RANDOM",
                                   "EQUIDISTANT",
                                        "GROMOS"],
        "seeding_max_cutoff": Real
}
```

k Number of clusters to be created.

seeding_type Method used for initial medoid seeding.

seeding_max_cutoff Only used when 'seeding_type' = GROMOS. Maximum cutoff radius for the GROMOS algorithm to generate the initial medoids.

The algorithm is very sensitive to the initial seeding configuration (the initial placement of the medoids). Three seeding methods are provided:

RANDOM Uses a random choice of elements from the dataset as initial medoids. If used, the parameter will be replicated 'tries' times (default: 10) with different random seeds.

EQUIDISTANT Divides the dataset in k consecutive parts and uses their central element as medoid. Useful if we suspect that sequence order and geometrical likeness are correlated (like in MD sequences).

GROMOS It will execute GROMOS algorithm with decreasing cutoffs until 'k' or more clusters are generated. Initial medoids will be the centers of this clusters. It is specially useful if clusters were well separated.

3.7.1.5 Spectral Clustering

This algorithm uses the spectral properties of the dataset (viewed as a graph). It is said that it can achieve better results than k-means or hierarchical clustering algorithms.

The implemented version of the algorithm is described in the detailed review written by Ulrike von Luxburg [169], based on the Normalized Spectral Clustering[170]). It needs to perform six steps:

Algorithm 3 Spectral clustering algorithm

Input: M, a distance matrix
Input: D, initial data set
Input: k, number of clusters
Output: C, a set of clusters
1: $A \leftarrow$ adjacencyMatrix(M)
2: $Deg \leftarrow$ degree(A)
3: $L \leftarrow$ laplacian(A, Deg)
4: $eigvec \leftarrow$ caclEigenvectors(L,k)
5: $C_e \leftarrow$ kMedoids($eigvec$, k)
6: $C \leftarrow$ map(C_e, D)
7: **return** C

adjacencyMatrix() Constructs a similarity graph by applying a kernel to the distance matrix.

degree() Computes the degree matrix and adjacency matrix.

laplacian() Computes the (unnormalized) Laplacian.

caclEigenvectors() Computes the first k generalized eigenvectors.

kMedoids() Clusters the eigenvector matrix as if each row was a point.

map() Maps each of the eigenvectors to the elements of the dataset (row i corresponds to element i).

3.7.1.5.1 Parameters generation strategy

The sigma global parameter, used to calculate the adjacency matrix, can be set by the user. If not, pyProCT will use a local sigma calculation strategy [171] to build it. 'k' parameter is generated using the same approach than in the k-medoids case.

3.7.1.5.2 Parameters object structure

```
{
        "k": Integer,
        "use_k_medoids": Boolean
}
```

use_k_medoids
: If set to true, the k-medoids algorithm will be used to cluster eigenvalues. If set to false, k-means will be used instead.

3.7.1.6 Random Grouping

This algorithm assigns random cluster labels to the different elements in the dataset. It cannot be considered a real clustering algorithm, but it is useful to compare the behaviour of certain ICVs with more sphisticated algorithms. It has been implemented in two ways:

FakeDistributionRandomClusteringAlgorithm
: Generates clusters with certain per cluster population distribution (e.g. First cluster 70% of the elements, second 20%, third 5% and so on)

RandomClusteringAlgorithm
: Generates a totally random clustering (random number of clusters and random cluster assignment) or a random clustering with a predefined number of clusters.

3.7.1.7 Parameters generation strategy

The same strategy used in k-means to calculate the k value will be used here for num_clusters.

3.7.1.8 Parameters object structure

Only RandomClusteringAlgorithm is accessible through the script, and it only needs one parameter:

```
{
        "num_clusters": List(Integer)
}
```

num_clusters
Number of clusters to be created (analogue to 'k' in other algorithms).

3.7.2 Clustering properties and quality functions

pyProCT implements functions which goal is to retrieve information from the generated clusterings to evaluate them. Some of these functions retrieve properties from the clustering ("properties"), other functions evaluate the quality and form the objective part of the clustering hypothesis. In this section we will define and formalize both of them.

3.7.2.1 General definitions

1. D is a set containing all the elements of the dataset with $|D|$ number of elements (number of datum in the dataset e.g. conformations, distances, etc).

2. $C = \{C_1, C_2, \dots C_k\}$ is a clustering of k clusters, where $|C|$ is its number of clusters ($|C| = k$ in this case) and $|C_i|$ is the number of elements of cluster i. Also:

$$\forall C_i, C_j \in C, \quad i \neq j \quad C_i \cap C_j \equiv \emptyset \tag{3.7}$$

3. $D_C \subseteq D$ is the set of clustered elements, which can differ from the initial set if some elements were considered as noise and thus discarded.

$$D_C \equiv \bigcup_{i=1}^{k} C_i, \tag{3.8}$$

4. $d(a, b)$ is a distance metric (dissimilarity function) applied to an element $a \in D$ and an element $b \in D$.

5. m_i is the medoid of cluster C_i so that:

$$m_i \in C_i \tag{3.9}$$

$$\forall a, b \neq m_i \quad a, b \in C_i \quad d(a, b) \geq d(a, m) \wedge d(a, b) \geq d(b, m) \tag{3.10}$$

6. As stated before singleton clustering is a clustering with one cluster that holds all the elements of the dataset. In a trivial clustering, each element

of the dataset forms its own clustering.

$$|C|_{singleton} = 1 \tag{3.11}$$

$$|C|_{trivial} = |D_C| \tag{3.12}$$

3.7.2.2 Properties

The properties are functions that allow users to query about simple traits and statistics of the clustering. The information returned by this functions is purely descriptive and, in general, not usable for evaluation purposes. Eight properties have been defined:

Details (String) Returns a string containing information about the type of clustering algorithm and the parameters used to generate it.

NumClusters (Integer) Returns the number of clusters ($|C|$).

MeanClusterSize (Real) Mean number of elements per cluster:

$$mcs(C) = \frac{\sum_{i=1}^{k} |C_i|}{|C|} \tag{3.13}$$

NumClusteredElems (Integer) Returns the number of elements that were clustered. It can be lower than the initial number of elements if noise was eliminated. Is defined as:

$$nce(C) = \sum_{i=0}^{|C|} |C_i| \equiv |D_C| \tag{3.14}$$

NoiseLevel (Real) Calculates the ratio of clustered elements over the number of initial elements:

$$nl(C, D) = \frac{\sum_{i=1}^{k} |C_i|}{|D|} \equiv \frac{|D_C|}{|D|} \tag{3.15}$$

ClustersTo90 (Real) Returns the minimum number of clusters needed to accumulate 90% of the clustered elements.

PercentInTop (Real) Calculates the percentage of clustered elements owned by the biggest cluster.

PercentInTop4 (Real) Calculates the percentage of clustered elements owned by the four larger clusters.

3.7.2.3 Quality functions

Sometimes referred to as Clustering Validity Indices (CVI), are functions which aim is to tell at which degree clusterings are an artificial partition or reflect the real inner structure of the dataset (assuming that it exists). Quality functions must use only internal information of the clustering, that is, not to be based on a solution that is considered correct.

The following are some initial definitions that will help us to characterize them:

Within cluster distance
The sum of all distances inside a cluster.

$$wd(C_i) = \sum_{a,b \in C_i} d(a,b) \tag{3.16}$$

Between cluster distance
Sum of the pairwise distances between elements belonging to two different clusters.

$$bd(C_i, C_j) = \sum_{a \in C_i} \sum_{b \in C_j} d(a,b) \tag{3.17}$$

Average distance
Average of all pairwise distances between the elements inside a cluster.

$$avg(C_i) = \frac{wd(C_i)}{|C_i|} \tag{3.18}$$

Standard deviation of distance
The standard deviation of all pairwise distances of the elements inside a cluster.

$$stdev(C_i) = \sqrt{\frac{1}{|C_i|} \sum_{a \in C_i} d^2(a, m_i)} \tag{3.19}$$

3.7.2.3.1 Cohesion [172]

Is a measure of cluster compactness. The cohesion factor measures the inner similarity of a cluster. Its value for one cluster is calculated by summing up the

distance of all elements belonging to that cluster. Its value for a clustering can be defined as the sum of its clusters partial cohesions weighted by the inverse of the cluster size. Due to the use of dissimilarity metrics, the interpretation of cohesion can be misleading: the smaller its value, the more compact the clusters are.

$$Ch(C) = \sum_{C_i \in C} \frac{1}{C_i} wd(C_i) \tag{3.20}$$

A cohesion value of 0 can be obtained with the singleton clustering. It reaches its maximum value in a trivial clustering ($|D_C|^{-1} wd(D_C)$).

Te actual implementation in pyProCT is a more intuitive redefinition of Cohesion, as an increment of cluster compactness will also increase this index value:

$$Ch(C) = 1 - \frac{Ch(C)}{|D_C|^{-1} wd(D_C)} \tag{3.21}$$

Its value ranges between 0 and 1.

3.7.2.3.2 Separation [172]

Measures how distinct a cluster is from other clusters (how isolated a cluster is from the others). In practice, it calculates the sum of distances weighted by its cohesion:

$$Sep(C) = \sum_{\substack{i=1 \\ j>i}}^{k} \frac{bd(C_i, C_j)}{Ch(C_i)} \tag{3.22}$$

When its value increases, cluster separation increases. Its value ranges from 0 (trivial clustering) to infinity (singleton clustering, which implies $Ch(C_i) \equiv 0$.

3.7.2.3.3 Compactness [173]

Compares the standard deviation (std. dev.) of the clustered dataset (std. dev. of its clusters) with the std. dev. of the whole dataset. Note that the std. dev. calculation function is defined using the medoid of the cluster instead of the mean point.

$$Cmp(C) = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{stdev(C_i)}{stdev(D_C)} \tag{3.23}$$

Maximizing its value minimizes compactness.

3.7.2.3.4 Gaussian Separation [173]

Is a prototype-based separation index where distances are attenuated using an exponential. This intends to produce the same behaviour that some exponential kernels used to calculate the adjacency matrix in graph-like representations of the dataset: diminish long range distances and sharpen subgraphs contours.

$$Gsep = \frac{1}{|C|(|C|-1)} \sum_{\substack{i,j=1,\dots,|C| \\ j \neq i}} e^{\frac{-d^2(m_i,m_j)}{2\sigma^2}} \tag{3.24}$$

Maximizing its value maximizes separation.

3.7.2.3.5 Davies-Bouldin [174]

A prototype-based measure that compares compactness (represented by the average of intra-cluster distances) and separation (distance between the prototypes).

$$Db(C) = \frac{1}{|C|} \sum_{i=1}^{|C|} max^{j \in 1,\dots,|C| \atop j \neq i} \left(\frac{avg(C_i) + avg(C_j)}{d(m_i, m_j)} \right) \tag{3.25}$$

Measures compactness and separation. The smaller the value is, the better overall quality the clustering has.

3.7.2.3.6 Dunn [175, 176]

Ratio of the minimum inter-cluster distance and the maximum intra-cluster distance.

$$mind(C_i) = min_{r,t \in C_i} d(r,t) \tag{3.26}$$

$$maxd(C_i, C_j) = max_{\substack{r \in C_i \\ t \in C_j}} d(r,t) \tag{3.27}$$

$$Dunn(C) = \frac{min_{C_i \in C}(mind(C_i))}{max_{\substack{C_i, C_j \in C \\ i \neq j}}(maxd(C_i, C_j))} \tag{3.28}$$

Dunn index evaluates compactness and separation simultaneously. Quality clusterings should have high values for this function.

3.7.2.3.7 Calinski-Harabasz [177]

Another variation of the intra and inter-cluster distances ratio calculation.

$$A_k(C) = \frac{1}{|D_C| - |C|} \sum_{i=1}^{|C|} (|C_i| - 1)(avg(D_C) - avg(C_i)) \tag{3.29}$$

$$CH(C) = \frac{avg(D_C) + \frac{|D_C|-|C|}{|C|-1} A_k}{|D_C| - A_k(C)} \tag{3.30}$$

It also measures compactness and separation.The higher the value is, the better quality the clustering has.

3.7.2.3.8 Silhouette [178]

Useful when distances are on a ratio scale (for instance euclidean distance), allowing to measure compactness and separation altogether by calculating the pairwise difference of inter and intracluster distances. The Silhouette index for a single element of a cluster can be calculated as:

$$S(e) = \begin{cases} \frac{b(e)-a(e)}{max(a(e),b(e))} & \text{if } |C_i| > 1 \\ 0 & \text{if } |C_i| \leq 1 \end{cases} \tag{3.31}$$

Where $a(e)$ is the average inner dissimilarity of its cluster and $b(e)$ is the outer dissimilarity of this element with the other clusters, calculated as follows:

$$e \in C_e \tag{3.32}$$

$$a(e) = \sum_{\substack{t \in C_e \\ t \neq e}} d(e, t) \tag{3.33}$$

$$b(e) = \sum_{\substack{\forall t \in C_i \\ t \neq e \\ C_f \neq C_e}} d(e, t) \tag{3.34}$$

Cluster and clustering values for this index can be calculated as the cluster average and global average of their per-element values. Its value ranges from -1 (worst quality) to 1 (best quality).

3.7.2.3.9 PCAanalysis [179]

Calculates the axes of variance and gives an estimation of the amount of variance in each axis for each cluster. The final value of this index will be the average value of the variance of the major variance axis for each cluster. As with other compactness measures, it depends on the size of the clusters. The higher the value is, the less compact the clusters are. Measures compactness (by means of variance).

3.7.2.4 Graph cut indices

The distance relationships between elements of the clustering can be viewed as a similarity graph were each vertex is an element and edges are weighted by their distances. In general, small values for this functions mean good quality of the partition (almost all are a sum of adjacency weights). Clustering can then be seen as a partitioning problem where one objective graph cut function is optimized. First, we will need to define some helper functions. Note that, in this case, the distances are the edge values, and are related to the adjacency matrix:

3.7.2.4.1 Degree of a node

Sum of the weights of the edges that contain this node.

$$deg(a) = \sum_{\forall i, b \in C_i} d(a, b) \tag{3.35}$$

3.7.2.4.2 Internal volume

Is defined as the sum of all the weights of the edges of one partition, including those that connect with other partitions). As each edge must be counted only once and internal edges are counted twice, final must be multiplied by 0.5. Can be seen as the sum of degrees too.

$$vol(C_i) = \frac{1}{2} \sum_{n \in C_i} deg(n) \tag{3.36}$$

3.7.2.4.3 Cut

Sum of the weights of the edges that have to be removed in order to separate two components of the graph. Its value is 0 if both subgraphs are not connected.

$$cut(C_i) = \frac{1}{2} \sum_{i \in C_i, j \in \overline{C_i}} d(i, j) \tag{3.37}$$

3.7.2.4.4 Ncut (Normalized Cut) [170]

It is mainly a separation measure. Some authors divide it by the number of clusters.

$$NCut(C) = \frac{\sum_{i=1}^{k} \frac{cut(C)}{vol(C_i)}}{k} \tag{3.38}$$

3.7.2.4.5 MinMaxCut [180]

It is mainly a separation measure.

$$MinMaxCut(C) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(C_i)}{vol(C_i)} \tag{3.39}$$

3.7.2.4.6 RatioCut [181]

It is mainly a separation measure.

$$RatioCut(C) = \sum_{i=1}^{k} \frac{cut(C_i)}{|C_i|} \tag{3.40}$$

3.7.3 Input script

pyProCT uses as input a human-readable JSON text file. Its file extension can be arbitrarily chosen, however pyProCT will generally produce this kind of files with a '.json' extension.

The input file describes how the software interacts with the underlying hardware where it is being executed, how the clustering exploration will be performed, and finally, which kind of post process operations will be performed after the clustering has been obtained. This section will be structured in 4 subsections, one for each of the main subsections of the script ('global', 'data', 'clustering' and 'postprocess').

Unless otherwise specified, each property or object will be represented by a descriptor. This descriptor contains the label of the property or JSON object (subsections), information about its data type, and possible dependencies.

A label consists of a double colon separated list of the names of the objects that encompass the property or subsection, followed by its name. For instance, **x::y::z** corresponds to the JSON object:

JSON 3.1: Simple JSON object.

```
"x":{
        "y":{
                "z":{}
        }
}
```

The nature of z will be specified after the label. Possible tags are:

Subsection The item is another JSON object that can contain another subsections (JSON objects) and properties.

Integer The item is an integer value property.

Real The item is a real value property.

String The item is a string property.

List The item is an array of elements.

If the value of the property has to be chosen from a limited list of possible choices, this values will be enumerated in a "value in [choices]" clause. "Optional" will be added to the end of the descriptor if the property is optional.

Finally, if a property or subsection needs it, a dependency clause will be added in front of the label. This dependency clause contains the label it depends on and the value it needs to have, or, if it just depends on its previous definition, an "is defined" clause.

Examples:

[**x::y::z** is defined] **x::y::t**

Means that x::y::t value will be ignored unless x::y::z has a value.

[**x::y::z** == "m"] **x::y::t**

Means that x::y::t will not be used unless x::y::z is defined and has value "m".

The rest of this section has been intentionally hidden as it no longer describes pyProCT input script. For updated information please visit the repository at `https://github.com/victor-gil-sepulveda/pyProCT`.

3.7.4 Results file

The results file summarizes the whole clustering process and its results. It contains extra information that can be processed afterwards with the results viewer in order to gain more insight about the used quality functions and criteria.

3.7.4.1 Best clustering

This section contains only one property that holds the identification string of the best clustering.

JSON 3.2: Best clustering result ID

```
{
        "best_clustering": String
}
```

best_clustering Id of the best clustering.

3.7.4.2 Files

This section stores a list of file objects, containing the details of all generated files (included the results file itself):

JSON 3.3: The file details section of the results file

```
{
        "files": List(FileDetailsObject)
}
```

A FileDetails object has the following structure:

JSON 3.4: FileDetails object

```
{
        "path": String,
        "type": String in ['text','image'],
        "description": String
}
```

path Complete path of the file

type 'text' if the file is a human-readable txt file, or 'image' if it is a viewable image file.

description Contains a very brief description of the file contents.

3.7.4.3 Trajectories

A list containing objects with details of the trajectories used in this clustering:

JSON 3.5: The trajectory details section of results file

```
{
        "trajectories": List(
           ↪ TrajectoryDetailsObject)
}
```

Each TrajectoryDetails object has the following structure:

JSON 3.6: TrajectoryDetails object

```
{
        "source": String,
        "conformations": Integer,
        "atoms": Integer
}
```

source Complete path of this trajectory.

conformations Number of conformations ('model' sections) of the pdb.

atoms Number of atoms of each conformation.

3.7.4.4 Workspace

Is a copy of the same section in the parameters file.

3.7.4.5 Scores

Contains the score value of each criterion for all non-filtered clusterings.

JSON 3.7: Possible score section for a clustering process that used two criteria and ended with two candidates

```
"scores": {
        "criterion_0": {
                "clustering_0001": 0.9822,
                "clustering_0002": 0.9935
        },
        "criterion_1": {
                "clustering_0001": 0.9150,
                "clustering_0002": 0.9229
        }
}
```

3.7.4.6 Timing

Compiles the values of all timer objects in a list. These can be used to rapidly assess the performance of the execution.

A timer object has the following structure:

JSON 3.8: Timer object

```
{
        "name": String,
        "elapsed": Real
}
```

name Name of the step being checked.

elapsed The duration of the step in seconds.

3.7.4.7 Clustering information

The clustering information section is indeed composed of 2 similar sections: 'selected' stores all non-filtered clusterings, and 'not_selected' holds the ones that were filtered.

Both sections contain JSON objects with details of the generated clustering. A clustering object is indexed by its id, and contains the following subsections:

clustering::clusters List(ClusterObject)

A list of the clusters forming the clustering. See the input file section **clustering::generation::clusters** for a description of the format of a cluster object.

clustering::total_number_of_elements Integer

The amount of clustered elements.

clustering::number_of_clusters Integer

The length of the clusters list.

evaluation Evaluation

An evaluation object containing the values of all calculated queries and quality functions for this clustering.

type String in ['dbscan', 'gromos', 'hierarchical', 'kmedoids', 'spectral', 'random']

Indicates which algorithm generated this clustering.

parameters ParametersObject

Parameters used by the algorithm to generate this clustering (as detailed in the Algorithms section).

Clusterings under the '_selected' key have slightly different contents. They share all fields but the 'evaluation' one, which changes to 'reasons'.

reasons List(ReasonObject)

Holds a list of reasons why the clustering was not considered for evaluation. each reason object looks like this:

JSON 3.9: Reason object

```
{
    "reason": String in ["TOO_FEW_CLUSTERS",
                         "TOO_MUCH_CLUSTERS",
                            "TOO_MUCH_NOISE",
                "EQUAL_TO_OTHER_CLUSTERING"],
    "data": ReasonDataObject
}
```

'reason' The reason to exclude this clustering from the evaluation step. 'TOO_FEW_CLUSTE
and 'TOO_MUCH_CLUSTERS' mean that the number of clusters is not into the allowed range. 'TOO_MUCH_NOISE' means that the clustering had too much noise. 'EQUAL_TO_OTHER_CLUSTERING' means that an exact clustering has been already generated (with other algorithm or parameters).

'data' Gives details about the reason to eliminate this clustering. For the first 3 cases, it will store the maximum/minimum value of the range and the current value for that clustering.

JSON 3.10: This clustering was not used because it had fewer clusters 1 than the minimum number of clusters allowed 6

```
{
        "reason": "TOO_FEW_CLUSTERS",
        "data": {
                "current": 1,
                "minimum": 6
        }
}
```

Data objects for 'EQUAL_TO_OTHER_CLUSTERING' reasons will only store the id of the repeated clustering.

```
{
        "reason": "EQUAL_TO_OTHER_CLUSTERING",
        "data": {
                "id": "clustering_0003"
        }
}
```

3.7.5 Clustering of a long trajectory (proof of concept)

In a cluster analysis, the function distance must be applied many times to the different elements of the dataset. As pyProCT needs to generate numerous clusterings, it is unfeasible to use an 'on line' distance calculation approach. Instead, the symmetric pairwise distance matrix is calculated once and used throughout all the process, improving the overall performance. The size of the distance matrix grows quadratically in the size of the input (the number of conformations), and can rapidly consume all the RAM of a state-of-the-art workstation. Because of this, using pyProCT with large conformational ensembles supposes a technical challenge.

In Section 3.2 we have shown how this limitation can be overcome by performing a redundancy reduction on the input trajectory before analyzing the

dataset. Here we want to apply this technique to reduce a longer trajectory (more than 1million frames) to an easier to handle size.

3.7.5.1 The trajectory

We used the 206 μs trajectory of Trp-cage (PDB[5] id 2JOF) presented in an article from D. E. Shaw's group [182]. The details of the simulation can be found in the Supporting Online Materials for this article.

3.7.5.2 Redundancy elimination

The goal of this technique is to reduce the input trajectory by exchanging sets of similar conformations (clusters) by a choice of its most representative structures so that the final number of elements is proportional to the original size of the cluster.

The first step to apply the reduction is to produce a clustering. This makes us face the memory problem mentioned before. A workaround for this issue is to divide the trajectory into smaller parts so that each distance matrix can fit in memory, process each part separately and then merge them again. Unfortunately, arbitrarily partitioning the trajectory can separate elements that could form part of the same cluster (which is more prone to happen in the boundaries of each part). This can lead to an unevenness in the redundancy elimination process (see the 2D example in Fig. 3.14A). We have performed the reduction process iteratively to mitigate the problem. We start with 105 parts of almost 10k frames each. After reducing each part to 2k frames, we merged them into groups of 7 ($\sim$14k frames each). These groups were compressed to have around 1.3k frames each and merged again to form a $\sim$24k frames trajectory. This was finally reduced to 20k frames and then analyzed.

3.7.5.3 Results

3.7.5.3.1 Performance

pyProCT was run in an Intel Xeon CPU W3530 @ 2.80GHz workstation. Each run of pyProCT spawned a maximum of 6 processes. While it was being executed the workstation was normally used, occasionally triggering operating system's swap mechanisms, which slowed down the process. Therefore, the calculated execution time has merely a qualitative meaning (see Table 3.3).

[5]Protein Data Bank (http://www.rcsb.org)

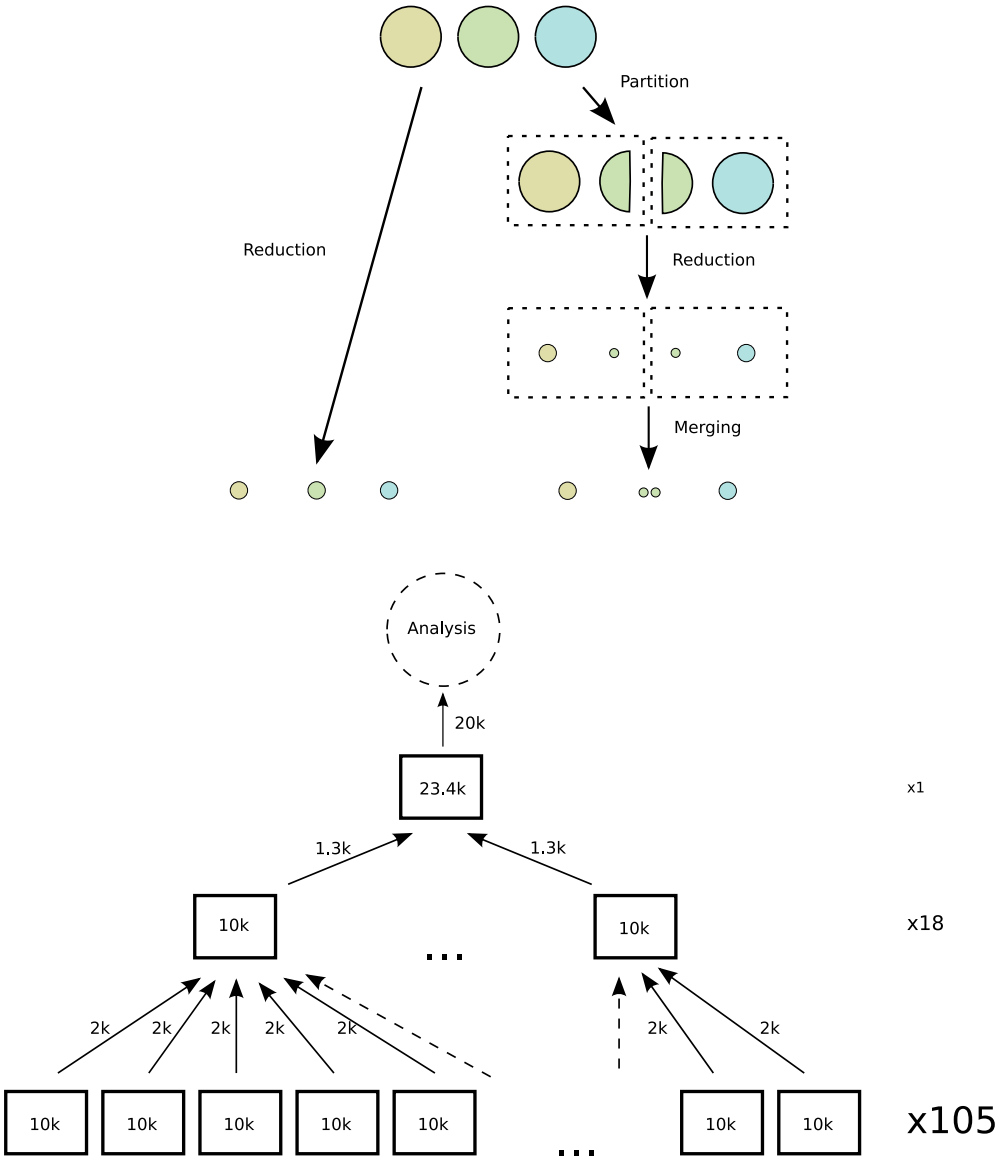


Figure 3.14: A) Global reduction of the size of the dataset vs. merging local reductions. B) Different levels of compression including the number of frames used in each level.

Level	Runs	Time per run (s)	Clusterings per run
Third (10k→2k)	105	~1500	300-400
Second (20k→1.3k)	18	~2200	300-400
First (24k→20k)	1	12395	452

Table 3.3: Around 40k clusterings were produced in almost 58h (1 clustering each 5s).

Also, the lack of knowledge of the system forced us to use a very general hypothesis, increasing the number of clusterings that had to be generated and thus the total execution time.

3.7.5.3.2 Clustering

The clustering chosen by pyProCT was composed of a total of 19 clusters, one of them holding the 34% of the conformations. The Cα-RMSD with the experimental structure (PDB id 2JOF) is 1.8Å, which is similar to the 1.4Å RMSD calculated in the original article (see Fig. 3.15).

3.7.6 2D Validation

In order to improve the reliability of pyProCT we have performed two different quality assurance methods. The first was to ensure that the software itself was working correctly using a unit testing methodology (trying to get the best possible test coverage). The second was centered in the validation of the clustering algorithms and the protocol.

Clusterings are hard to validate, especially when using multidimensional data. Validating a 2D clustering, however, can be an easier task as it can be visually checked. To this end we coded the scripts that can be found in the folder `pyproct/validation/bidimensional/`.

3.7.6.1 Datasets

To perform the validation, we downloaded some of Helmuth Spaeth's[183] datasets. These datasets have different characteristics that make them difficult to cluster :

1. In this dataset 3 to 5 clusters can be seen. It looks like some of them can be subdivided. In general, these clusters are compact.

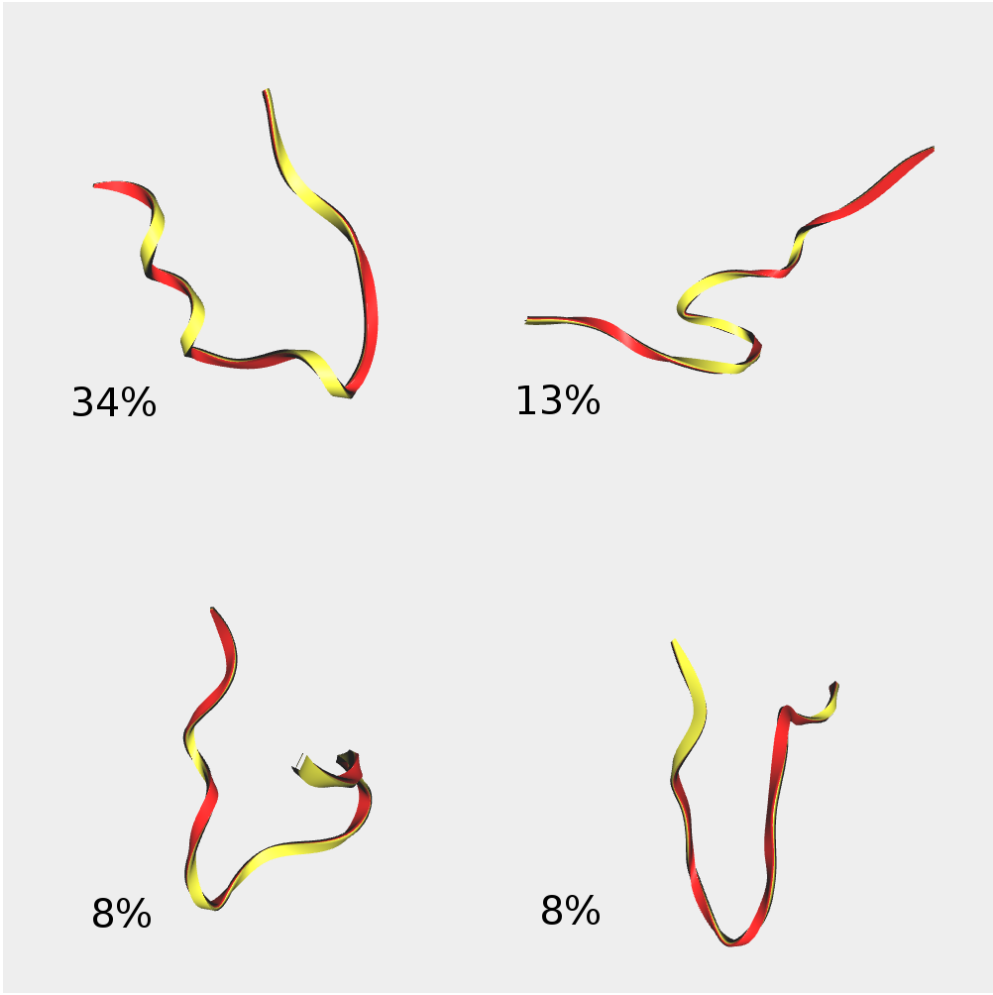


Figure 3.15: Representative conformations for the 4 most populated clusters, holding a 34%, 13%, 8 % and 8% of the elements of the dataset.

2. It shows a set of points homogeneously covering the plane. There is not noticeable density variations.

3. In this case there are two different density regions. In the bottom-right corner there is a compact cluster. The remaining points are sparsely distributed in the remaining space .

4. Two compact clusters to the left, one big cluster (which seems to be composed of other clusters) sits on the right.

5. Three parallel elongated clusters of different sizes and densities.

6. Three elongated clusters with similar densities sharing the same origin.

7. Two overlapped elongated clusters with different densities.

8. Three elongate clusters with similar densities. All three are overlapped.

We also used a code adapted from Jochen Wersdrfer's blog [184] to generate a 9th dataset, which contains 450 points lying in three concentric circles.

3.7.6.2 Protocol validation

In the first version of the validation script we used the datasets to validate the algorithms, that is, we coded some algorithm-parameter pairs and checked a picture of the resulting clusterings. Since the algorithms were working as expected, we upgraded the script to fully test the HCE protocol.

For each dataset, two hypotheses about the noise, cluster size and number of clusters were defined (see 3.4) based on our observations of the datasets. Also, we used two different criteria to describe the expected clusterings:

"default_criteria" Uses Silhouette and Cohesion ICVs. Is the default criteria of pyProCT and fosters both separation and compactness.

"graph_criteria" Uses the 'NCut' ICV. It tries to separate a graph representation of the dataset into connected components so that the sum of inner edge weights is optimized.

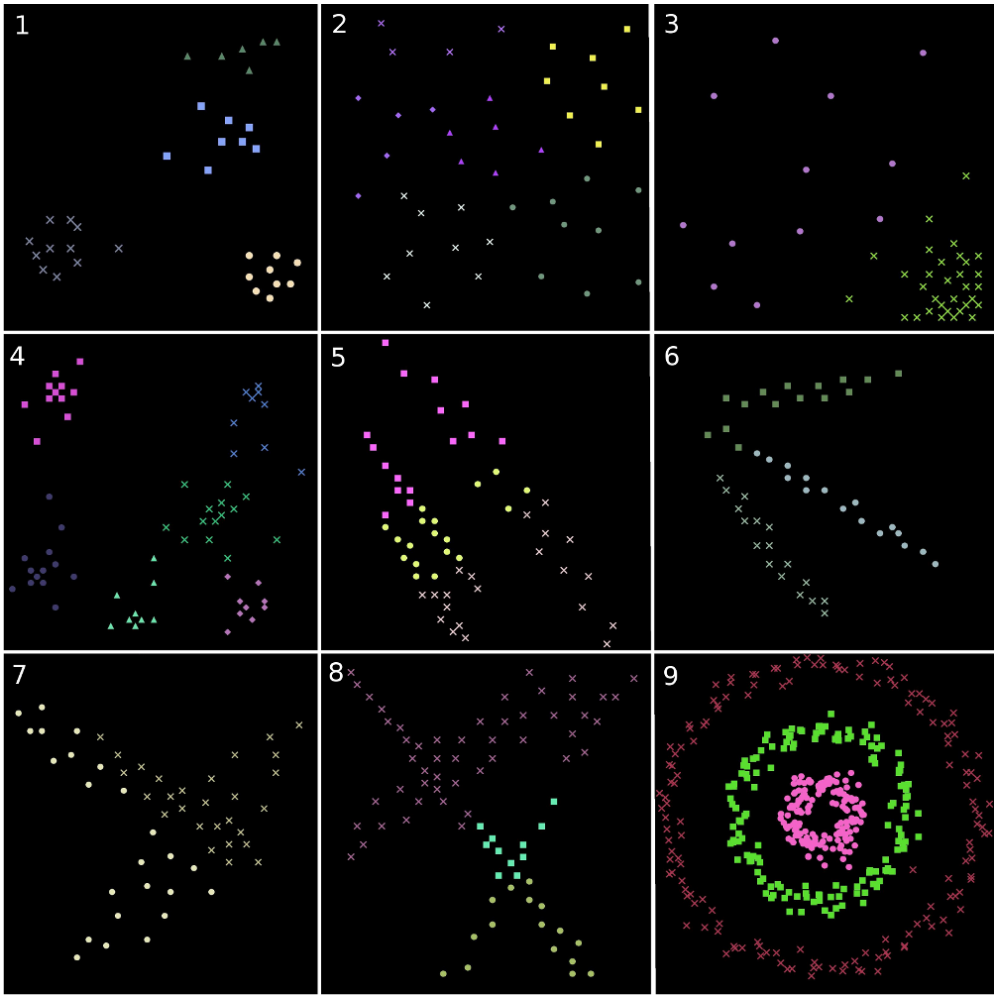


Figure 3.16: Results of the application of pyProCT to nine 2D datasets. Clusters are plotted using different colors and symbols.

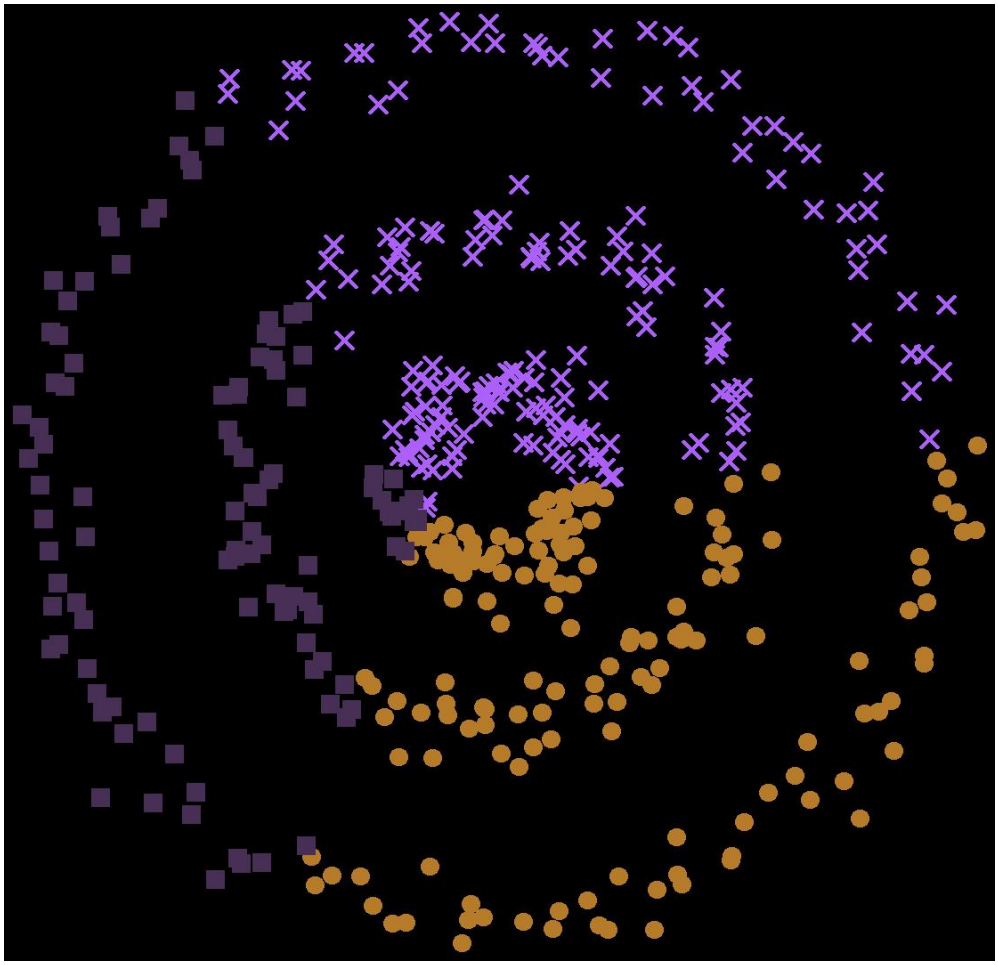


Figure 3.17: An incorrect choice of the ICVs to express the desired resulting clustering traits can drastically modify the results. In this case the criteria was changed from "graph_criteria" to "default_criteria", favoring one of the clusterings generated by the K-Medoids algorithm.

Dataset	Min. Clusters	Max. Clusters	Min. Cluster Size	Max. Noise	Criteria
1	2	10	3	10%	"default_criteria"
2	2	10	2	10%	"default_criteria"
3	2	10	10	10%	"default_criteria"
4	2	10	8	10%	"default_criteria"
5	3	10	10	5%	"default_criteria" and "graph_criteria"
6	3	10	13	10%	"graph_criteria"
7	2	10	10	10%	"default_criteria" and "graph_criteria"
8	3	8	5	10%	"default_criteria" and "graph_criteria"
9	3	4	100	5%	"graph_criteria"

Table 3.4: Clustering hypothesis for each of the datasets.

3.7.6.3 Results

Clusterings 1, 3, 4, 6 and 9 are in full accordance with our expectations (see table 3.5 and Fig. 3.16). We thought that the optimum solution for dataset 2 could be to use one single cluster encompassing all elements. However the final partition in 5 clusters looks reasonable.

Clustering 5, 7 and 8 are different of what our intuition dictates. The main problem that pyProCT has when dealing with a dataset like 7 or 8 is that their "natural" clusters overlap i.e. there are elements that belong to more than one cluster at the same time. This could be overcome by adding fuzzy algorithms to the algorithms pool. Despite this, results would look counterintuitive in any case, as its usefulness in most scenarios implies to discretize the membership values.

Clustering 5 highlights a weakness of the HCE methodology: its success depends on the ability of the user to convey their goals in the clustering hypothesis. If the user is not able to express it using pyProCT built-in ICVs (see Fig. 3.17) or the needed ICVs to define the hypothesis are not yet implemented, it would be impossible for users to get the best-fitted result for their problems. It is clear that, in this case, none of the used criteria suffices to choose the type of result we would like to obtain.

Dataset	Algorithm	Num. Clusters	Noise	Criteria
1	Gromos	4	8.10%	“default_criteria”
2	Spectral Clust.	5	0%	“default_criteria”
3	K-Medoids	2	0%	“default_criteria”
4	K-Medoids	6	9.59%	“default_criteria”
5	K-Medoids	3	0%	“graph_criteria”
6	DBSCAN	3	4%	“graph_criteria”
7	K-Medoids	2	0%	“graph_criteria”
8	K-Medoids	3	5.19%	“graph_criteria”
9	Spectral Clust.	3	0%	“graph_criteria”

Table 3.5: Details of the results. Last column indicates the criteria that obtained the best score.

4

Summary of the results

In the present chapter, we aim to give a brief summary of the results obtained for each of the proposed objectives. A more detailed discussion of each of them can be found in the next chapter.

4.1 Technical improvement of PELE

As stated before, the results related to the first objective consists, mainly, in the production of new software, which is not open source. The code base includes the rewriting of PELE in C++, the related Python scripts, and the parallelized kernels.

As one of the main objectives of this thesis is the development of faster sampling techniques for VHTS, a significant amount of work was focused on speeding up the software. This was partly achieved by GPU acceleration of the heavier routines, where we obtained up to 24x kernel speedups.

4.2 Algorithmic improvement of PELE

We have presented a new perturbation method for PELE using torsional normal modes (icNMA). We have tested the new methodology in two different systems (ubiquitin and an Src Kinase) and compared the results of the current method (ccNMA) and the new method with molecular dynamics simulations in explicit

solvent. The results show that this approach is able to produce more energetically favorable perturbations than the Cartesian coordinates-based method, thus allowing to work at 300 K without the need of a system-wide minimization. The root mean square fluctuation of the residues indicates that icNMA reproduces protein flexibility better than ccNMA; however both fluctuate less than MD. The measurements of the solvent accessible surface and radius of gyration show that icNMA is able to better capture the variations of volume of the protein. Furthermore, the way it simulates the inter-domain movements of the Src kinase is more similar to MD. Finally, each icNMA iteration is faster than a PELE iteration, as it does not include the side chain prediction and global minimization steps.

Some parts of the icNMA code are open source and can be found in its GitHub repository[1].

4.3 Efficient and reliable analysis of large conformational ensembles

4.3.1 Implementation of an efficient solution for the calculation of collective superimposition operations

We have introduced the Python package pyRMSD. It provides the Python programmer with three superimposition algorithms and up to 4 fully parallelized collective operations. One of the examples shown, the calculation of an RMSD matrix, reaches a speedup of 5x when using 6 cores and 11x using a GPU.

The code of pyRMSD is open source (under MIT license) and, to the best of our knowledge, it is the first open source CUDA parallelization of this kind. Readers interested in downloading or contributing to the code can find it in its GitHub repository[2]. Moreover, some compiled packages are hosted in the PyPI package repository[3] and can be easily installed using the `pip`[4] tool, which manages the downloading, compilation, installation and the handling of dependencies.

[1] `https://github.com/victor-gil-sepulveda/PhD-ANMInternalCoordinates`
[2] `https://github.com/victor-gil-sepulveda/pyRMSD.git`
[3] `https://pypi.python.org/pypi`
[4] `https://pip.pypa.io/en/stable/`

4.3.2 Implementation of a reliable cluster analysis protocol

We have presented the Python software pyProCT which aims to be a reliable cluster analysis alternative when used as a black box. We have described how the meta-algorithm works and shown some of its features in two representative use cases. In the first one, we have been able to correctly separate the conformations of two synthetic conformational ensembles without any knowledge of their generation process. Moreover, an iterative analysis method to refine the working hypothesis has been introduced. In the second use case, we have used pyProCT to eliminate the redundancy of a large DNA-ligand simulation and obtain the best ligand clusters. Our results correlate well with the clusters obtained in previous works using a kinetic analysis. Finally, we have shown how pyProCT can be used to reduce the size of a huge conformational ensemble (around 1 million structures) to find the most biologically relevant conformations of a protein folding simulation.

On the technical side, pyProCT also takes advantage of parallel architectures (multicore or distributed architectures) by using a parallel task scheduler. The code is also open source (under MIT license) and can be found in its public GitHub repositories[5]. Both pyProCT and its GUI are also available in the PyPI repository and can be installed easily using the `pip` tool.

[5] `https://github.com/victor-gil-sepulveda/pyProCT` and `https://github.com/victor-gil-sepulveda/pyProCT-GUI`

5 Discussion

5.1 Technical improvement of PELE

5.1.1 From PELE to PELE++

On its previous incarnation, PELE was using the functions of the Protein Local Optimization Program (PLOP) [185] software package as a library. Since PLOP had been written in FORTRAN 77-95, the most natural choice for PELE was to use the same programming language. The limitations of FORTRAN, together with the pragmatic but chaotic nature of academic developing, drove it to a maintainability dead-end. The refactoring needed to pay the accumulated "technical debt" was so huge that we decided to go through a complete rewriting of the code .

The author of these lines worked on this rewriting steadily for two years and a half, making more sparse contributions since then, and was in charge of the design of the software core and of its successive iterations. Given the magnitude of the project, a small group of technicians soon joined the development team which was leadered by Mr. Manuel Rivero González for more than three years. This group has been renewed lately, and is now leadered by Dr. Jorge Estrada. The rewriting has been named PELE++ and has been the base of all the developments mentioned in this work.

The old version of PELE was an academic software, and so is PELE++. This means that the software was not only planned to be used to perform *in silico* experiments but also as a tool to test and develop new algorithms. As a consequence, the new code needed to be well documented, easy to learn, easy

to maintain and robust to experimental changes. The academical nature of the software, indeed, explains some of the design decisions taken. As a result, PELE++:

- Uses an Object-Oriented Programming (OOP) paradigm instead of the previous single file per module approach, which also fosters reusability. C++ has been chosen as the language for this new version as it implements the OOP paradigm and compiles to very efficient executables.
- Is better designed for maintainability: Design patterns and other Software Engineering techniques have been wittingly applied (see Fig. 5.1). SOLID[1] principles have been honored. As a side effect, testability has been improved.
- Is better documented: The constant turnover of new students has made it necessary to pay special attention to a proper documentation of code functions and classes. This shortens the time needed to train new developers so that they can start to contribute to the project earlier.
- Is more reliable: the code correctness can be tested at any phase of the project. An in-house testing library was coded, and each class owns a test suite. An automated testing protocol was implemented.
- Has better performance than the old version: several algorithms were replaced with more readable and efficient alternatives.
- Is easier to extend and modify, a desirable feature as many people would use the code to test their own algorithms.

To exemplify the design changes provided, we include here a simplified UML (Unified Modelling Language) diagram (see Fig. 5.1). The diagram shows two packages: the first describes the handling of structural data, while the second models the energy calculation subsystem. We have chosen to neglect several minor classes and methods for the sake of clarity. We can observe how creation patterns have been profusely used in this new design: the Builder pattern for Potential objects, the Simple Factory pattern for ForceField objects. Consider that the Abstract Factory pattern, used to create the diverse AtomSet descendants, is not shown here.

The AtomSet object, a group of atoms with defined geometry (coordinates) and topology (interactions), is of particular interest as it is involved in two different hierarchies. The first one is semantic and comes from the use

[1] Acronym for Single responsibility principle, Open/closed principle, Liskov substitution principle, Interface segregation principle and Dependency inversion principle

of inheritance (e.g. a Residue is a type of Link which is a more general case of an AtomSet). The second one reveals a tree-like organization that emerges from a loose interpretation of the Composite pattern: the root is the Complex, a singleton object made of Chains which, in turn, are made of Links.

The rewriting of the code has allowed us to add new features to PELE, such as the possibility of using different force fields and various solvent models, supporting the simulation of other biopolymers (e.g. DNA [186]), using arbitrary sized structures and coarse-grained models and even performing an effective parallelization of the code. In general, this has converted PELE in a tool that is more useful for academical experimentation and efficient and reliable enough for industrial use.

5.1.2 Performance vs. maintainability

Scientific software is the object of an open debate on performance versus readability. The development of PELE++ has shown us that fostering readability can save developers time and efforts at almost no performance cost, in contrast with the theories supported by other authors [187]. Readable code, for instance, allows building a better structured code base. Thanks to this, developers can have a better global view of the software and apply optimizations (e.g. avoid unnecessary actions). It also allows them to react faster in front of specification changes and augments the resiliency of the developing team to staff adjustments.

One of the most important issues we faced is the mutability of specifications, in spite of the thorough initial use case studies. This can be caused by the communication gap between developers and users (or stakeholders), due to their different domains of knowledge. As this situation can pose a great problem for long-term design, we consider that domain-driven design techniques, using experts' feedback to refine the model, might be of great help.

Finally, this development has taught us that, in order to succeed in such big projects, it is important to perform short and abundant refactoring cycles and, ideally, to assign development tasks to optimal-sized teams of specialized software engineers.

5.1.3 Optimization and Parallelization

During the last decade, Moore's law predictive power has reached its limits, since semiconductor manufacturers have found the physical limits of miniaturization; CPU frequency scaling beyond that point was not possible and, as a consequence, cluster-based computers were the only solution left to increase

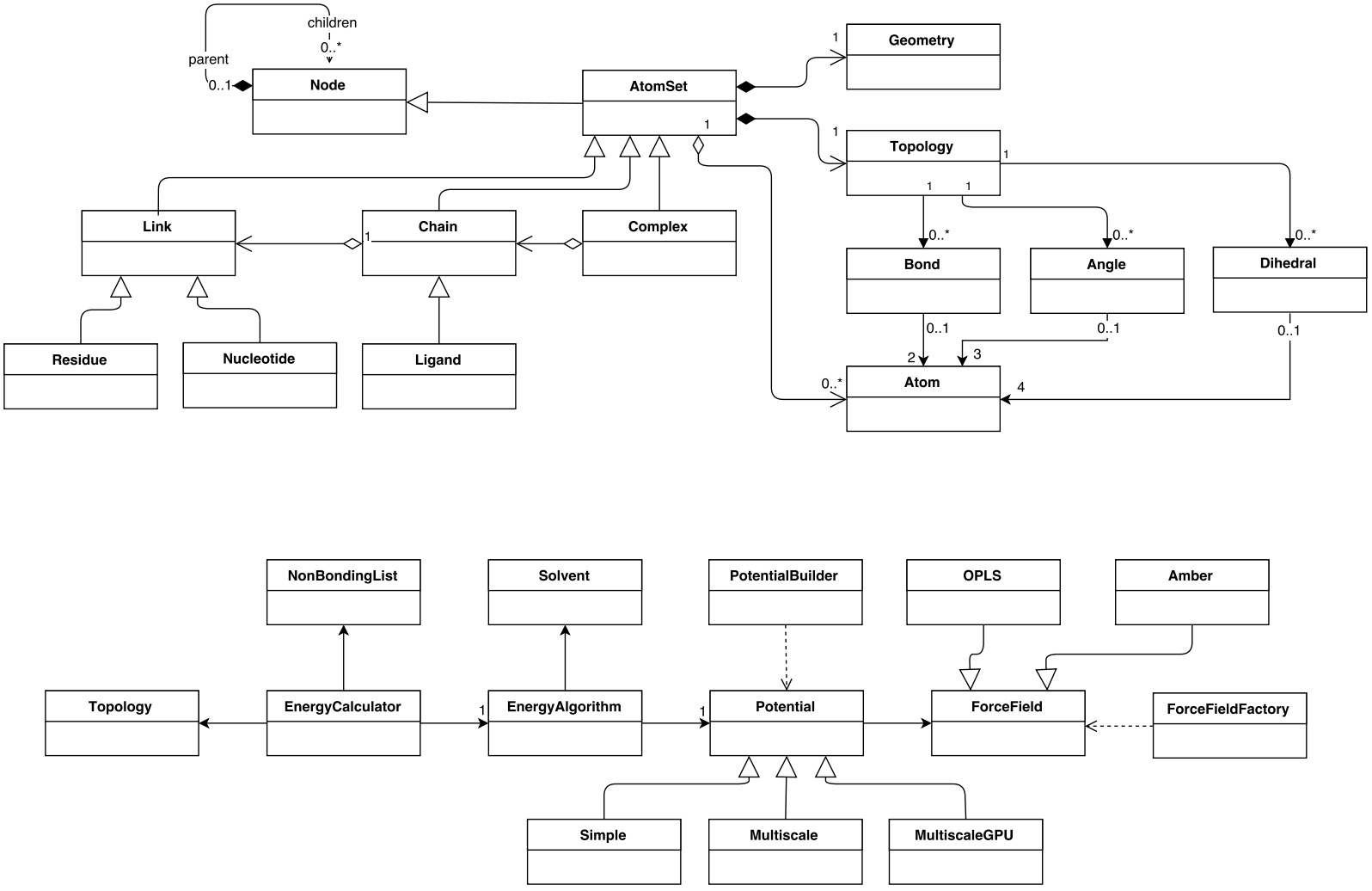


Figure 5.1: Pseudo-UML diagram showing the most relevant classes in PELE++ core: the AtomSet tree which allows the definition of different types of molecules, the topology subsystem and the energy/potential subsystem. The last uses geometry (atom coordinates) and topological information to calculate the energy of an AtomSet.

Size	Protein		Ligand
	Residues	Atoms	Atoms
Small	120	1731	19
Medium	583	9300	16
Big	710	11222	45

Table 5.1: Size-related details of the systems used in the initial profilings.

the global computational power. This was undoubtedly the starting shot for the development of multicore CPUs in commodity hardware. Some years later, the graphic coprocessors of such CPUs evolved to standalone graphic cards (GPUs[2]), which manycore chips were eventually adapted to perform highly parallel computations. Soon after, Intel presented their MIC[3] architecture. Easier to program than GPUs and also able to perform highly parallel computations, it has rapidly become as prominent as GPUs were some years ago. Computational hardware evolves rapidly, and software must be able to take advantage of this evolution.

One of the improvements of the PELE rewriting has been offering a code base that can be parallelized more easily than the old FORTRAN version.

5.1.3.1 Initial profilings

We selected three real (under study) protein-ligand systems with different sizes to use them in our test simulations. The number of atoms and residues of such systems is summarized in Table 5.1.

The initial profilings used the two currently implemented implicit solvent models in order to identify the most efficient one. Executions were performed in a Mare Nostrum [188] node (Intel SandyBridge-EP E5–2670 @2.6 GHz processor). Nodes were used exclusively so that no other processes could interfere with the results. A control script was written for each of the systems and solvent models, and PELE++ was run for 100+ steps in order to enter the convergence regime. The profiling was performed using the same script for ten steps, being the initial conformation the last frame of previous simulations. Profiling data was then extracted using gprof [189] and then analyzed using the visual analysis tool gprof2dot[4].

The results showed that PELE++ spent most of the time in two kinds of functions (see Fig. 5.2):

[2]Graphics Processing Unit

[3]Many Integrated Core

[4]https://github.com/jrfonseca/gprof2dot

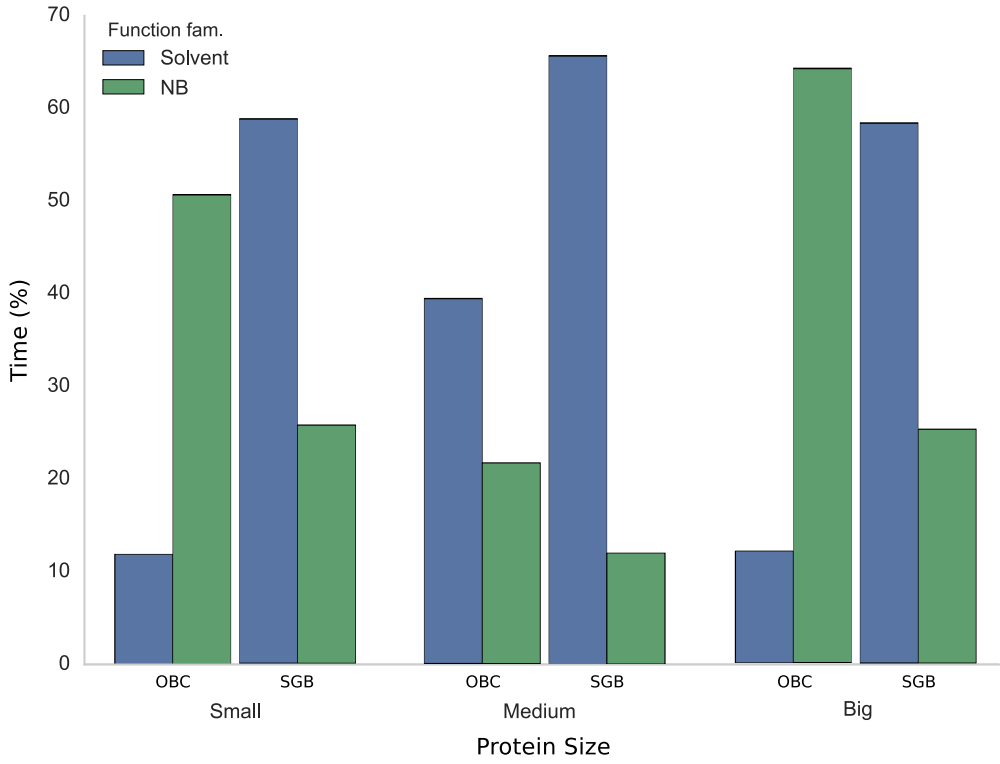


Figure 5.2: A set of profiles was performed for different protein sizes and using OBC and SGB solvent. This bar plot shows the percentage of time spent in non-bonding and solvent-related calculations.

- Functions related to the solvent model, more specifically the alpha and surface elements. It is worth noting that simulations using the OBC solvent model were typically faster.

- Functions related to the calculation and the evaluation of non-bonding lists in order to calculate the electrostatic and van der Waals potential energy terms and gradient. This is common in MD, among other methods, and has been the topic of several other studies [190–192].

From this profiling exercise, we also learned that the behaviour of PELE++ can totally change depending on the place of the ligand. Two different scenarios were identified: free ligand diffusion (the ligand explores the protein surface freely) and binding refinement (the ligand is already in the binding site, and a better pose is searched).

5.1.3.2 Non-bonding energy parallelization using GPUs

According to the profile results, the next logical step would have been improving the performance of the solvent-related functions, but the difficulty of parallelizing the SGB[5] algorithm (due mainly to its data dependencies) lead us to focus on the performance improvement of NB functions first [148].

The main contribution of NB energy/gradient calculations to the overall time does not reside in the computational cost of evaluating a single interaction calculation, which is indeed quite fast, but in the huge number of calculated interactions ($\propto N^2$ where N is the number of atoms).

The evaluation of these huge lists of interactions looked to be a computationally-bound problem and fit well with the GPGPU (General-Purpose computing on Graphics Processing Units) paradigm. Graphic Processing Units have many core architectures with numerous parallel calculation units that allow them to perform high-throughput calculations with ease. They usually have good bandwidth but elevated latencies, which can be shaded thanks to the extensive use of lightweight threads that allow alternating calculations with memory operations.

We wanted to adapt PELE++ code to work with this kind of accelerators using the OpenCL (Open Computing Language) and CUDA (Compute Unified Device Architecture) programming models. The first step was to design two new structures to pack the data that would eventually be sent to the device. Afterwards, we coded the GPU kernels for the energy and gradient calculations.

[5]First attempts were made at an early stage of the code, when OBC solvent was not yet available.

The energy case is quite trivial, as each thread just calculates several NB interactions to a per-thread accumulator variable. The final value of the energy is calculated through a reduction of the partial energy values.

Conversely, parallelizing the NB gradient calculation using a GPU is not that easy. The fundamental problem is that all threads can potentially end writing in the same positions of the gradient (race condition) and regular strategies to protect concurrent access would only end penalizing performance noticeably. To illustrate this, the reader can think of three particles A,B and C so that particle A interacts only with particle B, and B with C. If two different threads are calculating AB and BC interactions, it is very likely that both try to access B's gradient positions at the same time.

The solution starts by storing the gradient partial contributions in one temporary array. As the gradient contributions of each atom in the interaction pair are equal but opposite, it is possible to use only half of the memory to store intermediate results. Once all the interactions are evaluated and the array is filled, it is ordered by atom using two mapping tables (one for each interacting atom) so that each GPU thread is able to reduce all the iterations of one atom at a time.

In order to test our code, we had two different GPUs available: an NVIDIA Tesla M2090 card (for CUDA implementation only) and an AMD Radeon HD 6870 card. Both cards have different technical specifications[6] and their results are not comparable. Another set of three proteins with a larger number of atoms was used with ∼1k atoms, ∼14k atoms and ∼30 atoms so that the calculation times of a single non-bonding list were significant.

Our first estimates showed that the calculation of kernels was so fast that transference and precalculation overheads were making the overall performance worse than the serial version. Both data transfer and kernel execution were made asynchronous, thus alleviating the relative impact of that overheads.

The comparison of performance results was made at kernel and program levels. The execution time of serial and GPU kernels (Fig. 5.3) showed speedups of up to 24x for the energy function and 12x for the gradient.

At that time, the code was not mature enough to introduce the changes that would allow serial and accelerated versions to coexist. In order to obtain values for the execution of the whole program, we had to build a model of its behaviour based on the profiling studies, typical executions and our knowledge of the code.

The Truncated Newton minimization is the part of the code where energy and gradient functions are called the most. The method consists in several iterations of a five stage protocol [193], called the "outer loop", which includes

[6]E.g. Nvidia card has 512 threads per block and the AMD card has 256 and no double precision support

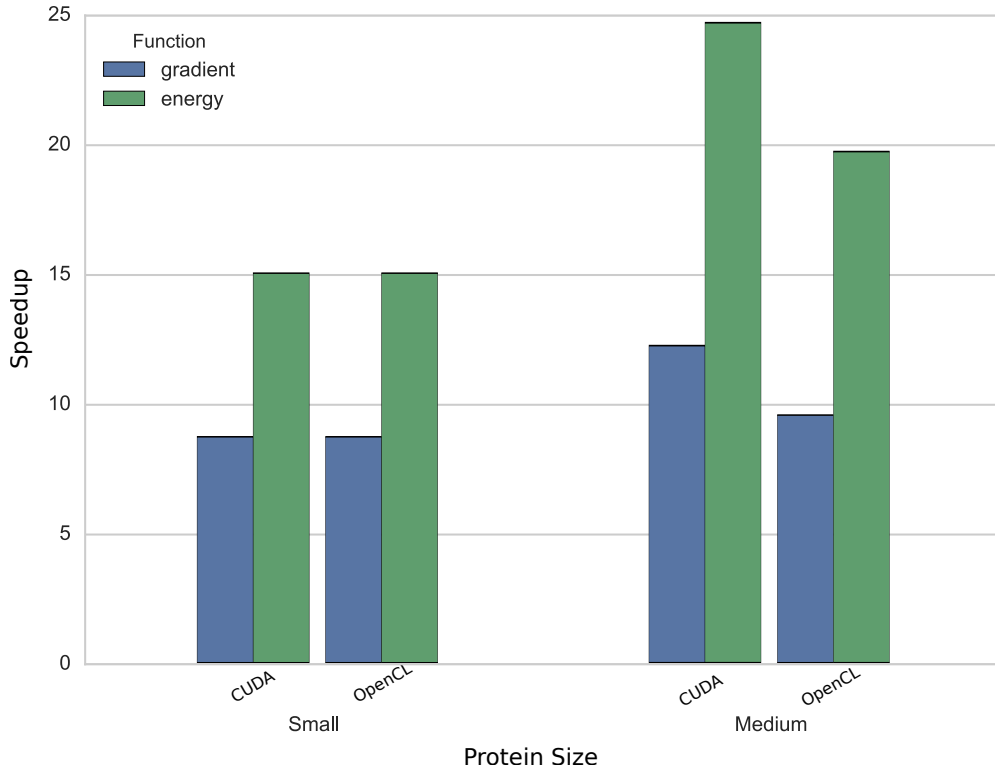


Figure 5.3: Comparison of the energy and gradient functions speedup for different protein sizes and the two programming models used. One of the reasons why the speedup for the medium protein is higher is because the relative weight of the non-bonding calculations has increased with the number of atoms.

some preparatory steps and the evaluation of the Hessian. The "inner loop" is the stage in which a preconditioned conjugated gradient is performed. Each minimization is usually run three times with fixed alpha values[7] for performance reasons. The time needed to perform a minimization in the serial case can be modelled as:

$$T = 3(N(t_5 + Mt_5)) \tag{5.1}$$

and for the parallel case:

$$T = 3(t_2 + N(t_1 + t_3 + t_4 + t_5 + M(t_1 + t_5))) \tag{5.2}$$

where N and M are the number of times the "outer" and "inner loops" are executed, t_1 and t_2 are the time needed to create (if applicable) and transfer atomic data structures to GPU, t_3 is the time required to generate the vector of interactions in the GPU, t_4 is the time needed to order the atom interaction maps and t_5 is the time needed to calculate the gradient. It is worth mentioning that non-bonding calculations are to be performed in both the "outer" and "inner loops", and that coordinates will be changed at each "inner loop" iteration, which makes it mandatory to update their GPU memory representation.

The model allowed us to obtain a rough estimate of the execution of the whole program (Fig. 5.4) in the serial and parallel case. The results showed that adding the CUDA implementation to PELE++ would imply a 16.3 - 27.8% faster execution than the serial version. The expected performance increase for the OpenCL code was smaller, with a 13.5 - 26.7% speed increase. However, it is not possible to make a fair comparison of both methods, as hardware platforms are not equivalent and, also, CUDA kernels are using an optimized sorting algorithm from a library, while the OpenCL sorting algorithm had to be coded from scratch.

5.1.3.3 Solvent parallelization

The performance improvement through the parallelization of solvent-related functions is currently an open project lead by our research group's technicians Pedro Riera and Jorge Estrada, the company Pharmacelera[8] and BSC's Montblanc[9] project team. The most costly functions are the ones that calculate surface elements and atom alpha attributes (which are used when calculating energy in order to take solvation contributions into account).

[7] Atomic property regarding the solvation model.

[8] http://www.pharmacelera.com/

[9] https://www.montblanc-project.eu/

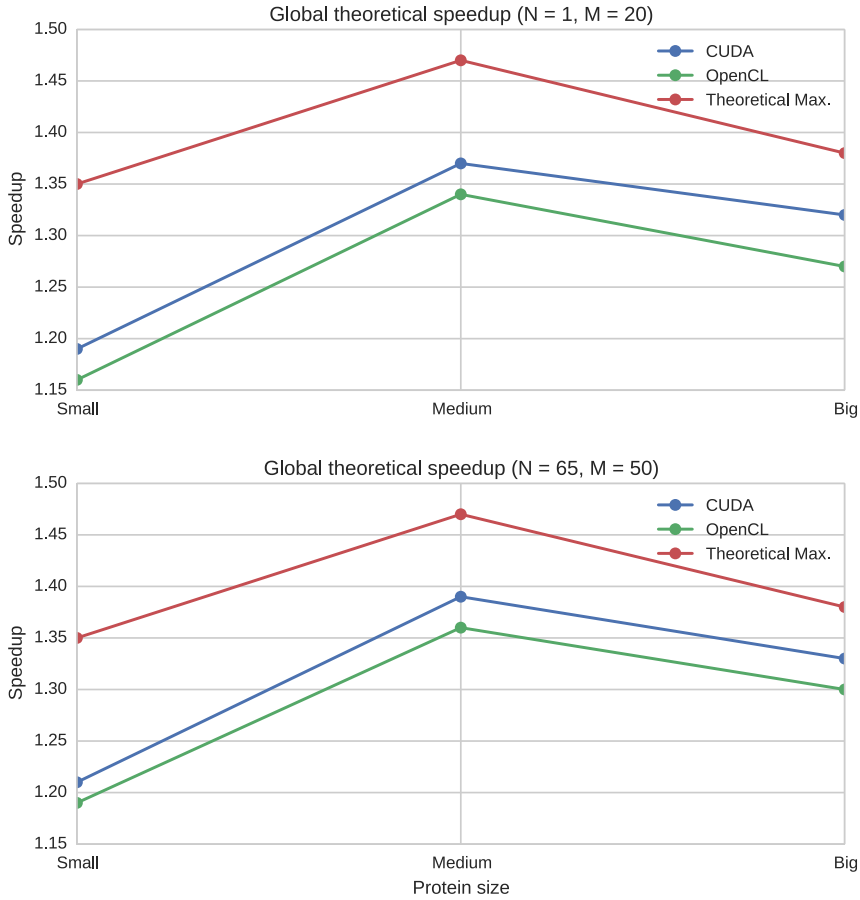


Figure 5.4: The global speedups have been calculated using a model. As parameters N and M range from 1 to 65 and from 20 to 50 respectively, we decided to study the best case (faster serial execution, where N = 1 and M = 20) and the worst case (slower serial execution, where N = 65 and M = 50). Theoretical speedup increases to decrease afterwards. This happens as a result of the changes in the relative weight of the non-bonding calculations: the weight first increases due to the increment in the number of atoms and then decreases since other functions, such as the covalent energy calculations, start to require more time. The difference between implementations is not significative.

As mentioned before, there are currently two different solvent models included in PELE++, namely, the SGB [116] and OBC models [119]. The algorithm for the SGB alpha calculation function is detailed in the pseudocode below:

```
updateAlphasSGB :
        updateSurfaceElements  O(Rn^2)
        updateAtomAlphas  O(n)
        updateSASA  O(Rn^2)
```

The calculation of the atom alpha property depends on the previous calculation of the surface elements. This is the most costly function, since it depends on the number of atoms, their neighbours and a resolution parameter (a worst-case complexity of $O(Rn^2)$).

```
updateAlphasOBC :
        for each atom :
        ComputeOtherAtomsContribution  O(n)
```

The OBC alpha updating method is implemented as a double loop over all atoms (a complexity of $O(n^2)$) which makes it simpler than SGB's (and thus easier to parallelize) and explains why its serial performance is better.

5.1.3.3.1 Solvent parallelization: OpenMP

Some initial attempts of parallelizing the SGB alpha calculation functions were performed using the OpenMP programming model. Two functions were selected: `computeSurfaceElementsOfAtom` ($O(Rn)$) and `updateAtomsSurface` ($O(n^2)$) both called by the `updateSurfaceElements` function.

Tests were performed using a medium size system (4284 protein atoms and 69 ligand atoms) and an Intel SandyBridge-EP E5–2670 (@2.6 GHz) processor. The best results were obtained using eight threads and threw overall speedups of ~1.5x for the refinement and free ligand diffusion simulations.

5.1.3.3.2 Solvent parallelization: OpenCL

The parallelization of SGB solvent functions has been currently abandoned in favour of OBC functions due to their greater simplicity and better serial performance. Pharmacelera engineers have focused on parallelizing it using the OpenCL programming model and have devised 3 parallelization strategies:

- Parallelization of the inner loop (`ComputeOtherAtomsContributions` function) (method 1).

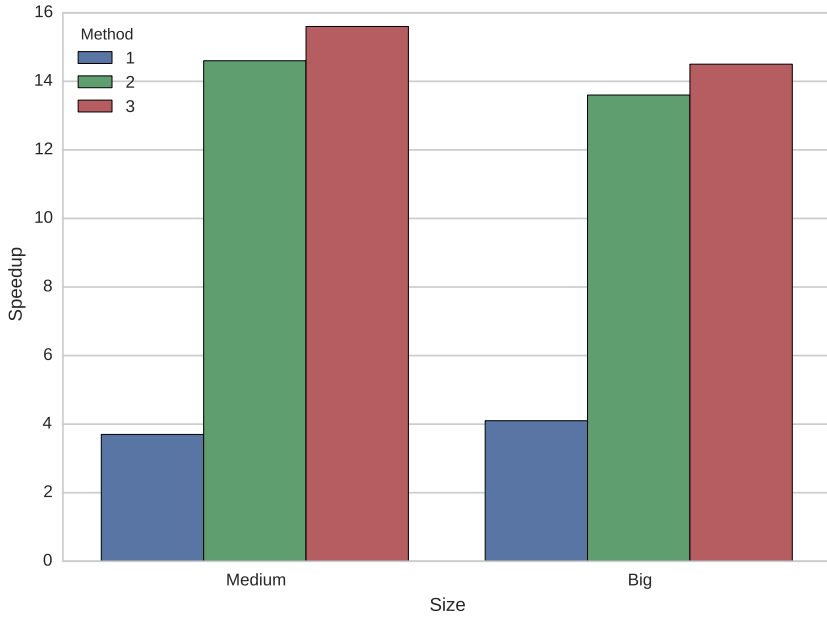


Figure 5.5: Kernel speedup for each of the methods and proteins tested. Methods 2 and 3 seem to obtain an equivalent efficiency improvement.

- Parallelization of the outer loop (`updateAlphas` function) (method 2).
- Precalculation of atom pairs and parallelization over these pairs (method 3).

The first and second strategies have a lower impact on the code, while the third makes a better use of the GPU by improving the load balance.

The tests have been performed on a workstation equipped with an AMD FirePro W5100 GPU card. The results, using the medium and big size proteins employed in the initial profilings, are promising, especially in the case of the third strategy which seems to have the best performance (see Fig. 5.5).

5.1.3.4 Montblanc and other projects

Nowadays, the main obstacle to projecting Exascale systems is energy consumption. The Montblanc project aims at the creation of an energy-efficient and scalable supercomputer using ARM[10] technology. Some of the parallelization techniques explained before are currently being tested in this novel architecture.

[10] Advanced RISC (Reduced Instruction Set Computing) Machine

Finally, some parts of the code are currently being ported to the CUDA programming model as part of the CUDA Center of Excellence program from NVIDIA.

5.2 Algorithmic improvement of PELE

Most of the limitations of PELE enumerated in the introduction (Section 1.5.3.3) are common to all methods using NMA or more specifically ANM, and therefore not exclusive to PELE. Some of the issues are related to the NMA model itself, while the rest are related to the specific ways normal modes are applied. This made us think that finding an alternative to the NMA algorithm would produce a noticeable improvement in PELE sampling robustness and performance. To this end, we have chosen to implement an internal coordinate (IC) based NMA.

5.2.1 Switching to a different coordinates space

Internal coordinate NMA (icNMA) is not a novelty. Indeed, the pioneering NMA works of Wilson [194] were already performed in the internal coordinates space. This is not surprising, given that internal coordinates (e.g. bond distance, bond angle, and dihedral torsion angles) are the most natural way of representing chemical molecules (e.g. with a z-matrix [195]). Actually, several methods use internal coordinates instead of Cartesian coordinates. Some examples are the Multiple Minima Monte Carlo [196] method, the Monte Carlo sampling software MCPRO (Monte Carlo for PROteins) [197] and ICM [46], or the MD software X-PLOR [198] and DYANA [199].

The truth is that Cartesian coordinate NMA methods (ccNMA) are more common than icNMA-based methods, possibly because the mathematical background of the former is more simple. However, this fact has not prevented researchers from introducing icNMA in their simulation software. First examples can be found in the early works of Noguti *et al.* [200, 201] and Kidera *et al.* [202, 203] where the scaled collective variables (SCV) MC and related algorithms are presented. Trosset *et al.* used later a similar methodology [108] in the implementation of PRODOCK. Levitt and Stern [204] suggested an MD method using IC NMA modes. Finally, Lin and coworkers [70] developed a different method using icNMA followed by an energy minimization.

There exist reports claiming that a significantly smaller number of modes are needed to reproduce conformational changes using torsional NMA [205], and that this method improves sampling quality [206] and the results of ligand binding simulations [207] compared to ccNMA-based methods. Changing the coordinate system implies not only changing the way normal modes are calculated, but also the way modes are applied, which could help to mitigate many of the NMA-related issues.

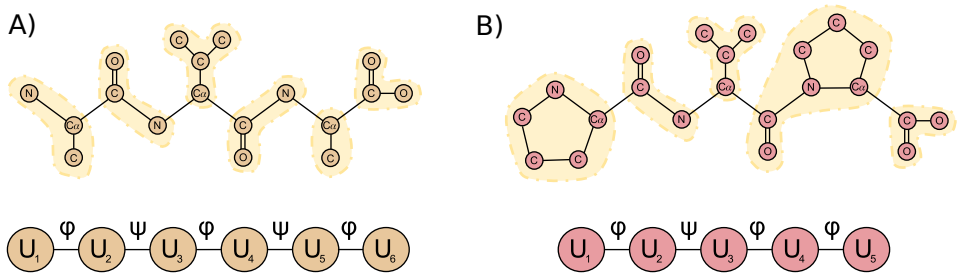


Figure 5.6: CG model of a peptide (A) using the $[C_\alpha]^3$ distribution of atoms. Each residue is composed of two units which are delimited by the ϕ and ψ torsions. The case of proline residues (B) is special, as they only form one unit.

5.2.2 Coarse grain model

The CG model used in our icNMA implementation describes rigid units that encompass all the heavy atoms among rotatable backbone torsions ($[C_\alpha]^3$ [208, 209] model). This means that we will define $\sim 2R$ units for each chain (where R is the number of residues), which can be of 4 different types (see Fig. 5.6):

N-terminal unit Contains the N-terminus, C_α and first residue side chain.

C_α unit Contains the C_α atom and side chain.

C-terminal unit Contains the atoms that form the final carboxyl group.

Peptide bond unit Contains the carboxyl and amino groups involved in the peptide bond.

To account for the rigidity of the ϕ torsion in prolines, its peptide bond and C_α units will be fused together (see Fig. 5.6B).

5.2.3 icNMA theory

5.2.3.1 Hessian calculation

Again, we model our system as a spring network whose potential is the sum of all Hookean interactions between atoms (or CG units) with the form $Vij = \frac{k_i j}{2}(r_{ij} - r^0_{ij})$[11]. If we express these interactions using generalized internal co-

[11] As a convention, units and dihedrals will be numbered correlatively from N-terminal to C-terminal. Regular letters will be used to index atoms and units, and Greek letters to index dihedrals.

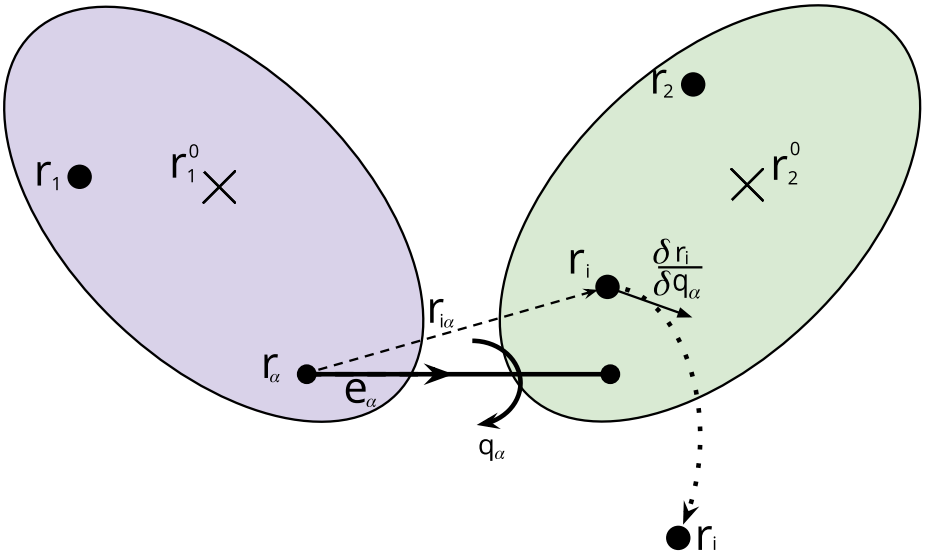


Figure 5.7: Representation of the rotation of two rigid bodies (green and purple) around axis q_α.

ordinates (which in our case will be limited to the torsion angles of backbone dihedrals), we can write the potential energy as:

$$V = \frac{1}{2}(q - q^0)H(q - q^0)^T \tag{5.3}$$

and the Hessian in terms of q [207] as:

$$H_{\alpha,\beta} = \frac{\partial^2 V}{\partial q_\alpha \partial q_\beta} = \sum_{i<j} \frac{f_{ij}}{\left|r_{ij}\right|^2} \left\langle r_i j, \frac{\partial r_i - \partial r_j}{\partial q_\alpha} \right\rangle . \left\langle r_i j, \frac{\partial r_i - \partial r_j}{\partial q_\alpha} \right\rangle \tag{5.4}$$

which depends on the values of the inverse of Wilson's B matrix [194] (with elements $\frac{\partial r_i}{\partial q_\alpha}$). If we impose Eckart conditions [210] ($\sum_i m_i \frac{\partial r_1}{\partial q_\alpha} = 0$ and $\sum_i m_i r_i^0 \times \frac{\partial r_1}{\partial q_\alpha} = 0$) and that the origin of the molecule is the center of mass, the partial derivatives can be calculated as [211]:

$$\frac{\partial r_1}{\partial q_\alpha} = e_\alpha \times \left(\frac{M_2}{M} r_\alpha + \frac{M_1}{M} r_1^0\right) - r_1 \times \frac{M_1 r_1^0 \times (e_\alpha \times r_\alpha) + I_2 e_\alpha)}{I} \tag{5.5}$$

$$\frac{\partial r_2}{\partial q_\alpha} = -e_\alpha \times \left(\frac{M_1}{M} r_\alpha + \frac{M_2}{M} r_2^0\right) + r_2 \times \frac{M_2 r_2^0 \times (e_\alpha \times r_\alpha) + I_1 e_\alpha)}{I} \tag{5.6}$$

which describes the change of Cartesian coordinates when one part of the chain is kept fixed and the other part moves around torsion q_α and vice versa. Both rigid bodies must be taken into consideration for a single rotation, as Eckart conditions imply the conservation of momentum. In these and in the following equations, M is the summed mass and I is the inertia tensor for all the units that compose the molecule. M_1, M_2, I_1, I_2, r_1^0 and r_2^0 are the summed masses, inertia tensor and center of mass of the set of units to the left (subindex 1) or to the right (subindex 2) of dihedral q_α ; e_α is a unit vector with the direction of the bond; r_α and $r_{\alpha+1}$ are the positions of the atoms at both ends of the rotatable bond. Finally, r_1 and r_2 are arbitrary atoms in any of the left or right units of q_α torsion (see Fig. 5.7).

From these equations, a naive expression for the Hessian calculation can be deduced with a $\Theta(n^4)$ computational cost (assuming that the number of dihedrals is roughly proportional to the number of atoms). We are again in debt to Noguti and Go [200] and Abe and coworkers [212] for the development of a recurrent method to calculate the Hessian, which can be implemented as a faster recursive algorithm ($\theta(n^2)$) that also turns out to be more memory efficient. First, Cartesian and internal coordinate-dependent terms are separated so that we can write each Hessian element as

$$H_{\alpha,\beta} = (e_\alpha, e_\alpha \times r\alpha)\, R_{\alpha,\beta} \begin{pmatrix} e_\alpha \\ e_\alpha \times r\alpha \end{pmatrix} \tag{5.7}$$

where the matrix R is calculated as

$$R_{\alpha,\beta} = \sum_i {}_\alpha \sum_j {}_\beta D_{ij} \tag{5.8}$$

D matrix models the interaction of atom i and j:

$$D_{ij} = \frac{f_{ij}}{\left|r_{ij}\right|^2} \begin{pmatrix} r_i \times r_j \\ r_{ij} \end{pmatrix} \left(r_i \times r_j, r_{ij}\right) \tag{5.9}$$

where the distance-dependent force constant is calculated as:

$$f_{ij} = \frac{k_0}{1.0 + \left(\frac{r_{ij}}{x0}\right)^6}. \tag{5.10}$$

In our implementation, $k = 1$ and $x0 = 3.8$, which are the same definitions used by the iNMA software [98].

To calculate the elements of R we may need to store (or recalculate, which would affect computational efficiency) all D_{ij} values. The number of matrices stored can be lowered by defining a matrix T, whose elements T_{ab} summarize the interactions of all the atoms of rigid units a and b:

$$T_{a,b} = \sum_{j \in a} \sum_{j \in b} D_{i,j} \tag{5.11}$$

It is possible to calculate R from T by determining a new matrix U so that:

$$U_{ab} = \sum_{i \leq a} \sum_{j > b} T_{ab} \tag{5.12}$$

which contains all atomic interactions from the first unit to the a unit, and from the b unit to the last one (which can be interpreted as fixing units a to b and allowing units 0 to a and $b + 1$ to M to rotate along dihedrals $\alpha = a$ and $\beta = b$). This makes it possible to calculate R with only one index change:

$$R_{\alpha,\beta} = U_{a,b+1} \tag{5.13}$$

Finally, a recursive solution can be written so that:

$$U_{a,b} = U_{a,b+1} + U_{a-1,b} + U_{a-1,b+1} + T_{a,b} \tag{5.14}$$

being the value of U equal to 0 if any of its indexes is outside the range [0, M-1].

5.2.3.2 Calculation of the metric tensor

We can also express the kinetic energy in terms of the generalized coordinates q so that

$$K = \frac{1}{2} \dot{q}^T K \dot{q} \tag{5.15}$$

and the metric tensor K becomes

$$K_{\alpha,\beta} = \frac{\partial^2 K}{\partial \dot{q}_\alpha \partial \dot{q}_\beta} = \sum_i^n m_i \left\langle \frac{\partial r_i}{\partial q_\alpha}, \frac{\partial r_i}{\partial q_\beta} \right\rangle \tag{5.16}$$

which can be calculated as [211]:

$$K_{\alpha\beta} = \frac{M_1 M_3}{M}\left[e_\alpha \times (r_\alpha - r_1^0)\right]\left[e_\beta \times (r_\beta - r_3^0)\right] + \left[M_1 r_1^0 \times (e_\alpha \times r_\alpha) - I_1 e_\alpha\right] I^{-1} \left[M_3 r_3^0 \times (e_\beta \times r_\beta) - I_3 e_\beta\right] \tag{5.17}$$

We are currently using the definition of the inertia tensor provided by Noguti and Go [211], also employed by Braun *et al.* [213] as defined by Lu and coworkers [146] so that

$$I = \sum_i m_i P^T P \tag{5.18}$$

and

$$P_i = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \tag{5.19}$$

Once we have obtained the metric tensor and the Hessian, we can calculate the normal modes using Eq. 1.4. The resulting eigenvalues are still related to mode frequencies. The meaning of the eigenvectors, however, changes compared to their Cartesian coordinates counterparts; their size decreases from $3N$ to $\sim 2R$ (where N is the number of atoms or CG units) and each single element α represents a differential rotation around torsion q_α.

In a preliminary study [149] we were able to demonstrate that the internal coordinate and Cartesian coordinate mode spaces are in good agreement.

5.2.4 From internal to Cartesian coordinates and back

In order to compare IC modes with CC modes, we convert one coordinate system to the other using the Jacobian (J, inverse of Wilson's B matrix) in the equation

$$J_{i,\alpha} = \frac{\partial r_i}{\partial q_\alpha} \tag{5.20}$$

We can calculate Cartesian coordinate displacements from torsional rotations (and thus, from IC modes):

$$\Delta\vec{r}_{i,\alpha} = \sum_\alpha^N \vec{J}_{i,\alpha} v_i^\alpha. \tag{5.21}$$

We can also obtain torsional rotations from Cartesian displacements by using:

$$\Delta\Theta = (J^T MJ)^{-1} J^T M \Delta r. \tag{5.22}$$

5.2.5 Description of the Internal coordinate NMA-based algorithm

We have introduced a computationally affordable new IC NMA-based algorithm that aims at providing a fast traversal of the conformational space. The algorithm consists of two independent stages: the backbone perturbation and the side chain perturbation.

The backbone perturbation is implemented as an MC algorithm where each iteration comprises four steps:

1. **Calculation of the target angular increments:** a normal mode and a sense for the rotations is randomly selected. The mode is scaled so that the maximum rotation amplitude is inside a user-defined range. The user can chose to specify a maximum (a_{max}) and a minimum (a_{min}) value for the amplitude instead. In this case the maximum amplitude value is sampled from a truncated normal distribution with mean $(a_{min}+a_{max})/2$ and standard deviation $(a_{min}-a_{max})/4$. Normal modes are calculated only at the beginning of the first iteration.

2. **Application of the angular increments:** the geometry of the protein is updated using Choi's [214] quaternion method.

3. **Side chain relaxation:** the rotations in the previous step treat side chains as rigid bodies. This may introduce atom clashes that must be freed. To this end, the side chains with any atom involved in a steric clash are selected and minimized. Note that, as only side chains are minimized, the conformations with clashes involving backbone atoms will be discarded in the next step because of their high potential energy.

4. **Acceptance criterion:** the Boltzmann criterion is tested and the new conformation is accepted or rejected.

The side chain perturbation is again implemented as an MC algorithm where, at each iteration, a side chain is randomly selected and modified. To change the side chain, its rotatable bonds are found and a random increment is applied to each of them. The new side chain conformation will be accepted or rejected depending on the result of the Boltzmann criterion.

5.2.6 Alternative implementation

We also worked on a solution that allowed us to fuse the icNMA step with PELE original scheme [149] (i.e. performing an icNMA step plus the relaxation phase). As in the regular PELE algorithm, the last minimization must be constrained so that the backbone proposal is maintained. In this case, we have applied harmonic dihedral constraints ($U^c_\alpha(q_\alpha) = {}^k/_2(q_\alpha - q^0_\alpha)$) to a percentage of the most flexible torsions. Restricting the number of constraints makes it is easier for the minimizer to free backbone stress and it helps it to converge faster. The gradient and Hessian derivations have been calculated and translated to C using the symbolic algebra software Maxima [215]. This alternative was studied during the preliminary development stage providing analogous results to those of ccNMA (the protein was collapsing into a compact form). This lead us to conclude that minimizations had a role in the sampling bias.

5.3 Obtention of the best set of parameters

To compare IC and CC methods, we first need to obtain the set of parameters that maximizes the performance of both method simulations. To this end, we chose the most relevant parameters for each method and analyzed how their changes were affecting performance.

5.3.1 Characterization of the NMA step

We started by isolating the mode application procedure in both methods. The ccNMA step comprises two smaller substeps:

1. The calculation of the translation vectors, which depends on the C_α displacement factor parameter (*displacementFactor* defines the maximum displacement for the C_α atoms).

2. The application of these translations through a minimization. The "strength" or "intensity" of this minimization depends mainly on two parameters, the force constant of the spring pulling the C_α atoms (*steeringForce*) and the root mean square of the gradient (*MinimumRMS*) that modulates the convergence of the minimization.

In the icNMA case, the NMA step coincides with a whole iteration of the backbone perturbation stage, and is performed in two consecutive substeps:

Method	Movement amplitude	Minimization strength	
CC	displacementFactor (Å)	steeringForce (kT/Å)	MinimumRMS ($^{kcal}/_{mol\ Å}$)
	0.25, 0.66, 1.08, 1.5, 1.92	20, 40, 60, 80, 100	0.01, 0.05, 0.1
IC	displacementFactor (radians)	relaxMinimRmsg ($^{kcal}/_{mol\ Å}$)	
	0.02, 0.05, 0.075 ,0.1, 0.12, 0.15	0.01, 0.05, 0.1	

Table 5.2: Choice of parameters affecting the mode application step in the CC and IC methods, including the values that will be used in characterization tests.

1. The calculation of the torsional increments, which depends on the displacement factor parameter (*displacementFactor*, which, in turn, defines the maximum rotation of any backbone torsional angle).

2. The application of the rotations, which has no parameter dependencies.

3. The side chain relaxation step, which again depends on the root mean square of the gradient (*relaxMinimRmsg*).

Table 5.2 summarizes the parameters that govern both methods.
To characterize the results of each step, we have focused on three features:

- The C_α RMSD between the initial conformation and the conformation after the mode application step, which shows the extent of the backbone deformation.

- The increment of internal energy between the starting conformation and the conformation after the NMA step, which shows how likely this deformation would be.

- The time the step takes.

Ideally, a good set of parameters is the one that produces big RMSD displacements at a low energy cost.

To study the effect and relevance of our parameter choice, we performed several simulations at 3000 K with a cutoff distance for the EN of 9 Å , random selection of pure modes (which means that modes will not be combined) and different values of movement amplitude and minimization strength (see Table 5.2. We have used the c-Src kinase structure (PDB id: 1y57) as test system. This protein performs wide inter-domain conformational transitions as well as loop rearrangements involving the temporary creation of secondary structures. As we have seen in Section 3.5.1 and Section 1.5.3.3, the choice of the initial

conformation can give place to very different mode spaces, which has an effect on the conformational search. This is the reason why we have used two different starting conformations in this test: an open P-loop structure (with a 17.32 Å distance between CYS:277:CA and LEU:387:CA) and a semi-closed P-loop one (with 13.5 Å between the same atoms). Both of them come from previous PELE simulations, which means that they have been previously minimized. The temperature of the simulation ensures high acceptance rates and, therefore, a wider exploration of the conformational space (we have not taken into account the biophysical correctness of the exploration at this point).

We run a first set of short CC simulations (~100 steps) in order to determine which minimization strength parameter we should modify and, therefore, further delimit the number of parameters to analyze. As both *steeringForce* and *MinimumRMS* influence minimization, we decided to fix *steeringForce* to 20 $kT/$Å when changing *MinimumRMS*, and *MinimumRMS* to 0.05 $^{\text{kcal}}/_{\text{mol Å}}$ when changing *steeringForce*. These are commonly used values for these parameters.

The plots in Fig. 5.8 show a strong positive relationship between the *displacementFactor* parameter (colored clusters), the amount of deformation (RMSD) and the energy increment (ΔU). The RMSD averages are similar for both parameters and structures, while the average energy for the simulations where the *steeringForce* is being modified is, in general, higher.

However, the relationship of the *relaxMinimRmsg* or *steeringForce* with the studied features is not that clear. In order to gain more insight, we measured the association strength of these variables using the Spearman rank correlation. This coefficient can be calculated as

$$\rho = 1 - \frac{6\Sigma d_i^2}{n(n^2-1)}, \tag{5.23}$$

where d is the difference between ranks and n is the number of samples.

The Spearman rank correlation is a non-parametric test that works with ranked ordinal data with monotonic relationships and does not make any assumption about data distribution (e.g. Pearson product-moment correlation assumes distributions to be normal). We consider that coefficients between 0.10 and 0.29 represent a small association; coefficients between 0.30 and 0.49 represent a medium association; and coefficients above 0.50 represent a tight relationship. The results, summarized in Table 5.3, confirm that both minimization strength parameters play an important role in the final energy increment. Given this, and the fact that the changes in *steeringForce* produce larger energy averages, we decided to keep this parameter fixed to its default value in the next tests.

	Open		Closed	
	steeringForce	MinimumRMS	steeringForce	MinimumRMS
RMSD	0.044 (0.019)	-0.015 (0.421)	0.042 (0.019)	-0.052 (0.004)
ΔU	0.223 (<0.001)	0.180 (<0.001)	0.175 (<0.001)	0.163 (<0.001)

Table 5.3: Association strength of the studied parameters and simulation features. Each value has been colored depending on its category: Green for high association, yellow for medium association, orange for low association and red for no association.

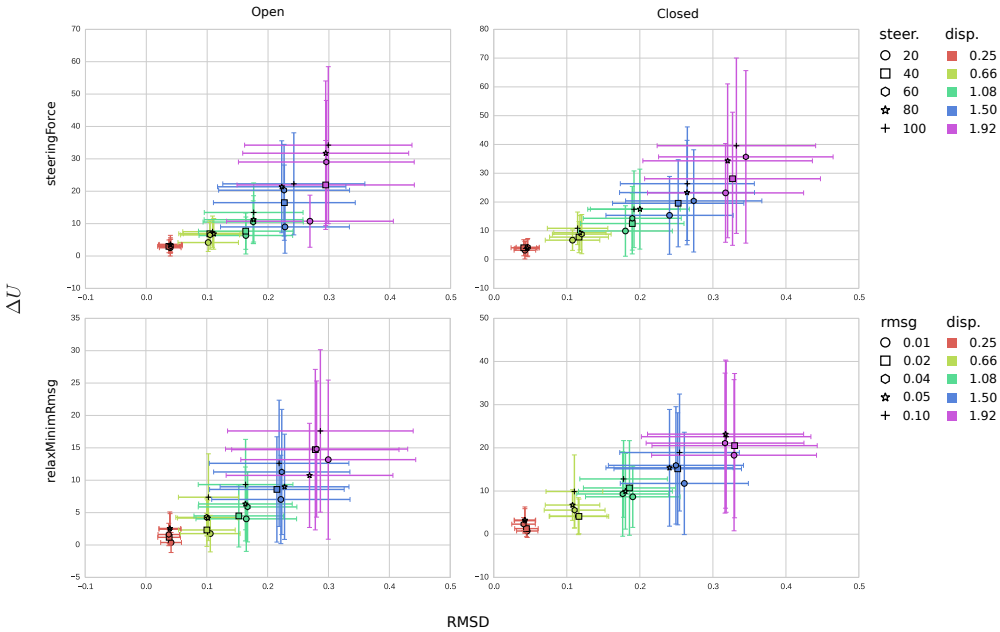


Figure 5.8: Plot showing the relationship of the two studied parameters (*steeringForce* and *MinimumRMS*) with the RMSD and energy increments of the Cartesian coordinate ANM step. Each point shows the average and standard deviation of the RMSD and energy increments for a given combination of parameters.

Having discarded to modify the *steeringForce* parameter, we proceeded to perform other longer simulations (~1500 steps) by varying the remaining parameters. Since the simulations starting from the closed structure looked more static, this time we started all simulations from the open structure. The reason for this behavior could be the higher overlap of the open conformation modes with the open to close transition. The simulations have been performed using temperatures between 300 K and 2568 K and the Spearman rank correlation for the (computational) step time, energy and RMSD increments (always of the NMA step) has been calculated (see Table 5.4).

		CC		IC	
T		ρ	p-value	ρ	p-value
300	RMSD \ ΔU	0.79	<0.001	0.69	<0.001
	Step time \ Displacement	0.563	<0.001	0.615	<0.001
	Step time \ Relax. strength	-0.591	<0.001	0.002	0.783
	RMSD \ Displacement	0.786	<0.001	0.78	<0.001
	RMSD \ Relax. strength	-0.057	<0.001	-0.003	0.594
	ΔU \ Displacement	0.656	<0.001	0.717	<0.001
	ΔU \ Relax. strength	0.214	<0.001	-0.005	0.405
583	RMSD \ ΔU	0.79	<0.001	0.63	<0.001
	Step time \ Displacement	0.548	<0.001	0.591	<0.001
	Step time \ Relax. strength	-0.598	<0.001	-0.006	0.274
	RMSD \ Displacement	0.803	<0.001	0.785	<0.001
	RMSD \ Relax. strength	-0.051	<0.001	0.001	0.895
	ΔU \ Displacement	0.66	<0.001	0.659	<0.001
	ΔU \ Relax. strength	0.213	<0.001	0.007	0.216
866	RMSD \ ΔU	0.79	<0.001	0.6	<0.001
	Step time \ Displacement	0.562	<0.001	0.588	<0.001
	Step time \ Relax. strength	-0.605	<0.001	0.001	0.808
	RMSD \ Displacement	0.794	<0.001	0.788	<0.001
	RMSD \ Relax. strength	-0.03	<0.001	-0.005	0.374
	ΔU \ Displacement	0.656	<0.001	0.629	<0.001
	ΔU \ Relax. strength	0.195	<0.001	0.011	0.059
1150	RMSD \ ΔU	0.79	<0.001	0.58	<0.001
	Step time \ Displacement	0.569	<0.001	0.618	<0.001
	Step time \ Relax. strength	-0.579	<0.001	0.002	0.753
	RMSD \ Displacement	0.799	<0.001	0.79	<0.001
	RMSD \ Relax. strength	-0.033	<0.001	0.005	0.372
	ΔU \ Displacement	0.65	<0.001	0.607	<0.001

	ΔU \ Relax. strength	0.182	<0.001	0.003	0.603
1432	RMSD \ ΔU	0.79	<0.001	0.56	<0.001
	Step time \ Displacement	0.543	<0.001	0.579	<0.001
	Step time \ Relax. strength	-0.609	<0.001	-0.016	0.006
	RMSD \ Displacement	0.787	<0.001	0.79	<0.001
	RMSD \ Relax. strength	-0.046	<0.001	-0.005	0.38
	ΔU \ Displacement	0.656	<0.001	0.589	<0.001
	ΔU \ Relax. strength	0.187	<0.001	0.001	0.822
2000	RMSD \ ΔU	0.75	<0.001	0.53	<0.001
	Step time \ Displacement	0.563	<0.001	0.559	<0.001
	Step time \ Relax. strength	-0.566	<0.001	0.003	0.608
	RMSD \ Displacement	0.782	<0.001	0.791	<0.001
	RMSD \ Relax. strength	-0.044	<0.001	0.001	0.884
	ΔU \ Displacement	0.615	<0.001	0.561	<0.001
	ΔU \ Relax. strength	0.198	<0.001	0.008	0.169
2568	RMSD \ ΔU	0.75	<0.001	0.51	<0.001
	Step time \ Displacement	0.57	<0.001	0.395	<0.001
	Step time \ Relax. strength	-0.556	<0.001	-0.046	<0.001
	RMSD \ Displacement	0.784	<0.001	0.792	<0.001
	RMSD \ Relax. strength	-0.047	<0.001	0.002	0.707
	ΔU \ Displacement	0.598	<0.001	0.53	<0.001
	ΔU \ Relax. strength	0.199	<0.001	-0.003	0.547

Table 5.4: Association strenght of the RSMD and energy increments, and step time between themselves and the chosen parameters.

The results show a strong association between the RMSD increment and the energy increment. This is somewhat expected, as big deformations can, in the CC case, distort the topology and, in the IC case, produce more steric clashes. It is remarkable, however, that the association strength is always smaller for the icNMA step, supporting our hypothesis that the icNMA method is able to deform the protein producing smaller energy increments. In this last case we see a slight anticorrelation with the temperature. This is surprising, since all these measures have been taken inside the NMA step and, thus, may be independent of the temperature.

RMSD increments have a strong association with the displacement magnitude, as expected, and no association with relaxation strength in both cases. The energy increment during the NMA step and the displacement magnitude

are strongly associated, the explanation behind this is the same that relates RMSD and energy increment. Energy increment and relaxation strength have a loose association in the ccNMA method. This relationship does not exist in the IC case, showing that early convergence should be enough to get rid of steric clashes. All these measures seem to be independent of the temperature.

The computational time needed to fulfil the NMA step is, in both cases, strongly related to the displacement magnitude. The reasons are different for each method: In the ccNMA step, the more distant the target positions are, the more difficult the convergence of the minimization becomes, as it has to fulfill the force field constraints plus the ANM constraints. In the icNMA case, the wider the rotation is, the easier it is to generate steric clashes, which are harder to relieve using the minimization.

Finally, we can observe that the computational time the ccNMA needs to perform a step is also strongly associated with the ANM minimization strength, while this association is almost nonexistent in the IC case, giving us more freedom to change the value of this parameter without time penalties.

5.3.2 Obtention of the best simulations and comparison with MD

One of the goals of this project is to evaluate to which extent the icNMA implementation improves PELE sampling capabilities. To this end, we have decided to compare simulations using both methods with a third reference method: MD. We have tested two different systems, ubiquitin (PDB id: 1UBQ) a small and very static globular protein, and a c-Src kinase (PDB id: 1Y57) as an example of a bigger and more flexible protein.

5.3.2.1 Best values for the parameters at 300 K

The value of the parameters used in a simulation can dramatically alter its outcome. As the icNMA-based method is new, and the ccNMA-based method is not commonly used at 300, we do not have a predefined set of default values for the parameters to use in the simulations. As a consequence, we have worked to obtain the best parameterizations in order to make the comparison fairer. Here, we will illustrate the procedure we followed for the parameterization of the c-Src kinase simulation.

First, we have fixed the parameters common to both methods e.g. the temperature was set to 300 K and the distance cutoff for the construction of the EN was set to 9 Å . Then, using the simulations we had already run with the parameters detailed in Table 5.2, we have calculated the values of the three indices

we were going to use to discriminate the best values for the parameters:

- The average RMSD increment of the ANM step, which gives us an idea of the extent of the deformations.
- The root mean square of the root mean square fluctuations (RMSF) for each simulation vs. the RMSF of the MD simulation. This can tell us if the flexibility of our simulations resemble that of the MD simulations.
- The acceptance, which we wanted to keep in the 20-40% range.

After applying the acceptance restriction, the list of possible parameterizations went down to three possibilities for each of the simulations (see Table 5.5) and we were able to choose a good set of parameters for the regular PELE simulations: 0.66 Å for the *displacementFactor* parameter and 0.1 for the *MinimumRMS* parameter. Note that, as results are so similar, we decided to select the higher value of *MinimumRMS*, as we knew it would shorten the execution time of each step.

For the icNMA method, we saw a high difference in acceptance between the simulations with *displacementFactor* equal to 0.05 rad and *displacementFactor* equal to 0.08 rad, indicating that our angular step choice was too big. To verify this, we refined the icNMA simulations using values of the *displacementFactor* between 0.05 rad and 0.08 rad. The optimum value turned to be around the 0.065 rad. As the method uses a maximum and minimum value for the rotations, we performed further tests using values around 0.065 rad. as the lower limit and an arbitrary maximum value of 0.02 rad. The best combination we found was the range from 0.07 rad to 0.14 rad.

Once we got the best parameter choice for the icNMA step, we still needed to parameterize the rotation amplitude range of the side chain perturbation step. To this end, we performed several simulations using different values for the maximum and minimum rotation and choosing those which yield acceptance values between 20 and 40%. The optimum range happened to be 0.02 to 0.024 radians.

5.4 Comparison with MD

We performed 12 independent 24 h simulations for each method and compared different measurements with MD. These measurements include:

Energy increments The increments of the potential energy due to the NMA step perturbation.

	Acceptance	RMS(RMSF)	RMSD	Disp. mag. (Å)	Rel. str.
CC	0.316	2.467	0.094	0.66	0.05
	0.308	2.492	0.098	0.66	0.01
	0.281	2.447	0.09	0.66	0.1
IC	0.334	2.618	0.126	0.5	0.05
	0.328	2.642	0.127	0.5	0.1
	0.298	2.651	0.126	0.5	0.01
IC	0.296	2.637	0.139	0.55	0.1
(refinement)	0.295	2.639	0.137	0.55	0.05
	0.29	2.635	0.138	0.55	0.01
	0.284	2.629	0.148	0.6	0.1
	0.264	2.63	0.149	0.6	0.01
	0.26	2.588	0.165	0.65	0.05
	0.249	2.608	0.168	0.65	0.01
	0.248	2.614	0.153	0.6	0.05
	0.245	2.612	0.164	0.65	0.1
	0.229	2.592	0.179	0.7	0.01

Table 5.5: Values for the parameters that produced simulations with acceptance between 20 and 40%.

SASA and radius of gyration The SASA measures the area of the protein accessible to the solvent, while the radius of gyration calculates the dispersion of the atoms to its center of mass. Both measurements are related to the compactness of the protein.

RMSF The RMSF is a measure of the fluctuation per residue (C_α atom). We expect that similar methods produce similar conformational ensembles and, therefore, similar RMSF profiles. However, it is not guaranteed that two different conformational ensembles have different RMSF profiles. That is why we must combine this measure with others in order to have a real picture of the ensembles/methods similarity.

Conformational space overlap We used a similar approach to the ones shown by Lyman *et al.* [216] and Lindorff-Larsen [217] to calculate the extension of the exploration in the conformational space overlap. This strategy consists in clustering the structures of all the methods together using a geometrical distance measure (RMSD). Then the square root of the Jensen-Shannon divergence of the cluster populations (converted to a probability distribution) is calculated, yielding the desired overlap value.

5.4.1 Advantages of the new method

We have observed that the ccNMA step is not able to generate energetically favorable proposals, i.e. proposals that maintain or lower the potential energy and thus will be accepted by means of a Boltzmann criterion. This is, indeed, the reason that makes the relaxation phase a necessary part of the PELE approach. The icNMA algorithm, however, is able to generate energetically favorable proposals ∼18-27% of the times and therefore does not need the computationally costly relaxation step. As a consequence, the icNMA-based algorithm is able to generate MC proposals faster than the ccNMA approach (∼5-7x), while keeping similar RMSD increments.

The measurements of the SASA and radius of gyration for both algorithms reveal a tendency towards generating more compact structures than MD. This is not unexpected, as our methods use implicit solvent, which is known to produce a bias in favor of compact structures [119, 218, 219], while our reference MD simulations were run in explicit solvent. There is a remarkable difference between the results for icNMA and ccNMA, being the former in better accordance with MD. The increased compaction of the ccNMA-based algorithm seems to be caused by the repeatedly use of minimizations, which enforce the solvent bias. The absence of backbone minimizations in the icNMA method limits the severity of the bias.

We also studied the relative fluctuations of the residues by scaling the RMSF profiles and superimposing them. The RMSF profiles of the icNMA method show, in general, good agreement with MD. The ccNMA method shows a poorer performance in the ubiquitin case, specially in the β2-α-helix and the β4-β5 loop regions.

The study of the conformational space overlap shows that, in general, the icNMA-based method is populating regions of the conformational space closer to MD than to the ccNMA-method. Besides, we performed additional analyses of the sampling of the c-Src kinase P-loop. Correctly sampling this loop is of crucial importance as it is involved in the binding mechanisms of the kinase [220]. We measured the distances between the atoms CYS:277:CA and LEU:387:CA, which is related to the inter-domain distance and the P-loop sampling. The icNMA algorithm is able to sample a similar range of distances to MD, however, ccNMA rapid compaction prevents it from obtaining similarly good results. This highlights the advantage of the icNMA method, which is less prone to get trapped in energy minima corresponding to compact structures.

5.5 Limitations of the new method and future perspectives

In general, we observed that the RMSF baselines of the NMA-based methodologies is always lower MD baselines. This points to the main limitations of the new method: the lack of anharmonic backbone movements and mapping higher (local) frequency modes, which reduces the amplitude of fluctuations. We wonder if this could be overcome by using IC Principal Component Analysis (PCA) modes, which are known to have a higher anharmonic component [143]. Also, both methods show high RMSD differences with the structures coming from the MD trajectories. This differences are more evident in the ubiquitin case, where the flexibility of the protein is concentrated on the loops connecting the secondary structure. The cause of these RMSD differences reflects, indeed, the inability of our set of low-frequency NMA modes to model the high-frequency movements occurring in these loops.

We are currently working to yet lower the execution times and keep improving the simulation flexibility. Our aim is to make this method a good alternative to current VHTS software by finding a good trade off between simulation detail and speed. Moreover, by coupling this procedure with a ligand perturbation step, as done in the PELE algorithm, local fluctuations (not present in the NMA) can be recovered through the response of the protein to the ligand move.

We believe that the improvements in simulation reliability shown here can suppose a gain in the long term. Although it is true that it involves a considerable increase in the calculation times, it can save time (and money) at a later stage of the drug discovery process by the early elimination of false positives and negatives.

5.6 Efficient and reliable analysis of huge conformational ensembles

In the computational biology field, the evolution of hardware during the last decade has allowed tools to run faster, thus making them able to produce larger output in the same amount of time. Nowadays, the efficient storage and analysis of these increasingly big results can become a problem.

This is indeed our case, since one of our objectives is the development of faster conformational sampling tools, capable of being applied in VHTS. That is why we have focused on the improvement of the efficiency and robustness of the analysis of large ensembles of biomolecular structures.

5.6.1 Efficient calculation of collective superimposition operations

Collective superimposition operations are fundamental routines in structural biology related analysis methods and can become a bottleneck if data sets are large. We have implemented an efficient alternative to performing these operations and RMSD calculations: pyRMSD [221]. We have focused on three common use cases:

- Superimposition of a given conformation vs. the others in the ensemble (e.g. the first frame of a trajectory is usually compared to the others in order to calculate an RMSD profile).

- Pairwise superimposition of all the conformations of the ensemble. A pairwise superimposition of all conformations of the ensemble would be necessary to build an RMSD matrix to be used by a clustering algorithm

- Iterative superimposition of all the conformations of the ensemble. All conformations can be iteratively superimposed before calculating the covariance matrix for a PCA.

Python is an object-oriented interpreted language that seems to be currently gaining momentum among researchers. This may be due to the great availability of scientific libraries or to the possibility of creating software prototypes without effort.

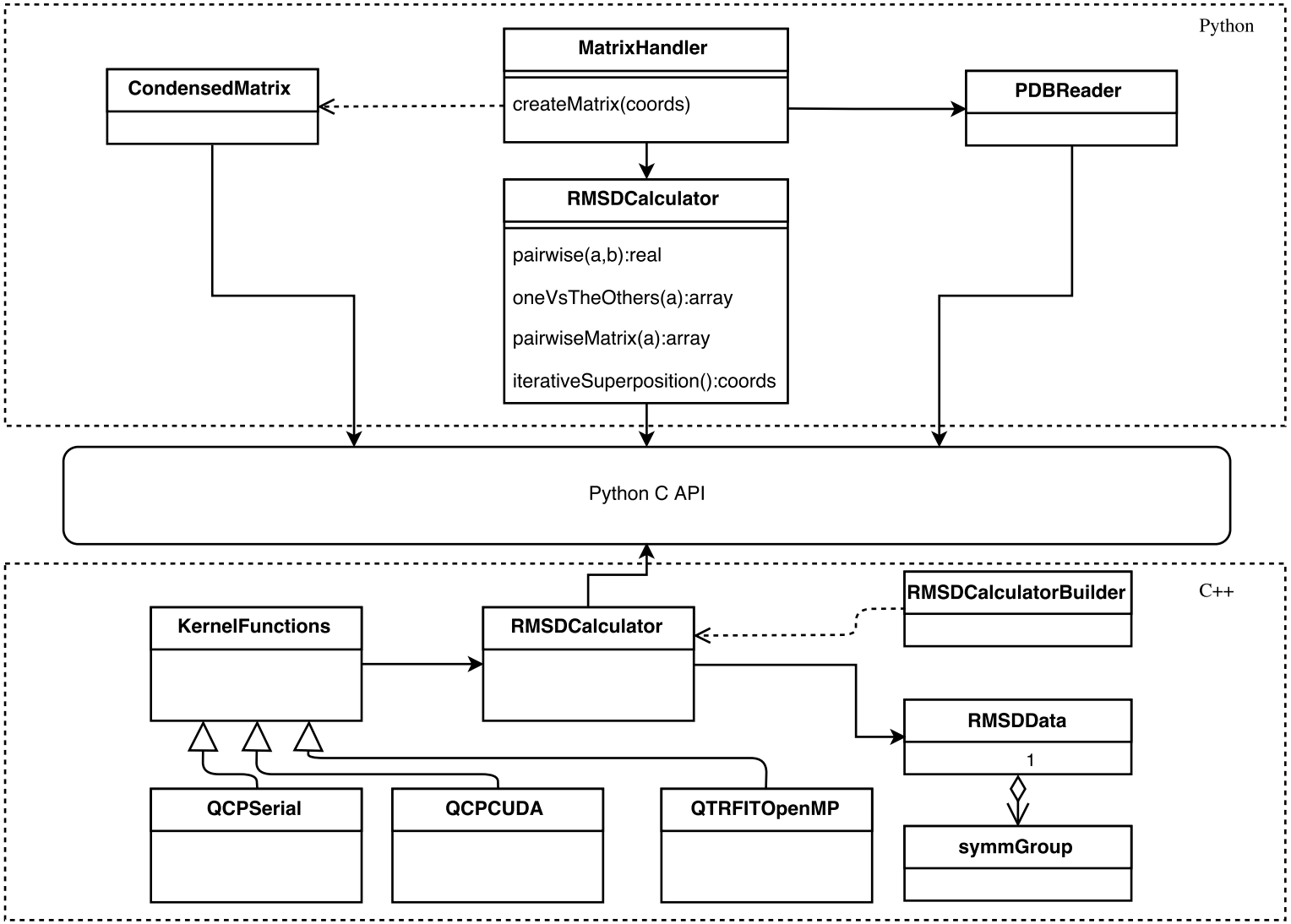


Figure 5.9: Pseudo-UML (Unified Modelling Language) diagram showing some key classes of pyRMSD as well as the three-layer design (Python classes, Python C interface and C++ classes).

5.6.2 Superimposition algorithms

pyRMSD has been designed as a Python library implementing a three-layer structure (see Fig. 5.9). The first layer is written in pure Python and contains high-level objects and functions. The second layer describes the C Python API interface, which links the low-level functions of the third layer with the high-level functionalities of the first one. The third layer is a full C++ library implementing the superimposition and RMSD calculation functions. Thanks to this structure, pyRMSD can be included effortlessly in other Python or C++ projects, ensuring the reuse of the code.

pyRMSD implements Kabsch's algorithm [222], Heisterberg's algorithm (aka QTRFIT) [223] and Theobald's algorithm (QCP) [224]. These three methods try to solve the superimposition problem by different means, but with the common goal of finding a rotation matrix (or equivalent quaternion) that minimizes the error function:

$$e^2 = \frac{1}{n} \sum_n \left| x_n - U y_n \right|^2 . \tag{5.24}$$

The inclusion of these three algorithms in one package obeys two reasons: the need to compare algorithm performance, and the necessity to provide users with an alternative in case they need to overcome any of the problems associated with these algorithms (e.g. Kabsch's can potentially produce rotoreflection matrices [225] and QCP is known to have bad convergence).

5.6.3 Parallelization and performance

The core functions of the third layer have been parallelized in order to boost performance. All three algorithms have a serial and OpenMP version to take advantage of multicore CPUs. In addition, QCP also implements a CUDA version that can run in Nvidia GPUs with both single and double floating point precision. This is has been the first open source CUDA implementation of this algorithm so far.

When testing the "one conformation versus the others" use case in a 30k snapshots trajectory, the faster algorithm showed a ~2.6x speedup using OpenMP with six threads. Improvements are more noticeable when testing the calculations of RMSD matrices, with speedups ranging from 5x (OpenMP) to 11x faster (CUDA using the "in-memory matrix" alternative).

Furthermore, the comparison of pyRMSD with g_rmsd [157], a RMSD matrix calculation specialized piece of software, shows that the former is more than four times faster. We have also implemented several RMSD matrix calcu-

lation examples using non-specific software (see Supplementary materials S6 of the publication). The best-performing function uses Prody, a well consolidated Python package, and is 50x slower than pyRMSD.

Our profilings have revealed that, when using distance matrices, the contribution of the matrix access time starts to become prominent if the superimposition bottleneck is eliminated. This behaviour can be explained based on Python's inherent overhead in array element access operations, e.g. Python checks internally that these indices are integers or that they are pointing to a legal position of the array. As these checks are an automatic part of the runtime, we needed to create the "CondensedMatrix" object. This is a C Python object that stores a square symmetric matrix in a memory efficient way and bypasses indexing checks to decrease access overhead to almost zero, adding a 6x automatic speed up to any function using it.

5.6.4 Distribution

pyRMSD is distributed as open source software and is maintained in a public GitHub repository[12]. It can be easily installed using customary Python installation methods (setup.py and pip remote installation). We have also added a new makefile-like setup script that allows defining different hardware parameters and environments so that users can make the most of their machine architectures. This installation script also overcomes some current Python limitations related to CUDA code compilation.

[12] `https://github.com/victor-gil-sepulveda/pyRMSD`

5.7 A reliable cluster analysis protocol

Cluster analysis is a powerful non-supervised analysis technique which aims to group the elements of a data set so that its underlying structure gets unveiled. This allows discovering properties of data that would be impossible to obtain through other means.

It has been extensively used in several fields, and computational biology is not an exception. For instance, it is used in population analysis of ensembles [226], to summarize simulation output [227, 228], finding native-like structures in homology modeling refinement processes [69] or as a key part of Markov State Model (MSM) analysis [229].

5.7.1 Looking for the best clustering

The results of a cluster analysis heavily depend on the choice of the algorithm and its parameters (see Fig. 5.10). Cluster analysis is also very sensitive to the distance metric used. It is known, for instance, that RMSD sensibility can make clustering more difficult, especially when structural dissimilarity is high. Also, the so-called "curse of dimensionality" can suppose a problem here. Consequently, using cluster analysis as a black box can be dangerous: as cluster analysis is usually part of other more complex analyses or algorithms, erroneous partitions would directly compromise all derived results.

A cluster analysis algorithm is usually considered to be good if it is able to classify the data set so that each of them is more similar to the elements of their own group than to the elements in the other groups. However, this definition does not always apply. In Fig. 5.10 we have used two algorithms and two parameter sets, thus obtaining four different clusterings. The two upper clusterings show perfect equipartitions of the space; the two lower clusters are able to capture the circular shape to some extent. This leads to a question: which is the correct result? For instance, the first solution would be a good outcome if we embrace the definition above, while the second one would be especially useful in a computer vision scenario where the shape of objects must be identified. The best clustering is, indeed, the one that best fits the user's goals [169].

5.7.2 pyProCT

We have created pyProCT [230] in order to have a robust cluster analysis method that can be used without understanding how cluster analysis algorithms work. The pyProCT work flux starts with users defining a working hypothesis by

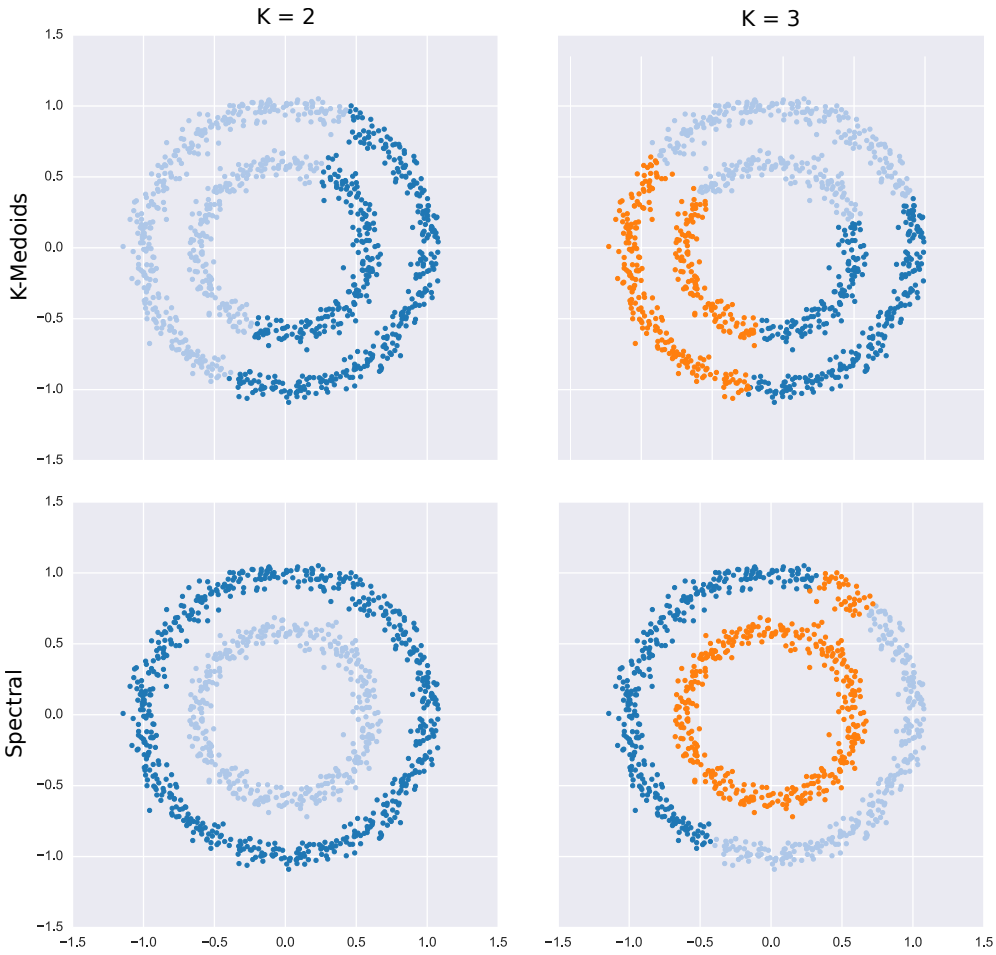


Figure 5.10: Four cluster analysis have been performed over the same data set. Results can change dramatically depending on the algorithm (k-medoids or spectral clustering here) and parameters used (k = 2 or k = 3).

using their domain expertise and their knowledge about the problem they are facing. This hypothesis describes the results that they would consider useful using just a few parameters, and obliges them to identify the specific goals of their clustering efforts. For instance, a user trying to analyze an MD trajectory of a system he or she is studying may know the following: the approximate number of expected clusters, the expected size of populations, or whether the method is prone to produce "noisy" ensembles.

The second part of the hypothesis is built around the definition of one or more scoring criteria. These are based on simple concepts, such as the degree of cluster separation or cohesion. The hypothesis will be used by pyProCT to perform a clustering exploration with five different clustering algorithms (K-means, Spectral clustering, DBSCAN, hierarchical complete linkage and GROMOS), and different sets of automatically generated parameters. Thanks to the previously defined hypothesis and scoring criteria, the software will choose the result that better fulfills the user's expectations.

5.7.3 Hypothesis refinement

It is very likely that the first hypothesis does not define the user's goals completely. Users may gain knowledge from result inspection, and this knowledge can be used to refine the hypothesis and the cluster again. Although this step is not mandatory, the preferred and more reliable way of working with pyProCT is performing these iterative hypothesis refinement cycles.

5.7.4 Software flow detail

We have implemented pyProCT as a flow of 5 stages:

1. Distance matrix calculation. All clustering algorithms need to calculate the distances (or similarities) of the elements in the data set. In the case of protein conformations, this calculation implies a costly superimposition. Therefore, precalculating the distances can be advantageous as the superimposition step can be time-consuming. pyProCT allows the use of RMSD and Euclidean distances. RMSD is calculated using pyRMSD and extends it by adding a way to define groups with symmetries, as well as automatic matching of symmetric chains in oligomers in order to obtain the minimum RMSD value every time. The Euclidean distance option calculates distances between the center of mass of the selected groups.

2. Algorithm parameterization. A set of parameters is calculated using a different methodology for each algorithm based on the working hypothesis.

3. Cluster analysis. All the algorithms (or a user-defined subset) are used in combination with the parameterizations found in the previous stage. This produces an ensemble of clustering solutions that will be filtered in order to avoid repetitions and solutions that do not agree with the user's hypothesis.

4. Remaining clusterings are scored using both hypothesis and the defined scoring criteria.

5. The best result is chosen and analyzed

pyProCT can take advantage of parallel architectures thanks to pyRMSD. As the workflow is composed of well-defined independent tasks, it has been possible to parallelize it using a naive parallelization scheme. A scheduler has been added so that tasks can be distributed to more than one processor (at node level, using native Python parallel functions) or more than one node (at cluster level, using MPI). A new scheduler type using pyCOMPSs [231] has recently been developed [150]. pyCOMPSs is an annotation-based parallel programming model for Python based on COMPSs [232]. The inclusion of this scheduler improves the load balance and opens the door to the execution of pyProCT in cloud architectures.

5.7.5 Scoring criteria

A crucial step in the craft of the working hypothesis is the definition of the scoring criteria. Each scoring criterion is a weighted sum of a set of clustering quality functions. Only internal clustering validation indices (ICVs) have been used, such as the Silhouette index, Cohesion or Separation, as there is not a correct clustering available to use as a reference. The implemented ICVs have been defined and discussed in Section 3.7.2. By defining more than one criterion, users can prepare the analysis for more than one scenario (e.g. the example in Section 3.7.6 uses one criterion that defines the common description of cluster analysis and another criterion that defines it in terms of graph components). In this case, the scores for each criterion will be normalized and the best clustering will be the one with the higher score for any of the criteria.

5.7.6 Use cases

Two main use cases have been presented:

- Cluster analysis: We have used pyProCT to analyze two different synthetic conformational ensembles. Populations, representatives, and global and per-cluster RMSFs have been automatically extracted from these ensembles. We have also used pyProCT to analyze a DNA-ligand simulation where ligands were in the bulk of the solvent most of the time. We have allowed pyProCT to treat them as noise, which lead us to obtain the clusters of the interaction sites. The clusters found were in good agreement with the ones obtained using MSM analysis in a prior work [233].
- Redundancy elimination: Storing data, and especially conformational ensembles, is becoming an issue of concern. We have introduced a method that is able to eliminate redundant conformations while preserving ensemble cluster populations. This makes it possible to process data sets whose distance matrix can not be fitted in memory. We have tested it with a ~100,000 elements trajectory and a ~1,000,000 elements ensembles with good results (see Section 3.2 of the article, and Section 3.7.5 for more information). The method works by iteratively partitioning the data set, cluster analyzing the partitions order to eliminate redundancy, and merging the results, until the desired number of frames is reached.

5.7.7 Graphical User Interface (GUI)

pyProCT includes a wizard-like html-based GUI that can assist users in order to generate a correct execution script, launch pyProCT executions and visualize results. The result viewer processes pyProCT's result files and shows them in a more understandable way making it a very useful tool for hypothesis refinement.

5.7.8 Distribution

pyProCT is a Python standalone program that can also be integrated into other projects as a library. It is distributed as open source software and is maintained in a public GitHub repository[13]. It supports the setup.py installation method and can also be installed using the pip package manager (which is the most convenient way of installing it as it handles package dependencies automatically). pyProCT is far from being finished; code refactorings and new features

[13] `https://github.com/victor-gil-sepulveda/pyProCT`

are included regularly (e.g. it has recently been upgraded to use numeric data sets).

6

Conclusions

6.1 Technical improvement of PELE

– We have succeeded in implementing all the core functionalities of PELE using C++. The use of modern software engineering techniques has allowed us to better organize the code base and to introduce a test subsystem. Nowadays, PELE++ has become a piece of software which is easier to modify, understand, and extend. It is also more robust and reliable. The rewriting the code has helped us to overcome some of its previous technical limitations, such as the restrictions on the size of the systems. Also, it has allowed us to extend PELE with new solvent models, force fields, and types of biomolecules.

– The rewriting has make it possible to adapt the code in order to take advantage of new parallel architectures and accelerators. The parallelization of the non-covalent interactions and solvent model related functions, which represent hot spots in our performance analyses, has yielded promising speedup results. These new pieces of code are currently being integrated in PELE++.

– PELE++ is, nowadays, the first choice for our research group. The new code has opened the doors to more ambitious developments involving machine learning and virtual reality. Also, a GUI for PELE++ is currently being created.

– PELE++ has seriously been considered for pharmaceutical R&D, which illustrates the success of the project. As a matter of fact, it is currently

being tested in the Department of Medicinal Chemistry of the renowned pharmaceutical company AstraZeneca.

6.2 Algorithmic improvement of PELE

– We have proposed a new method using torsional normal mode analysis to improve the conformational sampling of PELE. We have found that internal coordinates normal modes are more collective and robust to the changes of the elastic network.

– The method is able to produce more energy favorable perturbations than the current ANM-based strategy. This allows us to run simulations at 300 K without the need of complex relaxation protocols. As a consequence, the overall computational performance of the sampling is significantly improved (~5-7x). We are currently working on the methodology to yet lower the execution times and further improving the simulation of flexibility. Thanks to the new methodology, we are attaining a good trade-off between speed and detail that will turn PELE into a good alternative to current VHTS software.

– The new internal coordinates-based methodology is able to capture the flexibility of the backbone better than the old method. The relative magnitudes of these fluctuations correlate well with those of our (explicit solvent) MD reference simulations.

– The new method better preserves the volume of the protein during the simulation, thus overcoming the tendency of PELE to produce compact structures in certain situations.

– Finally, the inter-domain movements of c-Src kinase obtained with this method are closer to those obtained with MD.

6.3 Efficient and reliable analysis of large conformational ensembles

– We have developed a Python package which is able to perform collective superposition (and RMSD) operations of conformational ensembles one order of magnitude faster than the serial version.

- We have developed pyProCT, a cluster analysis software which is able to provide good results without any knowledge of cluster analysis techniques.

- pyProCT can be successfully used in the most common use cases: to retrieve clusters from ensembles of conformations (using RMSD) and to retrieve ligand clusters (using the distance to the center of mass). As requested by users, it isnow able to load and analyze other types of data, such as numeric arrays, allowing it to perform tasks beyond its original scope.

- We also present a novel application to reduce the size of huge conformational ensembles by eliminating redundant structures.

- pyProCT has already been used in a good number of biomedical publications for the analysis of results. It is currently being applied to improve PELE ligand sampling performance. Besides, it has been used in a Partnership for Advanced Computing in Europe (PRACE) project aiming at finding innovative ways to visually represent clusters of protein conformations.

- The code and compiled versions of both packages are available to the whole scientific community in free-access repositories.

Bibliography

1. WHO. *World Health Organization : Global Health Observatory (GHO) data* World Health Organization : Global Health Observatory (GHO) data. <http://www.who.int/gho/mortality_burden_disease/life_tables/situation_trends/en/> (2015).
2. Ostwald, D. A. & Knippel, J. *Measuring the Economic Footprint of the Pharmaceutical Industry* 2013.
3. EFPIA. *The Pharmaceutical Industry in Figures* 2014.
4. PhRMA. *2015 biopharmaceutical research industry profile* 2015.
5. Kristopher Hult & Tomas Philipson. *Should Investors Pay Attention To The Alleged Productivity Crises In Pharma?* Forbes. <http://www.forbes.com/sites/tomasphilipson/2015/04/03/should-investors-pay-attention-to-the-alleged-productivity-crises-in-pharma/> (2015).
6. CDER. *2013 Novel New Drugs* 2014.
7. DiMasi, J., Hansen, R. W. & Grabowski, H. G. The price of innovation: new estimates of drug development costs. *J. Health. Econ.* **22,** 151–185 (2003).
8. Goozner, M. The $800 million pill: The truth behind the cost of new drugs. *J. Clin. Invest.* **114,** 1182 (2004).
9. Paul, S. M. *et al.* How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nat. Rev. Drug Discovery* **9,** 203–214 (2010).

10. Light, D. W. & Warburton, R. Demythologizing the high costs of pharmaceutical research. *Biosocieties* **6,** 34–50 (2011).

11. Mestre-Ferrandiz, J., Sussex, J. & Towse, A. *The R&D Cost of a New Medicine* Monograph. 2012.

12. TIBCO. *Transforming the Drug Development Process* (TIBCO).

13. Peter Gwynne & Gary Heebner. *Pharmaceutical and biopharmaceutical firms need to reduce the cost and lead time of drug development. A range of recently developed technologies makes that goal possible.* <http://www.sciencemag.org/site/products/ddbt_0207_Final.xhtml> (2015).

14. PhRMA. *2011 annual report* 2011.

15. O'Connor, A. *Football - By the Numbers* <http://lillypad.lilly.com/entry.php?e=1424> (2016).

16. DiMasi, J. A., Grabowski, H. G. & Hansen, R. W. The Cost of Drug Development. *N. Engl. J. Med.* **372,** 1972–1972 (2015).

17. Mullin, R. *Cost to Develop New Pharmaceutical Drug Now Exceeds $2.5B* Scientific American. <http://www.scientificamerican.com/article/cost-to-develop-new-pharmaceutical-drug-now-exceeds-2-5b/> (2016).

18. Dreyfus, N. *Keeping the Lid on R&D Costs* Drug Discovery & Development. <http://www.dddmag.com/articles/2014/06/keeping-lid-r-d-costs> (2016).

19. Hughes, E. F., Michael Hu, Karl Schultz, Jack Sheu & Daniel Tschopp. *The Innovation Gap in Pharmaceutical Drug Discovery & New Models for R&D Success* 2007.

20. Chong, C. R. & Sullivan, D. J. New uses for old drugs. *Nature* **448,** 645–646 (2007).

21. Cutler, P. & Voshol, H. Proteomics in pharmaceutical research and development. *Proteomics Clin. Appl.* **9,** 643–650 (2015).

22. Grenet, O. Significance of the human genome sequence to drug discovery. *Pharmacogenomics J.* **1,** 11–12 (2001).

23. Hopkins, A. L. & Groom, C. R. The druggable genome. *Nat. Rev. Drug Discovery* **1,** 727–730 (2002).

24. Garnier, J.-P. *Rebuilding the R&D Engine in Big Pharma* Harvard Business Review. <https://hbr.org/2008/05/rebuilding-the-rd-engine-in-big-pharma> (2015).

25. PhRMA. *2013 biopharmaceutical research industry profile* 2013.

26. Austin, D. H. *Research and Development in the Pharmaceutical Industry* 1 vols. (Congressional Budget Office, 2006).

27. Tollman, P., Guy, P., Altshuler, J., Flanagan, A. & Steiner, M. *A Revolution in R&D: How Genomics and Genetics Are Transforming The Biopharmaceutical Industry* 2001.

28. Dror, O., Schneidman-Duhovny, D., Inbar, Y., Nussinov, R. & Wolfson, H. J. Novel Approach for Efficient Pharmacophore-Based Virtual Screening: Method and Applications. *J. Chem. Inf. Model.* **49,** 2333–2343 (2009).

29. Clark, R. Prospective Ligand- and Target-Based 3D QSAR: State of the Art 2008. *Curr. Top. Med. Chem.* **9,** 791–810 (2009).

30. Bajorath, J. Integration of virtual and high-throughput screening. *Nat. Rev. Drug Discovery* **1,** 882–894 (2002).

31. Berman, H. M. *et al.* The Protein Data Bank. *Nucleic Acids Res.* **28,** 235–242 (2000).

32. Tanrikulu, Y. & Schneider, G. Pseudoreceptor models in drug design: bridging ligand- and receptor-based virtual screening. *Nat. Rev. Drug Discovery* **7,** 667–677 (2008).

33. Schaerfe, C. *A Critical Assessment of the Impact of Computational Methods on the Productivity in Pharmaceutical Research and Development* Master Thesis (University of Warwick, 2011).

34. Gil-Redondo, R. *et al.* VSDMIP: virtual screening data management on an integrated platform. *J. Comput. Aided Mol. Des.* **23,** 171–184 (2009).

35. Prathipati, P. & Mizuguchi, K. Integration of Ligand and Structure Based Approaches for CSAR-2014. *J. Chem. Inf. Model.* (2015).

36. Le Guilloux, V., Schmidtke, P. & Tuffery, P. Fpocket: An open source platform for ligand pocket detection. *BMC Bioinformatics* **10,** 168 (2009).

37. Huang, B. & Schroeder, M. LIGSITEcsc: predicting ligand binding sites using the Connolly surface and degree of conservation. *BMC Struct. Biol.* **6,** 19 (2006).

38. Tuffery, P. & Derreumaux, P. Flexibility and binding affinity in protein-ligand, protein-protein and multi-component protein interactions: limitations of current computational approaches. *J. R. Soc. Interface* **9,** 20–33 (2012).

39. Kuntz, I. D., Blaney, J. M., Oatley, S. J., Langridge, R. & Ferrin, T. E. A geometric approach to macromolecule-ligand interactions. *J. Mol. Biol.* **161,** 269–288 (1982).

40. Morris, G. M., Goodsell, D. S., Huey, R. & Olson, A. J. Distributed automated docking of flexible ligands to proteins: parallel applications of AutoDock 2.4. *J. Comput. Aided Mol. Des.* **10,** 293–304 (1996).

41. Morris, G. M. *et al.* AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility. *J. Comput. Chem.* **30,** 2785–2791 (2009).

42. Jones, G., Willett, P. & Glen, R. C. Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *J. Mol. Biol.* **245,** 43–53 (1995).

43. Friesner, R. A. *et al.* Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy. *J. Med. Chem.* **47,** 1739–1749 (2004).

44. Rarey, M., Kramer, B., Lengauer, T. & Klebe, G. A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.* **261,** 470–489 (1996).

45. Yun, M.-R., Lavery, R., Mousseau, N., Zakrzewska, K. & Derreumaux, P. ARTIST: An activated method in internal coordinate space for sampling protein energy landscapes. *Proteins: Struct., Funct., Bioinf.* **63,** 967–975 (2006).

46. Abagyan, R., Totrov, M. & Kuznetsov, D. ICM - A new method for protein modeling and design: Applications to docking and structure prediction from the distorted native conformation. *J. Comput. Chem.* **15,** 488–506 (1994).

47. Jain, A. N. Surflex: fully automatic flexible molecular docking using a molecular similarity-based search engine. *J. Med. Chem.* **46,** 499–511 (2003).

48. Mobley, D. L. & Dill, K. A. Binding of Small-Molecule Ligands to Proteins: "What You See" Is Not Always "What You Get". *Structure* **17,** 489–498 (2009).

49. Weis, A., Katebzadeh, K., Soederhjelm, P., Nilsson, I. & Ryde, U. Ligand Affinities Predicted with the MM/PBSA Method: Dependence on the Simulation Method and the Force Field. *J. Med. Chem.* **49,** 6596–6606 (2006).

50. Moustakas, D. T. Application of docking methods to structure-based drug design. *RSC Biomol. Sci.* 155–180 (2007).

51. Totrov, M. & Abagyan, R. Flexible ligand docking to multiple receptor conformations: a practical alternative. *Curr. Opin. Struct. Biol.* **18,** 178–184 (2008).

52. De Beer, S., Vermeulen, N. & Oostenbrink, C. The Role of Water Molecules in Computational Drug Design. *Curr. Top. Med. Chem.* **10,** 55–66 (2010).

53. Doman, T. N. *et al.* Molecular docking and high-throughput screening for novel inhibitors of protein tyrosine phosphatase-1B. *J. Med. Chem.* **45,** 2213–2221 (2002).

54. Hartshorn, M. J. *et al.* Diverse, high-quality test set for the validation of protein-ligand docking performance. *J. Med. Chem.* **50,** 726–741 (2007).

55. Verkhivker, G. M. *et al.* Complexity and simplicity of ligand-macromolecule interactions: the energy landscape perspective. *Curr. Opin. Struct. Biol.* **12,** 197–203 (2002).

56. Zavodszky, M. I. & Kuhn, L. A. Side-chain flexibility in protein-ligand binding: The minimal rotation hypothesis. *Protein Sci.* **14,** 1104–1114 (2005).

57. Fischer, E. Einfluss der Configuration auf die Wirkung der Enzyme.(1894) Ber. *Dtsch. Chem. Ges.* **27,** 2984.

58. Koshland, D. E. Application of a Theory of Enzyme Specificity to Protein Synthesis. *Proc. Natl. Acad. Sci. U.S.A.* **44,** 98–104 (1958).

59. Monod, J., Wyman, J. & Changeux, J. P. On the nature of allosteric transitions: a plausible model. *J. Mol. Biol.* **12,** 88–118 (1965).

60. Rubin, M. M. & Changeux, J. P. On the nature of allosteric transitions: implications of non-exclusive ligand binding. *J. Mol. Biol.* **21,** 265–274 (1966).

61. Wright, P. E. & Dyson, H. J. Linking folding and binding. *Curr. Opin. Struct. Biol.* **19,** 31–38 (2009).

62. Wlodawer, A. & Vondrasek, J. Inhibitors of HIV-1 protease: a major success of structure-assisted drug design. *Annu. Rev. Biophys. Biomol. Struct.* **27,** 249–284 (1998).

63. Bystroff, C. & Kraut, J. Crystal structure of unliganded Escherichia coli dihydrofolate reductase. Ligand-induced conformational changes and cooperativity in binding. *Biochemistry* **30,** 2227–2239 (1991).

64. Wilson, D. K., Tarle, I., Petrash, J. M. & Quiocho, F. A. Refined 1.8 A structure of human aldose reductase complexed with the potent inhibitor zopolrestat. *Proc. Natl. Acad. Sci. U.S.A.* **90,** 9847–9851 (1993).

65. Erickson, J. A., Jalaie, M., Robertson, D. H., Lewis, R. A. & Vieth, M. Lessons in molecular recognition: the effects of ligand and protein flexibility on molecular docking accuracy. *J. Med. Chem.* **47,** 45–55 (2004).

66. Yildirim, I., Stern, H. A., Tubbs, J. D., Kennedy, S. D. & Turner, D. H. Benchmarking AMBER Force Fields for RNA: Comparisons to NMR Spectra for Single-Stranded r(GACC) Are Improved by Revised χ Torsions. *J. Phys. Chem. B* **115,** 9261–9270 (2011).

67. Cino, E. A., Choy, W.-Y. & Karttunen, M. Comparison of Secondary Structure Formation Using 10 Different Force Fields in Microsecond Molecular Dynamics Simulations. *J. Chem. Theory Comput.* **8,** 2725–2740 (2012).

68. Lange, O. F., van der Spoel, D. & de Groot, B. L. Scrutinizing Molecular Mechanics Force Fields on the Submicrosecond Timescale with NMR Data. *Biophys. J.* **99,** 647–655 (2010).

69. Raval, A., Piana, S., Eastwood, M. P., Dror, R. O. & Shaw, D. E. Refinement of protein structure homology models via long, all-atom molecular dynamics simulations. *Proteins* **80,** 2071–2079 (2012).

70. Lin, T.-L., Vammi, S. K. & Song, G. *Evaluating the Quality of Conformation Sampling Methods Using Experimental Residual Dipolar Coupling Data* in *Proceedings of the 2Nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine* (ACM, New York, NY, USA, 2011), 514–518.

71. Karplus, M. & McCammon, J. A. Molecular dynamics simulations of biomolecules. *Nat. Struct. Mol. Biol.* **9,** 646–652 (2002).

72. Dodson, G. G., Lane, D. P. & Verma, C. S. Molecular simulations of protein dynamics: new windows on mechanisms in biology. *EMBO Rep.* **9,** 144–150 (2008).

73. Dror, R. O., Dirks, R. M., Grossman, J. P., Xu, H. & Shaw, D. E. Biomolecular Simulation: A Computational Microscope for Molecular Biology. *Annu. Rev. Biophys.* **41,** 429–452 (2012).

74. Fine, R., Dimmler, G. & Levinthal, C. FASTRUN: a special purpose, hardwired computer for molecular simulation. *Proteins* **11,** 242–253 (1991).

75. Shinjiro Toyoda, H. M. Development of MD Engine: High-speed accelerator with parallel processor design for molecular dynamics simulations. *J. Comput. Chem.* **20,** 185–199 (1999).

76. Taiji, M. *et al. Protein Explorer: A Petaflops Special-Purpose Computer System for Molecular Dynamics Simulations* in *Supercomputing, 2003 ACM/IEEE Conference* Supercomputing, 2003 ACM/IEEE Conference (2003), 15–15.

77. Shaw, D. E. *et al.* Anton, a special-purpose machine for molecular dynamics simulation. *Commun. ACM* **51,** 91 (2008).

78. Shan, Y. *et al.* How Does a Drug Molecule Find Its Target Binding Site? *J. Am. Chem. Soc.* **133,** 9181–9183 (2011).

79. Gorfe, A. A. & Caflisch, A. Functional plasticity in the substrate binding site of beta-secretase. *Structure* **13,** 1487–1498 (2005).

80. Wong, C. F., Kua, J., Zhang, Y., Straatsma, T. P. & McCammon, J. A. Molecular docking of balanol to dynamics snapshots of protein kinase A. *Proteins* **61,** 850–858 (2005).

81. Bohacek, R., Mcmartin, C., Glunz, P. & Rich, D. H. in *Rational Drug Design* (eds Truhlar, D. G., Howe, W. J., Hopfinger, A. J., Blaney, J. & Dammkoehler, R. A.) *The IMA Volumes in Mathematics and its Applications* 108, 103–114 (Springer New York, 1999).

82. Hurst, T. Flexible 3D searching: The directed tweak technique. *J. Chem. Inf. Comput. Sci.* **34,** 190–196 (1994).

83. Nicklaus, M. C., Wang, S., Driscoll, J. S. & Milne, G. W. Conformational changes of small molecules binding to proteins. *Bioorg. Med. Chem.* **3,** 411–428 (1995).

84. Yue, S. Y. Distance-constrained molecular docking by simulated annealing. *Protein Eng.* **4,** 177–184 (1990).

85. Taylor, J. S. & Burnett, R. M. DARWIN: a program for docking flexible molecules. *Proteins* **41,** 173–191 (2000).

86. Baxter, C. A., Murray, C. W., Clark, D. E., Westhead, D. R. & Eldridge, M. D. Flexible docking using Tabu search and an empirical estimate of binding affinity. *Proteins* **33,** 367–382 (1998).

87. Del Carpio-Muñoz, C. A., Ichiishi, E., Yoshimori, A. & Yoshikawa, T. MIAX: A new paradigm for modeling biomacromolecular interactions and complex formation in condensed phases. *Proteins* **48,** 696–732 (2002).

88. Jiang, F. & Kim, S. H. "Soft docking": matching of molecular surface cubes. *J. Mol. Biol.* **219,** 79–102 (1991).

89. Goldstein, H. *Classical mechanics* (Addison-Wesley Pub. Co., 1950).

90. Tirion, M. M. Large Amplitude Elastic Motions in Proteins from a Single-Parameter, Atomic Analysis. *Phys. Rev. Lett.* **77,** 1905–1908 (1996).

91. Bahar, I., Atilgan, A. R. & Erman, B. Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential. *Fold Des.* **2,** 173–181 (1997).

92. Atilgan, A. R. *et al.* Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophys. J.* **80,** 505–515 (2001).

93. Hinsen, K. Analysis of domain motions by approximate normal mode calculations. *Proteins* **33,** 417–429 (1998).

94. Sen, T. Z. & Jernigan, R. L. in *Normal Mode Analysis* 171–186 (Chapman and Hall/CRC, 2005).

95. Orellana, L. *et al.* Approaching Elastic Network Models to Molecular Dynamics Flexibility. *J. Chem. Theory Comput.* **6,** 2910–2923 (2010).

96. Tama, F., Gadea, F. X., Marques, O. & Sanejouand, Y. H. Building-block approach for determining low-frequency normal modes of macromolecules. *Proteins* **41,** 1–7 (2000).

97. Li, G. & Cui, Q. A coarse-grained normal mode approach for macromolecules: an efficient implementation and application to Ca(2+)-ATPase. *Biophys. J.* **83,** 2457–2474 (2002).

98. Lopez-Blanco, J. R., Garzón, J. I. & Chacón, P. iMod: multipurpose normal mode analysis in internal coordinates. *Bioinformatics* **27,** 2843–2850 (2011).

99. Ahmed, A., Villinger, S. & Gohlke, H. Large-scale comparison of protein essential dynamics from molecular dynamics simulations and coarse-grained normal mode analyses. *Proteins* **78,** 3341–3352 (2010).

100. Rueda, M., Chacón, P. & Orozco, M. Thorough Validation of Protein Normal Mode Analysis: A Comparative Study with Essential Dynamics. *Structure* **15,** 565–575 (2007).

101. Yang, L., Song, G., Carriquiry, A. & Jernigan, R. L. Close Correspondence between the Essential Protein Motions from Principal Component Analysis of Multiple HIV-1 Protease Structures and Elastic Network Modes. *Structure* **16,** 321–330 (2008).

102. Delarue, M. & Sanejouand, Y.-H. Simplified normal mode analysis of conformational transitions in DNA-dependent polymerases: the elastic network model. *J. Mol. Biol.* **320,** 1011–1024 (2002).

103. Yang, L., Song, G. & Jernigan, R. L. Comparisons of Experimental and Computed Protein Anisotropic Temperature Factors. *Proteins* **76,** 164–175 (2009).

104. Thomas, A., Hinsen, K., Field, M. J. & Perahia, D. Tertiary and quaternary conformational changes in aspartate transcarbamylase: a normal mode study. *Proteins* **34,** 96–112 (1999).

105. Lee, J., Joo, K., Brooks, B. R. & Lee, J. The Atomistic Mechanism of Conformational Transition of Adenylate Kinase Investigated by Lorentzian Structure-Based Potential. *J. Chem. Theory Comput.* **11,** 3211–3224 (2015).

106. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.* **21,** 1087–1092 (1953).

107. Jorgensen, W. L. & Tirado-Rives, J. Monte Carlo vs Molecular Dynamics for Conformational Sampling. *J. Phys. Chem.* **100,** 14508–14513 (1996).

108. Trosset, J.-Y. & Scheraga, H. A. Prodock: Software package for protein modeling and docking. *J. Comput. Chem.* **20,** 412–427 (1999).

109. Liu, M. & Wang, S. MCDOCK: A Monte Carlo simulation approach to the molecular docking problem. *J. Comput. Aided Mol. Des.* **13,** 435–451 (1999).

110. Borrelli, K. W., Vitalis, A., Alcantara, R. & Guallar, V. PELE: Protein Energy Landscape Exploration. A Novel Monte Carlo Based Technique. *J. Chem. Theory Comput.* **1,** 1304–1311 (2005).

111. Madadkar-Sobhani, A. & Guallar, V. PELE web server: atomistic study of biomolecular systems at your fingertips. *Nuc. Acids Res.* **41,** W322–W328 (W1 2013).

112. Schlick, T. & Fogelson, A. TNPACK—a Truncated Newton Minimization Package for Large-scale Problems: II. Implementation Examples. *ACM Trans. Math. Softw.* **18,** 71–111 (1992).

113. Jorgensen, W. L. & Tirado-Rives, J. The OPLS [optimized potentials for liquid simulations] potential functions for proteins, energy minimizations for crystals of cyclic peptides and crambin. *J. Am. Chem. Soc.* **110,** 1657–1666 (1988).

114. Jorgensen, W. L., Maxwell, D. S. & Tirado-Rives, J. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *J. Am. Chem. Soc.* **118,** 11225–11236 (1996).

115. Ponder, J. W. & Case, D. A. Force fields for protein simulations. *Adv. Protein Chem.* **66,** 27–85 (2003).

116. Ghosh, A., Rapp, C. S. & Friesner, R. A. Generalized Born Model Based on a Surface Integral Formulation. *J. Phys. Chem. B* **102,** 10983–10990 (1998).

117. Romanov, A. N. *et al.* Surface Generalized Born Method: A Simple, Fast, and Precise Implicit Solvent Model beyond the Coulomb Approximation. *J. Phys. Chem. A* **108,** 9323–9327 (2004).

118. Zhu, K., Shirts, M. R. & Friesner, R. A. Improved Methods for Side Chain and Loop Predictions via the Protein Local Optimization Program: Variable Dielectric Model for Implicitly Improving the Treatment of Polarization Effects. *J. Chem. Theory Comput.* **3,** 2108–2119 (2007).

119. Onufriev, A., Bashford, D. & Case, D. A. Exploring protein native states and large-scale conformational changes with a modified generalized born model. *Proteins* **55,** 383–394 (2004).

120. Andrec, M., Harano, Y., Jacobson, M. P., Friesner, R. A. & Levy, R. M. Complete protein structure determination using backbone residual dipolar couplings and sidechain rotamer prediction. *J. Struct. Func. Genom.* **2,** 103–111 (2002).

121. Jacobson, M. P., Kaminski, G. A., Friesner, R. A. & Rapp, C. S. Force Field Validation Using Protein Side Chain Prediction. *J. Phys. Chem. B* **106,** 11673–11680 (2002).

122. Xiang, Z., Steinbach, P. J., Jacobson, M. P., Friesner, R. A. & Honig, B. Prediction of Side-Chain Conformations on Protein Surfaces. *Proteins* **66,** 814–823 (2007).

123. Cossins, B. P., Hosseini, A. & Guallar, V. Exploration of Protein Conformational Change with PELE and Meta-Dynamics. *J. Chem. Theory Comput.* **8,** 959–965 (2012).

124. Eyal, E., Yang, L.-W. & Bahar, I. Anisotropic network model: systematic evaluation and a new web interface. *Bioinformatics* **22,** 2619–2627 (2006).

125. Kitao, A., Hirata, F. & Gō, N. The effects of solvent on the conformation and the collective motions of protein: normal mode analysis and molecular dynamics simulations of melittin in water and in vacuum. *Chem. Phys.* **158,** 447–472 (1991).

126. Meireles, L., Gur, M., Bakan, A. & Bahar, I. Pre-existing soft modes of motion uniquely defined by native contact topology facilitate ligand binding to proteins. *Protein Sci.* **20,** 1645–1658 (2011).

127. Florence Tama & Charles L. Brooks. in *Normal Mode Analysis* 111–135 (Chapman and Hall/CRC, 2005).

128. Hosseini, A. *et al.* Molecular Interactions of Prodiginines with the BH3 Domain of Anti-Apoptotic Bcl-2 Family Members. *PLoS ONE* **8,** e57562 (2013).

129. Fernández-Fueyo, E. *et al.* Structural implications of the C-terminal tail in the catalytic and stability properties of manganese peroxidases from ligninolytic fungi. *Acta Crystallogr., Sect D: Biol. Crystallogr.* **70,** 3253–3265 (Pt 12 2014).

130. Linde, D. *et al.* Catalytic surface radical in dye-decolorizing peroxidase: a computational, spectroscopic and site-directed mutagenesis study. *Biochem. J.* **466,** 253–262 (2015).

131. Hosseini, A. *et al.* Atomic picture of ligand migration in toluene 4-monooxygenase. *J. Phys. Chem. B* **119,** 671–678 (2014).

132. Takahashi, R., Gil, V. A. & Guallar, V. Monte Carlo Free Ligand Diffusion with Markov State Model Analysis and Absolute Binding Free Energy Calculations. *J. Chem. Theory Comput.* **10,** 282–288 (2014).

133. Edman, K. *et al.* Ligand Binding Mechanism in Steroid Receptors: From Conserved Plasticity to Differential Evolutionary Constraints. *Structure* **23,** 2280–2290 (2015).

134. Babot, E. D. *et al.* Steroid hydroxylation by basidiomycete peroxygenases: A combined experimental and computational study. *Appl. Environ. Microbiol.* AEM.00660–15 (2015).

135. Lucas, F. *et al.* Molecular determinants for selective C25-hydroxylation of vitamins D2 and D3 by fungal peroxygenases. *Catal. Sci. Technol.* **6,** 288–295 (2015).

136. Jones, E. M. *et al.* Differential Control of Heme Reactivity in Alpha and Beta Subunits of Hemoglobin: A Combined Raman Spectroscopic and Computational Study. *J. Am. Chem. Soc.* **136,** 10325–10339 (2014).

137. Bakken, V. & Helgaker, T. The efficient optimization of molecular geometries using redundant internal coordinates. *J. Chem. Phys.* **117,** 9160–9174 (2002).

138. Meyer, T. *et al.* MoDEL (Molecular Dynamics Extended Library): a database of atomistic molecular dynamics trajectories. *Structure* **18,** 1399–1409 (2010).

139. Humphrey, W., Dalke, A. & Schulten, K. VMD: Visual molecular dynamics. *J. Mol. Graph. Model.* **14,** 33–38 (1996).

140. Bakan, A., Meireles, L. M. & Bahar, I. ProDy: protein dynamics inferred from theory and experiments. *Bioinformatics* **27,** 1575–1577 (2011).

141. Tama, F. & Sanejouand, Y. H. Conformational change of proteins arising from normal mode calculations. *Protein Eng.* **14,** 1–6 (2001).

142. Leo-Macias, A., Lopez-Romero, P., Lupyan, D., Zerbino, D. & Ortiz, A. R. An Analysis of Core Deformations in Protein Superfamilies. *Biophys. J.* **88,** 1291–1299 (2005).

143. Hayward, S., Kitao, A. & Gō, N. Harmonic and anharmonic aspects in the dynamics of BPTI: A normal mode analysis and principal component analysis. *Prot. Sci.* **3,** 936–943 (1994).

144. Zheng, W. Anharmonic Normal Mode Analysis of Elastic Network Model Improves the Modeling of Atomic Fluctuations in Protein Crystal Structures. *Biophys. J.* **98,** 3025–3034 (2010).

145. Frappier, V. & Najmanovich, R. J. A Coarse-Grained Elastic Network Atom Contact Model and Its Use in the Simulation of Protein Dynamics and the Prediction of the Effect of Mutations. *PLoS Comput. Biol.* **10,** e1003569 (2014).

146. Lu, M., Poon, B. & Ma, J. A New Method for Coarse-Grained Elastic Normal-Mode Analysis. *J. Chem. Theory Comput.* **2,** 464–471 (2006).

147. Grebner, C. *et al.* Binding Mode and Induced Fit Predictions for Prospective Computational Drug Design. *J. Chem. Inf. Model.* (2016).

148. Oró Gay, X., Rosa M Badia & Victor A Gil. *Paral·lelització del software de simulació PELE++ utilitzant GPUs* 2012. <http://upcommons.upc.edu/handle/2099.1/15498> (visited on 17/11/2015).

149. Rincón Muñoz, A., Rosa M Badia & Víctor A. Gil. *Análisis vibracional de proteínas en coordenadas internas mediante el modelo ANM* 2014. <http://upcommons.upc.edu/handle/2099.1/24417> (visited on 18/11/2015).

150. Alvarez Vecino, P., Rosa M Badia & Victor A Gil. *Optimization of the cluster analysis tool pyProCT with pyCOMPSs* 2015. <https://upcommons.upc.edu/handle/2117/79993> (visited on 17/11/2015).

151. Eyer, L. *et al.* Nucleoside Inhibitors of Tick-Borne Encephalitis Virus. *Antimicrob. Agents Chemother.* **59,** 5483–5493 (2015).

152. Hosseini, A. *et al.* Computational Prediction of HIV-1 Resistance to Protease Inhibitors. *J. Chem. Inf. Model.* doi:10.1021/acs.jcim.5b00667.

153. Brüschweiler, R. Collective protein dynamics and nuclear spin relaxation. *J. Chem. Phys.* **102,** 3396–3403 (1995).

154. Amadei, A., Ceruso, M. A. & Di Nola, A. On the convergence of the conformational coordinates basis set obtained by the essential dynamics analysis of proteins' molecular dynamics simulations. *Proteins* **36,** 419–424 (1999).

155. BSC-CNS. *MinoTauro User's guide* 2015.

156. Diamond, R. A Note on the Rotational Superposition Problem. *Acta Crystallogr., Sect. A: Found. Crystallogr.* **A,** 211–216 (1988).

157. Berendsen, H., van der Spoel, D. & van Drunen, R. GROMACS: A message-passing parallel molecular dynamics implementation. *Comp. Phys. Comm.* **91,** 43–56 (1995).

158. Cock, P. J. A. *et al.* Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25,** 1422–1423 (2009).

159. Fedorovsky, M. *PyVib2, a program for analyzing vibrational motion and vibrational spectra* <`http://pyvib2.sourceforge.net`> (2007).

160. Ester, M., Kriegel, H.-p., S, J. & Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise, 226–231 (1996).

161. Sander, J., Ester, M., Kriegel, H.-P. & Xu, X. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Min. Knowl. Discov.* **2,** 169–194 (1998).

162. Ankerst, M., Breunig, M. M., Kriegel, H.-p. & Sander, J. *OPTICS: Ordering Points To Identify the Clustering Structure* in (ACM Press, 1999), 49–60.

163. Zhou, H., Wang, P. & Li, H. Research on Adaptive Parameters Determination in DBSCAN Algorithm. *J. Info. Comput. Sci.* **9 (7),** 1967–1973 (2012).

164. Daura, X. *et al.* Peptide Folding: When Simulation Meets Experiment. *Angew. Chem., Int. Ed. Engl.* **38,** 236–240 (1999).

165. Daura, X., van Gunsteren, W. F. & Mark, A. E. Folding-unfolding thermodynamics of a beta-heptapeptide from equilibrium simulations. *Proteins* **34,** 269–280 (1999).

166. Mullner, D. fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *J. Stat. Soft.* **53,** 1–18 (2013).

167. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28,** 129–137 (1982).

168. Kaufman, L. & Rousseeuw, P. J. *Finding groups in data: an introduction to cluster analysis* (John Wiley and Sons, New York, 1990).

169. Von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **17,** 395–416 (2007).

170. Shi, J. & Malik, J. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22,** 888–905 (2000).

171. Zelnik-manor, L. & Perona, P. *Self-tuning spectral clustering* in *Advances in Neural Information Processing Systems 17* (MIT Press, 2004), 1601–1608.

172. Michael, S., Kumar, V. & Tan, P.-N. in *Introduction to Data Mining* 487–568 (Addison-Wesley).

173. He, J., Tan, A.-H., Tan, C.-L. & Sung, S.-Y. in *Clustering and Information Retrieval Network Theory and Applications* 11, 105–133 (Springer US, 2004).

174. Davies, D. L. & Bouldin, D. W. A Cluster Separation Measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-1,** 224–227 (1979).

175. Dunn, J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Cybernet. Syst.* **3,** 32–57 (1973).

176. Kryszczuk, K. & Hurley, P. in *Multiple Classifier Systems* (eds Gayar, N. E., Kittler, J. & Roli, F.) *Lecture Notes in Computer Science* 5997, 114–123 (Springer Berlin Heidelberg, 2010).

177. Calinski, T. & Harabasz, J. A dendrite method for cluster analysis. *Commun. Stat.* **3,** 1–27 (1974).

178. Rousseeuw, P. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20,** 53–65 (C 1987).

179. Amadei, A., Linssen, A. B. & Berendsen, H. J. Essential dynamics of proteins. *Proteins* **17,** 412–425 (1993).

180. Ding, C. H. Q., He, X., Zha, H., Gu, M. & Simon, H. D. *A Min-max Cut Algorithm for Graph Partitioning and Data Clustering* in *Proceedings of the 2001 IEEE International Conference on Data Mining* (IEEE Computer Society, Washington, DC, USA, 2001), 107–114.

181. Hagen, L. & Kahng, A. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst* **11,** 1074–1085 (1992).

182. Kresten Lindorff-Larsen1, R. O. D. Stefano Piana & Shaw, D. E. How Fast-Folding Proteins Fold. *Science* **334,** 517–520 (2011).

183. Spaeth, H. *SPAETH Cluster Analysis Tools* <http://people.sc.fsu.edu/%0020jburkardt/f_src/spaeth/%0020spaeth.html> ().

184. Wersdorfer, J. *spectral clustering with python* <http://wersdoerfer.com/%0020jochen/s9y/index.php?/archives/%0020109-spectral-clustering-with-python.html> ().

185. Jacobson, M. P., Friesner, R. A., Xiang, Z. & Honig, B. On the role of the crystal environment in determining protein side-chain conformations. *J. Mol. Biol.* **320,** 597–608 (2002).

186. Cabeza de Vaca, I., Lucas, M. F. & Guallar, V. New Monte Carlo Based Technique To Study DNA-Ligand Interactions. *J. Chem. Theory Comput.* (2015).

187. Larsson, P., Hess, B. & Lindahl, E. Algorithm improvements for molecular dynamics simulations. *WIREs Comput. Mol. Sci.* **1,** 93–108 (2011).

188. BSC-CNS. *MareNostrum III User's Guide* 2015.

189. Graham, S. L., Kessler, P. B. & Mckusick, M. K. *Gprof: A Call Graph Execution Profiler* in *Proceedings of the 1982 SIGPLAN Symposium on Compiler Construction* (ACM, New York, NY, USA, 1982), 120–126. <http://doi.acm.org/10.1145/800230.806987> (visited on 17/11/2015).

190. Myung, H.-J. *et al.* Accelerating Molecular Dynamics Simulation Using Graphics Processing Unit. *Bull. Korean Chem. Soc.* **31,** 3639–3643 (2010).

191. Schmid, N., Bötschi, M. & van Gunsteren, W. F. A GPU solvent-solvent interaction calculation accelerator for biomolecular simulations using the GROMOS software. *J. Comput. Chem.* **31,** 1636–1643 (2010).

192. Jin, X., Zhao, L. & Yang, J. *A CUDA based solute interaction calculation of biomolecular simulation for GROMOS* in *2011 International Conference on Computer Science and Network Technology (ICCSNT)* 2011 International Conference on Computer Science and Network Technology (ICCSNT). **2** (2011), 820–824.

193. Xie, D. & Schlick, T. Remark on Algorithm 702—the Updated Truncated Newton Minimization Package. *ACM Trans. Math. Softw.* **25,** 108–122 (1999).

194. Wilson, E. B., Decius, J. C. & Cross, P. C. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra* 414 pp. (Courier Corporation, 2012).

195. Gordon, M. S. & Pople, J. A. Approximate Self-Consistent Molecular-Orbital Theory. VI. INDO Calculated Equilibrium Geometries. *J. Chem. Phys.* **49,** 4643–4650 (1968).

196. Li, Z. & Scheraga, H. A. Monte Carlo-minimization approach to the multiple-minima problem in protein folding. *Proc. Natl. Acad. Sci. U.S.A.* **84,** 6611–6615 (1987).

197. Jorgensen, W. L. & Tirado-Rives, J. Molecular modeling of organic and biomolecular systems using BOSS and MCPRO. *J. Comput. Chem.* **26,** 1689–1700 (2005).

198. Stein, E. G., Rice, L. M. & Brünger, A. T. Torsion-Angle Molecular Dynamics as a New Efficient Tool for NMR Structure Calculation. *J. Magn. Reson.* **124,** 154–164 (1997).

199. Güntert, P., Mumenthaler, C. & Wüthrich, K. Torsion angle dynamics for NMR structure calculation with the new program DYANA. *J. Mol. Biol.* **273,** 283–298 (1997).

200. Noguti, T. & Gō, N. A Method of Rapid Calculation of a Second Derivative Matrix of Conformational Energy for Large Molecules. *J. Phys. Soc. Jpn.* **52,** 3685–3690 (1983).

201. Noguti, T. & Gō, N. Efficient monte carlo method for simulation of fluctuating conformations of native proteins. *Biopolymers* **24,** 527–546 (1985).

202. Kidera, A. Enhanced conformational sampling in Monte Carlo simulations of proteins: application to a constrained peptide. *Proc. Natl. Acad. Sci. U.S.A.* **92,** 9886–9889 (1995).

203. Kidera, A. Smart Monte Carlo simulation of a globular protein. *Int. J. Quantum Chem* **75,** 207–214 (1999).

204. Levitt, M., Sander, C. & Stern, P. S. Protein normal-mode dynamics: Trypsin inhibitor, crambin, ribonuclease and lysozyme. *J. Mol. Biol.* **181,** 423–447 (1985).

205. Bray, J., Weiss, D. & Levitt, M. Optimized Torsion-Angle Normal Modes Reproduce Conformational Changes More Accurately Than Cartesian Modes. *Biophys. J.* **101,** 2966–2969 (2011).

206. Mendez, R. & Bastolla, U. Torsional Network Model: Normal Modes in Torsion Angle Space Better Correlate with Conformation Changes in Proteins. *Phys. Rev. Lett.* **104,** 228103 (2010).

207. Kovacs, J. A., Cavasotto, C. N. & Abagyan, R. Conformational Sampling of Protein Flexibility in Generalized Coordinates: Application to Ligand Docking. *J. Comput. Theor. Nanosci.* **2,** 354–361 (2005).

208. Ghysels, A. *et al.* Mobile Block Hessian Approach with Adjoined Blocks: An Efficient Approach for the Calculation of Frequencies in Macromolecules. *J. Chem. Theory Comput.* **5,** 1203–1215 (2009).

209. Ghysels, A. *et al.* Comparative Study of Various Normal Mode Analysis Techniques Based on Partial Hessians. *J. Comput. Chem.* **31,** 994–1007 (2010).

210. Eckart, C. Some Studies Concerning Rotating Axes and Polyatomic Molecules. *Phys. Rev. A* **47,** 552–558 (1935).

211. Noguti, T. & Gō, N. Dynamics of Native Globular Proteins in Terms of Dihedral Angles. *J. Phys. Soc. Jpn.* **52,** 3283–3288 (1983).

212. Abe, H., Braun, W., Noguti, T. & Gō, N. Rapid calculation of first and second derivatives of conformational energy with respect to dihedral angles for proteins general recurrent equations. *Comput. Chem.* **8,** 239–247 (1984).

213. Braun, W., Yoshioki, S. & Gō, N. Formulation of Static and Dynamic Conformational Energy Analysis of Biopolymer Systems Consisting of Two or More Molecules. *J. Phys. Soc. Jpn.* **53,** 3269–3275 (1984).

214. Choi, V. On updating torsion angles of molecular conformations. *J. Chem. Inf. Model.* **46,** 438–444.

215. *Maxima, a Computer Algebra System. Version 5.34.1* <`http://maxima.sourceforge.net/`> (2014).

216. Lyman, E. & Zuckerman, D. M. Ensemble-Based Convergence Analysis of Biomolecular Trajectories. *Biophys. J.* **91,** 164–172 (2006).

217. Lindorff-Larsen, K. & Ferkinghoff-Borg, J. Similarity Measures for Protein Ensembles. *PLoS ONE* **4,** e4203.

218. Chen, J. & Brooks III, C. L. Implicit modeling of nonpolar solvation for simulating protein folding and conformational transitions. *Phys. Chem. Chem. Phys.* **10,** 471–481 (2008).

219. Zhang, W., Ganguly, D. & Chen, J. Residual Structures, Conformational Fluctuations, and Electrostatic Interactions in the Synergistic Folding of Two Intrinsically Disordered Proteins. *PLoS Comput. Biol.* **8,** e1002353.

220. Boggon, T. J. & Eck, M. J. Structure and regulation of Src family kinases. *Oncogene* **23,** 7918–7927 (2004).

221. Gil, V. A. & Guallar, V. pyRMSD: a Python package for efficient pairwise RMSD matrix calculation and handling. *Bioinformatics* **29,** 2363–2364 (2013).

222. Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallogr., A, Found. Crystallogr.* **32,** 922–923 (1976).

223. Heisterberg, D. QTRFIT algorithm for superimposing two similar rigid molecules. *The Ohio Supercomputer Center Ohio State University, Columbus, OH* (1990).

224. Theobald, D. L. Rapid calculation of RMSDs using a quaternion-based characteristic polynomial. *Acta Crystallogr., A, Found. Crystallogr.* **61,** 478–480 (Pt 4 2005).

225. Umeyama, S. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **13,** 376–380 (1991).

226. Shao, J., Tanner, S. W., Thompson, N. & Cheatham, T. E. Clustering Molecular Dynamics Trajectories: 1. Characterizing the Performance of Different Clustering Algorithms. *J. Chem. Theory Comput.* **3,** 2312–2334 (2007).

227. Fraccalvieri, D., Bonati, L. & Stella, F. Self Organizing Maps to efficiently cluster and functionally interpret protein conformational ensembles. *Electron. Proc. Theor. Comput. Sci.* **130,** 83–86 (2013).

228. Phillips, J. L., Colvin, M. E. & Newsam, S. Validating clustering of molecular dynamics simulations using polymer models. *BMC Bioinformatics* **12,** 445 (2011).

229. Pande, V. S., Beauchamp, K. & Bowman, G. R. Everything you wanted to know about Markov State Models but were afraid to ask. *Methods* **52,** 99–105 (2010).

230. Gil, V. A. & Guallar, V. pyProCT: Automated Cluster Analysis for Structural Bioinformatics. *J. Chem. Theory Comput.* **10,** 3236–3243 (2014).

231. Tejedor, E. *et al.* PyCOMPSs: Parallel computational workflows in Python. *Int. J. High Perform. Comput. Appl.* 1094342015594678 (2015).

232. Tejedor, E. *et al.* A High-productivity Task-based Programming Model for Clusters. *Concurr. Comput. : Pract. Exper.* **24,** 2421–2448 (2012).

233. Lucas, M. F., Cabeza de Vaca, I., Takahashi, R., Rubio-Martínez, J. & Guallar, V. Atomic level rendering of DNA-drug encounter. *Biophys. J.* **106,** 421–429 (2014).